A Kleene Theorem for Timed Automata[*]

Eugene Asarin[†] Paul Caspi[‡] Oded Maler[‡]

Abstract

In this paper we define timed regular expressions, an extension of regular expressions for specifying sets of dense-time discrete-valued signals. We show that this formalism is equivalent in expressive power to the timed automata of Alur and Dill by providing a translation procedure from expressions to automata and vice versa. The result is extended to ω-regular expressions (Büchi's theorem).

1. Introduction

Timed automata, i.e. automata equipped with clocks [AD94], have been studied extensively in recent years as they provide a rigorous model for reasoning about the quantitative temporal aspects of systems. Together with real-time logics and process algebras they constitute the underlying theoretical basis for the specification and verification of real-time systems.

Kleene's theorem [K56], stating that the regular (or rational) subsets of Σ^* are exactly the recognizable ones (those accepted by finite automata), is one of the cornerstones of automata theory. No such theorem has been established for timed automata.[1] Numerous real-time extensions have been suggested for process algebras (i.e. linear grammars plus concurrency operators), but we failed to trace the desired simple characterization, i.e. a class of algebraic objects[2] equivalent to timed automata. Various real-time logics have been proposed and proved to be equivalent to certain classes of timed automata (e.g. [W94] and the references therein), but a characterization analogous to Kleene theorem has not been found.

The essence of Kleene's theorem is first and foremost in the definition of regular sets, constructed from letters by concatenation, union and the star operation, and in showing the equivalence of this class of sets to those recognizable by finite-state automata. Later, Büchi [B60] extended this result to ω-languages (sets of infinite sequences) by using expressions involving the ω-exponentiation operator and automata on infinite words.

In this paper we apply these old-fashioned recipes to timed automata. We use the timed automata introduced in [AD94] but slightly change the notion of their corresponding languages from timed sequences to *continuous-time discrete-valued signals*. We have chosen this semantics because we think it captures the appropriate intuition for dealing with automata operating in dense time. We define *timed regular* and *ω-regular expressions* to denote sets of signals and show that these sets are exactly what timed automata can recognize.

There are two departures from the original theorems concerning the translation from automata to expressions. First, we must sometimes employ expressions with intersection (while in the classical theorems union is sufficient). We prove that this is necessary for timed automata. Secondly, when translating an automaton over the alphabet Σ, we may create an expression over a larger alphabet Σ' such that the language of the automaton is obtained from the language of the expression via a renaming $g : \Sigma' \to \Sigma$. Whether or not this is necessary is an open question.

The rest of the paper is organized as follows: in section 2 we introduce the syntax and semantics of timed regular expressions. In section 3 we review the basic definitions of timed automata and show how to associate with them sets of signals. The translation of expressions into timed automata is presented in section 4. In section 5 we perform the more involved transformation of automata into expressions. Section 6 consists of an extension of the results to timed ω-regular expressions and signals of infinite length, while in section 7 we prove that intersection is necessary in order to express the language of certain timed automata. Some contemplations on past and future work conclude the paper.

[*]This research was supported in part by the European Community projects HYBRID EC-US-043 and INTAS-94-697.

[†]Institute for Information Transmission Problems, 19 Bol. Karetnyi per., 101447 Moscow, Russia. **asarin@ippi.ac.msk.su** The results were obtained while the author was a visiting professor at ENSIMAG, INPG, Grenoble.

[‡]VERIMAG, Centre Equation, 2, av. de Vignate, 38610 Gières, France, {**caspi,maler**}**@imag.fr**

[1]In the reference paper of Alur and Dill [AD94] the *regular* sets are defined to be those acceptable by timed automata which makes the formulation of such a theorem a tautology.

[2]We exclude, of course, algebraic objects that refer explicitly to states and clocks, and which are equivalent to timed automata almost by definition.

2. Timed Regular Expressions

Let Σ be a finite *alphabet* and let R_+ denote the set of positive reals. A *signal* over Σ is a left-continuous piecewise-constant function $\xi : (0,k] \to \Sigma$ for some $k \in \mathsf{R}_+ \cup \{0\}$ such that ξ has a *finite* number of discontinuities. Every signal can be written as

$$\xi = a_1^{r_1} a_2^{r_2} \cdots a_n^{r_n}$$

where $a_i \in \Sigma$, $r_i \in \mathsf{R}_+$, $a_i \neq a_{i+1}$ and $\sum r_i = k$. We call k the *length* of ξ and denote it by $|\xi|$. The *signature* of the signal is the string $sig(\xi) = a_1 a_2 \cdots a_n$. The set of all signals is denoted by $\mathcal{S}(\Sigma)$. For every $\xi_1, \xi_2 \in \mathcal{S}(A)$ such that $|\xi_1| = k_1$ and $|\xi_2| = k_2$ we define their concatenation as $\xi = \xi_1 \circ \xi_2$ where $\xi(t) = \xi_1(t)$ at the interval $(0, k_1]$ and $\xi_2(t - k_1)$ at the interval $(k_1, k_1 + k_2]$. This notion can be extended naturally to concatenation of sets of signals by letting

$$L_1 \circ L_2 = \{\xi_1 \circ \xi_2 : \xi_1 \in L_1 \wedge \xi_2 \in L_2\}.$$

A function $g : \Sigma_1 \to \Sigma_2$ induces in a natural way a function $g_* : \mathcal{S}(\Sigma_1) \to \mathcal{S}(\Sigma_2)$ which we call a *renaming* function.

An *integer-bounded interval* is either $[l,u]$, $(l,u]$, $[l,u)$, or (l,u) where $l \in \mathsf{N}$ and $u \in \mathsf{N} \cup \{\infty\}$ such that $l \leq u$. We exclude $\infty]$ and use l for $[l,l]$.

Definition 1 (Timed Regular Expressions) *The set* $\mathcal{E}(\Sigma)$ *of timed regular expressions over an alphabet* Σ, (expressions, for short) *is defined recursively as either* a, $\varphi_1 \cdot \varphi_2$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, φ^* *or* $\langle\varphi\rangle_I$ *where* $a \in \Sigma$, $\varphi, \varphi_1, \varphi_2 \in \mathcal{E}(\Sigma)$ *and* I *is an integer-bounded interval.*

The semantics of timed regular expressions, $[\![\,]\!] : \mathcal{E}(\Sigma) \to 2^{\mathcal{S}(\Sigma)}$, is given by:

$$\begin{array}{lcl}
[\![a]\!] & = & \{a^r : r \in \mathsf{R}_+\} \\
[\![\varphi_1 \vee \varphi_2]\!] & = & [\![\varphi_1]\!] \cup [\![\varphi_2]\!] \\
[\![\varphi_1 \wedge \varphi_2]\!] & = & [\![\varphi_1]\!] \cap [\![\varphi_2]\!] \\
[\![\varphi_1 \cdot \varphi_2]\!] & = & [\![\varphi_1]\!] \circ [\![\varphi_2]\!] \\
[\![\varphi^*]\!] & = & \bigcup_{i=0}^{\infty}([\![\varphi^i]\!]) \\
[\![\langle\varphi\rangle_I]\!] & = & [\![\varphi]\!] \cap \{\xi : |\xi| \in I\}
\end{array}$$

The novel features here with respect to classical untimed regular expressions is that we use intersection (we will show in section 7 that this is necessary for timed automata, although it is optional for untimed ones) and the $\langle\varphi\rangle_I$ operator which restricts the length of the signals in $[\![\varphi]\!]$ to be in I.

Examples:

φ	$[\![\varphi]\!]$
$\langle a\rangle_{(0,3]}$	$\{a^x : x \in (0,3]\}$
$\langle(a \cdot b)^*\rangle_{(0,3]}$	$\bigcup_k\{a^{r_1}b^{s_1} \cdots a^{r_k}b^{s_k} :$ $r_i, s_i \in \mathsf{R}_+ \wedge$ $\sum_{i=1}^k (r_i + s_i) \in (0,3]\}$
$\langle a \cdot b\rangle_3 \cdot c \wedge a \cdot \langle b \cdot c\rangle_3$	$\{a^x b^y c^z :$ $(x+y=3) \wedge (y+z=3)\}$

The subsets of $\mathcal{S}(\Sigma)$ which are expressible using timed regular expressions are called timed regular languages.

3. Timed Automata

For the sake of readers not familiar with timed automata we start with an informal illustration of the behavior of these objects. Consider the timed automaton of figure 1. It has two states and two clocks c_1 and c_2. Suppose it starts operating in the configuration $(q_1, 0, 0)$ (the two last coordinates denote the values of the clocks). When the automaton stays at q_1, the values of the clocks grow with the unit slope. After one second, the condition $c_1 \geq 1$ (the guard of the transition from q_1 to q_2) is satisfied and the automaton can move to q_2 while resetting c_2 to 0. Having entered q_2 at a configuration $(q_2, t, 0)$ for some t, the automaton can either stay there or can unconditionally move to q_1 and reset the two clocks. By fixing some initial and final states, and by assigning a letter from Σ to each state, we can turn timed automata into generators/acceptors of timed languages, i.e. sets of signals.

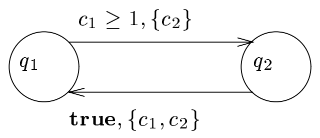

Figure 1. A timed automaton.

Definition 2 (Timed Automaton) *A timed automaton is a tuple* $\mathcal{A} = (Q, C, \Delta, \Sigma, \lambda, S, F)$ *where* Q *is a finite set of states,* C *is a finite state of clocks,* Σ *is an output alphabet,* Δ *is a transition relation (see below),* $\lambda : Q \to \Sigma$ *is an output map,* $S \subseteq Q$ *an initial set and* $F \subseteq Q$ *an accepting set. An element of the transition relation is of the form* (q, ϕ, ρ, q') *where* q *and* q' *are states,* $\rho \subseteq C$ *and* ϕ *(the transition guard) is a boolean combination of formulae of the form* $(c \in I)$ *for some clock c and some integer-bounded interval* I.

A *clock valuation* is a function $\mathbf{v} : C \to \mathsf{R}_+ \cup \{0\}$ (which is the same a vector $\mathbf{v} \in (\mathsf{R}_+ \cup \{0\})^{|C|}$). We denote the set of all clock valuations by $\mathcal{H}$. For a clock valuation $\mathbf{v}$ and a set $\rho \subseteq C$ we put for any clock variable $c \in C$

$$Reset_\rho\, \mathbf{v}(c) = \begin{cases} 0 & if \quad c \in \rho \\ \mathbf{v}(c) & if \quad c \notin \rho \end{cases}$$

That is, $\mathrm{R}eset_\rho$ resets to zero all the clocks in ρ and leaves the other clocks unchanged. We use $\mathbf{1}$ to denote the unit vector $(1,\ldots,1)$.

A *finite run* of the automaton is a sequence

$$(q_0,\mathbf{v}_0) \xrightarrow[t_1]{\delta_1} (q_1,\mathbf{v}_1) \xrightarrow[t_2]{\delta_2} \ldots \xrightarrow[t_n]{\delta_n} (q_n,\mathbf{v}_n),$$

where $q_i \in Q, \mathbf{v}_i \in \mathcal{H}, \delta_i \in \Delta, t_i \in \mathsf{R}_+$, and which satisfies the following conditions:

Time progress: $0 < t_1 < \ldots < t_n$ (for convenience we put $t_0 = 0$);

Succession: If $\delta_i = (q,\phi,\rho,q')$ then $q_{i-1} = q, q_i = q'$, the condition $\phi(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$ holds and $\mathbf{v}_i = Reset_\rho(\mathbf{v}_{i-1} + (t_i - t_{i-1})\mathbf{1})$.

An *accepting* run is a run satisfying the additional conditions:

Initialization: $q_0 \in S; \mathbf{v}_0 = \mathbf{0}$;

Termination: $q_n \in F$.

The *trace* of such a run is the signal

$$\lambda(q_0)^{t_1} \circ \lambda(q_1)^{t_2-t_1} \circ \cdots \circ \lambda(q_{n-1})^{t_n-t_{n-1}}$$

which is defined on $(0,t_n]$. Notice that $\lambda(q_n)$ is not included. The *language of a timed automaton*, $L(\mathcal{A})$, consists of all the traces of its accepting runs.

4. From Expressions to Automata

We construct automata by induction on the structure of the expression. This construction is straightforward and mimics the classical one [MY60]. We show that for every timed regular expression we can build a timed automaton with the following additional property: there are no transitions outgoing from any accepting states, and hence the definition of $\lambda(q)$ for $q \in F$ is not important.

Before giving the formal definition let us explain the construction intuitively (see also figure 2). The automaton for a can move at any time from an initial a-state to a final state. For union of two languages we just run the automata in parallel. For concatenation, we replace any transition to a final state of the first automaton by transitions to the initial states of the second automaton (while resetting its clocks). Similarly for the $*$-operation we add transitions to initial state for every transition leading to F. For the $\langle\varphi\rangle_I$ operator we introduce a new clock c and add a test $(c \in I)$ to the guard of every transition leading to F. Finally for intersection we do the usual Cartesian product (with a slight variation due to the fact that letters are assigned to states and not to transitions).

Let $\mathcal{A}_1 = (Q_1, C_1, \Delta_1, \Sigma, \lambda_1, S_1, F_1)$ and $\mathcal{A}_2 = (Q_2, C_2, \Delta_2, \Sigma, \lambda_2, S_2, F_2)$ be the timed automata accepting the languages $[\![\varphi_1]\!]$ and $[\![\varphi_2]\!]$ respectively.

- The automaton for $[\![a]\!]$ for every $a \in \Sigma$ is $(\{q_1,q_2\}, \emptyset, \Delta, \Sigma, \lambda, \{q_1\}, \{q_2\})$ where $\Delta = \{(q_1, \mathbf{true}, \emptyset, q_2)\}$ and $\lambda(q_1) = a$.
- The automaton for $[\![\varphi_1 \vee \varphi_2]\!]$ is $(Q_1 \cup Q_2, C_1 \cup C_2, \Delta_1 \cup \Delta_2, \Sigma, \lambda_1 \cup \lambda_2, S_1 \cup S_2, F_1 \cup F_2)$.
- The automaton for $[\![\varphi_1 \wedge \varphi_2]\!]$ is $(Q, C_1 \cup C_2, \Delta, \Sigma, \lambda, S, F)$ where $Q = \{\langle q_1, q_2\rangle \in Q_1 \times Q_2 : \lambda_1(q_1) = \lambda_2(q_2)\} \cup F_1 \times F_2$, $\Delta = \{(\langle q_1,q_2\rangle, \phi_1 \wedge \phi_2, \rho_1 \cup \rho_2, \langle q_1', q_2'\rangle) : (q_1, \phi_1, \rho_1, q_1') \in \Delta_1 \; and \; (q_2, \phi_2, \rho_2, q_2') \in \Delta_2\}$, $\lambda(\langle q_1,q_2\rangle) = \lambda_1(q_1) = \lambda_2(q_2)$, $S = Q \cap (S_1 \times S_2)$ and $F = (F_1 \times F_2)$.
- The automaton for $[\![\varphi_1 \cdot \varphi_2]\!]$ is $(Q_1 \cup Q_2 - F_1, C_1 \cup C_2, \Delta, \Sigma, \lambda_1 \cup \lambda_2, S_1, F_2)$ where Δ is constructed from $\Delta_1 \cup \Delta_2$ as follows: every transition (q_1, ϕ, ρ, q_1') in Δ_1 such that $q_1' \in F_1$ is removed and replaced by a set of transitions of the form $(q_1, \phi, \rho \cup C_2, q_2)$ for every $q_2 \in S_2$.
- The automaton for $[\![\varphi_1^*]\!]$ is $\mathcal{A} = (Q_1, C_1, \Sigma, \Delta, \lambda_1, S_1 \cup F_1, F_1)$ where Δ is constructed from Δ_1 by adding for every transition of the form (q, ϕ, ρ, q') in Δ_1 such that $q' \in F_1$ a set of transitions of the form (q, ϕ, C, q'') for every $q'' \in S_1$.
- The automaton for $[\![\langle\varphi_1\rangle_I]\!]$ is $\mathcal{A} = (Q_1, C_1 \cup \{c\}, \Delta, \Sigma, \lambda_1, S_1, F_1)$ where Δ is obtained from Δ_1 by replacing every transition of the form (q, ϕ, ρ, q') in Δ_1 such that $q' \in F_1$ by $(q, \phi \wedge (c \in I), \rho, q')$.

This concludes the construction which gives one side of Kleene theorem:

Theorem 1 (Exp $\Rightarrow$ Aut) *Every timed regular language can be accepted by a timed automaton.*

5. From Automata to Expressions

5.1. From Timed Automata to One-Clock Automata

In order to prove the main result we first perform a sequence of language-preserving transformations on the automaton. Each of these transformations eliminates an undesirable feature of the automaton as a preparation for the translation into expressions. Then we “determinize” the automaton by assigning a distinct letter to every state. After

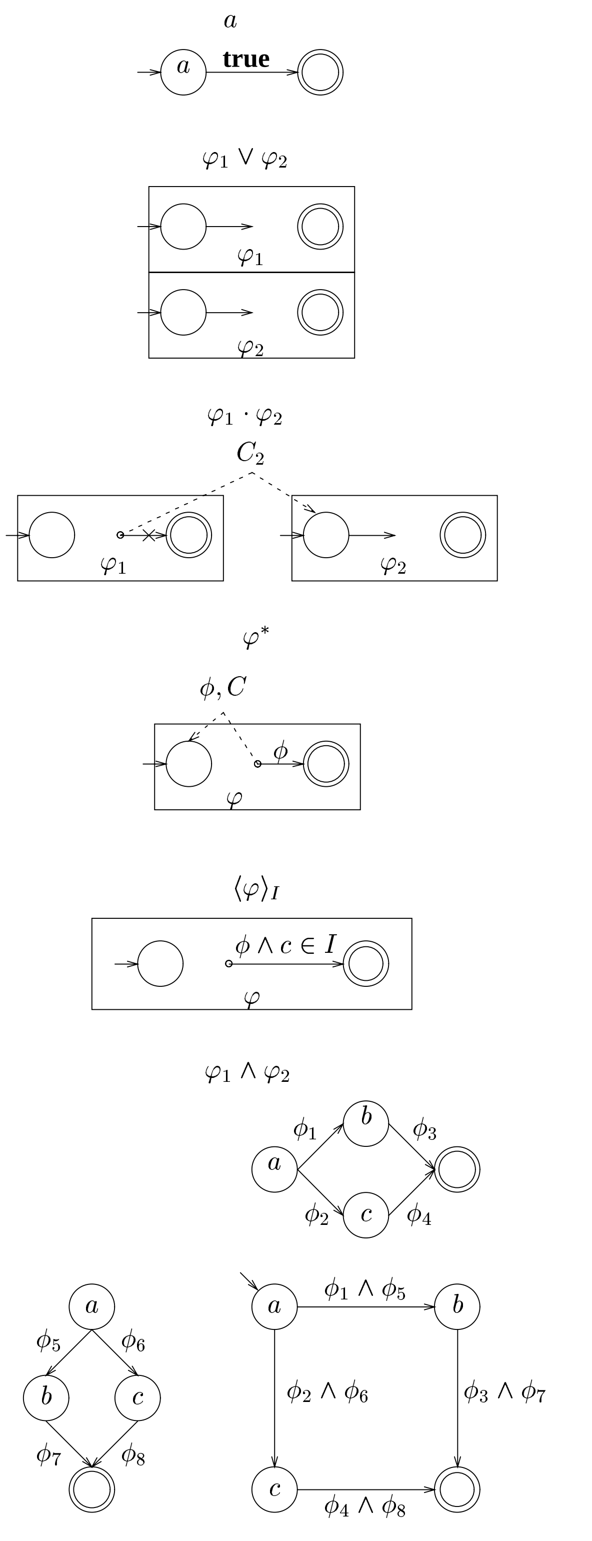

Figure 2. Constructing automata from expressions.

having done all this we can separate the automaton into several one-clock automata for which it is (relatively) easy to prove the theorem.

State-reset automata: A timed automaton is *state-reset* if all the transitions entering a given state reset exactly the *same set of clocks*. In order to make an automaton state-reset we split every state not satisfying this property into several copies and redirect the incoming transitions to each copy according to the sets of clocks reset by those transitions.

Self-loop free automata: An automaton is *self-loop free* if it contains no transitions of the form (q, ϕ, ρ, q). An automaton becomes self-loop free by splitting every self-looping state into two copies.

Disjunction free automata: An automaton is *disjunction free* if for every transition (q, ϕ, ρ, q'), the formula ϕ is a conjunction of simple tests ($c \in I$) and their negations.[3] In order to get rid of the disjunctions we first convert every transition guard into a disjunctive normal form (DNF) $\phi = \phi_1 \vee \phi_2 \vee \ldots \phi_k$ where every ϕ_i is a conjunction. We split every state q into m copies $q_1, \ldots, q_m$ where m is the maximum (over all the incoming transitions) of the number of disjuncts in the DNF formula for a guard. We then replace every transition $\delta = (q, \phi, \rho, q')$, where $\phi = \phi_1 \vee \phi_2 \vee \ldots \phi_k$ by k transitions of the form (q, ϕ_i, ρ, q'_i), $i = 1, \ldots, k$.

In all the state-splittings described above, the copies retain the same outgoing transitions, the same letter and the same initial/final status as their original state. Hence the original automaton is "homomorphic" to the new one and accepts the same language.

State-output automata: An automaton is *state-output* if $\Sigma = Q$ and λ is the identity function on Q. Given a state-reset, self-loop free and disjunction free automaton $\mathcal{A} = (Q, C, \Delta, \Sigma, \lambda, S, F)$ we convert it into a state-output $\mathcal{A}' = (Q, C, \Delta, Q, Id, S, F)$. Clearly, $L(\mathcal{A}) = \lambda_*(L(\mathcal{A}'))$.

From now on we assume that all automata have already been transformed into a state-reset, self-loop free, disjunction free and state-output form.

One-clock automata: A one-clock automaton, as its name implies, has only one clock.

Lemma 1 *Let $\mathcal{A} = (Q, C, \Delta, Q, Id, S, F)$ be a timed automaton where $C = \{C_1, \ldots, C_k\}$. One can build k one-clock automata $\mathcal{A}_1, \ldots, \mathcal{A}_k$ such that $L(\mathcal{A}) = \bigcap_{i=1}^{k} L(\mathcal{A}_i)$.*

Proof: We separate $\mathcal{A}$ into k automata $\mathcal{A}_i = (Q, \{c_i\}, \Delta_i, Q, Id, S, F)$ such that for every $(q, \phi, \rho, q') \in \Delta$ there is $(q, \phi_i, \rho_i, q') \in \Delta_i$ such that $\rho_i = \rho \cap \{c_i\}$ and ϕ_i is obtained from ϕ by substituting **true** in every occurrence of $c_j \in I$ or its negation for $j \neq i$. In other words,

[3]In fact, it is sufficient to eliminate only disjunctions of conditions referring to *different* clocks such as $(c_1 \in I_1) \vee (c_2 \in I_2)$ but our current goal is to optimize readability rather than the number of states.

every $\mathcal{A}_i$ respects only the constraints imposed by the clock c_i and ignores the rest of the clocks. Since every state has a different letter, every accepted word is a trace of exactly one run, and this is the same run in every $\mathcal{A}_i$. A run is possible in every $\mathcal{A}_i$ iff it is possible in $\mathcal{A}$. ∎

5.2. Reset-free Automata and Sampling

We prove regularity of languages accepted by one-clock automata in two stages, first we show regularity of runs that do not make transitions that reset the clock. For this we need yet another important construct.

Let $\mathcal{A} = (Q, \{c\}, \Delta, Q, Id, S, F)$ be a one-clock automaton and let $\tau_1 < \ldots < \tau_n$ be all the constants appearing in the transition guards (*critical points*). With every signal $\xi \in \mathcal{S}(Q)$ we associate its *sampling* $\sigma(\xi) = s_1 s_2 \ldots s_m \in (Q \times Q)^*$ where $m = \max\{j : |\xi| \geq \tau_j\}$. The sampling specifies what the signal is doing at every critical point. In particular $s_i = (q, q)$ means that the signal does not change its value at τ_i and keeps it equal to q. On the other hand $s_i = (q, q')$ where $q \neq q'$ means that the signal does change its value from q to q' exactly at τ_i (and q' is the left limit of ξ at τ_i). Formally $s_i = (\xi(\tau_i), \xi(\tau_i +))$ where

$$\xi(t+) = \begin{cases} \lim_{\varepsilon \downarrow 0} \xi(t + \varepsilon) & \mathrm{i}f \quad t < |\xi| \\ f & \mathrm{i}f \quad t = |\xi| \end{cases}$$

(The second case covers signals whose domain of definition is exactly some $(0, \tau_i]$ and can be ignored by the reader).

Since the length of the sampling is at most n, there are finitely many possible samplings and sampling equivalence induces a finite partition on $\mathcal{S}(Q)$. For any language M and a sampling ς let $M_\varsigma = M \cap \{\xi | \sigma(\xi) = \varsigma\}$. Clearly $M = \bigcup\limits_{\varsigma} M_\varsigma$.

Claim 2 *The language accepted by a reset-free one-clock automaton with one initial and one accepting state is regular.*

Proof: Let $\mathcal{A} = (Q, \{c\}, \Delta, Q, Id, \{s\}, \{f\})$ be such an automaton and let $M = L(\mathcal{A})$. We will show that for every sampling $\varsigma = s_1 s_2 \ldots s_m$, M_ς is regular. Note that since there are no resets, the value of the clock at time t is always t. For every i let $s_i = (q_i, q'_i)$. If for some i such that $q_i \neq q'_i$ the transition $q_i \to q'_i$ is impossible at time τ_i (i.e. there is no $\delta = (q, \phi, \rho, q') \in \Delta$ such that $\phi(\tau_i) =$ **true**), we say that ς is *inconsistent* with $\mathcal{A}$, the language M_ς is empty and we are done. Suppose this is not the case and ς is consistent with $\mathcal{A}$. Constructing a regular expression for M_ς is a 2-step procedure. The first step consists in slicing any signal in M_ς into parts corresponding to time intervals $(\tau_i, \tau_{i+1}]$. Formally, let L_i, $i = 0, \ldots, m-1$ be the language consisting of all the signals ζ such that

- $|\zeta| = \tau_{i+1} - \tau_i$ (we put $\tau_0 = 0$ and $\tau_{n+1} = \infty$);
- $\zeta(0+) = q'_i$ (we put $q'_0 = s$);
- $\zeta(\tau_{i+1} - \tau_i) = q_{i+1}$;
- ζ is a trace of a run of $\mathcal{A}$ (not necessarily accepting) starting with the clock value τ_i.

The last slice corresponding to the time interval $(\tau_m, |\xi|] \subset (\tau_m, \tau_{m+1})$ needs a special treatment. L_m is the language consisting of all the signals ζ such that

- $|\zeta| < \tau_{m+1} - \tau_m$;
- $\zeta(0+) = q'_m$;
- ζ is a trace of a run of $\mathcal{A}$ starting with the clock value τ_m and terminating with a transition to f .

The following result is now immediate from the definitions.

Lemma 2 *If ς is consistent with $\mathcal{A}$ then*

- *if $q'_m \neq f$ (which means that the lengths of signals in M_ς are in the interval (τ_m, τ_{m+1}) and never equal τ_m), then*
$$M_\varsigma = \bigcirc_{i=0}^{m} L_i;$$
- *if $q'_m = f$ (which means that some signals in M_ς may have a duration exactly τ_m), then*
$$M_\varsigma = \bigcirc_{i=0}^{m-1} L_i \cup \bigcirc_{i=0}^{m} L_i;$$

Consider now any of the languages L_i (this is the second step of the procedure). The signals $\zeta \in L_i$ are traces of the runs of $\mathcal{A}$ such that during all the run the clock stays in the interval (τ_i, τ_{i+1}). Note that (because of the appropriate choice of the critical points τ_i) the truth-values of the guards of $\mathcal{A}$'s transitions remain constant during all the (τ_i, τ_{i+1}) interval, which means that the runs of $\mathcal{A}$ referred to in the definition of L_i are essentially runs of an *untimed* automaton and the only timing restriction concerns the length of the signals. The following lemma holds.

Lemma 3 *Let $\bar{L}_i$ be the signature of L_i. Then*

$$L_i = \{\zeta : sig(\zeta) \in \bar{L}_i \wedge |\zeta| \in I_i\},$$

where

$$I_i = \begin{cases} \tau_{i+1} - \tau_i & \mathrm{i}f \quad i = 0, \ldots, m-1 \\ (0, \tau_{m+1} - \tau_m) & \mathrm{i}f \quad i = m. \end{cases}$$

The fact that $\mathcal{A}$ behaves like an untimed automaton when its clock stays in (τ_i, τ_{i+1}) allows to apply the untimed version of Kleene theorem.

Lemma 4 *The untimed languages $\bar{L}_i$ are regular.*

Proof: Consider the untimed automaton $\mathcal{B}_i = (Q, \Delta_i)$ where $(q, q') \in \Delta_i$ iff there exists a $\delta = (q, \phi, \emptyset, q') \in \Delta$ such that $\phi(t) = \textbf{true}$ for $t \in (\tau_i, \tau_{i+1})$. Then $\bar{L}_i$, $i = 0, 1, \ldots, m-1$ is the set of traces of all the runs of $\mathcal{B}_i$ starting in q'_i and finishing in q_{i+1} and it is regular by virtue of Kleene theorem. Regularity of $\bar{L}_m$ can be stated similarly. ▟

Now let φ_i be an untimed regular expression such that $[\![\varphi_i]\!] = \bar{L}_i$. Applying Lemmata 2–4 we obtain the regular expression for M_ς:

$$M_\varsigma = [\![\langle\varphi_0\rangle_{I_0} \cdot \langle\varphi_1\rangle_{I_1} \cdots \langle\varphi_m\rangle_{I_m}]\!]$$

in the first case of Lemma 2 and

$$M_\varsigma = [\![\langle\varphi_0\rangle_{I_0} \cdots \langle\varphi_{m-1}\rangle_{I_{m-1}} \vee \langle\varphi_0\rangle_{I_0} \cdots \langle\varphi_m\rangle_{I_m}]\!]$$

in the second one. This concludes the proof of the claim as we can write the expression for the whole automaton as a disjunction of the expressions for the consistent samplings. ▟

5.3. Automata with Resets

Having stated the regularity of the set of signals whose runs never reset the clock we can proceed by an inductive argument that adds one more resetting state at a time. The idea is adapted from one of the proofs of the untimed Kleene theorem (see [HU79]) described shortly below. Suppose the states of the automaton are $\{q_1, \ldots, q_n\}$. We let $R^k_{i,j}$ denote the set[4] of sequences leading from q_i to q_j without passing at states in $\{q_{k+1}, \ldots, q_n\}$ (q_i and q_j themselves *can* be outside $\{q_1, \ldots, q_k\}$). Clearly $R^n_{i,j}$ consists of *all* the sequences leading from q_i to q_j. The basis of the induction, $R^0_{i,j}$, is simply the set of letters that lead directly from q_i to q_j. The induction is via the formula

$$R^{k+1}_{i,j} = R^k_{i,j} \cup R^k_{i,k+1} \cdot (R^k_{k+1,k+1})^* \cdot R^k_{k+1,j}.$$

The first term stands for sequences leading from q_i to q_j without ever visiting q_{k+1}. The second term denotes words that lead from q_i to q_{k+1} (without visiting q_{k+1}), followed by an arbitrary number of words that induce a cycle from q_{k+1} to itself and ending with a word leading from q_{k+1} to q_j. This procedure derives the regular expression for every $R_{i,j}$.

We apply this idea to one-clock automata in the following way. As a basis we have the languages that lead from every q_i to q_j without passing in any resetting state. Then we inductively add a passage in one more resetting state (which goes well with the star operation) until we get the language of the whole automaton.

[4]Here we do not make the purist distinction between sets and the expressions they denote.

Claim 3 *The language accepted by any one-clock automaton is regular.*

Proof: Let $\mathcal{A} = (Q, \{c\}, \Delta, Q, Id, S, F)$ have n non-resetting and r resetting states. Without loss of generality suppose that $q_1, \ldots, q_n$ are non-resetting and $q_{n+1}, \ldots, q_{n+r}$ are resetting.

We introduce some auxiliary languages. Let $R_{i,j}$ be the set of the traces of all the runs of $\mathcal{A}$ starting in q_i and finishing by a transition to q_j, and $R^{(k)}_{i,j}$ the set of the traces of all such runs using only $q_1, \ldots, q_k$ as intermediate states. Clearly $R_{i,j} = R^{(n+r)}_{i,j}$ and

$$L(\mathcal{A}) = \bigcup_{q_i \in S, q_j \in F} R_{i,j},$$

so it is sufficient to prove regularity and to find timed regular expression for $R^{(k)}_{i,j}$.

To construct this expression we use the induction on k starting from the level $k = n$ (recall that n is the number of non-resetting states).

For the base of induction we construct a non-resetting automaton $\mathcal{A}_{i,j}$ such that $R^{(n)}_{i,j} = L(\mathcal{A}_{i,j})$. The automaton is just the non-resetting part of $\mathcal{A}$ and two special states s and f representing q_i and q_j respectively. Formally

$$\mathcal{A}_{i,j} = (Q', \{c\}, \Delta', Q, Id, \{s\}, \{f\}),$$

where $Q' = \{q_1, \ldots q_n\} \cup \{s, f\}$ and $\delta = (q, \phi, \emptyset, q') \in \Delta'$ iff one of the following holds:

- $q, q' \in \{q_1, \ldots q_n\}$ and $\delta \in \Delta$;
- $q = s, q' \in \{q_1, \ldots q_n\}$ and $(q_i, \phi, \emptyset, q') \in \Delta$;
- $q \in \{q_1, \ldots q_n\}, q' = f$ and $(q, \phi, \rho, q_j) \in \Delta$ for some ρ;
- $q = s, q' = f$ and $(q_i, \phi, \rho, q_j) \in \Delta$ for some ρ.

It is easy to see that $R^{(n)}_{i,j} = L(\mathcal{A}_{i,j})$, hence by virtue of claim 2 this language is regular.

The inductive step is easy: when $k \geq n$ the state q_{k+1} is resetting and hence the following equality holds

$$R^{(k+1)}_{i,j} = R^{(k)}_{i,j} \cup R^{(k)}_{i,k+1} \cdot (R^{(k)}_{k+1,k+1})^* \cdot R^{(k)}_{k+1,j}.$$

This concludes the proof of the claim. ▟

Collecting all the results in the section we obtain

Theorem 4 (Aut ⇒ Exp) *Every language accepted by a timed automaton is a renaming of a timed regular language.*

Corollary 5 (Kleene theorem for timed automata) *Timed automata and timed regular expressions have the same expressive power modulo renaming.*

Example: Consider the automaton $\mathcal{A}$ in figure 3. This automaton is already state-reset and self-loop free. Getting rid of disjunctions and making it state-output we obtain $\mathcal{A}'$. By splitting b into b_1 and b_2 we get the state-output automaton $\mathcal{A}''$ which is separated into two one-clock automata $\mathcal{A}_1$ and $\mathcal{A}_2$.

For $\mathcal{A}_1$ the non-resetting states are b_1 and b_2. Applying the procedure of claim 2 we obtain the following languages:

R	a	b_1	b_2
a	$\langle a\rangle_{(3,\infty)} \cdot b_1 \vee a \cdot b_2$	$\langle a\rangle_{(3,\infty)}$	a
b_1	b_1	$\emptyset$	$\emptyset$
b_2	b_2	$\emptyset$	$\emptyset$

For $\mathcal{A}_2$ the non-resetting state is a. Applying the procedure of claim 2 we obtain the following languages:

R	a	b_1	b_2
a	$\emptyset$	a	$\langle a\rangle_{(0,5)}$
b_1	b_1	$b_1 \cdot a$	$\langle b_1 \cdot a\rangle_{(0,5)}$
b_2	b_2	$b_2 \cdot a$	$\langle b_2 \cdot a\rangle_{(0,5)}$

Then by applying the procedure of claim 3 we have:

$$L(\mathcal{A}_1) \quad = \quad [\![(\langle a\rangle_{(3,\infty)} \cdot b_1 \vee a \cdot b_2)^*]\!]$$

and

$$\begin{aligned} L(\mathcal{A}_2) \quad = \quad & [\![(a \cdot b_1)^* \vee \\ & (\langle a\rangle_{(0,5)} \vee a \cdot (b_1 \cdot a)^* \langle b_1 \cdot a\rangle_{(0,5)}) \cdot \\ & (\langle b_2 \cdot a\rangle_{(0,5)} \vee b_2 \cdot a(b_1 \cdot a)^* \langle b_1 \cdot a\rangle_{(0,5)})^* \cdot \\ & (b_2 \cdot (a \cdot b_1)^*)]\!] \end{aligned}$$

While $L(\mathcal{A}) = g_*(L(\mathcal{A}_1) \cap L(\mathcal{A}_2))$ where $g(a) = a$ and $g(b_1) = g(b_2) = b$.

6. Büchi Theorem

First we have to extend the definitions of signals, expressions and accepting runs:

Signals

An ω-*signal* over Σ is a left-continuous piecewise-constant function $\xi : \mathsf{R}_+ \to \Sigma$ such that ξ has a finite number of discontinuities in any bounded sub-interval of R_+. The set of all such signals is denoted by $\mathcal{S}_\omega(\Sigma)$. The concatenation $\xi_1 \circ \xi_2$ where $\xi_1 \in \mathcal{S}(A)$ and $\xi_2 \in \mathcal{S}_\omega(A)$ is defined almost as before, resulting in an ω-signal. For an infinite sequence $\xi_1, \xi_2, \ldots$ of signals such that $\sum_{i=1}^{\infty} |\xi_i| = \infty$, their infinite concatenation $\bigcirc_{i=1}^{\infty} \xi_i$ is the ω-signal which coincides with

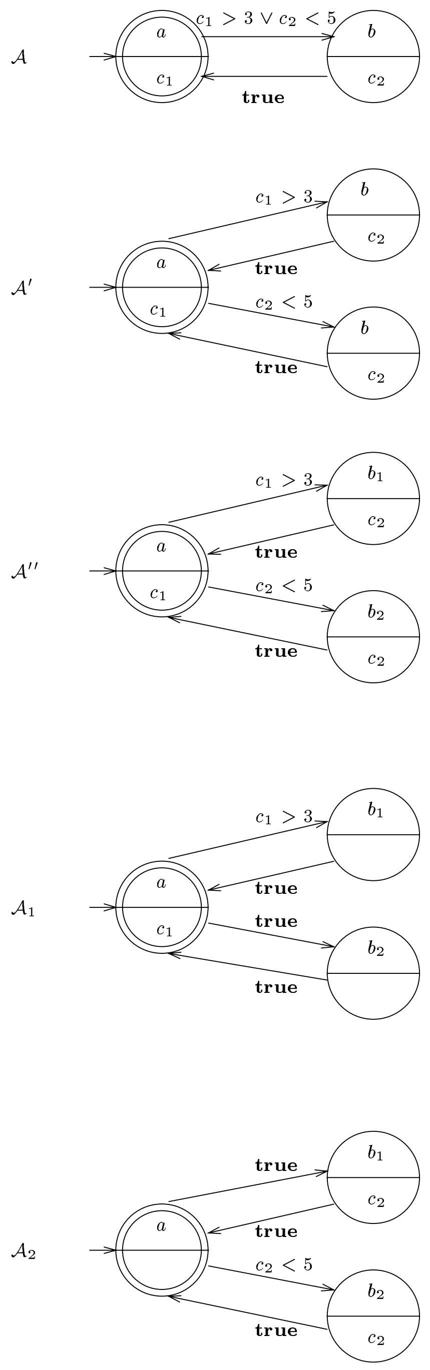

Figure 3. Constructing an expression from an automaton. The output letters appear on the upper part of the state while the clocks reset upon entering appear in the lower part.

each ξ_i in the interval $(l_i, l_{i+1}]$ where $l_i = \sum_{j=1}^{i-1} |\xi_j|$. Note that when extending this definition to sets of ω-signals, by letting

$$\bigodot_{i=1}^{\infty} L_i = \{\bigodot_{i=1}^{\infty} \xi_i : \xi_i \in L_i\}$$

we do not allow an arbitrary choice of ξ_i's but only those whose sum of lengths diverges. This prevents the so-called Zeno phenomenon where an infinite concatenation results in a signal of bounded length.

Expressions

The set of timed ω-regular expressions $\mathcal{E}_\omega(\Sigma)$ is defined as either φ^ω, $\varphi \cdot \psi$, $\psi_1 \vee \psi_2$ or $\psi_1 \wedge \psi_2$ where $\varphi \in \mathcal{E}(\Sigma)$ and $\psi, \psi_1, \psi_2 \in \mathcal{E}_\omega(\Sigma)$.

For timed ω-regular expressions we define the semantic function $[\![]\!]_\omega : \mathcal{E}(\Sigma) \to 2^{\mathcal{S}_\omega(\Sigma)}$ as:

$$\begin{array}{lcl}
[\![\psi_1 \vee \psi_2]\!]_\omega & = & [\![\psi_1]\!]_\omega \cup [\![\psi_2]\!]_\omega \\
[\![\psi_1 \wedge \psi_2]\!]_\omega & = & [\![\psi_1]\!]_\omega \cap [\![\psi_2]\!]_\omega \\
[\![\varphi \cdot \psi]\!]_\omega & = & [\![\varphi]\!] \circ [\![\psi]\!]_\omega \\
[\![\varphi^\omega]\!]_\omega & = & \bigodot_{i=1}^{\infty} [\![\varphi]\!]
\end{array}$$

Acceptance

An *ω-run* of the automaton is an infinite sequence

$$(q_0, \mathbf{v}_0) \xrightarrow[t_1]{\delta_1} (q_1, \mathbf{v}_1) \xrightarrow[t_2]{\delta_2} (q_2, \mathbf{v}_2) \ldots,$$

where $q_i \in Q, \mathbf{v}_i \in \mathcal{H}, \delta_i \in \Delta, t_i \in \mathsf{R}_+$, and which satisfies the following conditions:

Time progress: $0 < t_1 < t_2 < \ldots$ and $\lim_{i\to\infty} t_i = \infty$,

Succession: same as for finite runs;

An accepting run satisfies in addition the following conditions:

Initialization: same as for finite runs;

Büchi acceptance: $q_i \in F$ for infinitely many i.

The *trace* of this ω-run is the ω-signal defined on $(0, \infty)$ and equal to $\lambda(q_i)$ on $(t_i, t_{i+1}]$. The *ω-language* of a timed automaton, $L_\omega(\mathcal{A})$, consists of all the traces of its accepting ω-runs.

Let us extend the results of the paper ω-languages and ω-automata.

From ω-expressions to ω-automata

As in the finitary case the inductive construction is rather straightforward. As a basis we take the automaton for any finitary timed regular expression. From theorem 1 we can assume that timed regular languages are accepted by automata without transitions outgoing from accepting states. The ω-exponentiation is similar to the $*$ where each transition to an accepting state is replaced by transitions to special copies of the initial states that serve as new accepting states. These states are visited infinitely-often in the ω-automaton iff infinitely many finite factors of the signals lead from S to F in the finitary automaton. The concatenation of a language and an ω-language, as well as the union of two ω-languages are almost identical to the finitary case. Intersection requires some more details (because the visits of a signal in the accepting states of two automata need not be simultaneous). All the constructions are minor adaptations of their untimed analogues ([T90]).

Let $\mathcal{A}_1 = (Q_1, C_1, \Delta_1, \Sigma, \lambda_1, S_1, F_1)$ and $\mathcal{A}_2 = (Q_2, C_2, \Delta_2, \Sigma, \lambda_2, S_2, F_2)$ be the timed ω-automata accepting the ω-languages $[\![\psi_1]\!]$ and $[\![\psi_2]\!]$ respectively, and let $\mathcal{A} = (Q, C, \Delta, \Sigma, \lambda, S, F)$ accept the language $[\![\varphi]\!]$.

- The automaton for $[\![\varphi^\omega]\!]$ is $((Q \cup S') - F, C, \Sigma, \Delta', \lambda', S, S')$ where S' is a new set of states $\{s'_1, \ldots s'_k\}$ having one-to-one correspondence with S. Δ' is constructed from Δ by replacing every transition of the form (q, ϕ, ρ, q') in Δ such that $q' \in F$ by a set of transitions of the form (q, ϕ, C, s') for every $s' \in S'$. In addition, for every transition $(s_i, \phi, C, q) \in \Delta$ such that $s_i \in S$ we add to Δ' a transition (s'_i, ϕ, C, q), $s'_i \in S'$. For every $s'_i \in S'$ we let $\lambda'(s'_i) = \lambda(s_i)$.

- The automaton for $[\![\psi_1 \vee \psi_2]\!]$ is $(Q_1 \cup Q_2, C_1 \cup C_2, \Delta_1 \cup \Delta_2, \Sigma, \lambda_1 \cup \lambda_2, S_1 \cup S_2, F_1 \cup F_2)$.

- The automaton for $[\![\varphi_1 \wedge \varphi_2]\!]$ is $(Q \times \{1,2,3\}, C_1 \cup C_2, \Delta, \Sigma, \lambda, S, F)$ where $Q = \{\langle q_1, q_2\rangle \in Q_1 \times Q_2 : \lambda_1(q_1) = \lambda_2(q_2)\}$, Δ is constructed from Δ_1 and Δ_2 in the following way: for every $(q_1, \phi_1, \rho_1, q'_1) \in \Delta_1$ and $(q_2, \phi_2, \rho_2, q'_2) \in \Delta_2$ we put in Δ the transitions $(\langle q_1, q_2, i\rangle, \phi_1 \wedge \phi_2, \rho_1 \cup \rho_2, \langle q'_1, q'_2, j\rangle)$ whenever $i = 3$ and $j = 1$ or $i \in \{1, 2\}$ and $j = i$, or $i = 1$, $q'_1 \in F_1$ and $j = 2$ or $i = 2$, $q'_2 \in F_2$ and $j = 3$. The labeling $\lambda(\langle q_1, q_2, i\rangle) = \lambda_1(q_1) = \lambda_2(q_2)$, $S = (Q \cap (S_1 \times S_2)) \times \{1\}$ and $F = Q \times \{3\}$.

- The automaton for $[\![\varphi \cdot \psi_2]\!]$ is $(Q \cup Q_2 - F, C \cup C_2, \Delta', \Sigma, \lambda_1 \cup \lambda_2, S, F_2)$ where Δ' is constructed from $\Delta \cup \Delta_2$ as follows: every transition (q, ϕ, ρ, q') in Δ such that $q' \in F$ is removed and replaced by a set of transitions of the form $(q, \phi, \rho \cup C_2, q_2)$ for every $q_2 \in S_2$.

Hence we have the first part of Büchi theorem.

Theorem 6 (ω-Exp $\Rightarrow$ ω-Aut) *Every timed ω-regular language can be accepted by a timed ω-automaton.*

From ω-automata to ω-expressions

This construction is based on theorem 4 and on the untimed Büchi theorem. The level of explanation is less detailed but can be understood and verified by readers who have fully grasped the ideas of the proofs of the abovementioned theorems. We assume that the automaton has gone through all the transformation described in section 5.1.

Let $\mathcal{A} = (Q, \{c\}, \Delta, \Sigma, Id, S, F)$ be a one-clock ω-automaton with $F = \{f_1, \ldots, f_n\}$. Clearly

$$L(\mathcal{A}) = \bigcup_{i=1}^{n} L(\mathcal{A}_i)$$

where $\mathcal{A}_i = (Q, C, \Delta, \Sigma, \lambda, S, \{f_i\})$. Hence it is sufficient to prove regularity for automata with one accepting state $F = \{f\}$. In case f is a resetting state we have

$$L_\omega(\mathcal{A}) = \bigcup_{s \in S} R_{sf} \cdot (R_{ff})^\omega$$

and apply theorem 4. This will not work for non-resetting states because f can be entered with different clock valuations. Thus we have to follow a variation of the path of theorem 4 and treat non-resetting automata first.

Claim 7 *Let $\mathcal{A}$ be a reset-free one-clock automaton and let q_i, q_j, f be states. Then the following holds:*

- *The set of all signals leading from q_i to q_j is regular.*
- *The set of all signals leading from q_i to q_j and visiting f is regular.*
- *The set of all ω-signals accepted by $\mathcal{A}$ when $S = \{q_i\}$ and $F = \{f\}$ is ω-regular.*

Proof: The first part is just a rephrasing of claim 2. The second part is a variation of the first (we spare the reader from the details). For the third part observe that after the last critical point τ_n the automaton behaves like an untimed one. Hence, by the virtue of the original Büchi theorem, for every state q_l there is an untimed ω-regular language $\bar{T}_{l,\omega}$ accepted by the automaton starting at q_i with $c = \tau_n$. Consider now the sampling (q_j, q_l) of a given signal at τ_n. If this sampling is consistent ($j = l$ or the transition from q_j to q_l is possible at τ_n) then the language accepted from q_i can be written as

$$T_{i,j} \cdot T_{l,\omega}$$

where $T_{i,j}$ is the set of signals leading from q_i to q_j whose length is exactly τ_n, and $T_{l,\omega}$ is the set of all signals whose signatures are in $\bar{T}_{l,\omega}$ (as expressions the two are identical). The regularity of $T_{i,j}$ is implied by claim 2. ∎

Claim 8 *The ω-language accepted by any one-clock automaton with one accepting state is ω-regular.*

Proof: As in claim 3 we take $q_1, \ldots, q_n$ to be non-resetting and $q_{n+1}, \ldots, q_{n+r}$ as resetting states. The three languages mentioned in claim 7 are denoted by $R^{(n)}_{i,j}$, $R^{(n)}_{i,f,j}$, and $R^{(n)}_{i,\omega}$. We would like to calculate $R_{i,\omega} = R^{(n+r)}_{i,\omega}$ inductively. To make the inductive step when $k \geq n$ note that when we add a new resetting state the language $R^{(k)}_{i\omega}$ changes as follows:

$$\begin{aligned} R^{(k+1)}_{i\omega} \quad = \quad & R^{(k)}_{i\omega} \cup R^{(k)}_{i,k+1} \cdot (R^{(k)}_{k+1,k+1})^* \cdot R^{(k)}_{k+1,\omega} \cup \\ & R^{(k)}_{i,k+1} \cdot ((R^{(k)}_{k+1,k+1})^* \cdot R^{(k)}_{k+1,f,k+1})^\omega . \end{aligned}$$

The first term consists of accepting runs not visiting q_{k+1} at all, the second contains those that visit q_{k+1} a finite number of times while the third involves those that visit it infinitely many times. As in claim 3, the liberty to put $R^{(k)}_{k+1,k+1}$ under the $*$ and $R^{(k)}_{k+1,f,k+1}$ under the ω is due to the fact that q_{k+1} resets the clock. ∎
This implies:

Theorem 9 (ω-Aut $\Rightarrow$ ω-Exp) *Every language accepted by a timed ω-automaton is a renaming of a timed ω-regular language.*

And we can conclude:

Corollary 10 (Büchi theorem for timed automata) *Timed ω-automata and timed ω-regular expressions have the same expressive power modulo renaming.*

7. Necessity of Intersection

Claim 11 *The language $M = \{a^x b^{1-x} c^x | x \in (0,1)\}$ is regular but it cannot be expressed without $\wedge$.*

Proof: The equality $M = [\![\langle a \cdot b \rangle_1 \cdot c \wedge a \cdot \langle b \cdot c \rangle_1]\!]$ guarantees regularity of the language. This language can be recognized by the automaton of figure 4.

We first associate with every language L all of whose signals having the signature abc, a set $P_L = \{(x,y,z) : a^x b^y c^z \in L\} \subseteq \mathsf{R}^3_+$. We then show that if L is $\wedge$-free then P_L must have a certain form, while P_M, the set associated with M, is not of that form. Hence M cannot be expressed without $\wedge$. Our first step is to get rid of the renaming.

Lemma 5 *If a language $L = g_*[\![\psi]\!]$ where g_* is a renaming and ψ is a $\wedge$-free expression, then there exists a $\wedge$-free φ such that $L = [\![\varphi]\!]$.*

This lemma is an immediate consequence of the fact that g_* commutes with all the operations occurring in ψ. ∎

Next we eliminate the star. A *monomial* regular language is a language definable by an expression using only atoms, $\cdot$ and $\langle\rangle_I$. Notice that all the signals in a monomial language have one and the same signature.

Lemma 6 *Let $L = [\![\varphi]\!]$ where φ is a $\wedge$-free expression, then L is a countable (or finite) union of monomial languages.*

Proof: Induction on the structure of φ. It is straightforward for atomic expressions. Suppose $L_1 = \bigcup\limits_m L_{1,m}$ and $L_2 = \bigcup\limits_n L_{2,n}$. Let us proceed with induction.

$L = L_1 \cup L_2$. This case is straightforward: $L = \bigcup\limits_m L_{1,m} \cup \bigcup\limits_n L_{2,n}$.

$L = L_1 \circ L_2$. In this case $L = \bigcup\limits_m \bigcup\limits_n (L_{1,m} \circ L_{2,n})$.

$L = L_1^*$. In this case $L = \bigcup\limits_{n=0}^{\infty} \bigcirc_{i=1}^{n} L_1$ and the previous case can be applied to each term.

$L = \langle L_1\rangle_I$. In this case $L = \bigcup\limits_m \langle L_{1,m}\rangle_I$. ▟

Monomial languages are useful because they correspond to convex polyhedra and operations on these languages have a simple geometrical sense. We will state these properties only for a subclass of monomial languages that could occur in a representation of the language M. Let $U = [\![a \cdot b \cdot c \vee a \cdot b \vee b \cdot c \vee a \vee b \vee c]\!]$.

An abc-language is a nonempty monomial sublanguage of U. For any language $L \subseteq U$ put $P_L = \{(x,y,z) \in \mathsf{R}^3 : a^x b^y c^z \in L\}$. For any two sets $S_1, S_2 \subseteq \mathsf{R}^3$ we denote by $S_1 \oplus S_2$ the set $\{(x_1+x_2, y_1+y_2, z_1+z_2) : (x_1,y_1,z_1) \in S_1 \wedge (x_2,y_2,z_2) \in S_2\}$.

Lemma 7 *Let L be an abc-language, L_1 and L_2 - monomial languages.*

1. *If $L = [\![a]\!]$, then P_L is the ray $\{(x,0,0) : x \in \mathsf{R}_+\}$, similarly for b and c.*

2. *If $L = L_1 \circ L_2$ then both L_1 and L_2 are abc-languages and $P_L = P_{L_1} \oplus P_{L_2}$.*

3. *If $L = \langle L_1\rangle_I$ then L_1 is an abc-language and $P_L = P_{L_1} \cap P_I$ where P_I is a layer between two parallel planes (maybe coinciding). If L has the signature abc then the normal vector to these planes is (1,1,1).*

4. *P_L is a convex polyhedron.*

Proof:

1. This is straightforward from the definitions.

2. Let $w \in L$ and $w = w_1 \circ w_2$ where $w_i \in L_i$. Since L is an abc-language, the signature of w should be one of abc, ab, bc, a, b, c and hence this is the case for w_i also. This implies that L_i are abc-languages. Further study depends of the signatures of the languages L, L_1 and L_2. Suppose, for example that signatures are abc, ab and bc respectively. In this case

$$\begin{aligned} L &= \{w_1 \circ w_2 : w_1 \in L_1 \wedge w_2 \in L_2\} = \\ &= \{a^{x_1} b^{y_1+y_2} c^{z_2} : a^{x_1} b^{y_1} \in L_1 \wedge b^{y_2} c^{z_2} \in L_2\}. \end{aligned}$$

Hence

$$\begin{aligned} P_L &= \{(x_1, y_1+y_2, z_2) : (x_1,y_1,0) \in P_{L_1} \wedge \\ &\quad (0,y_2,z_2) \in P_{L_2}\} = \\ &= \{(x_1+x_2, y_1+y_2, z_1+z_2) : \\ &\quad (x_1,y_1,z_1) \in P_{L_1} \wedge \\ &\quad (x_2,y_2,z_2) \in P_{L_2}\} = \\ &= P_{L_1} \oplus P_{L_2}. \end{aligned}$$

We have used the fact that the third coordinate of a point in P_{L_1} as well as the first coordinate of the point in P_{L_2} are always zero.

Other possible signatures of the languages are considered similarly.

3. The case study depending of possible signatures (abc, ab, bc, a, b, c) of L can be used. Suppose the signature is ab. It follows from the definitions that $P_L = P_{L_1} \cap \{(x,y,z) : x+y \in I\}$. The set $\{(x,y,z) : x+y \in I\}$ is a layer between two parallel planes with a normal vector (1,1,0). Other possible signatures are considered similarly.

4. Immediate by induction from the previous statements of lemma. ▟

For the language M the polyhedron P_M is the straight line interval (AB), where $A = (1,0,1)$ and $B = (0,1,0)$. Suppose now that there exists a representation $M = g_*[\![\psi]\!]$ where g_* is a renaming and ψ is a $\wedge$-free expression. Applying Lemma 5 we obtain a representation $M = [\![\varphi]\!]$ where φ is a $\wedge$-free expression. Applying Lemma 6 we obtain a representation $M = \bigcup\limits_n L_n$ where L_n are monomial, hence abc-languages. Hence $P_M = \bigcup\limits_n P_{L_n}$ which implies (together with Lemma 7) that at least one of polyhedra P_{L_n} is a nondegenerate (i.e non-singleton) subinterval of (AB). The following result states that this is impossible and concludes the proof of the claim.

Lemma 8 *There is no abc-language L such that P_L is a non-degenerate interval parallel to the vector $\vec{v} = (-1,1,-1)$.*

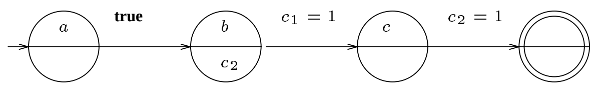

Figure 4. The automaton accepting $[\![\langle a \cdot b\rangle_1 \cdot c \wedge a \cdot \langle b \cdot c\rangle_1]\!]$.

Proof: Induction on the structure of L.

$L \in \{[\![a]\!], [\![b]\!], [\![c]\!]\}$ The statement is immediate from the statement 1 of Lemma 7.

$L = L_1 \circ L_2$ By Lemma 7 $P_L = P_{L_1} \oplus P_{L_2}$. P_L can be an interval in two cases: either when one of P_{L_1} and P_{L_2} is a point and the other is an interval or when they are both parallel intervals. In each of these cases P_L is parallel to an interval P_{L_1} or P_{L_2}, hence, by the inductive hypothesis it cannot be parallel to $\vec{v}$.

$L = \langle L_1 \rangle_I$ The only case when the statement could be violated is when L_1 has the signature abc. In this case by Lemma 7 $P_L = P_{L_1} \cap P_I$ where P_I is a layer between two parallel planes with the normal $\vec{n} = (1, 1, 1)$. But since $\vec{v} \not\perp \vec{n}$, the vector $\vec{v}$ is not parallel to these planes. This implies that P_L cannot be an interval parallel to $\vec{v}$ unless P_{L_1} is such an interval. ▟

This concludes the proof. ▟

8. Discussion

The main design decisions used to achieve the reported results were the following. First by taking the signal semantics (and not the one based on timed sequences), we believe that the definitions of concatenation and star became more natural and easier to manipulate.[5] The second issue was how to introduce dense real-time into regular expressions. After having tried some first-order extensions of regular expressions (which lead to a class incomparable to timed automata, and which suffers from undecidability problems) we have settled on the current solution, namely to take an untimed regular set and limit the length of its members to an integer-bounded interval. As the example in section 4 shows, this construct alone is not sufficient because in timed automata the order of clock resetting and the order of clock testing need not be compatible (leading to "unbalanced parentheses", so to speak). This motivated the separation of clocks and the use of intersection. This also forced us to make the automata state-output and to use renaming in the formulation of the result. We conjecture that renaming is necessary but we found no proof for it.

[5]However, the theory can be developed without significant modifications around timed sequences.

After reduction to one-clock automata we still had a problem with the star. Unlike untimed automata, a cycle in the transition graph of a timed automaton is not really a cycle in its configuration space unless the clock is reset. For this reason we had to treat the non-resetting part of the automaton separately, using the concept of sampling which is roughly the one-dimensional analog of the "region-graph" of [AD94]. Cutting the behavior according to the sampling interval can be seen as a concatenation of timed automata having a special kind of real-time constraints such that each of them is tested only once. After the non-resetting part was solved we could use standard proof techniques for Kleene theorem to add the resetting states.

We have tried to follow the path of classical regular expressions where the expression provides an "extrinsic" description (not mentioning internal states at all). While applying this attitude to timed automata, we have avoided any explicit references to clock values (internal variables) and our formalism is based only on external observable features of the signals. We hope that the result and the proofs will improve our understanding of the role of clocks in timed automata. Recently, the notion of *speed-independent operators* was introduced in [RT97]. This is an operator from signals to signals which is "stretching" invariant. It is immediate to see that languages defined using expressions without the $\langle\rangle_I$ operator are exactly the speed-independent languages where membership is determined only by signature.

For the future, the question of whether renaming is necessary should be settled. In addition bounds on the complexity of the translation should be established. Another interesting question is whether the techniques introduced here can be applied to various logic-based formalisms.

A more speculative domain for future research might be the extension of the theory to signals which are not necessarily piecewise-constant. For example, one may consider an "alphabet" consisting of a set of rational constants $\{k_1, \ldots, k_n\}$ where the semantics of an expression k is the set of real-valued signals whose time derivative is k. It would be interesting to see whether such a theory of regularity can contribute new insights to hybrid dynamical systems and signal processing.

References

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

[B60] J.R. Büchi, A decision method in restricted second order arithmetic, in E. Nagel et al (Eds.), *Proc. Int. Congr. on Logic, Methodology and Philosophy of Science*, Stanford University Press, 1960.

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

[K56] S.C. Kleene, Representations of events in nerve nets and finite automata, in C.E. Shannon and J. McCarthy (Eds.), *Automata Studies*, 3–42, Princeton University Press, 1956.

[MY60] R. McNaughton and H. Yamada, Regular expressions and state graphs for automata, *IRE Trans. Electronic Computers* EC-9, 39–47, 1960.

[RT97] A. Rabinovich and B.A. Trakhtenbrot, From finite automata toward hybrid systems, Unpublished manuscript, 1997.

[T90] W. Thomas, Automata on infinite objects, in J. Van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, 133-191, Elsevier, Amsterdam, 1990.

[W94] Th. Wilke, Specifying state sequences in powerful decidable logics and timed automata, in H. Langmaack et al (Eds.), *Proc. FTRTFT'94*, LNCS 863, 694-715, Springer, 1994.