Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE @ DIRECT®

Information Processing Letters 89 (2004) 237–245

Information Processing Letters

www.elsevier.com/locate/ipl

# A Kleene theorem for splitable signals

Jérôme Durand-Lose [a,b]

[a] *Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46, Allée d'Italie, 69364 Lyon Cedex 07, France*
[b] *Université de Nice-Sophia Antipolis, France*

Received 24 January 2003; received in revised form 24 November 2003

Communicated by L. Boasson

**Abstract**

In this paper, we consider timed automata for piecewise constant signals and prove that they recognize exactly the languages denoted by signal regular expressions with intersection and renaming. The main differences from the usual timed automata are: time elapses on transitions (passing through a state is instantaneous), signals may be split on a run on an automaton and constraints on transitions correspond to unions of open intervals but should be satisfied on closed intervals. This makes exact rendez-vous impossible. The paper stresses on the similarities and differences from the usual model.
© 2003 Elsevier B.V. All rights reserved.

*Keywords:* Timed automata; Piecewise constant signal; Signal regular expression; Formal languages

Classical automata deal with sequences of events, but they do not provide any explicit notion of time (e.g., delay between or duration of) and thus are of little use for, e.g., the verification of real-time systems. Timed automata were introduced to manipulate time explicitly [3]. They are classical automata enhanced with clocks such that each transition must satisfy a constraint over clocks and may reset some clocks. The two most popular semantics for timed automata are time event (sequence of action labels with time stamps) and signals (piecewise constant mappings from time intervals to labels). For both semantics, each transition is instantaneous and its constraint has to be satisfied when the transition occurs; corresponding clocks are reseted simultaneously. In this model, time elapses only on states while only transitions are constrained.

Signals, as inputs, are considered in [3] and a Kleene theorem is proved in [1,2]. But signals correspond to remaining in a state for some time (not in a transition) and splitting and splicing of signals are not addressed. In [7], it is considered that consecutive elementary signals with the same label can be split or spliced and then correspond to 1, 2 or more transitions; in contrast with the time event model, 2 consecutive identical inputs always correspond to 2 transitions. Splitting or splicing add non-determinism in the way a signal can be read by a signal automaton.

*E-mail address:* Jerome.Durand-Lose@ens-lyon.fr (J. Durand-Lose).
*URL:* http://www.i3s.unice.fr/~jdurand.

0020-0190/$ – see front matter © 2003 Elsevier B.V. All rights reserved.
doi:10.1016/j.ipl.2003.11.010

We consider here that everything has a duration and that time measurements and synchronizations cannot have infinite precision. This leads to the choices in the next paragraphs: non-instantaneous actions, constraint valid throughout a transition and constraints corresponding to open intervals.

The first difference is that actions are not instantaneous. Time elapses on transitions and passing through a state is instantaneous. Inputs are piecewise constant (pre-)signals, i.e., sequences of labels/symbols with durations. Splitting and splicing are considered as in [7] and a signal is an equivalence class for these operations.

The second difference is that any transition constraint has to be satisfied during the whole transition (closed time interval); clocks are reset at the end of it. In previous works, time elapses on states but no condition exists for remaining in a state (even if this can be handled with extra states and constraints on entering and leaving transitions). Having time elapse on transitions allows to constrain directly atomic durations.

The third difference is that, in the constraints, only open intervals of time are considered: constraints are finite unions of products of open intervals over clocks; constraints correspond to open sets. This means that instantaneous passing through a state correspond to an (intersection of) open time interval; thus it is impossible to build an automaton such that this corresponds exactly to a given date. Thus, no exact rendez-vous/synchronization can be set. Nevertheless, since these intervals can be made as small as wanted, approximations of dates can be achieved up to any given constant.

Although the signal automata presented here differ in many ways from the classical timed automata, they can mostly be manipulated the same way. Signal regular expressions like the ones of [1,2] are defined. We prove that they denote the same languages as signal automata up to renaming. The constructions presented here are not optimal, but the aim of this paper is to stress on the difference from the usual model. No formal proofs are given, they are straightforward from the constructions.

The paper is articulated as follows. All the definitions are gathered in Section 1. The inductive construction from a signal regular expression to a signal automaton is given in Section 2. The computation of a signal regular expression and a renaming corresponding to the language accepted by a signal automaton is given in Section 3. This is done in two phases: splitting the automaton into 1-clock automata and then considering each 1-clock automaton independently. A brief conclusion is presented in Section 4.

## 1. Definitions

Let $\Sigma$ be a finite set of symbols. A *pre-signal* is defined as a sequence of symbols associated with durations. It is denoted $\sigma_1^{\pi_1}\sigma_2^{\pi_2}\ldots\sigma_l^{\pi_l}$ where all $\sigma_i$ belong to $\Sigma$ and all $\pi_i$ to $\mathbb{R}_{>0}$. Its *duration* is $|m| = \sum_{i=1}^{l}\pi_i$. Any pre-signal can be *split* by replacing some factor $\sigma^\pi$ by $\sigma^{\pi_1}\sigma^{\pi_2}$ as long as $\pi_1 + \pi_2 = \pi$. The inverse operation (merging two factors with the same symbol) is called *splicing*. Two pre-signals are *equivalent* if one can go from one to the other by a finite sequence of splittings and splicings. A *signal* is an equivalence class of pre-signals. It admits a canonical pre-signal representation, its non-stuttering normal form obtained by making all possible splicings on any pre-signal of the class. A pre-signal $m'$ is a *sub-split* of $m$ if $m'$ can be obtained from $m$ by splitting. Any two pre-signals represent the same signal iff they have a common sub-split. The empty signal is denoted by $\varepsilon$; its duration is 0. An *elementary* signal is just a constant signal (some $\sigma^\pi$). For example, $a^{0.7}a^{0.3}c^1c^{1.5}c^1$ and $a^1c^{1.5}c^2$ are equivalent and the non-stuttering normal form associated to their class is $a^1c^{3.5}$ and $a^{0.7}a^{0.3}c^1c^{0.5}c^1c^1$ is one of their common sub-splits. Pre-signals can also be represented as partial mappings from $(0,t)$ to $\Sigma$ with finitely many discontinuity (where the value is a symbol if it is the same on both side, otherwise it is undefined) as illustrated by Fig. 1. Concatenation is defined as usual. Let us note that, for splicing reasons, the location of concatenation may be lost (e.g., $a^1.a^3 = a^4$). This splitting/splicing property justifies the exponent notation and the equivalence relation.

*Clocks and constraints.* A *clock*, $z$, is a mapping from the absolute time ($\mathbb{R}_{\geqslant 0}$) to non-negative real numbers. Its value regularly increases as time elapses. The only operations available on a clock are comparisons to rational

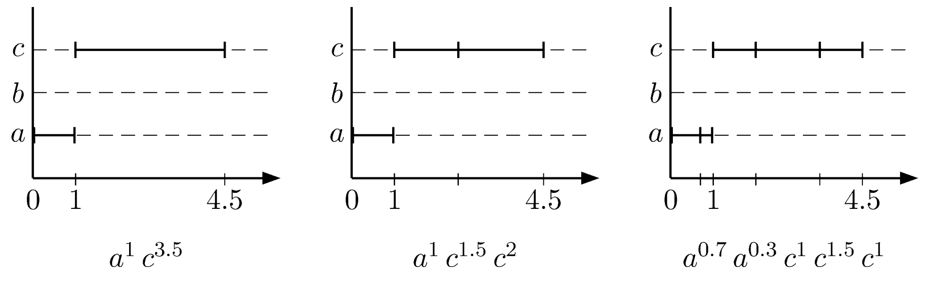


Fig. 1. Non-stuttering normal form and sub-splittings.

constants and resetting to 0. Let $Z$ be a finite set of clocks; $Z_t$ denotes the clocks valuation at time $t$ and $Z_t + d$ means $d$ added to each clock valuation.

A *constraint* over $Z$ is a logical formula using the connectors $\vee$ and $\wedge$ over atomic formulae of the form $z < c$ or $c < z$ where $z$ is a clock and $c$ is a constant in $\mathbb{Q}_{>0}$. The set of all constraints is denoted $\Phi(Z)$. It is possible to construct a constraint always satisfied, it is denoted *true* or just left blank. Since atomic formulae denote open intervals (of $\mathbb{R}_{>0}$), and only intersection and union are used, each formula represents a finite union of products (over possible values of clocks) of open intervals, thus an open subset of $\mathbb{R}^n_{>0}$ where $n$ is the number of clocks. It is impossible to create a constraint equivalent to $z = c$ or $z \leqslant c$.

*Signal automata.* A signal automaton is defined by: an alphabet $\Sigma$, a finite set of *states*, $Q$, a set of *initial states*, $I \subseteq Q$, a set of *accepting states*, $F \subseteq Q$, a finite set of *clocks*, $Z$, and a finite set of *transitions* $\Delta \subseteq Q \times \Sigma \times \Phi(Z) \times 2^Z \times Q$. A transition $\delta$ is denoted $(q, \sigma, \phi, \rho, q')$ and is represented as in Fig. 2 ($\phi$ is the constraint and $\rho$ is the set of the clocks that have to be reseted at the end of the transition). The mapping *Reset*, from clock valuations and a set of clocks to clocks valuations, resets to zero all the valuations of the clocks in the set, other valuations are unaffected. When no clock is reset, this is indicated by Ø or left blank.

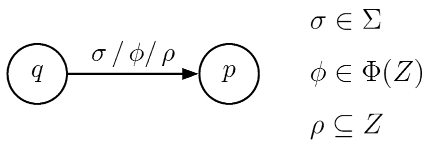


Fig. 2. Representation of transition $(q, \sigma, \phi, \rho, p)$.

A transition is valid from $t$ to $t'$ if the input is $\sigma$ on $(t, t')$ (open) and the clock constraint $\phi$ is satisfied on $[t, t']$ (closed!). This means that for each clock $z$, $[z_t, z_t + (t' - t)]$ has to be included in a finite union of products of open intervals, which means strictly included and away from the bounds.

Let $\mathcal{A} = (\Sigma, Q, I, F, Z, \Delta)$ be a signal automaton and $m$ a signal. Since $m$ can be split/spliced in infinitely many ways, it is useless to consider some precise representation for it; instead, $m$ is considered to be the piecewise constant function from $(0, |m|)$ to $\Sigma \cup \{\perp\}$ ($\perp$ stands for undefined) corresponding to its non-stuttering normal form. Let $m_t$ be the value of $m$ at time $t$. A *run* of $m$ over $\mathcal{A}$ is a finite sequence of transitions and strictly increasing dates $\{(\delta_i, t_i)\}_{1 \leqslant i \leqslant n}$. Let $t_0 = 0$ and $\delta_i = (q_i, \sigma_i, \phi_i, \rho_i, p_i)$. A run must satisfy:

(1) $Z_0 = \vec{0}$ (initialization),
(2) $\forall i$, $1 \leqslant i \leqslant n$ (passing through a transition)
  (a) $\forall t \in (t_{i-1}, t_i)$, $m_t = \sigma_i$,
  (b) $\forall \delta \in [0, t_i - t_{i-1}]$, $\phi_i(Z_{t_{i-1}} + \delta)$ is satisfied,
(3) $\forall i$, $1 \leqslant i < n$ (passing through a state)
  (a) $q_{i+1} = p_i$,

(b) $Z_{t_i} = Reset(Z_{t_{i-1}} + t_i - t_{i-1}, \rho_i)$,

(4) $t_n = |m|$ (complete reading of $m$).

Let us note that (3)(a) the value of $m$ at $t_i$ is not considered whereas (2)(b) clock constraints $\phi_{i-1}$ and $\phi_i$ have to be satisfied at $t_i$ (before resetting the clocks for $\phi_{i-1}$ and after resetting for $\phi_i$). The pre-signal corresponding to this run is $\sigma_1^{t_1}\sigma_2^{t_2-t_1}\ldots\sigma_n^{t_n-t_{n-1}}$. A run is *accepting* iff $q_1 \in I$ and $p_n \in F$. The *language accepted* by $\mathcal{A}$, $\mathcal{L}(\mathcal{A})$, is the set of all signals for each of which there exists an accepting run for a pre-signal of its class.

*Signal regular expressions.* They are defined inductively, over an alphabet $\Sigma$, using $\varepsilon$, $\sigma$, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \cdot \varphi_2$, $\varphi^*$ and $\langle\varphi\rangle_I$, s.t. $\sigma \in \Sigma$ and $\varphi, \varphi_1, \varphi_2$ are signal regular expressions and $I$ is an open interval of $\mathbb{R}_{\geqslant 0}$, either $(d, d')$ or $(d, \infty)$ s.t. $d, d' \in \mathbb{Q}$ and $0 \leqslant d < d'$. The semantic of regular expressions is:

(1) $[\![\varepsilon]\!] = \{\varepsilon\}$,
(2) $[\![\sigma]\!] = \{\sigma^r \mid r \in \mathbb{R}_{>0}\}$,
(3) $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$,
(4) $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!]$,
(5) $[\![\varphi_1 \cdot \varphi_2]\!] = \{m_1.m_2 \mid m_1 \in [\![\varphi_1]\!] \wedge m_2 \in [\![\varphi_2]\!]\}$,
(6) $[\![\varphi^*]\!] = \bigcup_{i=0}^{\infty}([\![\varphi^i]\!])$ s.t. $[\![\varphi^0]\!] = \{\varepsilon\}$, $\varphi^1 = \varphi$, $\varphi^{n+1} = \varphi^n \cdot \varphi$,
(7) $[\![\langle\varphi\rangle_I]\!] = \{m \in [\![\varphi]\!] \mid |m| \in I\}$.

Let $\Sigma$ and $\Sigma'$ be two alphabets, a *renaming function*, $\lambda$, is a function from $\Sigma$ to $\Sigma'$. It is extended to pre-signals in the following way: $\lambda(\sigma_1^{\pi_1}\sigma_2^{\pi_2}\ldots\sigma_l^{\pi_l}) = \lambda(\sigma_1)^{\pi_1}\lambda(\sigma_2)^{\pi_2}\ldots\lambda(\sigma_l)^{\pi_l}$ and to sets of signals pointwisely (it is not possible to do it inductively on rational expressions because of the intersection).

All these operations are compatible with the equivalence relation.

## 2. From regular expressions to automata

We show that basic regular expressions correspond to automata and then that the languages accepted by automata are closed for the corresponding operators. All is done following usual constructions for un-timed and timed automata. Emphasis is made on substantially different constructions: the automata product used for $\wedge$ (splittings induced by an automaton have to be considered) and duration restrictions. Fig. 3 shows the constructions for $\varepsilon$ and $a$.

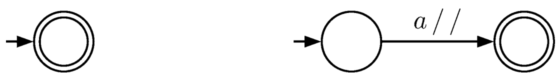


Fig. 3. Automata for $\varepsilon$ and $a$.

If more than one automaton are considered, the sets of states as well as sets of clocks are assumed to be disjoint (renaming is used if necessary) but the alphabets are identical (or the union is considered). In the pictures, the dotted parts are discarded in the constructions, the dashed parts are preserved but are not relevant and the dashed boxes delimit automata.

Since automata are non-deterministic, the automaton for an union is very easy to built: just gather the automata as in Fig. 4.

*Intersection.* The classical way to make an intersection is to make a product of automata and then to restrain initial and accepting states. The problem is that accepting runs may be different, e.g., $a^1a^3b^1b^2$ on one automaton and $a^4b^2b^1$ on the other. These pre-signals both represent $a^4b^3$ and have infinitely many common sub-splits (e.g.,

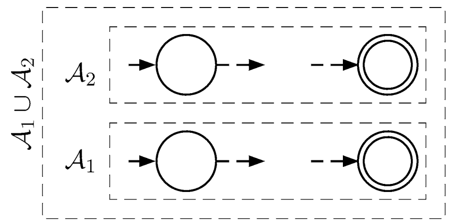


Fig. 4. Automaton for $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.

$a^1a^3b^1b^1b^1$), but it may happen that none of them corresponds to an accepting run on both automata. Each automaton is transformed in order that any sub-split of an accepting run also corresponds to an accepting run. This is done by adding for each transition 3 transitions and 1 new state as depicted on Fig. 5. The generated automaton accepts exactly the same language and, any sub-split of any pre-signal corresponding to an accepting run also corresponds to an accepting run.

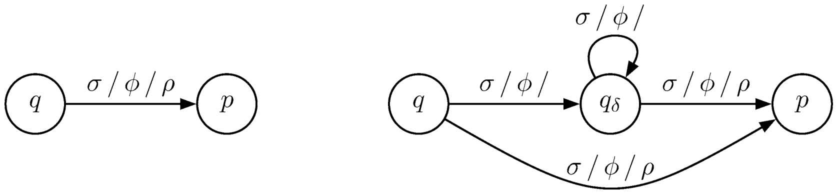
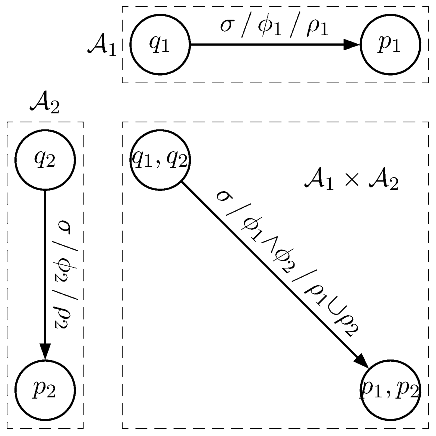


Fig. 5. Full split of transition $\delta = (q, \sigma, \phi, \rho, p)$.

With this *full split form*, it is easy to construct the product automaton and then the intersection. Fig. 6 shows the product of transitions; of course only and all pairs of transitions with the same symbol are considered. The set of initial (accepting) states is the product of initial (accepting) states. If mis accepted by both $\mathcal{A}_1$ and $\mathcal{A}_2$ split as $m_1$ and $m_2$, then any sub-split $m'$ common to both $m_1$ and $m_2$ corresponds to an accepting run in the product of the full split automata. If mis not accepted by $\mathcal{A}_1$ ($\mathcal{A}_2$), then neither it is by the full split form of $\mathcal{A}_1$ ($\mathcal{A}_2$) and nor by the product.



$\mathcal{A}_1$
$q_1$
$\sigma / \phi_1 / \rho_1$
$p_1$
$\mathcal{A}_2$
$q_2$
$\sigma / \phi_2 / \rho_2$
$p_2$
$q_1, q_2$
$\mathcal{A}_1 \times \mathcal{A}_2$
$\sigma / \phi_1 \wedge \phi_2 / \rho_1 \cup \rho_2$
$p_1, p_2$

Fig. 6. Product of transitions $(q_1, \sigma, \phi_1, \rho_1, p_1)$ and $(q_2, \sigma, \phi_2, \rho_2, p_2)$.

*Concatenation and iteration.* Concatenation of $\mathcal{A}_1$ and $\mathcal{A}_2$ is done by doubling each transition to an accepting state of $\mathcal{A}_1$ to an initial state of $\mathcal{A}_2$; these copies reset all the clocks of $\mathcal{A}_2$ as illustrated in Fig. 7. The initial (accepting) states are the ones of $\mathcal{A}_1$ ($\mathcal{A}_2$).

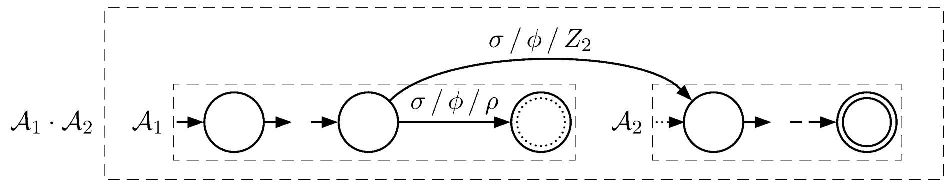


Fig. 7. Transition added for $\mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$.

For the finite iteration (or Kleene star), copies of each transition to a final state are made to each initial state; these copies reset all the clocks. An automaton recognizing $\varepsilon$ is added for zero iteration. This is summed up in Fig. 8.

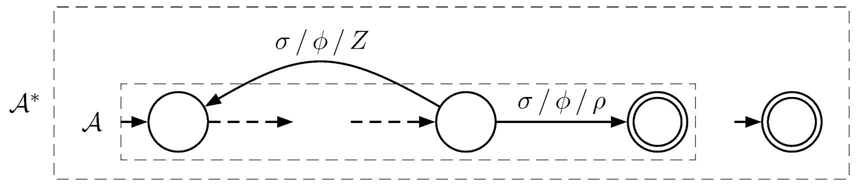


Fig. 8. Transitions and state added for $\mathcal{L}(\mathcal{A})^*$.

*Duration restriction.* It cannot be added directly to transitions leading to accepting states because it might be satisfied at the end of the last transition but not during the whole transition. This is handled by adding an extra state and adding for each transition leading to an accepting state one new state and two consecutive new transitions (corresponding to one extra split in the run); the duration restriction is only present in the last constraint as depicted on Fig. 9. A new clock, $z_0$ is added; it only appears in, and in every, final transition as $z_0 \in I$ (remember $I$ is open and its bounds are in $\mathbb{Q}$). Since $z_0$ is zero when the run starts and is never reset, it corresponds to the duration of the part already read; thus the total duration of any accepted input has to be in $I$.

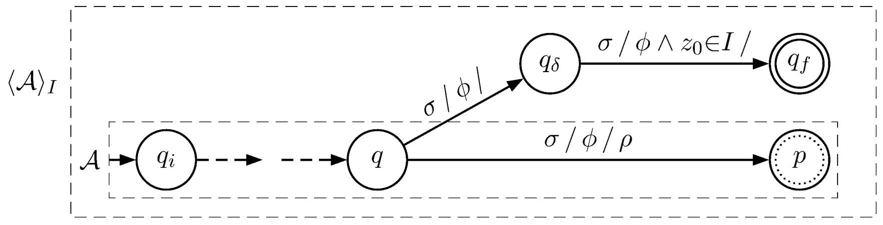


Fig. 9. Automaton for $\langle\mathcal{L}(\mathcal{A})\rangle_I$.

Finally, by induction:

**Lemma 1.** *The signal languages described by signal regular expressions are accepted by signal automata.*

If a renaming function is applied to $\mathcal{L}(\varphi)$, it should be applied to every transition. This is not a problem since intersections are encoded inside the automaton structure.

## 3. From automata to regular expressions

The construction is done in two steps: first separating a $n$-clock automaton $\mathcal{A}$ into $n$ 1-clock signal automata, one for each clock, $\{\mathcal{A}_z\}_{z\in Z}$, and a renaming function $\lambda$. These 1-clock automata are then considered independently.

*Separating the clocks.* Various manipulations are made in order to finally get an automaton $\mathcal{A}_z$ for each clock $z$ such that the intersection of the accepted languages is the one accepted by $\mathcal{A}$ up to some renaming.

(1) All disjunctions are removed. First all constraints are presented in normal disjunctive form. No transition can be simply separated in two transitions because the disjunction may be satisfied during the whole duration while no single term is. Transitions (and pre-signals) are split and disjunction are disposed of as in Fig. 10 where $\phi_1$ and $\phi_2$ may still contain $\vee$.

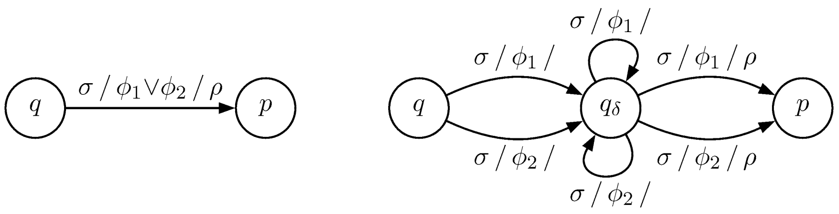


Fig. 10. Removing disjunction in $\delta = (q, \sigma, \phi_1 \vee \phi_2, \rho, p)$.

(2) All loops are removed by putting an extra state (and split) in each loop.
(3) The automaton is made deterministic by replacing each transition symbol by a new symbol appearing only in this transition. Each run corresponds to a run where transitions are indicated. The original run/pre-signal is recovered by projecting the transition symbols on the original symbols. This projection is denoted $\lambda$.
(4) One copy of the automaton is made for each clock, setting to true atomic constraints over other clocks. From the deterministic association of symbols to transitions, any run on one automaton can only correspond to one run (the same) for each copy. If a signal is accepted by $\mathcal{A}$, then it is accepted by all copies (constraints are only conjunctive). Conversely, if a signal is accepted by all copies, then it has to be with the same run which is also accepting for $\mathcal{A}$ (conjunctions of constraints satisfied for all clocks).

The automaton $\mathcal{A}$ is thus transformed into $\{\mathcal{A}_z\}_{z\in Z}$ and a renaming function $\lambda$ such that:

$$\mathcal{L}(\mathcal{A}) = \lambda\Bigl(\bigcap_{z\in Z} \mathcal{L}(\mathcal{A}_z)\Bigr), \tag{1}$$

and copies can be treated independently.

*From one-clock automaton to regular expression.* Let $\mathcal{A}_z$ be a signal automaton with only one clock z, $\mathcal{A}_z = (\Sigma, Q, I, F, \{z\}, \Delta)$. Let $\tau_0 = 0 < \tau_1 < \tau_2 < \cdots < \tau_\iota < \tau_{\iota+1} = \infty$ be the list of all the constants that appear in at least one clock constraint in $\mathcal{A}$ (plus 0 and $\infty$). Since constraints are conjunctions over atomic formulae $z < \tau_i$ or $\tau_j < z$, they correspond to $\tau_\alpha < z < \tau_\beta$ (or to false and are removed), since $\tau_0 = 0$ (transitions have strictly positive durations) and $\tau_{\iota+1} = \infty$ all kinds of intervals are covered. Constraints are "constant" on each $(\tau_i, \tau_{i+1})$, either satisfied or not.

Let $\Delta_1$ be the subset of reset-less transitions of $\mathcal{A}_z$ and $\Delta_2$ the subset of transitions resetting $z$ ($\Delta = \Delta_1 \cup \Delta_2$).

*One-clock automaton without reset.* Only the transitions in $\Delta_1$ are considered here. Let $L(q, p, i)$ denote the un-timed language corresponding to the runs from $q$ to $p$ using only transitions whose constraints are satisfied on $(\tau_i, \tau_{i+1})$. There is an un-timed regular expression for $L(q, p, i)$. Let $L^{\tau_i}_{q\to p}$ denote the signal language corresponding to the runs from $q$ to $p$ of total durations strictly less than $\tau_i$. The $L^{\tau_i}_{q\to p}$ can be computed recursively:

$$L^{\tau_1}_{q\to p} = \langle L(q,p,0)\rangle_{(0,\tau_1)} \quad \big(\cup\{\varepsilon\}\ \{\text{if } q = p\ \}\big), \tag{2}$$

$$\begin{aligned} L^{\tau_{i+1}}_{q\to p} = L^{\tau_i}_{q\to p} \cup \Bigg\langle \bigcup_{\substack{(r,\sigma,\tau_\alpha<z<\tau_\beta,\emptyset,p)\in\Delta_1\\ \tau_\alpha<\tau_i<\tau_\beta}} \langle L^{\tau_i}_{q\to r}\rangle_{(\tau_\alpha,\infty)}.\sigma \Bigg\rangle_{(\tau_{i-1},\tau_{i+1})} \\ \cup \Bigg\langle \bigcup_{\substack{(r,\sigma,\tau_\alpha<z<\tau_\beta,\emptyset,s)\in\Delta_1\\ \tau_\alpha<\tau_i<\tau_\beta}} \Big\langle \langle L^{\tau_i}_{q\to r}\rangle_{(\tau_\alpha,\infty)}.\sigma\Big\rangle_{(\tau_i,\infty)}.L(s,p,i+1)\Bigg\rangle_{(0,\tau_{i+1})}. \end{aligned} \tag{3}$$

Before $\tau_1$, the automaton is "constant", classical language theory gives an un-timed expression which only needs to have its duration restrained (2). The empty word has to be added because its duration is zero. Eq. (3) holds because duration could be less than $\tau_i$, or $\tau_i$ could be during the last transition (possibly finishing at $\tau_i$) or during a previous one. Eq. (2) directly gives a signal regular expression, (3) uses concatenation, finite union and regular expression. By induction, all $L^{\tau_i}_{q\to p}$ correspond to signal regular expressions.

Let $L_{q\to p}$ denote the language corresponding to all the runs from $q$ to $p$: $L_{q\to p} = L^{\infty}_{q\to p} = L^{\tau_{i+1}}_{q\to p}$.

*One-clock automata with reset.* The resetting transitions, i.e., the ones in $\Delta_2$, are now considered, together with the ones in $\Delta_1$. Let $\Delta_2 = \{\delta_1, \delta_2, \ldots, \delta_\kappa\}$ and $\delta_k = (q_k, \sigma_k, \tau_{\alpha_k} < z < \tau_{\beta_k}, \{z\}, p_k)$. Let $L^{\delta_1,\ldots,\delta_k}_{s,\delta_l}$ be the language corresponding to the runs starting from $s$, using only resetting transitions $\delta_1, \delta_2, \ldots, \delta_k$ and ending by $\delta_l$ $(l \leqslant k)$. The following recurrence equations are satisfied:

$$L^{\delta_1}_{s,\delta_1} = \Big\langle\langle L_{s\to q_1}\rangle_{(\tau_{\alpha_1},\infty)}.\sigma_1\Big\rangle_{(0,\tau_{\beta_1})}.\Big(\Big\langle\langle L_{p_1\to q_1}\rangle_{(\tau_{\alpha_1},\infty)}.\sigma_1\Big\rangle_{(0,\tau_{\beta_1})}\Big)^*, \tag{4}$$

$$\begin{aligned} L^{\delta_1,\ldots,\delta_k}_{s,\delta_k} = \Bigg(\Big\langle\langle L_{s\to q_k}\rangle_{(\tau_{\alpha_k},\infty)}.\sigma_k\Big\rangle_{(0,\tau_{\beta_k})} \cup \bigcup_{1\leqslant l<k} L^{\delta_1,\ldots,\delta_{k-1}}_{s,\delta_l}.\Big\langle\langle L_{p_l\to q_k}\rangle_{(\tau_{\alpha_k},\infty)}.\sigma_k\Big\rangle_{(0,\tau_{\beta_k})}\Bigg) \\ .\Bigg(\Big\langle\langle L_{p_k\to q_k}\rangle_{(\tau_{\alpha_k},\infty)}.\sigma_k\Big\rangle_{(0,\tau_{\beta_k})} \cup \bigcup_{1\leqslant l<k} L^{\delta_1,\ldots,\delta_{k-1}}_{p_k,\delta_l}.\Big\langle\langle L_{p_l\to q_k}\rangle_{(\tau_{\alpha_k},\infty)}.\sigma_k\Big\rangle_{(0,\tau_{\beta_k})}\Bigg)^*, \end{aligned} \tag{5}$$

$$L^{\delta_1,\ldots,\delta_k}_{s,\delta_l} = L^{\delta_1,\ldots,\delta_{k-1}}_{s,\delta_l} \cup L^{\delta_1,\ldots,\delta_k}_{s,\delta_k}.L^{\delta_1,\ldots,\delta_{k-1}}_{p_k,\delta_l} \quad (l < k). \tag{6}$$

Resetting transition $\delta_1$ has to be done at least once, then the run can go back through it any number of times (4). The same holds for a run ending by $\delta_k$, it has to be done once, and the run can go back through it, each time other allowed resetting transitions may be used or not (5). A run ending by $\delta_l$ does not use $\delta_k$ after the last passage, if any, through $\delta_k$ (6).

$$\mathcal{L}(\mathcal{A}_z) = \bigcup_{s\in I,\ t\in F}\Bigg(L_{s\to t} \cup \bigcup_{1\leqslant l\leqslant\kappa} L^{\delta_1,\ldots,\delta_\kappa}_{s,\delta_l}.L_{p_l\to t}\Bigg). \tag{7}$$

All these equations use signal regular expressions and their inductive operators. So, the set of signal languages accepted by one-clock signal automata is included in the set of signal languages described by signal regular expressions. From (1) and (7) comes:

**Lemma 2.** *The signal languages accepted by signal automata can be described by signal regular expressions and renaming.*

## 4. Conclusion

Altogether, we have proved that:

**Theorem 3.** *The set of signal languages accepted by signal automata is equal to the one described by signal regular expressions and renaming.*

Renaming seems to be unavoidable as proved in the usual model [8]. We believe that regular expressions for signals like the ones of [6] can also provide a Kleene theorem. We also believe that there is some algebraic context like in [4,2] and that infinite duration and Zeno configurations can be approached with techniques as in [5].

It should also be interesting to investigate the consequences of choosing for the transitions open interval constraints that should be verified on closed intervals.

## Acknowledgement

We wish to thank the anonymous referee for stressing out the importance of pre-signals.

## References

[1] E. Asarin, P. Caspi, O. Maler, A Kleene theorem for timed automata, in: Proc. 12th Annual IEEE Symp. on Logic in Computer Science, Warsaw, Poland, 1997, pp. 160–171.
[2] E. Asarin, P. Caspi, O. Maler, Timed regular expressions, J. ACM 49 (2) (2002) 172–206.
[3] R. Alur, D.L. Dill, A theory of timed automata, Theoret. Comput. Sci. 126 (2) (1994) 183–235.
[4] E. Asarin, Equations on timed languages, in: T.A. Henzinger, S. Sastry (Eds.), Hybrid Systems: Computation and Control, HSCC ’98, in: Lecture Notes in Comput. Sci., vol. 1386, Springer-Verlag, Berlin, 1998, pp. 1–12.
[5] B. Bérard, C. Picaronny, Accepting Zeno words: a way towards timed refinements, Acta Inform. 37 (1) (2000) 45–81.
[6] P. Bouyer, A. Petit, A Kleene/Büchi-like theorem for clock languages, J. Autom. Lang. Comb. 7 (2) (2002) 167–186.
[7] C. Dima, Real-time automata and the Kleene algebra of sets of real numbers, in: STACS ’00, in: Lecture Notes in Comput. Sci., vol. 1770, 2000, pp. 279–290.
[8] Ph. Herrmann, Renaming is necessary in timed regular expressions, in: FSTTCS ’99, in: Lecture Notes in Comput. Sci., vol. 1738, Springer-Verlag, Berlin, 1999, pp. 47–59.