Computing Reachability Relations in Timed Automata

Cătălin Dima,
Verimag, 2 avenue de Vignate, 38610 Gières, France
ctdima@imag.fr

Abstract

We give an algorithmic calculus of the reachability relations on clock values defined by timed automata. Our approach is a modular one, by computing unions, compositions and reflexive-transitive closure (star) of "atomic" relations. The essential tool is a new representation technique for n-clock relations – the $2n$-automata – and our strategy is to show the closure under union, composition and star of the class of $2n$-automata that represent reachability relations in timed automata.

1 Introduction

Timed automata [1] are a successful and widely used extension of finite automata for modeling real-time systems [17, 13]. They are finite automata augmented with clocks that measure time passage. Clocks evolve synchronously at rate 1, and may have any real value. Transitions are taken when some simple arithmetic conditions on the clocks are met, and some transitions might reset some clocks to 0.

Several problems in the domain of verification of real-time systems translate to checking whether a set of states is reachable in a timed automaton. The possibility to reach a state r, starting from a state q, depends essentially on the clock values with which we start in state q. Moreover, even if r is reachable by starting from q with a given vector v of clock values, the set of clock values when reaching r is in general uncountable. The dependency between clock values when starting from q and clock values when reaching r, that is, the *reachability relation* defined by q and r, is therefore an important characteristics for the timed automaton. Its computation might help in the verification of some clock constraints which cannot be modeled with timed automata transitions [11]. The symbolic computation of such relations is the subject of this paper.

Our approach is to build the reachability relations by computing unions, compositions and reflexive-transitive closures of "atomic" relations. To this end, we introduce $2n$-*automata* as a new representation technique for n-clock relations. The idea is that each tuple of clock values in the given relation is coded by a *run* in the automaton. We then show that these automata are closed under union, composition and (for mild conditions) star and have a decidable emptiness problem. We also show that the mild conditions are necessary since in general star closure is not possible, and that the $2n$-automata arising from the relational semantics of timed automata satisfy these conditions.

The representation of clock relations by $2n$-automata passes through several intermediary steps in which we make extensive use of the *difference bound matrix* (DBM) representation [3] of "diagonal" constraints (i.e., constraints like $x - y \in \,]2, 5])$. In fact, our $2n$-automata represent *unions of DBMs*, and therefore give a new technique for consistency checking of disjunctions of diagonal constraints [14].

Expressing reachability relations in timed automata by means of Presburger Arithmetic is the subject of [5]. The technique employed in that paper is to construct the reflexive-transitive closure of the constraint graph associated to simple loops in timed automata. The problem is that constraint graphs cannot represent constraints employing disjunctions, whereas the reflexive-transitive closure is an infinite disjunction. Therefore, in [5] the authors need to "flatten" each timed automaton, such that no nested loops be allowed, and then "accelerate" each simple loop. On the contrary, our $2n$-automata can represent disjunctions and therefore we may iterate the reflexive-transitive closure construction. In other words, our construction allows the iteration of "sets of loops", not only of simple loops. Another result on expressing clock relations in pushdown timed automata, which utilizes a technique for representing dense clock constraints with discrete clocks, similar to our representation technique, can be found in [6].

The paper runs as follows: in the second section we remind the definition of timed automata (without actions) and give a relational semantics for them. In the third section we give several properties of DBMs and investigate on the possibility to model relational composition and reflexive-transitive closure by DBM relations. We emphasize several problems that such a modeling cause, and our solution is to decompose DBMs into *region matrices* and to utilize set-based operations on region matrices. The operations on region matrices are introduced and studied in the fourth section. The fifth section contains the definition and properties of n-automata This section is devoted only to representing sets of *point* region matrices. The sixth section provides the central result of this paper, our star-closure theorem for $2n$-automata whose language satisfy a so-called *non-elasticity* property. In the seventh section we show how to represent sets of *nonpoint* region matrices too – and hence DBMs – with n-automata. We also show here that the $2n$-automata arising from the relational semantics of timed automata satisfy the non-elasticity condition that assures star closure.

2 Timed automata

In this section we remind the definition of timed automata and their timed transition semantics [1]. We then show how to transform this semantics into a relational semantics, by associating to each transition a relation on the clock values before, resp. after taking the transition. This relational semantics uses *reset points* instead of clock values, for reasons to be explained later in the paper. Let us note that, as we are not interested in computing *behaviors* of timed automata, we do not endow them with actions or state labels.

First, we give several notations: the set of real nonnegative numbers is denoted $\mathbb{R}_{\geq 0}$ while the sets of integers, resp. nonnegative integers, are denoted $\mathbb{Z}$, resp. $\mathbb{N}$. For any finite set A, *card*(A) denotes the cardinality of A. The term *interval* denotes any interval whose extremities are integers, or ∞ or $-\infty$ and whose parentheses are either open or closed. For example, $[-1, \infty[= \{x \in \mathbb{R} \mid -1 \leq x < \infty\}$. The empty set is an interval too, and is represented in a unique way – simply as $\emptyset$. That is, we do not allow representations like $[3, 2[$.

We use the set-lifting of summation of reals, that is, for $A, B \subseteq \mathbb{R}$, $A + B = \{a + b \mid a, b \in \mathbb{R}\}$. Note that for A, B intervals, $A + B$ is an interval whose limits and parentheses can be easily computed from A and B. E.g., $[2, 3[+ [1, 4] = [3, 7[$. For completeness, we give also the straightforward rule for summing up with the empty interval: $I + \emptyset = \emptyset + I = \emptyset$, for any interval I.

A timed automaton is a finite automaton [10] endowed with the possibility to measure time passage by means of real-valued clocks. The automaton has a finite set of *states* and a finite set of *transitions*, which are "commands" for state changing. The automaton may rest in some state for a finite amount of time t, during which the clocks values are incremented with t. Then it may "move" to another state by taking one of the "enabled" transitions. For a transition to be enabled, the clock values must satisfy a certain constraint "just before" taking the transition. Taking a transition has two effects: the state changes and some clocks are reset to zero.

Formally, a **timed automaton** with clocks $\{x_1, \ldots, x_n\}$ is a tuple $\mathcal{A} = (Q, \delta)$ where Q is a finite set of *states* and δ is a finite set of tuples (*transitions*) (q, C, X, q') where $q, q' \in Q$, $X \subseteq \{1, \ldots, n\}$ and C is a conjunction of *clock constraints* of the following two forms:

1. $x_i \in I$, where $i \in \{1, n, \ldots,\}$ and $I \subseteq [0, \infty[$ is an interval.

2. $x_i - x_j \in J$, where $i, j \in \{1, \ldots, n\}$, $i \neq j$ and $J \subseteq]-\infty, \infty[$ is an interval. We will call such two-variable constraints *diagonal constraints*.

Each timed automaton can be represented as a labeled graph, with states represented as nodes and transitions as labeled edges. An example is provided in Figure 1, in which $Q = \{q_1, q_2, q_3, q_4\}$ and

$$\delta = \{(q_1, x_1 = 2, \{x_2\}, q_2), (q_2, x_2 \in [1, 2], \{x_2\}, q_3), (q_3, \mathsf{true}, \emptyset, q_2), (q_3, x_1 \in [4, 6] \wedge x_2 \in [1, 3], \emptyset, q_4)\}.$$

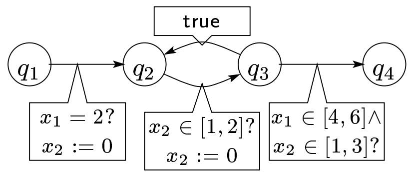


Figure 1. A timed automaton.

The standard way to give a semantics to a timed automaton $\mathcal{A}$ is to associate a *timed transition system* [17] to it, and to define the behaviors of the timed automaton as the runs in the timed transition system. The *configurations* of the timed transition system are tuples consisting of a state and n clock values. The *transitions* of the timed transition system are pairs of configurations of two kinds:

1. *Time passage transitions*, in which the state does not change but the clocks are incremented with the same value, and

2. *State change transitions*, in which the state changes, some clock values are reset to zero and some others remain unchanged.

The formal statement of these can be found in almost any paper on timed automata. We only give an example of behavior for our automaton in Figure 1:

$$(q_1, 1, 1) \to (q_1, 2, 2) \to (q_2, 2, 0) \to (q_2, 3.5, 1.5) \to (q_3, 3.5, 0) \to (q_3, 5, 1.5) \to (q_4, 5, 1.5). \quad (1)$$

Observe that, e.g., in configuration $(q_1, 2, 2)$ the transition $(q_1, x_1 = 2, \{x_2\}, q_2)$ is taken, which is possible since the clock values $x_1 = 2, x_2 = 2$ satisfy the clock constraint $x_1 = 2$. Formally, when an assignment of clock values $(x_1 := v_1, \ldots, x_n := v_n)$ satisfies a constraint C, we denote it by $(v_1, \ldots, v_n) \models C$.

In this paper we will focus on an **alternative semantics** of timed automata: the *reset points semantics*, first used in [4]. The idea is to utilize a new clock T which is never reset, and to record, at each moment t and for each clock x_i, the last moment before t when x_i was reset – denote this by r_i. Then the value of clock x_i at t equals the difference between the value of T and r_i. Further, time passage in a state amounts to incrementing T, while state change amounts to assigning the value of T to some of the r_i's.

Formally, the *reset transition system* associated to $\mathcal{A}$ is $\mathcal{T}(\mathcal{A}) = (Q \times \mathbb{R}^{n+1}_{\geq 0}, \theta)$ where $\mathbb{R}^{n+1}_{\geq 0}$ is the set of $(n+1)$-tuples of nonnegative reals and $\theta \subseteq Q \times \mathbb{R}^{n+1}_{\geq 0} \times Q \times \mathbb{R}^{n+1}_{\geq 0}$ is defined as follows:

$$\begin{aligned}\theta = \big\{(q, r_1, \ldots, r_n, t, q, r_1, \ldots, r_n, t') \mid t' \geq t,& \\ r_i \leq t \;\forall 1 \leq i \leq n\big\}& \\ \cup \big\{(q, r_1, \ldots, r_n, t, q', r'_1, \ldots, r'_n, t) \mid& \\ \exists (q, C, X, q') \in \delta \text{ such that if we put } v_i = t' - r_i& \\ \text{then } (v_1, \ldots, v_n) \models C \text{ and } \forall 1 \leq i \leq n, r_i \leq t,& \\ \text{and if } i \in X \text{ then } r'_i = t', \text{ if } i \notin X \text{ then } r'_i = r_i\big\}.&\end{aligned}$$

A *run* in $\mathcal{T}(\mathcal{A})$ is then a sequence of configurations connected by transitions in θ. The set of runs of $\mathcal{T}(\mathcal{A})$ gives the (reset points) semantics of $\mathcal{A}$. For our example in Figure 1, the following run in $\mathcal{T}(\mathcal{A})$ gives "the same" behavior as the run in 1 above, but in the reset time semantics:

$$\begin{aligned}(q_1, 0, 0, 1) &\to (q_2, 0, 0, 2) \to (q_2, 0, 2, 2) \to \\ &\to (q_2, 0, 2, 3.5) \to (q_3, 0, 3.5, 3.5) \to \\ &\to (q_3, 0, 3.5, 5) \to (q_4, 0, 3.5, 5).\end{aligned}$$

The relations that we want to compute in this paper are all the possible dependencies between initial and final reset values in a run that starts in some given state q and ends in a given state q', i.e., the relations $R_{qq'} \subseteq \mathbb{R}^{2n+2}_{\geq 0}$ defined by

$$\begin{aligned}R_{qq'} = \big\{(r_1, \ldots, r_n, t, r'_1, \ldots, r'_n, t') \mid \exists \text{ a run in } \mathcal{T}(\mathcal{A})& \\ \big(q^j, r^j_1, \ldots, r^j_n, t^j\big)_{j \in \{1, \ldots, k\}} \text{ s.t. } q^1 = q, q^k = q',& \\ t^1 = t, t^k = t' \text{ and } \forall i \in \{1, \ldots, n\}, r^1_i = r_i, r^k_i = r'_i\big\}.&\end{aligned}$$

Our aim is to give a characterization of the relations $R_{qq'}$ which can be computed in a modular manner – that is, which must be closed under union, composition and reflexive-transitive closure – and which must have a decidable emptiness problem. Here, by composition of relations we mean the following operation: given two relations $R, R' \subseteq A \times A$,

$$\begin{aligned}R \circ R' = \big\{(a, a'') \mid \exists a' \in A \text{ such that}& \\ (a, a') \in R \text{ and } (a', a'') \in R'\big\},&\end{aligned}$$

while the reflexive-transitive closure of R is

$$R^* = \bigcup_{n \geq 1} \underbrace{R \circ \ldots \circ R}_{n \text{ times}} \cup \big\{(a, a) \mid a \in A\big\}.$$

Let us show that $R_{qq'}$ can be represented in a modular way: first, we associate to each transition an "atomic" relation on reset points by "fusing" a time passage step with a state change step. Then, we recursively compute unions, compositions and reflexive-transitive closure of these atomic relations by a simple application of well-known algorithms for graphs – or, equivalently, by applying the Kleene theorem [10].

For our example in Figure 1, the atomic relation associated to the transition $\tau_{12} = (q_1, x_1 = 2, \{x_2\}, q_2)$ is

$$\begin{aligned}R_{\tau_{12}} = \big\{(r_1, r_2, t, r'_1, r'_2, t') \mid r_1, r_2, r'_1, r'_2, t, t' \in \mathbb{R}_{\geq 0},& \\ t' \geq t, r_1 \leq t, r_2 \leq t, t' - r_1 = 2, r'_1 = r_1, r'_2 = t'\big\}.&\end{aligned}$$

Observe that $(r_1, r_2, t, r'_1, r'_2, t') \in R_{\tau_{12}}$ iff in $\mathcal{T}(\mathcal{A})$ we have the run $(q_1, r_1, r_2, t) \to (q_1, r_1, r_2, t') \to (q_2, r'_1, r'_2, t')$.

Further in this example, if we denote by τ_{ij} the transition connecting state q_i with state q_j then

$$R_{q_1 q_4} = R_{\tau_{12}} \circ \big(R_{\tau_{23}} \circ R_{\tau_{32}}\big)^* \circ R_{\tau_{23}} \circ R_{\tau_{34}}.$$

In general, given a transition $\tau = (q, C, X, q')$ in a timed automaton $\mathcal{A}$, with $C = \bigwedge_{i \in \mathcal{I}} x_i \in U_i \wedge \bigwedge_{(i,j) \in \mathcal{J}} x_i - x_j \in U_{ij}$ for some $\mathcal{I} \subseteq \{1, \ldots, n\}$, $\mathcal{J} \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$, we associate the atomic relation $R_\tau \subseteq \mathbb{R}^{2n+2}_{\geq 0}$

$$\begin{aligned}R_\tau = \big\{(r_1 \ldots r_n, t, r'_1 \ldots r'_n, t') \mid t' \geq t \text{ and } \forall i \in \{1, \ldots, n\}& \\ r_i \leq t, \text{ if } i \in X \text{ then } r'_i = t', \text{ if } i \notin X \text{ then } r'_i = r_i,& \\ \text{and if we put } v_i = t' - r_i \text{ then } (v_1, \ldots, v_n) \models C\big\}.&\end{aligned}$$

This relation can be specified by the following formula (which can be rewritten as a formula of linear arithmetic over the reals):

$$\begin{aligned}f_\tau : x'_{n+1} \geq x_{n+1} \wedge \bigwedge_{i \in \mathcal{I}} x'_{n+1} - x_i \in U_i \wedge \bigwedge_{(i,j) \in \mathcal{J}} x_i - x_j \in U_{ij} \wedge& \\ \bigwedge_{i \in X} x'_i = x'_{n+1} \wedge \bigwedge_{i \notin X} x'_i = x_i \wedge \bigwedge_{i \in \{1, \ldots, n\}} x_i \leq x_{n+1}.& \qquad (2)\end{aligned}$$

Observe that this formula employs only diagonal constraints.

Composition of relations can be specified by using conjunction and quantification: given two transitions τ_1 and τ_2, specified by the formulas f_{τ_1}, resp. f_{τ_2}, the composition $R_{\tau_1} \circ R_{\tau_2}$ is specified by the formula

$$\begin{aligned}\exists x'_1 \ldots \exists x'_{n+1} . f_{\tau_1}(x_1, \ldots, x_{n+1}, x'_1, \ldots, x'_{n+1}) \wedge& \\ f_{\tau_2}(x'_1 \ldots, x'_{n+1}, x''_1, \ldots, x''_{n+1}).&\end{aligned}$$

In the resulting formula we may eliminate the existential quantifiers by arithmetic manipulations. The essential observation is that we will then obtain a *conjunction of diagonal constraints*! This would not be the case if we used the traditional clock valuation semantics – see [5, 7].

One of the problems of this approach is that the reflexive-transitive closure of a relation is not expressible in first-order logic [16]. We will therefore use a different formalism: we will code the formulas f_τ by *difference bound matrices* (DBMs) [3]. Our technique is to build a Kleene algebra [12] over the set of DBMs. The rest of the paper is dedicated to constructing operations on DBMs that model composition and reflexive-transitive closure. Remind that quantifier elimination in the formulas specifying compositions of clock relations yields conjunctions of diagonal constraints – this is the source of our idea of defining a composition operation on DBMs.

3 Difference bound matrices

Throughout this section we review some known facts about DBMs and emphasize some hurdles on the way to define a composition operation on DBMs. The solution to these problems is the subject of the next section.

Definition 1 *An n-**DBM** is an $n \times n$ matrix D of intervals.*

The set of n-DBMs is denoted Dbm_n.

DBMs represent sets of tuples of real numbers. The *semantics* of n-DBMs is the mapping $\|\cdot\| : \mathsf{Dbm}_n \to \mathbb{R}^n$ defined as follows: for all $D \in \mathsf{Dbm}_n$,

$$\|D\| \stackrel{def}{=} \big\{(a_1,\ldots,a_n) \in \mathbb{R}^n \mid \forall i,j \in \{1,\ldots,n\}, a_j - a_i \in D_{ij}\big\}.$$

An n-DBM $D \in \mathsf{Dbm}_n$ is said to be **in normal form** iff for each $i,j,k \in \{1,\ldots,n\}$, $D_{ii} = \{0\}$ and $D_{ik} \subseteq D_{ij} + D_{jk}$. Observe that, in an n-DBM D in normal form, $D_{ij} \neq \emptyset$ for all $i,j \in \{1,\ldots,n\}$.

Checking DBMs for emptiness is achieved with the following property:

Proposition 1 ([9, 7]) *1. For each $D \in \mathsf{Dbm}_n$, if $\|D\| \neq \emptyset$ then the following property holds:*

(*) For each sequence of indices $(i_j)_{j \in \{1,\ldots,k+1\}}$ with $i_j \in \{1,\ldots,n\}$ and $i_{k+1} = i_1$ we have that $0 \in \sum_{j=1}^{k} D_{i_j i_{j+1}}$.

2. For each DBM D with the property (*), there exists a DBM in normal form D' with $\|D\| = \|D'\|$.

3. Any n-DBM in normal form has a nonempty semantics.

Note also that an n-DBM in normal form is in some sense "minimal" w.r.t. the partial order '$\sqsubseteq$' $\subseteq \mathsf{Dbm}_n \times \mathsf{Dbm}_n$, defined by $D \sqsubseteq D'$ iff $D_{ij} \subseteq D'_{ij}$ for all $i,j \in \{1,\ldots,n\}$. That is, the set of all n-DBMs with the same non-empty semantics contains exactly one DBM in normal form, which is the least element of this set w.r.t. $\sqsubseteq$.

The basic operation which we need for DBMs is *composition* – which we will name here *composition*, in order not to produce ambiguities with the term "compositional". Remember that for formulas f_τ corresponding to a transition τ in a timed automaton, composition was implemented as conjunction followed by quantification. We therefore define an operation on DBMs which models conjunction, and an operation which models quantification. The first will be called *juxtaposition*, and the second *projection*, for reasons that will come up in their definition. We define these operations at the semantic level first, that is, on tuples of reals:

Definition 2 *Given an n-tuple $\overline{a} = (a_1,\ldots,a_n) \in \mathbb{R}^n_{\geq 0}$ and a set $X \subseteq \{1,\ldots,n\}$, with $X = \{i_1,\ldots,i_k\}$ where $i_j < i_{j+1}$ for all $j \in \{1,\ldots,k-1\}$, the X-**projection** of $\overline{a}$ is the tuple*

$$\overline{a}\big|_X \stackrel{def}{=} (a_{i_1},\ldots,a_{i_k}) \in \mathbb{R}^{card(X)}_{\geq 0}.$$

*Given $m,n,p \in \mathbb{N}$ with $p \leq \min(m,n)$, the p-**juxtaposition** of $\overline{a} = (a_1,\ldots,a_m) \in \mathbb{R}^m_{\geq 0}$ with $\overline{b} = (b_1,\ldots,b_n) \in \mathbb{R}^n_{\geq 0}$ is defined iff $a_{m-p+i} = b_i$ for all $i \in \{1,\ldots,p\}$ and is the $(m+n-p)$-tuple denoted $\overline{a} \mathbin{\square_p} \overline{b}$, whose i-th component is:*

$$\big(\overline{a} \mathbin{\square_p} \overline{b}\big)_i \stackrel{def}{=} \begin{cases} a_i & \text{for } i \in \{1,\ldots,m\} \\ b_{i-m+p} & \text{for } i \in \{m+1,\ldots,m+n-p\}. \end{cases}$$

*Given $\overline{a}, \overline{b} \in \mathbb{R}^{2n}_{\geq 0}$, the **composition** of $\overline{a}$ with $\overline{b}$ is the tuple denoted $\overline{a} \odot \overline{b}$ and obtained as follows:*

$$\overline{a} \odot \overline{b} \stackrel{def}{=} \big(\overline{a} \mathbin{\square_n} \overline{b}\big)\big|_{\{1,\ldots,n\} \cup \{2n+1,\ldots,3n\}} \in \mathbb{R}^{2n}_{\geq 0}.$$

It is trivial to check that juxtaposition is associative, that is, $(\overline{a} \mathbin{\square_q} \overline{b}) \mathbin{\square_r} \overline{c}$ is defined iff so is $\overline{a} \mathbin{\square_q} (\overline{b} \mathbin{\square_r} \overline{c})$, and then $(\overline{a} \mathbin{\square_q} \overline{b}) \mathbin{\square_r} \overline{c} = \overline{a} \mathbin{\square_q} (\overline{b} \mathbin{\square_r} \overline{c})$. *Composition* is then associative too (as a corollary of the associativity of juxtaposition), but does not have a unit.

The *powerset* of $2n$-tuples can then be endowed with an associative composition by lifting $\odot$ to sets: for $A, B \subseteq \mathbb{R}^{2n}_{\geq 0}$, $A \odot B = \big\{\overline{a} \odot \overline{b} \mid \overline{a} \in A, \overline{b} \in B\big\}$. The unit for set composition is the set

$$\mathbf{1}_{2n} \stackrel{def}{=} \big\{(\overline{a},\overline{a}) \mid \overline{a} \in \mathbb{R}^n_{\geq 0}\big\} = \big\{(a_1,\ldots,a_n,a_1,\ldots,a_n) \mid a_i \in \mathbb{R}_{\geq 0} \forall i \in \{1,\ldots,n\}\big\}.$$

Moreover, for each $L \subseteq \mathbb{R}^{2n}_{\geq 0}$, we may define the *star* of L as the set $L^{\circledast} = \bigcup_{k \in \mathbb{N}} L^{k\odot}$, where $L^{0\odot} = \mathbf{1}_{2n}$ and for all $k \in \mathbb{N}$, $L^{(k+1)\odot} = L^{k\odot} \odot L$.

Proposition 2 $\big(\mathcal{P}(\mathbb{R}^{2n}_{\geq 0}), \cup, \odot, \circledast, \emptyset, \mathbf{1}_{2n}\big)$ *is a Kleene algebra (in the sense of, e.g., [12]), that is, $\big(\mathcal{P}(\mathbb{R}^{2n}_{\geq 0}), \cup, \odot, \emptyset, \mathbf{1}_{2n}\big)$ is a semiring and $\circledast$ satisfies the following properties: for all $L_1, L_2 \subseteq \mathbb{R}^{2n}_{\geq 0}$,*

$$L_1 \odot L_2 \subseteq L_2 \Rightarrow L_1^{\circledast} \odot L_2 \subseteq L_2$$
$$L_2 \odot L_1 \subseteq L_2 \Rightarrow L_2 \odot L_1^{\circledast} \subseteq L_2$$
$$1_{2n} \cup \big(L_1 \odot L_1^{\circledast}\big) \subseteq L_1^{\circledast}$$
$$1_{2n} \cup \big(L_1^{\circledast} \odot L_1\big) \subseteq L_1^{\circledast}.$$

Our aim is to lift these operations to DBMs in a *compositional* manner – that is, to define operations $\big|_X$, $\Box_p$, $\odot$ and $\circledast$ on DBMs such that

$$\|D_1 \Box_p D_2\| = \|D_1\| \Box_p \|D_2\|, \quad \|D\big|_X\| = \|D\|\big|_X$$
$$\|D_1 \odot D_2\| = \|D_1\| \odot \|D_2\|, \quad \|D^{\circledast}\| = \|D\|^{\circledast}$$

Projection. The X-projection of an n-DBM D is, intuitively, the removal from D of the rows and columns whose indices are not in X:

Definition 3 *Given $D \in \mathsf{Dbm}_n$ and $X \subseteq \{1, \ldots, n\}$ with $X = \{k_1, \ldots, k_m\}$, where $k_i < k_{i+1}$ for all $i \in \{1, \ldots, m-1\}$, the X-**projection** of D is the $card(X)$-DBM $D\big|_X$ whose (i,j)-component is:*

$$\big(D\big|_X\big)_{ij} \stackrel{def}{=} D_{k_i k_j}.$$

Note that, on arbitrary DBMs, projection is *not compositional*: for example, the 3-DBM $D = \begin{pmatrix} \{0\} & \{1\} & \{1\} \\ \{-1\} & 0 & \{1\} \\ \{-1\} & \{-1\} & 0 \end{pmatrix}$ has an empty semantics, but its projection on the set $\{1,2\}$ has a nonempty semantics, since $(0,1) \in \|D\|\big|_{\{1,2\}}$.

Proposition 3 *Projection is compositional on n-DBMs in normal form, that is, $\|D\big|_X\| = \|D\|\big|_X$, for any $D \in \mathsf{Dbm}_n$ and $X \subseteq \{1, \ldots, n\}$.*

Juxtaposition. We would like to define a juxtaposition operation on DBMs, that would verify the equation

$$\|D \Box_p D'\| = \|D\| \Box_p \|D'\|, \tag{3}$$

for all $D \in \mathsf{Dbm}_m$, $D' \in \mathsf{Dbm}_n$ and $p \leq (m,n)$.

But the set of tuples $\|D\| \Box_p \|D'\|$ is the semantics of a DBM: take the block-decompositions $D = \begin{pmatrix} D_1 & D_2 \\ D_3 & D_4 \end{pmatrix}$, $D' = \begin{pmatrix} D'_1 & D'_2 \\ D'_3 & D'_4 \end{pmatrix}$, with $D_4, D'_1 \in \mathsf{Dbm}_p$. Then $D \Box_p D' = \begin{pmatrix} D_1 & D_2 & E \\ D_3 & D_4 \cap D'_1 & D'_2 \\ E^t & D'_3 & D'_4 \end{pmatrix}$ where E is a $(m-p) \times (n-p)$ matrix of intervals whose (i,j)-component is $E_{ij} = \bigcap_{k=m-p+1}^{m} D_{ik} + D'_{k-m+p, j+p}$, while E^t is the transpose of E. Observe that, when each component in D and D' is a finite interval, all the components of E are finite intervals too.

We may then define composition by combining juxtaposition with projection, similar to Definition 2: $D_1 \odot D_2 \stackrel{def}{=} \big(D_1 \Box_n D_2\big)\big|_{\{1,\ldots,n\} \cup \{2n+1, \ldots, 3n\}}$.

On the other hand, the equation

$$\|D^{\circledast}\| = \|D\|^{\circledast} \tag{4}$$

defines a *set-based* operation on DBMs in normal form. That is, for each DBM D, it defines the set of DBMs for which the union of their semantics gives $\|D\|^{\circledast}$. To see that, in general, this set is not the semantics of a single DBM, take e.g. $D = \begin{pmatrix} \{1\} & \{0\} \\ \{0\} & \{-1\} \end{pmatrix}$. We have that $\|D\|^{\circledast}$ is not the semantics of any DBM. In fact, $\|D\|^{\circledast}$ is the union of the semantics of an infinite family of DBMs in normal form.

The big problem raised by these definitions is the symbolic calculation of $D^{\circledast}$ – to actually compute it, we need a way to finitely represent $\|D\|^{\circledast}$, and we also need that this finite representation be endowed with an algorithm for emptiness checking.

Let us see, on the other hand, that Equation 3 can also be satisfied by a *set-based operation*, that is, by an operation $\blacksquare_p$ which associates to each pair (D_1, D_2) a *set* of DBMs whose union of semantics equals $\|D_1\| \blacksquare_p \|D_2\|$. This will be our choice in the next section: we decompose each DBM into the smallest DBMs in normal form, which we call *region matrices*. Then, $D_1 \blacksquare_p D_2$ will be the *set of region matrices* whose semantics is included in $\|D_1\| \blacksquare_p \|D_2\|$.

The idea behind the decomposition of each DBM into region matrices is to reduce the problem of representing infinite sets of DBMs to the problem of representing infinite sets of region matrices.

4 Region matrices

This section implements the idea of defining composition on DBMs by decomposing them into region matrices and employing a set-based composition on region matrices. This solution still induces further problems, connected to the fact that we may have to handle *infinite* sets of region matrices. These problems are discussed at the end of this section.

Definition 4 *An n-DBM $D \in \mathsf{Dbm}_n$ is called an n-**region matrix** if it has a nonempty semantics and, for each $i, j \in \{1, \ldots, n\}$, one of the following properties holds:*

- *Either $D_{ij} = \{\alpha\}$, for some $\alpha \in \mathbb{Z}$,*
- *Or $D_{ij} =]\beta, \beta+1[$, for some $\beta \in \mathbb{Z}$.*

Observe that each region matrix is in normal form. The set of region matrices is denoted Reg_n. Their name is drawn from the similarities with the regions of [1].

The following property shows the principle of the decomposition of DBMs into region matrices:

Proposition 4 *For all $D \in \mathsf{Dbm}_n$, $\|D\| = \bigcup_{R \sqsubseteq D} \|R\|$.*

This property can be transformed into a semi-algorithm as follows: given an n-DBM D, we decompose each of its components into point intervals or open unit intervals. Then we take all combinations of such intervals – one interval for each (i, j) – and check whether the resulting matrix is in normal form. The set of matrices which pass this test equals the set of regions $R \sqsubseteq D$. This semi-algorithm terminates iff all the intervals in D are finite.

Definition 5 *Given two region matrices $R_1 \in \mathsf{Reg}_m$, $R_2 \in \mathsf{Reg}_n$ and $p \leq \min(m, n)$, the **region p-juxtaposition** of R_1 and R_2 is the following* set *of $(m+n-p)$-regions:*

$$R_1 \blacksquare_p R_2 \stackrel{def}{=} \big\{R \in \mathsf{Reg}_{m+n-p} \mid R\big|_{\{1,\ldots,m\}} = R_1, \\ R\big|_{\{m-p+1,\ldots,m+n-p\}} = R_2\big\}.$$

Note that $R_1 \blacksquare_p R_2 = \emptyset$ iff $R_1\big|_{\{m-p+1,\ldots,m\}} \neq R_2\big|_{\{1,\ldots,p\}}$.

On the other hand, we have a simple algorithm for checking whether a juxtaposition $R_1 \blacksquare_p R_2$ is nonempty: first, we build the DBM $R_1 \square_p R_2$, as in Section 3, and then we apply the semi-algorithm for constructing the set $\{R \sqsubseteq R_1 \square_p R_2\}$. Since, by construction, all components of $R_1 \square_p R_2$ are finite intervals, our algorithm terminates.

Proposition 5 *Region juxtaposition is associative and compositional: for each $R_1 \in \mathsf{Reg}_m$, $R_2 \in \mathsf{Reg}_n$ and $p \leq \min(m, n)$, $\|R_1 \blacksquare_p R_2\| = \|R_1\| \blacksquare_p \|R_2\|$.*

Definition 6 *Given $R_1, R_2 \in \mathsf{Reg}_{2n}$, the **$2n$-region-composition** of R_1 and R_2 is the set*

$$R_1 \odot R_2 \stackrel{def}{=} \big\{R\big|_{\{1,\ldots,n\}\cup\{2n+1,\ldots,3n\}} \mid R \in R_1 \blacksquare_n R_2\big\}.$$

For example, $\begin{pmatrix} \{0\} &]0,1[\\ ]-1,0[& \{0\} \end{pmatrix} \odot \begin{pmatrix} \{0\} &]0,1[\\ ]-1,0[& \{0\} \end{pmatrix} =$
$= \left\{ \begin{pmatrix} \{0\} &]0,1[\\ ]-1,0[& \{0\} \end{pmatrix}, \begin{pmatrix} \{0\} & \{1\} \\ \{1\} & \{0\} \end{pmatrix}, \begin{pmatrix} \{0\} &]1,2[\\ ]-2,-1[& \{0\} \end{pmatrix} \right\}.$

As a corollary of Proposition 5, $\|R_1 \odot R_2\| = \|R_1\| \odot \|R_2\|$ for all $R_1, R_2 \in \mathsf{Reg}_{2n}$. Composition can then be lifted to sets of regions as usual, $\mathcal{R}_1 \odot \mathcal{R}_2 = \big\{R_1 \odot R_2 \mid R_1 \in \mathcal{R}_1, R_2 \in \mathcal{R}_2\big\}$. The unit for composition on *sets of region matrices* is the set

$$\mathbf{1}_{2n}^{reg} \stackrel{def}{=} \big\{R \in \mathsf{Reg}_{2n} \forall i, j \in \{1, \ldots, n\}, \\ R_{ij} = R_{n+i,j} = R_{i,n+j} = R_{n+i,n+j}\big\}.$$

The *star* operation $\circledast$ can then be defined as follows: for each $\mathcal{R} \subseteq \mathsf{Reg}_{2n}$, $\mathcal{R}^{\circledast} = \bigcup_{k\in\mathbb{N}} \mathcal{R}^{k\odot}$, where $\mathcal{R}^{0\odot} = \mathbf{1}_{2n}^{reg}$ and $\mathcal{R}^{(k+1)\odot} = \mathcal{R}^{k\odot} \odot \mathcal{R}$ for all $k \in \mathbb{N}$. Then star is compositional too, that is, $\|\mathcal{R}^{\circledast}\| = \|\mathcal{R}\|^{\circledast}$ for all $\mathcal{R} \subseteq \mathsf{Reg}_{2n}$. Hence, $\big(\mathcal{P}(\mathsf{Reg}_{2n}), \cup, \odot, \circledast, \emptyset, \mathbf{1}_{2n}^{reg}\big)$ is a Kleene algebra.

One of the classic way to handle infinite sets of objects which are endowed with a composition operation is to consider "regular expressions" over these objects. *Regular expressions over DBMs* are then the class of expressions generated by the grammar:

$$E ::= D \mid E \cup E \mid E \odot E \mid E^{\circledast}, \text{ where } D \in \mathsf{Dbm}_{2n}.$$

The semantics of these expressions is straightforward, e.g.:

$$\|E_1 \odot E_2\| = \|E_1\| \odot \|E_2\|, \ \|E^{\circledast}\| = \|E\|^{\circledast}.$$

Theorem 1 *The emptiness problem for regular expressions over $2n$-DBMs is undecidable for $n \geq 2$.*

The proof relies on the possibility to code runs of two-counter machines (see e.g., [10]), as regular expressions over DBMs.

This theorem emphasizes again the problem of working with infinite sets of region matrices. We therefore need to identify "regularities" in such infinite sets of region matrices, such that only a finite amount of pairs (R_1, R_2) be checked whether $\|R_1 \odot R_2\| \neq \emptyset$. We introduce in the next section the concept of n-automata as a means to code "regularity" in sets of region matrices.

5 n-automata

In this section we give the definition and the basic properties of n-automata. We show that $2n$-automata are closed under composition. We will identify, in the next section, a special property which assures star closure. To ease the understanding, this section shows how n-automata can be used to represent sets of *point* region matrices. The use of n-automata for representing *nonpoint* region matrices is the subject of Section 7.

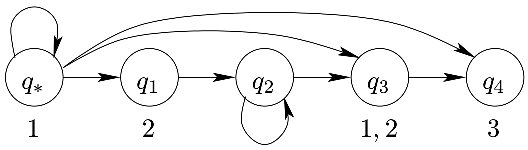
Let us denote by PReg_n the set of point n-regions. The nice feature of each point n-region is that its semantics contains points in $\mathbb{R}^n_{\geq 0}$ whose coordinates are all natural numbers. And if we represent these numbers in unary, they become words over a one-letter alphabet. This is the intuition behind our definition of n-automata.

Definition 7 *An **n-automaton** is a tuple $\mathcal{A} = (Q, \delta, q_*, Q_1, \ldots, Q_n)$ in which Q is a finite set of* states, *$\delta \subseteq Q \times Q$ is a set of* transitions, *and for each $i \in \{1, \ldots, n\}$, Q_i is a set of* accepting states for index i. *Finally, $q_* \in Q$ is called the* initial state *and we assume $(q_*, q_*) \in \delta$ and $\forall i \in \{1, \ldots, n\}$, $\forall q \in Q_i$, $(q_*, q) \in \delta$.*

An n-automaton accepts a tuple $(a_1, \ldots, a_n) \in \mathbb{N}^n$ in the following way: the automaton starts in the specially designated state q_*, and tries to build a run that passes through all the sets of accepting states. For each $i \in \{1,\dots,n\}$, one of the moments when the run passes through Q_i must be exactly when the length of the run is a_i.

More formally, a **run** is a sequence of states $\rho = (q_j)_{j\in\{0,\dots,k\}}$ connected by transitions, i.e., $(q_{j-1}, q_j) \in \delta$ for all $j \in \{1,\dots,n\}$, and with $q_0 = q_*$. The run is **accepting** if for each $i \in \{1,\dots,n\}$ there exists some $j \in \{1,\dots,k\}$ such that $q_j \in Q_i$. Given an n-tuple of naturals $\overline{a} = (a_1, \dots a_n) \in \mathbb{N}^n$ we say that $\overline{a}$ is **accepted by** ρ iff $q_{a_i} \in Q_i$ for all $i \in \{1,\dots,n\}$. The **point language** of $\mathcal{A}$ is then the set of points in $\mathbb{N}^n$ accepted by $\mathcal{A}$, and is denoted $L_p(\mathcal{A})$.

For example, consider the following 3-automaton $\mathcal{A}_0$



in which $Q_1 = \{q_*, q_3\}$, $Q_2 = \{q_1, q_3\}$, $Q_3 = \{q_4\}$ as suggested by the indices below the respective states. This automaton accepts the 3-tuple $(1, 4, 5)$: the accepting run for it is $q_* \to q_* \to q_1 \to q_2 \to q_3 \to q_4$. Intuitively, we have put all three components of $\overline{a}$ on the real axis and let $\mathcal{A}_0$ "parse" this axis, unit after unit. Observe that the tuple $(1, 4, 5)$ belongs to the semantics of the point region matrix $W = \begin{pmatrix} \{0\} & \{3\} & \{4\} \\ \{-3\} & \{0\} & \{1\} \\ \{-4\} & \{-1\} & \{0\} \end{pmatrix}$. We then say that $\mathcal{A}_0$ **accepts** W.

More formally, some $W \in \mathsf{PReg}_n$ is accepted by an n-automaton $\mathcal{A}$ if there exists an n-tuple in $\|W\|$ which is accepted by $\mathcal{A}$. The **region language** of $\mathcal{A}$ is the set of point region matrices accepted by $\mathcal{A}$ and is denoted $L_r(\mathcal{A})$.

One more observation: the requirement that $(q_*, q_*) \in \delta$ implies that if a tuple $\overline{a} = (a_1, \dots, a_n)$ is accepted by $\mathcal{A}$ then for any $k \in \mathbb{N}$, $\overline{a} + k = (a_1 + k, \dots, a_n + k) \in L_p(\mathcal{A})$.

Proposition 6 *The nonemptiness problem for n-automata is decidable, and its complexity class is NP-complete.*

The NP-hardness follows from a reduction of the Hamiltonian Path Problem (see [8]) to the nonemptiness problem for n-automata. For the NP-easiness, consider the following "nondeterministic" algorithm (i.e., the *pick* instructions are nondeterministic):

$X := \{i \mid q_* \in Q_i\}$; $S := \emptyset$; $T := \{(q_*, X)\}$;
`while` $X \neq \{1,\dots,n\}$ `and` $S \neq T$ `do`
  $S := T$;
  pick $(q, X) \in T$; pick $a \in \Sigma$; pick $r \in Q$;
  `if` $(q, a, r) \notin \delta$ `then stop`;
  $X := X \cup \{i \in \{1,\dots,n\} \mid r \in Q_i\}$;
  $T := \big(T \setminus (\{r\} \times \mathcal{P}(\{1,\dots,n\}))\big) \cup \{(r, X)\}$;
`endwhile`;
`if` $X = \{1,\dots,n\}$ then `write(``nonempty'')`
  `else write(``empty'')`.

The algorithm associates to each state $q \in Q$ an index set $X \subseteq \{1,\dots,n\}$ iff there exists a run that starts in q_*, ends in q and passes through each of the accepting sets whose indices are in X. This nondeterministic algorithm runs in polynomial time and linear space in the size of $\mathcal{A}$.

Proposition 7 *1) For each n-DBM $D \in \mathsf{Dbm}_n$, the set $\{W \in \mathsf{PReg}_n \mid W \sqsubseteq D\}$ can be accepted by an n-automaton.*

2) The class of region languages accepted by n-automata is closed under union and intersection.

3) Given an n-automaton $\mathcal{A}$ and $X \subseteq \{1,\dots,n\}$, the set of region matrices $L_r(\mathcal{A})\big|_X$ can be accepted by a $card(X)$-automaton.

4) Given two $2n$-automata $\mathcal{A}$, $\mathcal{B}$, there exists a $2n$-automaton $\mathcal{C}$ with $L_r(\mathcal{C}) = L_r(\mathcal{A}) \odot L_r(\mathcal{B})$.

The assumption $(q_*, q_*) \in \delta$ is essential for the closure under composition: with this assumption, for $W \in L_r(\mathcal{A})$, $W' \in L_r(\mathcal{B})$ for which $W \odot W'$ is defined, there exist $\overline{a}, \overline{b} \in \mathbb{N}^{2n}$ with $\overline{a} \in \|W\| \cap L_p(\mathcal{A})$ and $\overline{b} \in \|W'\| \cap L_p(\mathcal{B})$. This does not yet assure that $\overline{a}\big|_{\{n+1,\dots,2n\}} = \overline{b}\big|_{\{1,\dots,n\}}$ – it only assures that for all $i, j \in \{1,\dots,n\}$, $\overline{a}_{n+i} - \overline{a}_{n+j} = \overline{b}_i - \overline{b}_j$. But remind that, as $(q_*, q_*) \in \delta_{\mathcal{A}} \cap \delta_{\mathcal{B}}$, for any $a, b \in \mathbb{N}$, we have that $\overline{a} + a \in L_p(\mathcal{A})$ and $\overline{b} + b \in L_p(\mathcal{B})$. Hence we may find some a and b such that $\overline{a}_1 + a = \overline{b}_1 + b$, which assures that $(\overline{a} + a)\big|_{\{n+1,\dots,2n\}} = (\overline{b} + b)\big|_{\{1,\dots,n\}}$.

On the other hand, observe that Theorem 1 implies nonclosure under star for $2n$-automata.

6 Non-elasticity

We give here a property on $L_r(\mathcal{A})$ that assures that $L_r(\mathcal{A})^{\circledast}$ can be accepted by a $2n$-automaton. A motivation for this rather technical property comes from timed automata: if we look at the formula f_τ that specifies the relation R_τ for τ a transition in a timed automaton (formula 2) we see the following particularity: for each valuation of variables, say $x_i := a_i$, $x_{n+1} := a_{n+1}$, $x'_i := a_{n+i+1}$, $x'_{n+1} := a_{2n+2}$ which satisfies f_τ, whenever $a_{n+i+1} \neq a_i$ then $a_{n+i+1} \geq a_j$ for all $i, j \in \{1,\dots,n+1\}$.

We will show that a slightly more general property than this, translated to $2n$-tuples $\overline{a} = (a_1, \dots, a_{2n})$, suffices for star closure. This property is the following:

> For each $i, j \in \{1,\dots,n\}$, if $a_{n+i} \neq a_i$ and $a_{n+j} \neq a_j$ then $a_{n+i} - a_j \geq 0$ and $a_{n+j} - a_i \geq 0$.

Each $2n$-tuple with this property is called **non-elastic**. When the tuple satisfies the unconditional requirement $a_{n+i} - a_j \geq 0$ and $a_{n+j} - a_i \geq 0$ for all $i, j \in \{1,\dots,n\}$ we say it is a *strictly non-elastic tuple*.

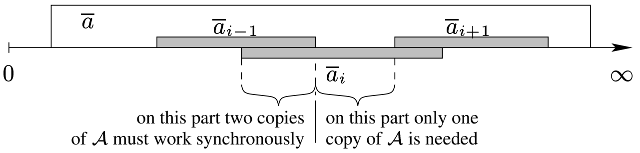
Translated to region matrices, non-elasticity gives the following property: a $2n$-region R is called **non-elastic** if for each $i, j \in \{1, \dots, n\}$, if $R_{i,n+i} \neq \{0\}$ and $R_{j,n+j} \neq \{0\}$ then $R_{j,n+i} \geq \{0\}$ and $R_{i,n+j} \geq \{0\}$. Here, for two intervals $I, J \subseteq \mathbb{R}$, $I \leq J \Longleftrightarrow \forall\alpha \in I, \forall\beta \in J, \alpha \leq \beta$.

The central theorem of this paper is the following:

Theorem 2 *Given a $2n$-automaton $\mathcal{A}$, suppose that, for all $k \in \mathbb{N}$, $L_r(\mathcal{A})^{k\odot}$ is composed of non-elastic region matrices only. Then $L_r(\mathcal{A})^{\circledast}$ can be recognized by a $2n$-automaton.*

The proof of this result can be found in [7]. For the simple case when each set $L_r(\mathcal{A})^{k\odot}$ is composed of *strictly* non-elastic tuples only, the idea is to use, at each moment, one or two replicas of $\mathcal{A}$, such that, each time one component has completed an accepting run, a new component starts checking for an accepting run. The two copies work "synchronously": for each $i \in \{1, \dots, n\}$, one of the passages of the first copy through Q_{n+i} must happen exactly when the second copy passes through Q_i.

A graphical presentation of the way a tuple $\overline{a} \in L(\mathcal{A})^{\circledast\geq 2}$ must be parsed is given in the following figure. Here, the big strip labeled with $\overline{a}$ shows the zone on the real axis where the components of $\overline{a}$ are located. This tuple is decomposed as $\overline{a} = \overline{a}_1 \odot \dots \odot \overline{a}_k$, with $\overline{a}_i \in L_p(\mathcal{A})$ for all $i \in \{1, \dots, k\}$, and the shadowed strips represent the segments where the components of each $\overline{a}_i$ lie on the real axis.



7 Representing region matrices with n-automata

Our approach to computing the reachability relation of a timed automaton is to build a $2n$-automaton for each relation R_τ, then to apply the union/composition/star constructions for $2n$-automata. To show that this approach works, we have to adapt $2n$-automata to represent also regular sets of arbitrary region matrices. And we also have to show that the sets of region matrices representing relations $R_{qq'}$ in timed automata satisfy the nonelasticity property. These are the topics of this section.

The adaptation of n-automata for nonpoint region matrices is based upon the following idea: we represent each nonpoint n-region with one of the point region matrices neighboring it, in pair with some information about the "direction" in which the representative is situated w.r.t. the original region matrix. For example, consider the region matrix $R = \begin{pmatrix} \{0\} &]2,3[&]1,2[\\ ]-3,-2[& \{0\} &]-1,0[\\ ]-2,-1[&]0,1[& \{0\} \end{pmatrix}$ Our idea is to represent R with the point region matrix $W = \begin{pmatrix} \{0\} & \{3\} & \{2\} \\ \{-3\} & \{0\} & \{-1\} \\ \{-2\} & \{1\} & \{0\} \end{pmatrix}$ in pair with the following "matrix of relational symbols": $M = \begin{pmatrix} `=` & `<` & `<` \\ `>` & `=` & `>` \\ `>` & `<` & `=` \end{pmatrix}$. The connection is the following: M_{12} is '$<$' because $R_{12} < W_{12}$, M_{22} is '$=$' because $R_{22} = W_{22}$, etc.

Two other representations of R by point region matrices can be found. However not all 9 combinations of upper and lower bounds of the intervals in R give a point region matrix – we have at most n neighboring point region matrices for each n-dimensional region matrix. Moreover, not all matrices of relational symbols may carry the information regarding the "direction of approximation". The actual matrices we use are the following:

Definition 8 *An **n-relation** is an $n \times n$ matrix M over the set $\Gamma = \{`<`, `=`, `>`\}$ satisfying the following property: there exists no sequence of indices $(i_1, i_2, i_3, i_4 = i_1)$ with $i_1, i_2, i_3 \in \{1, \dots, n\}$ for which, for all $j \in \{1, \dots, 3\}$, $M_{i_j,i_{j+1}} \in \{`<`, `=`\}$ and for some $j \in \{1, \dots, 3\}$, $M_{i_j,i_{j+1}} = `<`$, The set of n-relations is denoted Γ_n.*

*For $M \in \Gamma_n$ and $X \subseteq \{1, \dots, n\}$, the **$X$-projection** of M is the card(X)-relation resulting by deleting from M the rows and columns that are not in X. Formally, for $X = \{k_1, \dots, k_m\}$, where $k_i < k_{i+1}$ for all $i \in \{1, \dots, m-1\}$, and for all $i, j \in \{1, \dots, m\}$, $\left(M\big|_X\right)_{ij} \stackrel{def}{=} M_{k_i,k_j}$.*

*Given an m-relation $M_1 \in \Gamma_m$, an n-relation $M_2 \in \Gamma_n$ and a positive integer $p \leq \min(m, n)$, the **p-juxtaposition** of M_1 with M_2 is the set of $(m + n - p)$-relations:*

$$M_1 \blacksquare_p M_2 \stackrel{def}{=} \big\{M \in \Gamma_{m+n-p} \mid M\big|_{\{1,\dots,m\}} = M_1, \quad M\big|_{\{m-p+1,\dots,m+n-p\}} = M_2\big\}.$$

*The **composition** of $M_1, M_2 \in \Gamma_{2n}$ is*

$$M_1 \odot M_2 \stackrel{def}{=} \left(M_1 \blacksquare_n M_2\right)\big|_{\{1,\dots,n\}\cup\{2n+1,\dots,3n\}}.$$

Note that $M_1 \square_p M_2 = \emptyset$ iff $M_1\big|_{\{m-p+1,\dots,m\}} \neq M_2\big|_{\{1,\dots,p\}}$.

We may easily show that relation composition is associative. The unit for composition on *sets* of $2n$-relations is

$$\mathbf{1}_{2n}^{rel} \stackrel{def}{=} \big\{M \in \Gamma_{2n} \mid \forall i, j \in \{1, \dots, n\}, \quad M_{ij} = M_{n+i,j} = M_{i,n+j} = M_{n+i,n+j}\big\}.$$

Further, we may define the *star* of a set of $2n$-relations $\mathcal{M} \subseteq \Gamma_{2n}$ as $\mathcal{M}^{\circledast} = \bigcup_{k\in\mathbb{N}} \mathcal{M}^{k\odot}$, where $\mathcal{M}^{0\odot} = \mathbf{1}_{2n}^{rel}$ and $\mathcal{M}^{(k+1)\odot} = \mathcal{M}^{k\odot} \odot \mathcal{M}$. Then the structure $\mathcal{P}\big(\Gamma_{2n}, \cup, \odot, \circledast, \emptyset, \mathbf{1}_{2n}^{rel}\big)$ is a Kleene algebra.

Our representation of region matrices is the following:

Definition 9 *A tuple $(W, M) \in \mathsf{PReg}_n \times \Gamma_n$ is called an **n-region representation**. The n-region $R \in \mathsf{Reg}_n$ **represented** by (W, M) is the region matrix denoted $[W, M]$, for* *which, for all* $i, j \in \{1, \ldots, n\}$,

$$[W, M]_{ij} \stackrel{def}{=} \begin{cases} \{W_{ij}\} & \text{iff } M_{ij} = `=` \\ ]W_{ij}-1, W_{ij}[& \text{iff } M_{ij} = `<` \\ ]W_{ij}, W_{ij}+1[& \text{iff } M_{ij} = `>`. \end{cases}$$

The above definition would be incorrect unless we prove:

Proposition 8 *For each* $(W, M) \in \mathsf{PReg}_n \times \Gamma_n$, $[W, M]$ *has a nonempty semantics.*

The mapping $[\cdot] : \mathsf{PReg}_n \times \Gamma_n \to \mathsf{Reg}_n$ defines an equivalence relation on $\mathsf{PReg}_n \times \Gamma_n$ (the *kernel* of $[\cdot]$) denoted in the sequel $\equiv_n$ and defined as follows: $(W, M) \equiv_n (W', M')$ iff $[W, M] = [W', M']$. A set of n-region representations $\mathcal{M}$ is called **saturated** by $\equiv_n$ iff for each $(W, M) \in \mathcal{M}$, if $[W, M] = [W', M']$ for some $(W', M') \in \mathsf{PReg}_n \times \Gamma_n$ then also $(W', M') \in \mathcal{M}$. For each $\mathcal{W} \subseteq \mathsf{PReg}_n \times \Gamma_n$, we denote

$$[\mathcal{W}] \stackrel{def}{=} \{[W, M] \mid (W, M) \in \mathcal{W}\}.$$

We may now define projection, juxtaposition, composition and star on n-region representations:

$$(W, M)\big|_X \stackrel{def}{=} (W\big|_X, M\big|_X)$$

$$(W, M) \blacksquare_p (W', M') \stackrel{def}{=} \{W \blacksquare_p W', M'') \mid M'' \in M \blacksquare_p M'\}$$

$$(W, M) \odot (W', M') \stackrel{def}{=} \big((W, M) \blacksquare_n (W', M')\big)\big|_{\substack{\{1,\ldots,n\}\cup \\ \{2n+1,\ldots,3n\}}}$$

If we put $\mathbf{1}_{2n}^{preg} = \mathbf{1}_{2n}^{reg} \cap \mathsf{PReg}_{2n}$, then $\mathbf{1}_{2n}^{preg} \times \mathbf{1}_{2n}^{rel}$ is the unit for composition on $2n$-region representations. Further, we may build star of a set of $2n$-region representations $\mathcal{W} \subseteq \mathsf{PReg}_{2n} \times \Gamma_{2n}$ as usual, $\mathcal{W}^{\circledast} = \bigcup_{k \geq 0} \mathcal{W}^{k\odot}$, where $\mathcal{W}^{0\odot} = \mathbf{1}_{2n}^{preg} \times \mathbf{1}_{2n}^{rel}$ and $\mathcal{W}^{(k+1)\odot} = \mathcal{W}^{k\odot} \odot \mathcal{W}$.

Proposition 9 *For each* $(W, M) \in \mathsf{PReg}_n \times \Gamma_n$, $[W\big|_X, M\big|_X] = [W, M]\big|_X$.

For each two saturated sets $\mathcal{W}_1, \mathcal{W}_2 \subseteq \mathsf{PReg}_{2n} \times \Gamma_{2n}$, $\mathcal{W}_1 \odot \mathcal{W}_2$ *is a saturated set and* $[\mathcal{W}_1 \odot \mathcal{W}_2] = [\mathcal{W}_1] \odot [\mathcal{W}_2]$.

For each saturated set $\mathcal{W} \subseteq \mathsf{PReg}_{2n} \times \Gamma_{2n}$, $\mathcal{W}^{\circledast}$ *is saturated and* $[\mathcal{W}^{\circledast}] = [\mathcal{W}]^{\circledast}$.

Definition 10 *An* ***n-region automaton*** *is a tuple* $\mathcal{A} = (Q, \delta, Q_*, Q_1, \ldots Q_n, \lambda)$ *in which* Q, δ, *and* $Q_1, \ldots, Q_n$ *bear the same meaning and properties as in n-automata, while* $Q_* \subseteq Q$ *is a* set of initial states *and* $\lambda : Q \to \Gamma_n$ *is an* n-relation labeling function. *Additionally, it is required that if* $(q, q') \in \delta$ *then* $\lambda(q) = \lambda(q')$, *and that, for all* $q_* \in Q_*$, *for all* $i \in \{1, \ldots, n\}$ *and all* $q \in Q_i$, $(q_*, q_*), (q_*, q) \in \delta$.

The notion of accepting run is defined as in n-automata, with the only difference that it may start in any state from Q_*. Observe that, by definition, all the states in a run must be labeled with the same n-relation. An accepting run $\rho = (q_j)_{j \in \{0,\ldots,k\}}$ **accepts** an n-region representation (W, M) iff $\lambda(q_j) = M$ for all $j \in \{0, \ldots, k\}$ and there exists some n-tuple $(a_1, \ldots, a_n) \in \mathbb{N} \cap \|W\|$ such that $q_{a_i} \in Q_i$ for all $i \in \{1, \ldots, n\}$. We associate three languages to each n-region automaton $\mathcal{A}$:

- The **n-region representation language accepted by** $\mathcal{A}$ consists of the n-region representations accepted by some accepting run. and is denoted $L_{rep}(\mathcal{A})$.
- The **region language** of $\mathcal{A}$ is

$$L_{rgn}(\mathcal{A}) \stackrel{def}{=} \{[W, M] \mid (W, M) \in L_{rep}(\mathcal{A})\}.$$

- The **n-tuple language** of $\mathcal{A}$ is

$$L_{tup}(\mathcal{A}) \stackrel{def}{=} \bigcup \{\|R\| \mid R \in L_{rgn}(\mathcal{A})\}.$$

We say that an n-automaton is **saturated** if its n-region representation language is saturated.

Observe that, for each two n-region automata $\mathcal{A}$ and $\mathcal{B}$, $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ iff $L_{tup}(\mathcal{A}) = L_{tup}(\mathcal{B})$, but it might be possible that $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ and $L_{rep}(\mathcal{A}) \neq L_{rep}(\mathcal{B})$, due to the possibility to represent the same n-region by different n-region representations. But when $\mathcal{A}$ and $\mathcal{B}$ are saturated, then we also have $L_{rgn}(\mathcal{A}) = L_{rgn}(\mathcal{B})$ iff $L_{rep}(\mathcal{A}) = L_{rep}(\mathcal{B})$.

Proposition 10 *1) If $\mathcal{A}$ and $\mathcal{B}$ are two saturated n-region automata, then one can build a saturated n-region automaton for* $L_{tup}(\mathcal{A}) \cup L_{tup}(\mathcal{B})$ *and* $L_{tup}(\mathcal{A}) \cap L_{tup}(\mathcal{B})$.

2) For each saturated n-region automaton $\mathcal{A}$, and $X \subseteq \{1, \ldots, n\}$, $L_{tup}(\mathcal{A})\big|_X$ *can be accepted by a saturated* $card(X)$*-automaton* $\mathcal{B}$.

3) Given two saturated $2n$-region automaton $\mathcal{A}$ and $\mathcal{B}$, there exists a $2n$-region automaton $\mathcal{D}$ with $L_{tup}(\mathcal{D}) = L_{tup}(\mathcal{A}) \odot L_{tup}(\mathcal{B})$.

Theorem 3 *Given a saturated $2n$-region automaton $\mathcal{A}$, suppose that for all $k \in \mathbb{N}$, $L_{tup}(\mathcal{A})^{k\odot}$ is contains only non-elastic tuples. Then $L_{tup}(\mathcal{A})^{\circledast}$ can be recognized by a saturated $2n$-region automaton.*

The only thing that remains to be shown is that this theorem works for the relations $R_{qq'}$ in timed automata. To this end, remind that we have associated a conjunctive formula f_τ for each transition τ in a timed automaton. Each formula f_τ is representable by a $2n$-DBM $D(\tau)$. Let us denote $\mathcal{L}$ the union of the semantics of all $2n$-DBMs $D(\tau)$, over all possible transitions τ that may occur in some timed automaton.

Proposition 11 *$\mathcal{L}$ is composed only of non-elastic tuples and is closed under composition.*

Therefore, for any subset of $\mathcal{L}$ which is representable by a $(2n+2)$-region automaton we may apply Theorem 3. Observe that the closure under composition of $\mathcal{L}$ implies that all relations $R_{qq'}$ are included in $\mathcal{L}$.

8 Conclusions

We have presented a relational approach for computing reachability relations in timed automata, approach which is based upon representing clock relations by n-automata. We have seen that, contrary to DBMs, n-automata may represent "disjunctive" information, that is, formulas over diagonal clock constraints that utilize disjunctions. A nice feature of this approach is its modularity, which may help in building "parallel" model-checking algorithms.

Observe that there is no connection between the $2n$-automaton obtained for each relation $R_{qq'}$ in a timed automaton and the *region construction* of [1]. Each composition and star "forgets" the intermediary steps within a run, hence we cannot recover all the "discrete states" in the region automaton.

Unfortunately n-automata are essentially nondeterministic, hence the problem of finding small automata representations of timing constraints is somewhat related to the problem of finding small nondeterministic finite automata. In particular, the union construction of several n-automata has to be accompanied by a technique which identifies some states, in order to reduce the state space.

One direction of future research is to compare the non-elasticity property with the "reversal-bounded" properties that assure decidability of Turing machines [6].

Acknowledgments

The author wishes to thank Evgeni Asarin and Oded Maler for their support in doing this research. We are also indebted to Andreas Podelski and Andrei Voronkov whose valuable observations have essentially improved the contents and presentation of the paper.

References

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

[2] E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of LICS'97*, pages 160–171, 1997.

[3] R. Bellmann. *Dynamic Programming*. Princeton University Press, 1957.

[4] J. Bengtsson, B. Jonsson, J. Lilius, and Wang Yi. Partial order reductions for timed systems. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 485–500, 1998.

[5] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proceedings of CONCUR'99*, volume 1664 of *LNCS*, pages 242–257, 1999.

[6] Zhe Dang. Binary reachability analysis of pushdown timed automata with dense clocks. In *Proc. of CAV 2001*, volume 2102 of *LNCS*, pages 506 –517, 2001.

[7] C. Dima. *An algebraic theory of real-time formal languages*. PhD thesis, Université Joseph Fourier Grenoble, France, 2001.

[8] M. R. Garey and D. S. Johnson. *Computers and intractability*. W.H. Freeman & Co., 1979.

[9] S. Gaubert and Max Plus. Methods and applications of $(\max, +)$ linear algebra. In *Porceedings of STACS'97*, volume 1200 of *LNCS*, pages 261–282, 1997.

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

[11] Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Integration graphs: a class of decidable hybrid systems. In *Proceedings of "Workshop on Theory of Hybrid Systems"*, volume 736 of *LNCS*, pages 179–208, 1992.

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

[13] K. G. Larsen, Paul Petterson, and Wang Yi. Uppaal: Status & developments. In *Proceedings of CAV'97*, LNCS, pages 456–459, 1997.

[14] I. Meiri, R. Dechter, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, pages 61–95, 1991.

[15] S. Mukhopadhyay and A. Podelski. Beyond region graphs: Symbolic forward analysis of timed automata. In *Proceedings of FST&TCS'99*, volume 1738 of *LNCS*, pages 232–244, 1999.

[16] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[17] S. Yovine. Model-checking timed automata. In *Lectures on Embedded Systems*, volume 1494 of *LNCS*, pages 114–152, 1998.