Journal of Automata, Languages and Combinatorics **u** (v) w, x–y


# Real-time automata[1]

Cătălin Dima
*Bucharest University, Department of Fundamentals of Computer Science,*
*str. Academiei 14, Bucharest, RO 70109 Romania*
*and*
*INRIA Rhône-Alpes, 655 Av. de l'Europe, 38330, Montbonnot St. Martin, France.*
*e-mail:* cdima@funinf.math.unibuc.ro

ABSTRACT

We study here the class of timed automata with a single clock which is reset at each transition. We adapt for these automata the classical results for finite automata: the Kleene theorem, the closure under complementation and the Pumping Lemma. We provide an algorithm for the elimination of stuttering steps, which is essential in complementation. This algorithm relies upon the properties of the Kleene algebra of sets of real numbers, namely the existence of a normal form for sets of reals generated from intervals with rational bounds, using boolean operations, summation and star.

*Keywords:* Real-time automata, rational expressions, Kleene algebra.

## 1. Introduction

The search for an appropriate class that may bear the name of *real-time regular languages* has produced several results like [AD94, AFH94, HRS98]. The features sought for were: closure under complementation and/or relationship to some monadic logic of order over the real-time axis that generalizes MSO over words [Tho97, Wil94]. The relationship with some rational expressions was not under attention when assigning the label *regular* to a class of languages. It is already known [ACM97, BP99, Di99b] that rational (or regular) expressions do not easily fit the classes of timed automata or event-clock automata. This amounts to the problem of giving semantics for concatenation: in the presence of a total operation like in [ACM97] one direction of the Kleene theorem is missing [Her99]. For obtaining both directions of the Kleene theorem a *partial* operation on timed words or signals is needed for the semantics of concatenation, see [BP99, Di99b].

In this paper we study a class of automata which we claim to be the largest extension from finite automata still carrying the decidability of both the emptiness and the universality problems, a Pumping Lemma and, moreover, a Kleene theorem in which the semantics of the associated rational expressions is based upon a total "concatenation" operation. The automata we study, called Real-Time Automata (RTA), can be

[1] A preliminary version of this paper appeared as [Di00].

regarded as timed automata with a single clock which is reset at each transition, and they appeared (in a slightly different version) in connection to the so-called Simple Duration Calculus [DW96].

However, even at this lowest level of introduction of timing constraints into finite automata we find that complementation raises a specific problem: the subset construction can be adapted to handle timing constraints, but it works only if the automata are *stuttering-free*, i.e., two states labeled with the same symbol are not connected by any transition. We also find out that *language determinism*, i.e., the property that each signal is associated with a unique run that starts in an initial state, cannot be captured by local properties like state-determinism or stuttering-freeness: stuttering-free state-deterministic RTA are less expressive than RTA.

We solve this problem by introducing the Kleene algebra of sets of real numbers. The rôle of concatenation from Kleene algebras of languages is taken here by addition of sets of real numbers. This operation models the process of removing one stuttering transition by "fusing" the adjacent states. We then study the sub-Kleene algebra generated by finite unions of intervals with rational bounds and prove a normal form theorem for this subalgebra, result which is based on properties of integer division and roughly says that elements in this subalgebra are "ultimately periodic". This result is not a corollary of the normal form for rational languages over a one letter alphabet because the Kleene algebra that arises from intervals with nonnegative integer bounds has two generators whose generated subalgebras are not disjoint but which cannot generate one another.

We have chosen here the "signals" approach for modeling real-time systems, hence stuttering elimination might look a peculiarity of this semantics. However, if one chooses the "timed words" semantics [AD94] the stuttering-freeness property would translate to the *absence of epsilon-transitions* and the technique for removing these would be the one we develop here. The determinization construction would be the same and, roughly speaking, similar to the determinization construction for event-clock automata [AFH94].

It is worth mentioning that the aim of this study is not to compare the expressive power of real-time automata with the existing models [AD94, AFH94, HRS98], but rather to point out the properties carried by the class of languages defined by RTA. It is clear that real-time automata are insufficiently expressive to be used for modeling distributed real-time systems, see the nonrational language from section 5. However we may observe that they are incomparable to (recursive) event-clock automata [AFH94, HRS98] which are so far the largest determinizable subclass of timed automata. This situation occurs as real-time automata may accept languages consisting of signals in which two actions may be separated by an interval with integer length while event-clock automata may not.

The rest of the paper is divided as follows: in the next section we remind what RTA are, their associated expressions and what are the problem posed by their complementation. In the third section we introduce the Kleene algebra of sets of positive reals and we prove the normal form theorem. The fourth section contains the constructions that accomplish stuttering elimination and determinization and the fifth is reserved for a pumping lemma and expressivity issues concerning real-time automata.

We end with a section containing short comments and directions of further study.

## 2. Real-time automata and their rational expressions

We denote $\mathbb{R}_{\geq 0}$ and $\mathbb{R}_{>0}$ the sets of nonnegative, resp. positive numbers, $\mathbb{Q}_{\geq 0}$ the set of nonnegative rational numbers and, for each $n \in \mathbb{N}$, $[n] = \{1, \ldots, n\}$. $Int$ denotes the set of intervals having bounds in $\mathbb{Q}_{\geq 0} \cup \{\infty\}$ and including the empty set. For some arbitrary set $A$, some $I \in Int$, $\alpha \in I$ and a function $f : I \longrightarrow A$ we say that $f$ *has a discontinuity at* $\alpha$ iff there exist $a, b \in A$, $a \neq b$ and some $\epsilon > 0$ such that

$$\forall x \in (\alpha - \epsilon, \alpha),\ f(x) = a \quad \text{and} \quad \forall x \in (\alpha, \alpha + \epsilon),\ f(x) = b$$

The discontinuity[2] is *left* iff we also have that $f(\alpha) = b$.

**Definition 2.1** *A **signal** over a finite alphabet $\Sigma$ is a function $\sigma : [0, e) \longrightarrow \Sigma$ where $e$ is a nonnegative number, function which has finitely many discontinuities, all of them being left discontinuities.*

Hence the domain of a signal $\sigma$ splits into finitely many intervals $[e_{i-1}, e_i)$ on which $\sigma$ is constant. We denote by $dom(\sigma)$ the domain of $\sigma$, $endp(\sigma)$ its endpoint and $Sig(\Sigma)$ the set of signals over $\Sigma$. Subsets of $Sig(\Sigma)$ are called **real-time languages**.

For $\sigma_1, \sigma_2 \in Sig(\Sigma)$ with $dom(\sigma_i) = [0, e_i)$ $(i = 1, 2)$ define their *concatenation* $\sigma_1; \sigma_2 = \sigma$ as the signal with $dom(\sigma) = [0, e_1 + e_2)$ and such that

$$\sigma(t) = \begin{cases} \sigma_1(t) & \text{for } t \in [0, e_1), \\ \sigma_2(t - e_1) & \text{for } t \in [e_1, e_1 + e_2). \end{cases}$$

Hence $(Sig(\Sigma), \text{“;”}, \sigma_\epsilon)$ becomes a noncommutative monoid whose unit is the signal $\sigma_\epsilon$ with $dom(\sigma_\epsilon) = [0, 0) = \emptyset$. Then concatenation can be extended to sets of signals:

$$L_1; L_2 = \{\sigma_1; \sigma_2 \mid \sigma_1 \in L_1, \sigma_2 \in L_2\}$$

and gives rise to star:

$$L^* = \bigcup_{n \in \mathbb{N}} L^n$$

where $L^0 = \{\sigma_\epsilon\}$ and $L^{n+1} = L^n; L$.

### *2.1. Real-time automata defined*

**Definition 2.2** *A **real-time automaton** (RTA for short) is a tuple $\mathcal{A} = (Q, \Sigma, \lambda, \iota, \delta, Q_0, Q_f)$ where $Q$ is the (finite) set of* states*, $\Sigma$ is the (finite nonempty)* alphabet*, $\delta \subseteq Q \times Q$ is the* transition relation*, $Q_0, Q_f \subseteq Q$ are the sets of* initial*, resp.* final *states, $\lambda : Q \longrightarrow \Sigma$ is the* state labeling function *and $\iota : Q \longrightarrow Int$ is the* time labeling function.

[2]This definition of discontinuity amounts to considering the discrete topology on $A$.

We also call the pair $\big(\lambda(q), \iota(q)\big)$ the **label** of the state $q$.

RTA work over signals: a **run** of length $n \geq 1$ is a sequence of states $(q_i)_{i\in[n]}$ connected by $\delta$, i.e., $(q_i, q_{i+1}) \in \delta, \forall i \in [n-1]$. The run is **accepting** iff it starts in $Q_0$ and ends in $Q_f$.

A run is **associated with** a signal $\sigma$ iff there exist some sequence of "splitting" points $0 = e_1 \leq \ldots \leq e_{n+1} = endp(\sigma)$ such that $e_{i+1} - e_i \in \iota(q_i)$ and $\sigma(t) = \lambda(q_i)$ for all $t \in [e_i, e_{i+1})$ and all $i \in [n]$. Hence the "splitting" points must contain all the discontinuities in the signal but this inclusion might be strict, case in which we say that the run is *stuttering*.

When a signal $\sigma$ is associated with some accepting run we say that $\sigma$ is **accepted** by $\mathcal{A}$. The **language** of $\mathcal{A}$ is the set of signals associated with some accepting run of $\mathcal{A}$ and is denoted $L(\mathcal{A})$. Two RTA are **equivalent** iff they have the same language. If we denote the class of all RTA whose alphabet is $\Sigma$ as $\mathrm{RTA}(\Sigma)$, then we may define the class of **real-time recognizable languages** over $\Sigma$ as

$$TRec(\Sigma) = \{L \in Sig(\Sigma) \mid \exists \mathcal{A} \in \mathrm{RTA}(\Sigma) \text{ s.t. } L(\mathcal{A}) = L\}$$

As an example, the automaton in Figure 1 accepts the signal $\sigma : [0, 9) \longrightarrow \{a, b\}$ where $\sigma(x) = a$ for $x \in [0, 2.5)$ and $\sigma(x) = b$ for $x \in [2.5, 9)$. The accepting run which is associated with this signal is $(q, r, s, t)$ and the splitting points are $e_1 = 0$, $e_2 = 2.5$, $e_3 = e_4 = 6.5$ and $e_5 = 9$. Note that the run is stuttering.

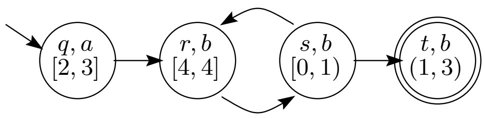


Figure 1: An example of a real-time automaton.

Real-time automata can be viewed as state-labeled timed automata [ACM97] with a single clock which is reset at each transitions. It is easy to define a class of real-time automata that work over timed words instead of signals and to transport to that setting the results of this study. Most notably stuttering steps would translate into epsilon-transitions in the timed words setting.

On the other hand, a RTA in $\mathrm{RTA}(\Sigma)$ is, from a syntactic point of view, a "finite presentation" of a classical automaton over the (uncountable) set of symbols $\Sigma \times \mathbb{R}_{\geq 0}$, where a state $q$ labeled $(a, I)$ embodies a whole family of states labeled with $(a, \alpha)$ for all $\alpha \in I$. However the comparison stops at this syntactic level since semantically $\Sigma \times \mathbb{R}_{\geq 0}$ comes with a structure which is unavailable for the set of symbols in a classical automaton, structure which allows the "fusion" of two symbols sharing the same state label. It is this structure that allows the acceptance of the signal $\sigma$ in the above example by splitting the symbol $(b, 6.5)$ into three symbols $(b, 4)$, $(b, 0)$ and $(b, 2.5)$.

We end this subsection with the following adaptation of the decidability of the emptiness problem for finite automata.

**Proposition 2.3** *The emptiness problem for RTA is decidable.*

The proof relies on the algorithm for computing the sets of accessible states and then checking whether a final state is accessible, which can be done in linear time w.r.t. $card(Q) \cdot card(\delta)$.

*2.2. Rational expressions and the Kleene theorem*

We have observed that labels in RTA are finite presentations of sets of symbols from $\Sigma \times \mathbb{R}_{\geq 0}$. This observation can be further extended to considering rational expressions over $\Sigma_{Int} := \Sigma \times Int$ with the aim of obtaining a Kleene theorem:

**Definition 2.4** *Consider* $\mathbf{Rat}(\Sigma_{Int}$*, the set of rational (or regular) expressions over* $\Sigma_{Int}$*, i.e., defined by the following grammar*

$$E ::= 0 \mid \mathbf{1} \mid a_I \mid E + E \mid E; E \mid E^*$$

*where the* atoms $a_I$ *are any symbols from* $\Sigma_{Int}$*.*

*A rational expression* $E \in \mathbf{Rat}(\Sigma_{Int})$ *will be called also as* ***real-time rational expression.***

There are two types of semantics for real-time rational expressions: the first one, called henceforth **abstract**, is the classic semantics in terms of words over the set of symbols $\Sigma_{Int}$ and is denoted $|\cdot|$. For this semantics, $|0|$ is the empty set and $|\mathbf{1}|$ is the set containing the empty word over $\Sigma_{Int}$, word which is denoted $\mathbf{1}$ too.

The second semantics, called the **real-time semantics** or simply **the semantics**, is in terms of signals and is denoted $\|\cdot\|$:

$$\|a_I\| = \{\sigma \in Sig(\Sigma) \mid dom(\sigma) = [0, e), e \in I \text{ and } \forall t \in [0, e) \;\; \sigma(t) = a\}$$

$$\begin{aligned} \|E+F\| &= \|E\| \cup \|F\| & \qquad \|0\| &= \emptyset \\ \|E; F\| &= \|E\|; \|F\| & \qquad \|\mathbf{1}\| &= \{\sigma_\epsilon\} \\ \|E^*\| &= \|E\|^* \end{aligned}$$

Note also that $\|a_{[0,0]}\| = \{\sigma_\epsilon\}$ for any $a \in \Sigma$.

The following straightforward property relates the two types of semantics:

**Proposition 2.5** *For each real-time rational expression* $E \in \mathbf{Rat}(\Sigma_{Int})$*,*

$$\|E\| = \bigcup \big\{ \|w\| \mid w \in |E| \big\}$$

Define the class of **real-time rational languages** over $\Sigma$ as

$$TReg(\Sigma) = \{L \in Sig(\Sigma) \mid \exists E \in \mathbf{Rat}(\Sigma_{Int}) \text{ such that } \|E\| = L\}$$

**Theorem 2.6 (Kleene theorem for RTA, [Di99a])** $TRec(\Sigma) = TReg(\Sigma)$.

The Kleene theorem would follow almost immediately from proposition 2.5 and the classical Kleene theorem [HU] if we would have transition-labeled real-time automata rather than state-labeled. Since this kind of automata will further show useful, we define them here and provide the straightforward translations from and to RTA:

**Definition 2.7** *A* ***transition-labeled RTA*** *(*t-RTA *for short) is a tuple* $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ *where* $Q$*,* $\Sigma$*,* $Q_0$ *and* $Q_f$ *have the same names and properties as in RTA and the transition relation* $\delta$ *satisfies* $\delta \subseteq Q \times \Sigma \times Int \times Q$ *with* $card(\delta) < \infty$*.*

Transitions of the form $(q, a, I, r) \in \delta$ will be called *a-transitions.*

Since a transition-labeled RTA is a finite automaton over a finite subset of $\Sigma_{Int}$, we may speak of its language in the classical sense, as the set of words over $\Sigma_{Int}$ which are concatenations of the labels of some accepting run. Let's call this the **abstract language** and denote it as $L_{abs}(\mathcal{A})$. The **real-time language accepted** by $\mathcal{A}$, or simply **the language** of $\mathcal{A}$, denoted $L(\mathcal{A})$, is then the union of the semantics of each word in $L_{abs}(\mathcal{A})$, with this abstract word viewed as a rational expression over $\Sigma_{Int}$, that is,

$$L(\mathcal{A}) = \bigcup \big\{ \|w\| \mid w \in L_{abs}(\mathcal{A}) \big\} \tag{1}$$

The translations between RTA and transition-labeled RTA are the usual transformations of a state-labeled automaton into a transition-labeled one and back, with a special case when the empty signal is accepted by the t-RTA:

- Given some RTA $\mathcal{A} = (Q, \Sigma, \lambda, \iota, \delta, Q_0, Q_f)$, a transition-labeled RTA with the same language is $\mathcal{B} = (Q \cup \{t_0\}, \Sigma, \theta, \{t_0\}, Q_f)$ where $t_0 \notin Q$ and

$$\theta = \big\{ \big(q, \lambda(r), \iota(r), r\big) \mid (q, r) \in \delta \big\} \cup \big\{ \big(t_0, \lambda(q), \iota(q), q\big) \mid q \in Q_0 \big\}$$

- For the reverse, given some transition-labeled RTA $\mathcal{B} = (Q, \Sigma, \theta, Q_0, Q_f)$ a RTA whose language is $L(\mathcal{B})$ is $\mathcal{A} = (\theta \cup \{q_\epsilon\}, \Sigma, \delta, \lambda, \iota, T_0, T_f)$ where
  - for each RTA state $(q, a, I, r) \in \theta$, $\lambda\big((q, a, I, r)\big) = a$ and $\iota\big((q, a, I, r)\big) = I$;
  - $\lambda(q_\epsilon) = a$ for some $a \in \Sigma$ (assumed nonempty) and $\iota(q_\epsilon) = [0, 0]$;
  - $T_0 = \{(q, a, I, r) \mid q \in Q_0\} \cup \{q_\epsilon\}$;
  - $T_f = \{(q, a, I, r) \mid r \in Q_f\} \cup \{q_\epsilon \mid \sigma_\epsilon \in L(\mathcal{B})\}$;
  - $\delta = \big\{ \big((q, a, I, r), (r, b, J, s)\big) \mid (q, a, I, r), (r, b, J, s) \in \theta \big\}$

Hence, when $\sigma_\epsilon \in L(\mathcal{B})$ we must add to $\mathcal{A}$ an initial and final state for accepting $\sigma_\epsilon$. Note that this state is neither the source nor the target of any transition.

The proof of the Kleene theorem is then the following: each RTA $\mathcal{A}$ is equivalent to some t-RTA, whose abstract language equals the abstract semantics of some real-time rational expression $E$ due to the classical Kleene theorem. Then, by combining properties 2.5 and equation 1 we obtain that the (timed) semantics of $E$ and the language of $\mathcal{A}$ are equal. The reverse implication is similar. □

We end this subsection with a procedure for removing the zeroes from the time labels in RTA which is a straightforward adaptation of the epsilon-elimination procedure for finite automata [HU]: for each t-RTA $\mathcal{A} = (Q, \Sigma, \delta, Q_0, Q_f)$ we split each transition $(q, a, I, r)$ with $0 \in I$ in two, the first being $\big(q, a, I \setminus \{0\}, r\big)$ and the second $\big(q, a, [0, 0], r\big)$. Then we may compare transitions $(q, a, [0, 0], r)$ with transitions with the empty word in finite automata. Hence, for each state $q$ in $\mathcal{A}$, we define

$$\epsilon(q) = \{q' \in Q \mid \text{ there exists a run } \big((q_{i-1}, a_i, [0, 0], q_i)\big)_{i\in[n]} \text{ with } q = q_0, q' = q_n\}$$

and then construct a t-RTA $\overline{\mathcal{A}} = (\overline{Q}, \Sigma, \overline{\delta}, Q_0, \overline{Q}_f)$ equivalent to $\mathcal{A}$, where

$$\begin{aligned} \overline{\delta} &= \big\{(q, a, I \setminus \{0\}, r) \mid \exists (q, a, I, s) \in \delta \text{ and } r \in \epsilon(s)\big\} \\ \overline{Q}_f &= Q_f \cup \{q \in Q \mid Q_f \cap \epsilon(q) \neq \emptyset\} \end{aligned}$$

Note that when translating transition-labeled RTA without zero labels into state-labeled RTA, we will get the special initial state $q_\epsilon$ whose time label is $[0, 0]$, needed for not loosing the empty signal from the accepted language. Hence:

**Proposition 2.8** *Each transition-labeled RTA is equivalent to some t-RTA in which the interval labels of the transitions do not contain* $0$*.*

*Each RTA is equivalent to some RTA in which there exists a single state whose interval label contains* $0$*, the label of this state is actually* $[0, 0]$ *and no transition enters or leaves this state.*

### *2.3. The problem of complementation of real-time automata*

The usual way of showing that a class of automata is closed under complementation is to prove that the automata can be transformed such that for each word there exists a unique run that starts in an initial state, for then complementation would be accomplished simply by complementing the set of final states. The notions that assure the *uniqueness* of the run for RTA are state-determinism combined with stuttering freeness:

**Definition 2.9** *A RTA* $\mathcal{A}$ *is* ***language deterministic*** *iff each signal in* $L(\mathcal{A})$ *is associated with a unique run that starts in an initial state.*

$\mathcal{A}$ *is* ***stuttering-free*** *iff*

- *there exists a state* $q_\epsilon \in Q_0$ *whose time label contains* $0$*; its time label is actually* $\iota(q_\epsilon) = [0, 0]$ *and it is not connected to any other state;*
- *the time labels of all the other states do not contain* $0$*;*
- *for each transition* $(q, r) \in \delta$*,* $\lambda(q) \neq \lambda(r)$*.*

$\mathcal{A}$ *is* ***state-deterministic*** *iff initial states have disjoint labels and transitions starting in the same state have disjoint labels too, i.e., whenever* $r \neq s$ *and either* $r, s \in Q_0$ *or* $(q, r), (q, s) \in \delta$ *then either* $\lambda(r) \neq \lambda(s)$ *or* $\iota(r) \cap \iota(s) = \emptyset$*.*

$\mathcal{A}$ *is called* ***deterministic*** *iff it is both state-deterministic and stuttering-free.*

The translations of these notions for transition-labeled RTA are the following:

**Definition 2.10** *A t-RTA $\mathcal{A}$ is* ***transition-deterministic*** *iff it has a single initial state and for each state $q \in Q$ and symbol $a \in \Sigma$, if two distinct $a$-transitions leave $q$ then their time labels are disjoint, i.e., if $(q, a, I, r), (q, a, J, s) \in \delta$ then either $I = J$ plus $r = s$ or $I \cap J = \emptyset$.*

*$\mathcal{A}$ is* ***stuttering-free*** *iff the time labels of the transitions do not contain zero and there are no two distinct adjacent transitions labeled with the same symbol, i.e., if $(q, a, I, r), (r, b, J, s) \in \delta$ then $a \neq b$.*

*$\mathcal{A}$ is* ***deterministic*** *iff it is state-deterministic and stuttering-free.*

**Proposition 2.11** *The translations between RTA and t-RTA provided in section 2.2 are such that*

- *state determinism in RTA is translated to transition determinism in t-RTA and vice-versa and*
- *stuttering-freeness in RTA is translated to stuttering-freeness in t-RTA and vice-versa.*

It is clear that determinism implies language determinism while state-determinism itself does not. But a more important observation is that stuttering-free RTA are *strictly less expressive* than general RTA: consider the language $L_{\mathbb{N}} = \{\sigma : [0, n) \longrightarrow \{a\} \mid n \in \mathbb{N}\}$ of constant signals with integer length which is accepted by the RTA in the Figure 2.

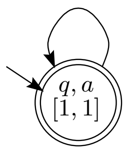


Figure 2: An RTA for the language $L_{\mathbb{N}}$.

**Proposition 2.12** *$L_{\mathbb{N}}$ cannot be accepted by any stuttering-free RTA.*

*Proof.* The proof is based on the intuition that a stuttering-free RTA for $L_{\mathbb{N}}$ would need an infinite number of states:

Suppose we had a stuttering-free automaton $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, T_0, T_f)$ which would recognize $L_{\mathbb{N}}$. We may consider $\Sigma = \{a\}$ since any state with other state-labels cannot be in an accepting run. Then, since the automaton is stuttering-free, $\delta = \emptyset$. Hence the number of accepting runs in $\mathcal{A}$ equals the number of initial and final states. Denote then $\mu$ the max in $\mathbb{R}_{\geq 0} \cup \{\infty\}$ of the time labels of these initial and final states. But then both the assumption $\mu = \infty$ and $\mu < \infty$ lead to a contradiction:

If $\mu = \infty$ then for some state $q \in Q_0 \cap Q_f$ we have that $\iota(q) = \llparenthesis l, \infty)$ where $\llparenthesis$ is any left parenthesis. Then any constant signal $\sigma : [0, \alpha) \longrightarrow \{a\}$ with $\alpha \in \llparenthesis l, \infty)$ would be accepted by $\mathcal{A}$, contradicting the assumption that only signals with natural endpoints are accepted.

On the other hand, if $\mu < \infty$ then any constant signal $\sigma : [0, n) \longrightarrow \{a\}$ with $n \in \mathbb{N}$, $n > \mu$, is not accepted by $\mathcal{A}$, again a contradiction. □

This proof can be easily adapted for showing that event-clock automata [AFH94, HRS98] cannot accept the language $L_\mathbb{N}$ or the following language:

$$|[b]_{(0,\infty)}|; L_\mathbb{N}; |[b]_{(0,\infty)}|$$

that contains all signals starting with $b$, continuing with $a$ and ending in $b$ and having the property that the distance between the $b \to a$ discontinuity and the $a \to b$ discontinuity is a natural number. Hence the class of languages accepted by event-clock automata does not contain the class of real-time rational languages as defined here.

Despite Proposition 2.12, there is no problem in building a RTA for the complement of $L_\mathbb{N}$: it is the RTA in Figure 3($a$) below. Also in Figure 3($b$) we have an example of some stuttering RTA which is equivalent with the stuttering-free RTA in the same figure, at ($c$).

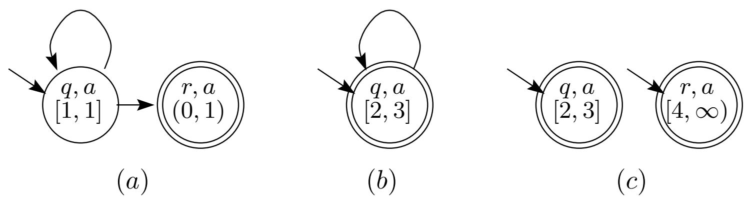


Figure 3: The complement of $L_\mathbb{N}$ is accepted by the RTA at ($a$). The stuttering RTA at ($b$) is equivalent to the stuttering-free one at ($c$).

Hence we discover the need of computing the "sum" of two intervals and the "star" of some interval, i.e., some operations that satisfy the following relations suggested by Figure 3:

$$\mathbb{R} \setminus \{1\}^* = \{1\}^* + (0, 1) \quad \text{and} \quad [2, 3]^* = \{0\} \cup [2, 3] \cup [4, \infty)$$

## 3. The Kleene algebra of sets of real numbers

The powerset of the nonnegative numbers $\mathcal{P}(\mathbb{R}_{\geq 0})$ is naturally endowed with an operation of "concatenation": it is addition extended over sets:

$$X + Y = \{x + y \mid x \in X, y \in Y\} \text{ for all } X, Y \subseteq \mathbb{R}$$

whose unit is $\mathbf{0} = \{0\}$.

Moreover we can define a star operation via the usual least fixpoint construction

$$X^* = \bigcup_{n \in \mathbb{N}} nX$$

where the multiples of $X$ are defined as usual: $0X = \mathbf{0}$ and $(n+1)X = nX + X$.

The following theorem can be easily verified:

**Theorem 3.1** *The structure* $\mathcal{P}(\mathbb{R}_{\geq 0}) = \big(\mathcal{P}(\mathbb{R}_{\geq 0}), \cup, +, (\cdot)^*, \emptyset, \mathbf{0}\big)$ *is a* ***commutative Kleene algebra****, i.e.,* $(\mathcal{P}(\mathbb{R}_{\geq 0}), \cup, \emptyset)$ *is an idempotent monoid,* $(\mathcal{P}(\mathbb{R}_{\geq 0}), +, \mathbf{0})$ *is a commutative monoid,* $+$ *distributes over* $\cup$ *and* $(\cdot)^*$ *satisfies the following equations [Con71, Koz94]:*

$$X + Y \subseteq Y \Rightarrow X^* + Y \subseteq Y \tag{2}$$

$$\mathbf{0} \cup (X + X^*) \subseteq X^* \tag{3}$$

$$X^* + Y^* = (X+Y)^* + (X^* \cup Y^*) \tag{4}$$

Because a complement operation is available, $\neg X = \mathbb{R}_{\geq 0} \setminus X$, we actually get a **commutative complemented Kleene algebra**, i.e., a boolean algebra which is also a commutative Kleene algebra.

Note also that summation with singletons distributes over intersection:

$$\{x\} + \big(Y \cap Z\big) = \big(\{x\} + Y\big) \cap \big(\{x\} + Z\big) \tag{5}$$

but distributivity of summation over intersection is not valid in general as the following example shows:

$$\big((2,3) + (4,5)\big) \cap \big((2,3) + (5,6)\big) = (7,8)$$

$$(2,3) + \big((4,5) \cap (5,6)\big) = \emptyset$$

### *3.1. Normal forms*

Denote $\mathcal{K}(Int)$ the sub-(commutative complemented Kleene) algebra generated by $Int$ in $\mathcal{P}(\mathbb{R}_{\geq 0})$, that is the family of sets which can be obtained from intervals of $Int$ by applying union, summation, star and complementation.

**Definition 3.2** *A set* $X \in \mathcal{K}(Int)$ *can be* ***written in normal form*** *iff there exist finite unions of rational intervals* $X_1, X_2$ *and some naturals* $k \in \mathbb{Q}_{\geq 0}$, $N \in \mathbb{N}$ *such that*

$$X = X_1 \cup (X_2 + \{k\}^*) \tag{6}$$

$$\text{and} \quad X_1 \subseteq [0, Nk), X_2 \subseteq [Nk, (N+1)k) \tag{7}$$

*We call* $N$ *the* ***bound*** *of the normal form.*

We will work with normal forms in which $X_1$ and $X_2$ are unions of *disjoint intervals*. It is straightforward how to transform some normal form such that this property holds.

Normal forms are not unique: for the normal form in the definition and some $p \in \mathbb{N}$, the following expression:

$$X = \Big(X_1 \cup \big(X_2 + \{0, k, 2k, \ldots, (p-1)k\}\big)\Big) \cup \big(X_2 + \{pk\} + \{k\}^*\big)$$

is a normal form too, but with bound $N+p$.

Any finite union of rational intervals $X$ can be put into normal form: when $X$ is bounded from above by $M$ then $X = X \cup \big(\emptyset + \{1\}^*\big)$ is a normal form with bound $\lceil M \rceil$. When $X$ is unbounded, suppose $X = X_1 \cup [M, \infty)$ is some decomposition of it into disjoint intervals where $M \notin \mathbb{N}$. Then

$$X = \Big(X_1 \cup \big[M, \lceil M \rceil\big)\Big) \cup \Big(\big[\lceil M \rceil, \lceil M \rceil + 1\big) + \{1\}^*\Big)$$

is a normal form with bound $\lceil M \rceil$.

**Proposition 3.3** *For each set $X$ written into normal form as $X = X_1 \cup \big(X_2 + \{k\}^*\big)$, $X = \emptyset$ iff both $X_1$ and $X_2$ are empty.*

This property, though trivial, has its own importance since we will use normal forms as time labels in automata and we still want to have a decidable emptiness problem for the resulting automata.

Sometimes, after the application of different operations to normal forms we might not be able to get directly a normal form; instead, we might get a **weak normal form**, which is a decomposition like equation 6 but without the additional requirement 7 on the existence of the bound. As an example we have the following:

- $X = \big((2,3) \cup (4,6)\big) \cup \Big(\big([6,7) \cup [8,9)\big) + \{3\}^*\Big)$ is written in normal form with bound 2 since we have $X_1 = (2,3) \cup (4,6) \subseteq [0,6)$, $X_2 = [6,7) \cup (8,9) \subseteq [6,9)$.
- $Y = \big((2,3) \cup (4,7)\big) \cup \big([5,7] + \{3\}^*\big)$ is a weak normal form which is not a normal form: there is no $N \in \mathbb{N}$ such that $[5,7] \subseteq \big[3N, 3(N+1)\big)$ and $(2,3) \cup (4,6) \subseteq [0, 3N)$.

However both expand to the same set:

$$X = Y = (2,3) \cup (4,7] \cup \bigcup_{n \geq 3} [3n-1, 3n+1]$$

**Lemma 3.4** *Weak normal forms can be transformed into normal forms.*

*Proof.* Throughout this proof, $\lbrace\!\!($ will denote any left parenthesis.

Take $X = X_1 \cup (X_2 + \{k\}^*)$ some weak normal form and define $M = \sup(\sup X_1, \sup X_2)$. Two cases arise:

1. $M = \infty$. This means that there exists some $L \in \mathbb{R}_{\geq 0}$ such that $(L, \infty) \subseteq X$. Define then $n = \left\lfloor \frac{L}{k} \right\rfloor$, that is $nk \leq L < (n+1)k$. Then for each $i \geq n+1$, $X_2 + \{ik\} \subseteq \big[(n+1)k, \infty\big) \subseteq (L, \infty)$. It follows that $X$ is a finite union of intervals
$$X = X_1 \cup \left(\bigcup_{i=0}^{n} \Big(X_2 + \{ik\}\Big)\right) \cup (L, \infty)$$
and thus we know how to transform it into normal form.

2. $M < \infty$. Define then $n = \left\lfloor \frac{M}{k} \right\rfloor + 1$, i.e., $(n-1)k \leq M < nk$ and further

$$Z_1 = X_1 \cup \left[ \left( \bigcup_{i=0}^{n-1} \big( X_2 + \{ik\} \big) \right) \cap [0, nk) \right]$$

$$Z_2 = \left( \bigcup_{i=1}^{n} \big( X_2 + \{ik\} \big) \right) \cap \big[nk, (n+1)k\big)$$

We claim that $X = Z_1 \cup \big(Z_2 + \{k\}^*\big)$ which is a normal form with bound $n$. To prove this, observe first that for each $i > j$, $i, j \in \mathbb{N}$,

$$\big(X_2 + \{ik\}\big) \cap [0, jk) = \emptyset$$

Moreover, for each $j \in \mathbb{N}$, using distributivity of summation of singletons over intersection (property 5) we get

$$\big(X_2 + \{jk\}\big) \cap \big[(n+j)k, \infty\big) = \big(X_2 \cap [nk, \infty)\big) + \{jk\} = \emptyset$$

due to the fact that $X_2 \subseteq [0, M] \subset [0, nk)$. This also implies that, for each $i \leq j$, $i, j \in \mathbb{N}$,

$$\big(X_2 + \{ik\}\big) \cap \big[(n+j)k, \infty\big) = \emptyset$$

Therefore, by the same distributivity property 5 we get

$$\begin{aligned}
&\big(X_2 + \{k\}^*\big) \cap \big[(n+j)k, (n+j+1)k\big) = \\
&\qquad = \left( \bigcup_{i=j+1}^{n+j} \big(X_2 + \{ik\}\big) \right) \cap [(n+j)k, (n+j+1)k) = \\
&\qquad = \left( \bigcup_{i=1}^{n} \big(X_2 + \{ik\}\big) \right) \cap [nk, (n+1)k) + \{jk\}
\end{aligned}$$

and further

$$\begin{aligned}
X &= X_1 \cup \big(X_2 + \{k\}^*\big) \\
&= X_1 \cup \left( \big(X_2 + \{k\}^*\big) \cap \left( [0, nk) \cup \bigcup_{j \geq 0} [(n+j)k, (n+j+1)k) \right) \right) \\
&= X_1 \cup \left( \bigcup_{i=0}^{n-1} \Big(X_2 + \{ik\}\Big) \cap [0, nk) \right) \cup \\
&\qquad \bigcup_{j \geq 0} \left( \left( \bigcup_{i=1}^{n} \Big(X_2 + \{ik\}\Big) \cap [nk, (n+1)k) \right) + \{jk\} \right) \\
&= Z_1 \cup \big(Z_2 + \{k\}^*\big)
\end{aligned}$$

□

*3.2. A normal form theorem*

The key result for normal forms is the following:

**Theorem 3.5** *Each $X \in \mathcal{K}(Int)$ can be written in normal form.*

*Proof.* We must show that the result of any operation applied to some normal forms can be put into normal form. We first list some useful identities valid in $\mathcal{P}(\mathbb{R})$ [Con71]:

$$X^{**} = X^* \tag{8}$$

$$(X \cup Y)^* = X^* + Y^* \tag{9}$$

$$(X^* + Y)^* = \{0\} \cup (X^* + Y^* + Y) \tag{10}$$

We employ the notations $lcm(p, q)$ and $gcd(p, q)$ where $p, q \in \mathbb{Q}_{\geq 0}$ as the generalization of *lcm* and *gcd* from integers. The formal definitions are:

$$lcm(p, q) = \min\{r \in \mathbb{Q}_{\geq 0} \mid \exists l, m \in \mathbb{N} \text{ such that } lp = r = mq\}$$

$$gcd(p, q) = \frac{pq}{lcm(p, q)}$$

We also use the following *ultimately periodicity property*:

> Given $n$ distinct positive rationals $a_i \in \mathbb{Q}_{\geq 0}$ we have that $\{a_1, \dots, a_n\}^*$ is *ultimately periodic*, i.e., there exist some *finite* set of rationals $B$ and some rationals $q, r \in \mathbb{Q}_{\geq 0}$ such that
>
> $$\{a_1, \dots, a_n\}^* = B \cup \big(\{q\} + \{r\}^*\big) \tag{11}$$

This property can be seen as an equivalent form of the normal form theorem for rational languages over a one letter alphabet. For a direct proof note first that, given two rationals $p, q \in \mathbb{Q}_{\geq 0}$,

$$\{p, q\}^* = B \cup \Big(\big\{lcm(p, q)\big\} + \big\{\,\mathrm{gcd}(p, q)\big\}^*\Big)$$

where $B = \big\{\alpha \in \mathbb{Q}_{\geq 0} \mid \alpha < lcm(p, q), \alpha = lp + mq, l, m \in \mathbb{N}\big\} = \{p, q\}^* \cap \big[0, lcm(p, q)\big)$. The property 11 follows from this by induction upon the number of elements in the starred set.

Fix now two normal forms $X = X_1 \cup (X_2 + \{k\}^*)$ with bound $M$ and $Y = Y_1 \cup (Y_2 + \{l\}^*)$ with bound $N$ and denote $m = lcm(k, l)$. We then get the following form for $X \cup Y$:

$$X_1 \cup Y_1 \cup \left(\left(\bigcup_{i=0}^{m/k-1} \Big(X_2 + \{ik\}\Big) \cup \bigcup_{i=0}^{m/l-1} \Big(Y_2 + \{il\}\Big)\right) + \{m\}^*\right)$$

This is a weak normal form and Lemma 3.4 shows how to transform it into normal form.

For $X + Y$ distributivity of $+$ over $\cup$ transforms it into:

$$\big(X_1 + Y_1\big) \cup \big(X_1 + Y_2 + \{l\}^*\big) \cup \big(X_2 + Y_1 + \{k\}^*\big) \cup \big(X_2 + Y_2 + \{k\}^* + \{l\}^*\big)$$

An instantiation of identity 9 gives $\{k\}^* + \{l\}^* = \{k,l\}^*$. The ultimately periodicity property 11 gives a normal form for this set and thence we have above a union of weak normal forms which we already know how to bring to normal form.

For $X^*$ we have two cases. The first one occurs when one of $X_1$ and $X_2$ contains a nonpoint interval. Then *the set $X^*$ is a finite union of rational intervals*, so it can be easily brought into normal form. To prove this claim, note that for each nonpoint interval, e.g., $(a,b]$ (that is $b-a>0$), denoting $m_0 = \left\lceil \frac{a}{b-a} \right\rceil$, we have that $(a,b]^* = \mathbf{0} \cup \bigcup_{i=1}^{m_0-1} (ia, ib] \cup (m_0 a, \infty)$ since the choice of $m_0$ assures that $(m_0+1)a < m_0 b$. Hence from the $m_0$-th iteration the intervals start to overlap. This observation can be easily adapted to prove our claim.

The second case is when both $X_1$ and $X_2$ consist of point intervals. Applying identity 9 we get $X^* = X_1^* + \left(X_2 + \{k\}^*\right)^*$. Then by the ultimately periodicity property 11 $X_1$ can be written into normal form, so we may concentrate on $\left(X_2 + \{k\}^*\right)^*$.

By identities 10 and 9 we further get

$$\left(X_2 + \{k\}^*\right)^* = \{0\} \cup \left(X_2 + X_2^* + \{k\}^*\right) \; = \; \{0\} \cup \Big(X_2 + \left(X_2 \cup \{k\}\right)^*\Big)$$

Finally the ultimately periodicity property 11 tells us that $\left(X_2 \cup \{k\}\right)^*$ can be put into normal form and therefore this case reduces to a summation of normal forms.

For $\neg X$ the strength of the normal form, i.e., the additional requirement on the existence of the bound $N$ helps us giving directly the result: it is

$$\neg X = \Big(\neg X_1 \cap [0, Nk)\Big) \cup \Big(\left(\neg X_2 \cap [Nk, (N+1)k)\right) + \{k\}^*\Big)$$

and the bound of this normal form is $N$ too. $\square$

In [CG98] it is proved that the set of finite unions of $n$-dimensional normal forms in which $X_1 = \emptyset$ forms a boolean algebra. The essential novelty in our result is the closure of 1-dimensional normal forms under star.

Though the theorem is based on the same technique that gives the normal form of rational languages over a one-letter alphabet it cannot be a simple corollary of that. Even if we restrict attention to the algebra generated by intervals with natural bounds, denote it $\mathbb{N}Int$, we find *two* generators: the point set $\{1\}$ and the nonpoint interval $(0,1)$. Neither of them may generate the other: $\{1\}$ generates just sets with isolated points or complements of such sets (i.e., countable or co-countable sets), while $(0,1)$ generates just finite unions of intervals (it cannot generate $(0,1) + \{1\}^*$).

One might also think that the result follows from Eilenberg's theory of automata with multiplicities [Ei74]. But this is not the case either since in that work star is defined via some formal power series and one cannot prove, unless defining some suitable equivalence on power series, that e.g. $[0,1)^* = [0,\infty)$.

Finally note the interesting relation which holds between the two generators of $\mathbb{N}Int$, showing they are not independent:

$$(0,1)^* = (\mathbf{0} \cup (0,1)) + \{1\}^* \tag{12}$$

*3.3. Matrices of normal forms*

At the end of this section we make a brief excursion into matrix theory. We construct, as in [Koz94] the Kleene algebra of $n \times n$ matrices over $\mathcal{P}(\mathbb{R}_{\geq 0})$ whose operations are based upon the operations in the Kleene algebra $\mathcal{P}(\mathbb{R}_{\geq 0})$:

$$(A \cup B)_{ij} = A_{ij} \cup B_{ij} \qquad (A + B)_{ij} = \bigcup_{k=1}^{n} (A_{ik} + B_{kj})$$

$$A^* = \bigcup_{n \in \mathbb{N}} nA$$

where $0A = I_n$ and $(n+1)A = nA + A$, $I_n$ denoting the unit for matrix summation.

If we write in detail the components of $A^*$ we have:

$$(A^*)_{ij} = \bigcup \big\{ A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j} \mid i_1, \ldots, i_m \in [n], m \in \mathbb{N} \big\} \cup \{0 \mid i = j\} \tag{13}$$

The star of a matrix $A$ can be computed by the following well-known Floyd-Warshall-Kleene algorithm [Con71, Ei74]: we recursively define a sequence of $n+1$ matrices $A(k)$ $(0 \leq k \leq n)$ with

$$\begin{aligned} A(0) &= A \cup I_n \\ A(k)_{ij} &= A(k-1)_{ij} \cup \Big( A(k-1)_{ik} + \big(A(k-1)_{kk}\big)^* + A(k-1)_{kj} \Big) \end{aligned} \tag{14}$$

**Proposition 3.6 ([Ei74])** *$A(n) = A^*$ for any matrix over $\mathcal{P}(\mathbb{R}_{\geq 0})$.*

The classical proof may run as follows: one proves first that $(nA)_{ik} + (mA)_{kj} \subseteq ((n+m)A)_{ij}$. This implies that $(A^*)_{ik} + \big((A^*)_{kk}\big)^* + (A^*)_{kj} \subseteq (A^*)_{ij}$. Then one shows that $A(k)_{ij} \subseteq (A^*)_{ij}$ by induction on $k$ and hence get the left-to-right inclusion.

The right-to-left inclusion follows by proving that

$$A(k)_{ij} = \bigcup \big\{ A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j} \mid i_1, \ldots, i_m \in [k], m \in \mathbb{N} \big\} \cup \{0 \mid i = j\}$$

by induction on $k$.

**Corollary 3.7** *If $A$ is a matrix of normal forms then $A^*$ can be transformed into a matrix of normal forms too.*

**Corollary 3.8** *For each matrix of normal forms $A$ if for all indices $i \neq j$ we have that $0 \notin A_{ij}$ then for all indices $i \neq j$, $0 \notin (A^*)_{ij}$.*

*Proof.* This is a corollary of relation 13: for any $i_1, \ldots, i_m \in [n]$ consider the sum $A_{i\,i_1} + A_{i_1 i_2} + \ldots + A_{i_m j}$. As we assumed $i \neq j$ we must have some $p \in [m]$ such that $i_p \neq i_{p+1}$. Thence $0 \notin A_{i_p i_{p+1}}$ and therefore the sum itself does not contain 0. $\square$

Note however that for any $i$, $(A^*)_{ii}$ will always contain 0.

**4. Determinization and complementation of RTA**

The above theory suggests that "periodic" constraints may replace intervals in the definition of RTA.

**Definition 4.1** *An **augmented real-time automaton** is a tuple* $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, Q_0, Q_f)$ *where* $Q$, $\Sigma$, $Q_0$, $Q_f$, $\delta$ *and* $\lambda$ *are the same as in RTA while* $\iota : Q \longrightarrow \mathcal{K}(Int)$ *(actually* $\iota$ *gives a normal form).*

Augmented RTA work similarly to RTA: *runs* have the same definition and a signal $\sigma$ with $dom(\sigma) = [0, e)$ is *associated with* a run of length $n$ iff there exist $0 = e_0 \leq \ldots \leq e_n = e$ with $e_i - e_{i-1} \in \iota(q_i)$ and $\sigma(t) = \lambda(q_i)$ for all $t \in [e_{i-1}, e_i)$ and all $i \in [n]$. The emptiness problem is again decidable in linear time w.r.t. $card(Q)$. Note that we need a preliminary step in which states $q$ whose interval label *denotes the empty set* are removed. It is here where we need Proposition 3.3.

The different notions of determinism remain unchanged for augmented RTA; hence we will speak of state-deterministic augmented RTA and stuttering-free augmented RTA in the sense of Definition 2.9.

We also have a transition-labeled version of augmented RTA, called in the sequel **augmented t-RTA**, which are tuples $\mathcal{B} = (Q, \Sigma, \delta, Q_0, Q_f)$ like t-RTA, the difference being that the transition relation is time-labeled with normal forms instead of just intervals: $\delta \subseteq Q \times \Sigma \times \mathcal{K}(Int) \times Q$. The different notions of determinism in Definition 2.10 are the same for augmented t-RTA, the translations between RTA and t-RTA and back from subsection 2.2 work with augmented automata too and Proposition 2.11 is valid for augmented automata.

The following theorem says that we do not increase the expressive power of RTA if we use normal forms instead of mere intervals:

**Theorem 4.2** $TReg(\Sigma)$ *equals the class of languages accepted by augmented RTA.*

The proof is very close to the one of Theorem 2.6 and is based on the following property of rational expressions:

$$\|[a]_{(a,b)\cup((c,d)+\{k\}^*)}\| = \|[a]_{(a,b)} + [a]_{(c,d)}; [a]^*_{\{k\}}\|$$

Of course, we also have to redefine rational expressions allowing normal forms as indices for atoms.

The first step in determinization is the achievement of stuttering-freeness and the proof runs smoother for augmented t-RTA:

**Theorem 4.3** *Each augmented t-RTA is equivalent to some stuttering-free augmented t-RTA.*

*Proof.* As a preliminary step, in the given augmented t-RTA we remove zeroes from the time labels by applying Proposition 2.8, slightly modified for handling normal forms instead of mere intervals. We also assume that all transitions with empty time label have been removed.

We achieve stuttering freeness by removing all stuttering $a$-transitions for some $a \in \Sigma$, and then repeating this for all the other letters in $\Sigma$. The idea is to find, for each pair of states $(q_1, q_2)$ the set of positive numbers which are the duration of a signal that is associated with some run starting in $q_1$, ending in $q_2$ and containing only $a$-transitions. For this we need to recursively add all the intervals of the transitions that may lie on such a run. This is the place where we apply the normal form theorem 3.5 and the algorithm for computing the star of a matrix of sets of positive numbers. The formalization is the following:

Start with some augmented t-RTA $\mathcal{A} = (Q, \Sigma, \delta, q_0, Q_f)$ and number its states as $Q = \{q_1, \ldots, q_p\}$. Construct a matrix $A$ whose elements are the interval labels of the $a$-labeled states:

$$A_{ij} = \begin{cases} X & \text{iff } (q_i, a, X, q_j) \in \delta \\ \emptyset & \text{otherwise} \end{cases}$$

Then $(A^*)_{ij}$ consists of the lengths of signals associated with runs starting in $q_i$, ending in $q_j$ and consisting of $a$-transitions only. More formally,

$$\begin{aligned}(A^*)_{jk} = \bigcup \big\{ X_1 + \ldots + X_m \mid (r_{i-1}, a, X_i, r_i)_{i \in [n]} \text{ is a run in } \mathcal{A} \text{ and} \\ r_0 = q_j, r_n = q_k \big\} \cup \{0 \mid j = k\} \end{aligned} \tag{15}$$

This fact is a corollary of identity 13.

Computation of $A^*$ is done by the Floyd-Warshall-Kleene algorithm (14). Note here the importance of Corollary 3.7: the elements of $A^*$ are still normal forms, hence they may be used for labeling some new transitions of an augmented RTA. Hence, while non-$a$-transitions will be preserved, the *nonempty* components of $A^*$ will replace all $a$-transitions: their time label will be $(A^*)_{ij}$ and they will be connected only to states from which no other $a$-transition is issued.

Formally, consider a disjoint copy of $Q$, $Q' = \{q'_i \mid q_i \in Q\}$; the primed states will be reached exactly after an $a$-transition. Build then $\mathcal{B} = (Q \cup Q', \Sigma, \overline{\delta}, Q_0, Q_f \cup Q'_f)$ where $Q'_f$ is the set of copies of final states and

$$\begin{aligned}\overline{\delta} = \big\{ (q, b, X, r), (q', b, X, r) \mid b \neq a, (q, b, X, r) \in \delta \big\} \cup \\ \Big\{ \big(q_i, a, (A^*)_{ij} \setminus \{0\}, q'_j\big) \mid \big(A^*\big)_{ij} \setminus \{0\} \neq \emptyset \Big\}\end{aligned}$$

The need for removing zero from the new transitions comes from the fact that we do not want to add stuttering steps involving the other symbols from $\Sigma$.

The equivalence of $\mathcal{A}$ and $\mathcal{B}$ follows from the observation that a run $\big((r_{i-1}, a_i, X_i, r_i)\big)_{i \in [n]}$ in $\mathcal{A}$ associated with some signal $\sigma$, can be transformed into a run in $\mathcal{B}$ for $\sigma$ by replacing all maximal sequences of $a$-transitions with the appropriate $a$-transition time-labeled from $A^*$ and by priming the state that follows this transition.

Observe that by construction no two $a$-transitions are directly connected. On the other hand, all non-$a$-transitions involving nonprimed states are just copied, hence no stuttering transitions are added on these states. Finally, the primed states are not involved in any stuttering transitions since they are targets of $a$-transitions and sources of non-$a$-transitions. This shows that by recursively applying this construction for all letters in $\Sigma$ we end with a stuttering-free augmented RTA. $\square$

The number of states of the final t-RTA is $2^{card(\Sigma)} \cdot card(Q)$, since at each step the states are at most duplicated. Concerning the number of transitions, note that, for each $a \in \Sigma$, at each step the number of $a$-transitions is either doubled (if $a$ is not chosen at that moment for stuttering elimination) or squared. Since there is a single step in which the number of $a$-transitions is squared, an upper bound for the number of $a$-transitions would be $2^{2 \cdot card(\Sigma)} \cdot m_a^2$, where $m_a$ is the initial number of $a$-transitions. Note that the earlier we choose to eliminate the stuttering $a$-transitions, the smaller the number of $a$-transitions we obtain. This is because squaring would apply to a smaller number of transitions. Of course the size of the resulting automaton would be smaller if we mix the above procedure with the procedure that removes inaccessible states.

The last step in the determinization process is the achievement of determinism in stuttering-free automata. This time, the construction works smoother for state-labeled automata:

**Theorem 4.4** *Each stuttering-free augmented RTA is equivalent to some deterministic augmented RTA.*

*Proof.* Note that, as we work with state-labeled RTA, the given stuttering-free RTA has the special initial state $q_\epsilon$ whose time-label is $[0,0]$ and which is not connected to any other state.

Start with a stuttering-free augmented RTA $\mathcal{B} = \big(Q \cup \{q_\epsilon\}, \Sigma, \delta, \lambda, \iota, Q_0 \cup \{q_\epsilon\}, Q_f\big)$ with $q_\epsilon \notin Q$. For some subset of states $S \subseteq Q$ we write $\lambda(S) = a$ as a shortcut for saying that all states in $S$ are identically labeled with $a$. If $\mathcal{B}$ were untimed, the states of the deterministic automaton would have been identically state-labeled subsets of $Q$ and we would draw a transition from some $S_1$ with $\lambda(S_1) = a$ to some $S_2$ with $\lambda(S_2) = b$ iff $S_2 = \{r \in Q \mid \exists q \in S_1 \text{ s.t. } (q,r) \in \delta\}$. Taking into account the time labels is done by splitting $S_2$ into several "smaller" sets such that the time labels of these give a partition of $\mathbb{R}_{>0}$.

To each $U \subseteq Q$ with $\lambda(U) = a$ we associate the set of time labels appearing in $U$:

$$Tl(U) = \{X \in \mathcal{K}(Int) \mid \exists q \in U \text{ s.t. } \iota(q) = X\}$$

Let $\mathcal{R}$ denote the set of triples $[S, S', a]$ where $a \in \Sigma$ and $S' \subseteq S \subseteq Q$ with $\lambda(S) = a$. Define then $\overline{\lambda}\big([S, S', a]\big) = a$ and

$$\overline{\iota}\big([S, S', a]\big) = \mathbb{R}_{>0} \cap \Big(\bigcap Tl(S')\Big) \cap \neg \Big(\bigcup Tl(S \setminus S')\Big)$$

where the usual conventions $\bigcap \emptyset = \mathbb{R}_{\geq 0}$ and $\bigcup \emptyset = \emptyset$ apply. Intuitively the control passes through $[S, S', a]$ iff in $\mathcal{B}$ the control may pass through some state in $S'$ but not through any of the states in $S \setminus S'$. We put $\mathbb{R}_{>0}$ in front of $\overline{\iota}([S, S', a])$ because otherwise we would lose stuttering-freeness. Also note that it is *here* where we need the result that normal forms are closed under complementation, because we need to put $\overline{\iota}([S, S', a])$ into normal form and $\overline{\iota}([S, S', a])$ contains complementation.

Hence we build $\mathcal{C} = (\mathcal{R} \cup \{q_\epsilon\}, \Sigma, \tilde{\delta}, \tilde{\lambda}, \tilde{\iota}, \mathcal{R}_0, \mathcal{R}_f)$ in which

- $\tilde{\delta}$ consists of transitions going from each $[S, S', a] \in \mathcal{R}$ to each tuple $[U, U', b]$ defined by

  $$U = \big\{q \in Q \mid \exists r \in S' \text{ s.t. } (r, q) \in \delta \text{ and } \lambda(q) = b\big\} \quad \text{and} \quad U' \subseteq U.$$

  Case $U' = \emptyset$ stands for the situation when the length of the current state in the signal is not in any of the sets from $Tl(U)$. Note how states $[\emptyset, \emptyset, a]$ time-labeled with $\mathbb{R}_{>0}$ play the role of the trap states in finite automata.
- initial and final states are

  $$\mathcal{R}_0 = \Big\{[S, S', a] \in \overline{Q} \mid S = \big\{q \in Q_0 \mid \lambda(q) = a\big\}\Big\} \cup \{q_\epsilon\}$$
  $$\mathcal{R}_f = \big\{[S, S', a] \in \overline{Q} \mid S' \cap Q_f \neq \emptyset\big\} \cup \big\{q_\epsilon \mid \sigma_\epsilon \in L(\mathcal{B})\big\}$$

The proof that $\mathcal{C}$ is equivalent to $\mathcal{B}$ proceeds by induction on the number of discontinuities in a signal. The construction assures that, at each discontinuity, exactly one state can be chosen such that the control goes to that state. □

The complexity of this construction is exponential in the number of states: by denoting $n = card(Q)$, observe first that the number of states $[S, S', a]$ where $card(S) = k$ is at most $2^k \cdot \begin{pmatrix} n \\ k \end{pmatrix}$ (at most due to the fact that some sets of states $S$ might not be consistently state-labeled). Therefore the cardinality of $\mathcal{R}$ is at most

$$\sum_{k=0}^{n} \left(2^k \cdot \begin{pmatrix} n \\ k \end{pmatrix}\right) = 3^n$$

**Theorem 4.5** *$TRec(\Sigma)$ is closed under complementation. The universality problem for $TRec(\Sigma)$ is decidable.*

*Proof.* This is a corollary of Theorem 4.4 and Proposition 2.3. The important property provided by the construction of the deterministic augmented RTA $\mathcal{C}$ in this theorem is that each signal (including the empty signal!) *is associated with a unique run that starts in $T_0$*. Hence the augmented RTA that accepts $Sig(\Sigma) \setminus L(\mathcal{C})$ is obtained by complementing the set of final states of $\mathcal{C}$. □

Let us finally underline the need for theorem 3.5 in determinization: in our construction, we actually build an automaton whose time labels are in fact *extended rational expressions (i.e., using complementation) over intervals.* In the absence of theorem 3.5, such an automaton would not be an augmented RTA any longer and we would be in no position to decide whether, after complementing the set of final states, the resulting automaton would still be an augmented RTA. This would make questionable the decidability of the universality problem.

It is actually this problem what stops the application of the determinization construction for RTA whose time labels lie in a class larger than $Int$ in which comparison of the time bounds is effective - for example, the class of intervals whose bounds are algebraic numbers. If this class of intervals is chosen for the time labels, it is unclear whether the universality problem remains decidable.

## 5. The Pumping Lemma and expressivity issues

**Lemma 5.1 (Pumping Lemma)** *If a language $L$ is accepted by a RTA then there exists $N \in \mathbb{N}$ such that each signal $\sigma$ having at least $N$ discontinuities can be factored into three signals $\sigma = \sigma_1; \sigma_2; \sigma_3$, such that $\sigma_2$ contains at least one discontinuity and for any $n \in \mathbb{N}$ we have $\sigma_1; \sigma_2^n; \sigma_3 \in L$.*

*Proof.* The proof of this lemma is almost the same as in the untimed case, the difference lying in the reference to discontinuities. Take $\mathcal{A} = (Q, \Sigma, \delta, \lambda, \iota, Q_0, Q_f)$ a *stuttering-free* augmented RTA accepting $L$ and define $N = card(Q) + 1$. It is clear that each signal $\sigma \in L$ having $N$ discontinuities must be accepted by some run having exactly $N + 1$ states, hence one of the state must be repeated throughout the run. Since we assumed that $\mathcal{A}$ is stuttering-free we cannot have self loops at the repeated state. Hence the part of the run which can be repeated must contain at least two distinctly state-labeled states and therefore $\sigma_2$ must contain a discontinuity. □

**Proposition 5.2** *The language $L_{nonreg} = \{\sigma \in Sig(\{a, b\}) \mid endp(\sigma) \in [1, 3]\}$ is not real-time rational.*

*Proof.* Supposing $L_{nonreg}$ was real-time rational, we may pick up a signal $\sigma \in L_{nonreg}$ such that its number of discontinuities is more than the natural number $N$ provided by the Pumping Lemma. An example is the signal for which, for each $k \in [N]$, $\sigma(t) = a$ for $t \in \left[\frac{2k-2}{N}, \frac{2k-1}{N}\right)$ and $\sigma(t) = b$ for $t \in \left[\frac{2k-1}{N}, \frac{2k}{N}\right)$.

Then by the Pumping Lemma $\sigma$ can be factored as $\sigma = \sigma_1; \sigma_2; \sigma_3$ such that $\sigma_2$ has at least a discontinuity, (and hence $endp(\sigma_2) > 0$) and $\sigma_1; \sigma_2^n; \sigma_3 \in L_{nonreg}$ for any $n \in \mathbb{N}$. But then $n \cdot endp(\sigma_2) \leq 3$ for all $n \in \mathbb{N}$, which is in obvious contradiction with $endp(\sigma_2) > 0$. □

It is easy to build a state-labeled timed automaton [ACM97] with a single clock accepting $L_{nonreg}$. Note also that the untiming of this language is a rational (untimed) language. Here by the untiming of a real-time language $L$ we mean the set of words $w$ for which there exist a signal $\sigma \in L$ such that $w$ represents the sequence of symbols that appear in $\sigma$.

## 6. Conclusions and further work

We have presented here a Kleene algebra of sets of positive numbers where elements have a finite representation and a class of real-time languages which is closed under complementation and is defined by some automata and by rational expressions.

One question to be asked is whether the results concerning normal forms can be applied to timed automata. Recently Bouyer and Petit [BP99] have proved a Kleene theorem for timed automata that does not require renaming as in [ACM97]. It is possible that this theorem, correlated with our ultimately periodicity result, might allow removing loops with silent transitions by induction on the structure of the automaton (see also [BDGP98]). Of course, timed automata would need to use periodic constraints instead of intervals.

Another question is whether a normal form can be found for sets of intervals which have *any* bounds (not necessarily rational) and are generated from finite sets of intervals by union, complementation, concatenation and star. This question can also be restated as follows: is emptiness decidable for expressions denoting sets of positive numbers which are constructed by applying boolean operations, summation and star to intervals whose bounds are in some computable subset of $\mathbb{R}$? Note that this does not follow from the decidability of the emptiness problem for extended rational expressions where concatenation is replaced by shuffle since $\mathcal{P}(\mathbb{R}_{\geq 0})$ is not a free commutative Kleene algebra. Neither we mean that our normal form result can be extended to intervals with any bounds, since it is clear that one may no longer employ the ultimately periodicity property 11.

## Acknowledgments

I acknowledge the high quality environment of UNU/IIST Macau; this research was initiated and mostly achieved during a fellowship offered by UNU/IIST. My thanks go to Xu Qiwen, Paritosh Pandya and Evgeni Asarin for their support in doing this research. I also thank the anonymous referees for their valuable reviews that helped improve the paper.

## References

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

[AFH94] R. Alur, L. Fix and T.A. Henzinger. A determinizable class of timed automata, in *Proceedings of CAV'94*, LNCS 818:1-13, Springer Verlag, 1994.

[ACM97] E. Asarin, P. Caspi and O. Maler, A Kleene theorem for timed automata, in *Proceedings of LICS'97*, p.160-171, 1997.

[BDGP98] B. Bérard, V. Diekert, P. Gastin and A. Petit. Characterization of the expressive power of silent transitions in timed automata, *Fundamenta Informaticæ*, 36:145-182, 1998.

[BP99] P. Bouyer and A. Petit, Decomposition and composition of timed automata, *Proceedings of ICALP'99*, LNCS series, 1999.

[CG98] Ch. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints, LIAFA report, 1998.

[Con71] J. H. Conway. *Regular Algebra and Finite Machines*, Chapman and Hall, 1971.

[DW96] Dang Van Hung and Wang Ji, On the design of Hybrid Control Systems Using Automata Models, in *Proceedings of FST&TCS'96*, LNCS 1180:156-167, 1996.

[DMP91] R. Dechter, I. Meiri and J. Pearl, Temporal constraint networks, *Artificial Intelligence*, 49:61-95, 1991.

[Di99a] C. Dima. Automata and regular expressions for real-time languages, *Proceedings of the AFL'99 workshop*, Vasszeczeny, Hungary, 9-13 august 1999.

[Di99b] C. Dima. Two Kleene theorems from event-clock automata, in *Proceedings of FCT'99*, LNCS 1684:215-225, 1999.

[Di00] C. Dima. Real-time automata and the Kleene algebra of sets of real numbers, in *Proceedings of STACS 2000*, LNCS 1770:279-289, 2000.

[Ei74] S. Eilenberg. *Automata, Languages, and Machines, Vol. A*, Academic Press, 1974.

[HZ97] Michael R. Hansen and Zhou Chaochen. Duration Calculus: Logical Foundations, in *Formal Aspects of Computing*, 3:1-49, 1997.

[HRS98] T.A. Henzinger, J.-F. Raskin and P.-Y. Schobbens. The regular real-time languages, in *Proceedings of ICALP'98*, LNCS series, Springer Verlag, 1998.

[Her99] P. Herrmann. Renaming is necessary in timed regular expressions, in *Proceedings of FST&TCS'99*, LNCS series, 1999.

[HU] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley/Narosa Publishing House, eighth edition, New Delhi, 1992.

[Koz94] D. Kozen. A completeness theorem for Kleene algebras and the algebra of regular events, *Information and Computation*, 110:366-390, 1994.

[Ras99] J.F. Raskin. *Logics, Automata and Classical Theories for Deciding Real-Time*, PhD Thesis, Facultés Universitaires Notre Dame de la Paix, Namur, 1999.

[Tho97] W. Thomas. Automata on infinite objects, in J. van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, vol. B, 133-191, Elsevier, Amsterdam, 1997.

[Wil94] T. Wilke. Specifying timed state sequences in powerful decidable logics and timed automata. In *Proceedings of FTRTFT'94*, LNCS 863:694-715, Springer Verlag, 1994.