HAL
open science

A Kleene Theorem for Piecewise Constant Signals Automata

Jérôme Durand-Lose

To cite this version:

Jérôme Durand-Lose. A Kleene Theorem for Piecewise Constant Signals Automata. Information Processing Letters, 2004, 89(5), pp.237-245. hal-00079824

HAL Id: hal-00079824

https://hal.science/hal-00079824

Submitted on 16 Jun 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON
n° 5668

A Kleene Theorem for Piecewise Constant Signals Automata (extended abstract)

Jérôme Durand-Lose

November 2002

Research Report N° 2002-43

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`

A Kleene Theorem for Piecewise Constant Signals Automata (extended abstract)

Jérôme Durand-Lose

November 2002

Abstract

In this paper, we consider timed automata for piecewise constant signals. In the model presented here, time elapses only during transitions; any constraint on clocks should be satisfied during all the duration of the transition. Signal automata are very different from un-timed and time-event automata because piecewise constant signals may be split (and spliced) in an infinite number of ways. We show that there exist signal regular expressions with renaming describing exactly the languages accepted by signal automata. The constructions show the similarities and differences from the time-event model.

Keywords: Timed automata,piecewise constant signals, regular expression

Résumé

Dans ce rapport, nous considrons les automates temporiss sur des signaux constants par morceaux. Dans le modle prsent, le temps s'coule durant les transitions ; toute contrainte sur un transition devant tre vrifie durant toute la transition. Les automates signaux sont trs diffrents de leurs homologues non temporiss ou vnement-date car les morceaux constants peuvent tres dcoups ou recolls d'une infinit de manires. Nous montrons qu'il existe des expressions rationnelles de signaux coupls des renommages qui dcrivent exactement les mmes langages que ceux accepts par les automates signaux. Les constructions montrent les similarits et les diffrences avec le modle vnement-date.

Mots-clés: Automates temporiss, signaux constants par morceaux, expressions rationnelles

A Kleene Theorem for Piecewise Constant Signals Automata

Jérôme Durand-Lose

Abstract

In this paper, we consider timed automata for piecewise constant signals. In the model presented here, time elapses only during transitions; any constraint on clocks should be satisfied during all the duration of the transition. Signal automata are very different from un-timed and time-event automata because piecewise constant signals may be split (and spliced) in an infinite number of ways. We show that there exist signal regular expressions with renaming describing exactly the languages accepted by signal automata. The constructions show the similarities and differences from the time-event model.

1 Introduction

Classical automata deals with sequence of event, but they do not provide any explicit notion of time (e.g. delay or duration). Timed automata are classical automata enhanced with clocks such that each transition must satisfy a constraint over clocks and may reset some clocks [AD94]. They are one of the canonical tools for the verification of real-time systems. In the usual, time-event, model, inputs correspond to instantaneous actions. If the constraint is satisfied by the clocks when the event comes, the transition can be instantaneously taken and the indicated clocks are reset. Time elapses only between events, on states.

We consider here that everything has a duration and that time measure and synchronization can not have infinite precision. This leads to the choices in the next two paragraphs: signals and open intervals.

Inputs are piecewise constant signals, i.e. sequences of values with durations; actions are not instantaneous. Time elapses during transitions and passing through a state is instantaneous. Any transition constraint has to be satisfied during the whole transition and clocks are reset at the end of the transition. (Piecewise constant) signals can be split in many ways, e.g. reading a a of duration 2 can be done in 1 transition or in many ones as long as their total duration is 2. Conversely two consecutive elementary signals, i.e. constant parts, with the same value may be spliced into the same transition. Signals which can be split and spliced are considered in [Dim00], but time elapses on state (and still there are only constraints on transitions) and there is only one clock (which is known to be a strictly weaker model).

In the constraints, only open intervals of time are considered: constraints are unions of products of open intervals over clocks (i.e. no $=$, $\leq$, $\geq$ nor $\neg$); constraints denote open sets. This means that no constant appearing in a constraint may have to correspond to the instantaneous passing through a state;

no exact rendezvous can be set. Perfect synchrony does not exist, nevertheless synchrony up to any constant can be achieved.

Although splicing and splicing are great differences from the time-event model, signal automata can mostly be manipulated the same way. Regular expressions for signals like the ones of [ACM97, ACM02] for time-event (we think that it also works with the ones of [BP02]) are defined. They denote the same languages as the signal automata up to renaming. The constructions are not optimal, but the aim of this paper is simplicity and to stress on signals singularities. No formal proof is given, they are straightforward from the constructions.

The paper is articulated as follows. All the definitions are gathered in Sect. 2. The inductive transformation from signal regular expression to signal automaton is given in Sect. 3. The computation of a signal regular expression and a renaming corresponding to the language accepted by a signal automaton is given in Sect. 4. It is done in two phases: splitting the automaton into 1-clock automata and then considering only these automata. A brief conclusion is presented in Sect. 5.

2 Definitions

Let Σ be a finite (non empty) set of signal values / letters.

2.1 Piecewise constant signals

A *piecewise constant signal*, or just signal, is defined by the sequence of states associated with durations. It is denoted $\sigma_1^{\pi_1}\sigma_2^{\pi_2}\dots\sigma_l^{\pi_l}$ where all σ_i belong to Σ and all π_i to $\mathbb{R}_+^*$. Its *duration* is $|m| = \sum_{i=1}^{l} \pi_i$. The empty signal is denoted ε; its duration is 0. An *elementary* (piecewise constant) signal is just a constant signal (some σ^π). The *starting date* of the ith elementary signal of m, ζ_i, is defined by: $\zeta_1 = 0$ and $\zeta_{i+1} = \zeta_i + \pi_i$ $(1 \leq i \leq l)$. If σ_i and σ_{i+1} are equal then the value at ζ_{i+1} is defined and equals σ_i, otherwise it is undefined. In case of equality, the two elementary signals can be spliced and it is still the same signal. Conversely, any elementary signal can be split into finitely many elementary signals with the same value and total duration. This is illustrated in Fig. 1 where three different decompositions of the same signal are presented. In the time-event model, 2 consecutive identical inputs correspond to 2 transitions; whereas consecutive elementary signals of the same value can be spliced and then correspond to 1, 2 or more transitions. This adds non determinism in the way a signal could be split / spliced by a signal automaton.

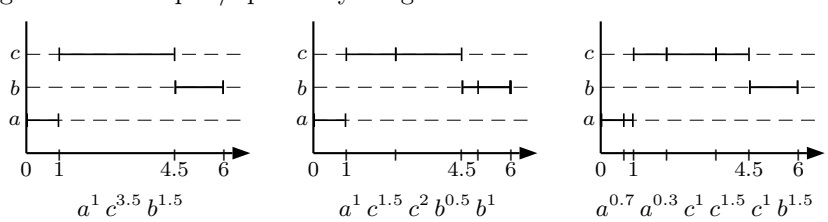

Figure 1: Normal form and sub-splittings.

A signal m' is a *sub-split* of m if m' can be obtained by splitting elementary

signals of m. Any two signals which represent the same signal, i.e. they are both sub-splits of it, have a common sub-split (by merging the sets of starting dates).

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.

2.2 (Piecewise constant) signal automata

Let Z be a finite set of *clocks*. Each clock has a value in $\mathbb{R}_+$ which increases regularly as time elapses. The only operations available on clock are comparisons to constants and resetting to 0. Let Z_t denote the values of the clocks at time t, $Z_t{+}d$ means d added to each clock.

2.2.1 Clock constraint.

A *constraint* over Z is a propositional 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 $\mathbb{Q}_+$. 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 $\mathbb{R}_+^*$), and only intersection and union are used, each formula represents a finite union of products (over clocks) of open intervals. It is impossible to create a constraint equivalent to $z{=}c$ or $z{\leq}c$. Exact rendezvous are impossible as shown below.

2.2.2 A (piecewise constant) signal automaton

is defined by:

- Σ, a finite set of values for elementary signals, the *signal alphabet*
- Q, a finite set of *states*,
- $I \subseteq Q$, the set of *initial states*,
- $F \subseteq Q$, the set of *accepting states*,
- Z, a finite set of *clocks*,
- $\Delta \subseteq Q \times \Sigma \times \Phi(Z) \times \mathcal{P}(Z) \times Q$, s.t. $|\Delta| < \infty$, is the set of *transitions*.

A transition δ is denoted $(q, \sigma, \phi, \rho, q')$ and is represented as in Fig. 2. ρ is the set of clocks which has to be reset at the end of the transition. The mapping *Reset*, from clocks and a set of clocks to clocks, resets to zero all the clocks in the set, other clocks are unaffected. When no clock is reset, this is indicated by $\emptyset$ or left blank.

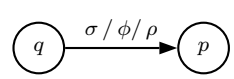

Figure 2: Representation of transition $(q, \sigma, \phi, \rho, p)$.

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

2.2.3 Run of an automaton.

Let $\mathcal{A} = (\Sigma, Q, I, F, Z, \Delta)$ be a signal automaton and m be a signal. Since m can be split / spliced in an infinity of ways, it is useless to consider some precise representation for it; instead, m is considered to be a piecewise constant function from $\mathbb{R}_+$ to Σ. Let m_t the value of m at time t.

A run of m over $\mathcal{A}$ is a finite sequence of transitions and dates $\{(\delta_i, t_i)\}_{1 \leq i \leq n}$. Let t_0=0 and $\delta_i = (q_i, \sigma_i, \phi_i, \rho_i, p_i)$. A run must satisfy:

1. $\forall i$, $1 \leq i < n$, $p_i = q_{i+1}$,
2. $\forall i$, $1 \leq i \leq n$, $\forall t \in (t_{i-1}, t_i)$, $m_t = \sigma_i$,
3. $Z_0 = \vec{0}$ (initialization),
4. $\forall i$, $1 \leq i \leq n$, $\forall \delta \in [0, t_i - t_{i-1}]$, $\phi_i(Z_{t_i} + \delta)$ is satisfied,
5. $\forall i$, $1 \leq i \leq n$, $Z_{t_i} = Reset(Z_{t_{i-1}} + t_i - t_{i-1}, \rho_i)$,
6. $\exists d > 0$, $\forall i$, $1 \leq i \leq n$, $d < t_i - t_{i-1}$ (monotony and progression).

Let us note that the value of m at t_i is not considered (2.) whereas (4.) clock constraints ϕ_{i-1} and ϕ_i have to be satisfied at t_i (without resetting the clocks for ϕ_{i-1}). Condition 6. implies that there is no Zeno configuration (i.e. no accumulation point).

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 signal for which there exists an accepting run.

2.3 (Piecewise constant) signal regular expression

The set of *signal regular expressions* over Σ, $\mathcal{R}(\Sigma)$, is defined inductively using ε, σ, $\varphi_1 \vee \varphi_2$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \cdot \varphi_2$, φ^* and $\langle \varphi \rangle_I$, s.t. $a \in \Sigma$, $\varphi, \varphi_1, \varphi_2 \in \mathcal{R}(\Sigma)$ and I is an open interval of $\mathbb{R}_+$ either (d, d') or (d, ∞) s.t. $d, d' \in \mathbb{Q}$ and $0 \leq d < d'$. The semantic of regular expressions is:

1. $[\![\varepsilon]\!] = \{\varepsilon\}$,
2. $[\![\sigma]\!] = \{\sigma^r \mid r \in \mathbb{R}_+^*\}$,
3. $[\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \bigcup [\![\varphi_2]\!]$,
4. $[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \bigcap [\![\varphi_2]\!]$,
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. $[\![\langle \varphi \rangle_I]\!] = \{\, m \in [\![\varphi]\!] \mid |m| \in I \,\}$.

3 From regular expressions to automata

Basic (signal) regular expressions are shown to correspond to (signal) automata and then that automata are closed for the same operators. All is done following usual constructions for the un-timed and time-event models. Emphasis is made on substantially different constructions: the automata product used for $\wedge$ (possible splittings have to be considered) and duration restrictions.

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

3.0.1 Basic constructions.

Figure 3 shows the constructions for ε and a.

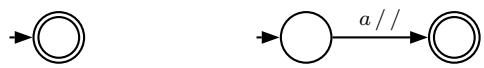

Figure 3: Automata for $\mathcal{L}(\varepsilon)$ and $\mathcal{L}(a)$.

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

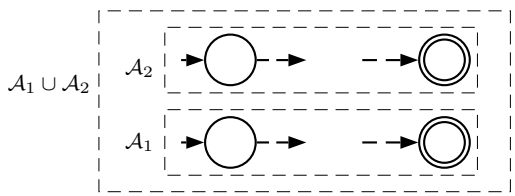

Figure 4: Automaton for $\mathcal{L}(\mathcal{A}_1) \cup \mathcal{L}(\mathcal{A}_2)$.

3.0.2 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. $a^1a^3b^1b^2b^1$ on one automaton and $a^4b^2b^2$ on the other. These signals both represent a^4b^4 and have infinitely many common sub-splits (e.g. $a^1a^3b^1b^1b^1b^1$), 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 a new state as depicted on Fig. 5. The generated automaton accepts exactly the same language and any sub-split of a sub-split corresponding to an accepting run also corresponds to an accepting run.

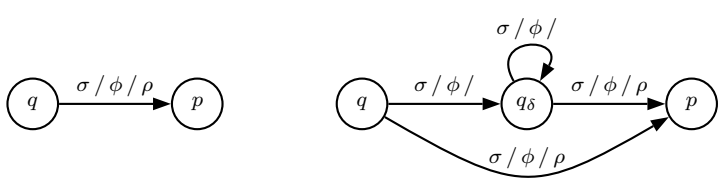

Figure 5: Full split of transition $\delta = (q, \sigma, \phi, \rho, p)$.

With this *full split form*, it is easy to construct the product and then the intersection. Figure 6 shows the product of transitions; of course only and all pairs of transitions with the same letter are considered. The set of initial (accepting) states is the product of initial (accepting) states. If m is 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 m is not accepted by $\mathcal{A}_1$ (or $\mathcal{A}_2$), then neither it is by the full split form of $\mathcal{A}_1$ and neither by the product.

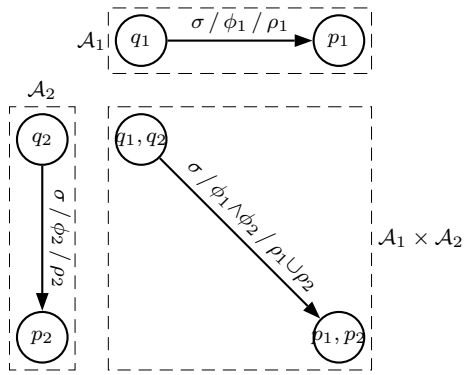

Figure 6: Product of transitions $(q_1, \sigma, \phi_1, \rho_1, p_1)$ and $(q_2, \sigma, \phi_2, \rho_2, p_2)$.

3.0.3 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 states are the ones of $\mathcal{A}_1$ and the accepting states are the ones of $\mathcal{A}_2$.

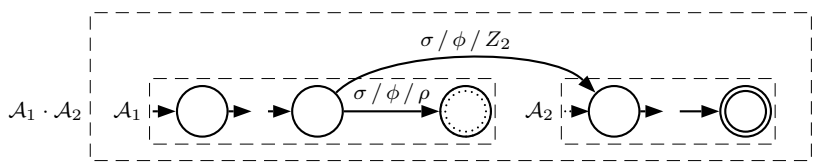

Figure 7: Transition added for $\mathcal{L}(\mathcal{A}_1) \cdot \mathcal{L}(\mathcal{A}_2)$.

For the finite iteration (or Kleene star), copies of each transition to a final state are made leading to each initial state; these copies reset all the clocks. A copy of an automaton which recognize ε is added for zero iteration. This is sumed up in Fig. 8.

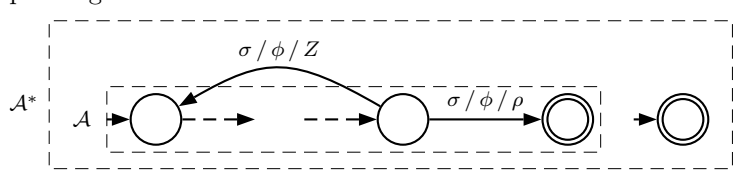

Figure 8: Transitions and state added for $\mathcal{L}(\mathcal{A})^*$.

3.0.4 Duration restriction.

It can not 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 a new state and two consecutive transitions (one extra split in the run); the time duration restriction is only added in the

last constraint as depicted on Fig. 9. A new clock, z_0 is added as well as a new (and only) final state q_f. 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, the total duration of any accepted input has to be in I.

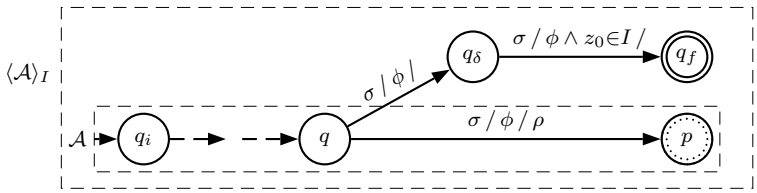

Figure 9: Automaton for $\langle\mathcal{L}(\mathcal{A})\rangle_I$.

Finally, by induction:

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

If a renaming λ is applied to $\mathcal{L}(\varphi)$, it remains to apply it to every transition.

4 From automata to regular expressions

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

4.1 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. Signals, and thus transitions, can be split and disjunction can be disposed of as in Fig. 10 where ϕ_1 and ϕ_2 may still contain $\vee$.

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 letter by a letter appearing only in this transition. Signals are lift to sub-splits

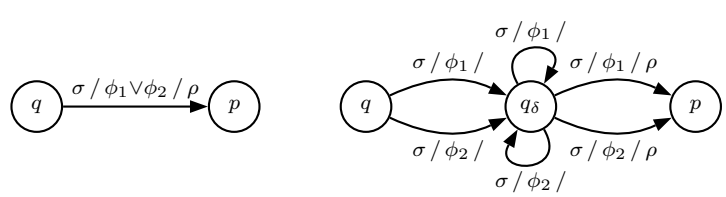

Figure 10: Removing disjunction in $\delta = (q, \sigma, \phi_1 \vee \phi_2, \rho, p)$.

where transitions are indicated. The inverse mapping from the large set of letters to the original one is denoted λ.

4. One copy of the automaton is made for each clock, setting to true atomic constraints over other clocks. From the deterministic association of letters to transitions, any run in one automaton could only correspond to one run (the same) for each copy. If a signal is accepted by $\mathcal{A}$ then it is accepted by all the copies (constraints are only conjunctive). Reciprocally, if a signal is accepted by all the copies, then it has to be with the same run which is also accepting for $\mathcal{A}$ (conjunctions of constraints satisfied for all clocks).

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

$$\mathcal{L}(\mathcal{A}) \;=\; \lambda\left(\bigcap_{z\in Z}\mathcal{L}(\mathcal{A}_z)\right) \quad . \tag{1}$$

Copies can be treated individually.

4.2 One-clock automaton to regular expression

Let $\mathcal{A}_z = (\Sigma, Q, I, F, \{z\}, \Delta)$ be a signal automaton with only one clock z. Let $0{=}\tau_0 < \tau_1 < \tau_2 \cdots < \tau_\iota < \tau_{\iota+1}{=}\infty$ be the list of all critical times (plus 0 and ∞), the constants that appear in at least one elementary clock constraint in $\mathcal{A}$.

Since constraints are conjunctions over atomic formulae $z{<}\tau_i$ or $\tau_j{<}z$, they corresponds to $\tau_\alpha < z < \tau_\beta$ (or to false and are removed), since $0{=}\tau_0$ and $\tau_{\iota+1}{=}\infty$ all kinds of intervals are covered. Constraints are "constant" on each (τ_i, τ_{i+1}), either satisfied or not.

Let Δ_1 be the subset of reset-less transitions and Δ_2 the subset of resetting transitions ($\Delta = \Delta_1 \cup \Delta_2$).

4.2.1 One-clock automaton without reset.

Only the transitions in Δ_1 are considered in this subsection. 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 (τ_i, τ_{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 τ_i. The $L^{\tau_i}_{q\to p}$ can be computed recursively:

$$L^{\tau_1}_{q\to p} \;=\; \langle L(q,p,0)\rangle_{(0,\tau_1)} \quad , \tag{2}$$

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

Before τ_1, the automaton is "constant", to that classical language theory gives an un-timed expression which only needs to have its duration restrained (2). Equation (3) holds because duration could be less than τ_i, equals to τ_i (covered

by the second line) or be strictly between τ_i and τ_{i+1}. Equation (2) directly gives a signal regular expression, (3) uses concatenation, finite union and regular expression as constants, so all $L^{\tau_i}_{q\to p}$ correspond to signal regular expressions.

Let $L_{q\to p}$ denote the language corresponding to the runs from q to p without any duration restriction: $L_{q\to p} = L^{\infty}_{q\to p} = L^{\tau_{\iota+1}}_{q\to p}$.

4.2.2 One-clock automata with reset.

The resetting transitions, i.e. the ones in Δ_2, are now considered, together with the ones in Δ_1. Let $\Delta_2 = \{\delta_1, \delta_2, \dots \delta_\kappa\}$ and $\delta_k = (q_k, \sigma_k, \tau_{\alpha_k}{<}z{<}\tau_{\beta_k}, \{z\}, p_k)$. Let $L^{\delta_1..\delta_k}_{s,\,\delta_l}$ be the language corresponding to the runs starting from s, using only resetting transitions $\delta_1, \delta_2, \dots \delta_k$ and ending by δ_l ($l \le k$). The following recurrence equations are satisfied:

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

$$L^{\delta_1..\delta_k}_{s,\,\delta_k} = \left(\left\langle \left\langle L_{s\to q_k} \right\rangle_{(\tau_{\alpha_k},\infty)} .\sigma_k \right\rangle_{(0,\tau_{\beta_k})} \cup \bigcup_{1\le l<k} L^{\delta_1..\delta_{k-1}}_{s,\,\delta_l} . \left\langle \left\langle L_{p_l\to q_k} \right\rangle_{(\tau_{\alpha_k},\infty)} .\sigma_k \right\rangle_{(0,\tau_{\beta_k})} \right)$$
$$. \left(\left\langle \left\langle L_{p_k\to q_k} \right\rangle_{(\tau_{\alpha_k},\infty)} .\sigma_k \right\rangle_{(0,\tau_{\beta_k})} \cup \bigcup_{1\le l<k} L^{\delta_1..\delta_{k-1}}_{p_k,\,\delta_l} . \left\langle \left\langle L_{p_l\to q_k} \right\rangle_{(\tau_{\alpha_k},\infty)} .\sigma_k \right\rangle_{(0,\tau_{\beta_k})} \right)^*, \quad (5)$$

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

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

Resetting transition δ_1 has to be done at least once, then the run can go back to it any number of times (4). The same holds for a run ending by δ_k, it has to be done once, and can come back to it, each time other allowed resetting transitions may be used or not (5). A run ending by δ_l do not use δ_k after the last passage, if any, through δ_k (6).

All these equations start with (and use) signal regular expressions, and use 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 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.*

5 Conclusion

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

Renaming seems to be unavoidable as proved in the time-event model [Her99]. The emptiness of a language accepted by a signal automaton should also be investigated, as well as special composition or verification of specifications given in term of another signal automaton.

We believe that complexity and feasibility are not too different from the ones in the time-event context. We also believe that there is some algebraic context (like in [Asa98, ACM02]) and that infinite duration and zeno configurations can be approached with techniques as in [BP00].

References

[ACM97] Eugene Asarin, Paul Caspi, and Oded Maler. A Kleene theorem for timed automata. In *IEEE Logic in Computer Science*, pages 160–171, 1997.

[ACM02] Eugene Asarin, Paul Caspi, and Oded Maler. Timed regular expressions. *Journal of the ACM*, 49(2):172–206, 2002.

[AD94] Rajeev Alur and David L. Dill. A Theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 25 April 1994.

[Asa98] Eugene Asarin. Equations on timed languages. In *HSCC*, pages 1–12, 1998.

[BP00] Béatrice Bérard and Claudine Picaronny. Accepting zeno words: a way towards timed refinements. *Acta Informatica*, 37(1):45–81, 2000.

[BP02] Patricia Bouyer and Antoine Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 2002. To appear.

[Dim00] Cătălin Dima. Real-time automata and the Kleene algebra of sets of real numbers. In *STACS '00*, volume 1770 of *Lecture Notes in Computer Science*, pages 279–290, 2000.

[Her99] Philippe Herrmann. Renaming is necessary in timed regular expressions. In *FSTTCS '99*, number 1738 in LNCS, pages 47–59, 1999.