Formalized Timed Automata

Simon Wimmer*

Institut für Informatik, Technische Universität München

Abstract. Timed automata are a widely used formalism for modeling real-time systems, which is employed in a class of successful model checkers such as UPPAAL. These tools can be understood as trust-multipliers: we trust their correctness to deduce trust in the safety of systems checked by these tools. However, mistakes have previously been made. This particularly regards an approximation operation, which is used by model-checking algorithms to obtain a finite search space. The use of this operation left a soundness problem in the tools employing it, which was only discovered years after the first model checkers were devised. This work aims to provide certainty to our knowledge of the basic theory via formalization in Isabelle/HOL: we define the main concepts, formalize the classic decidability result for the language emptiness problem, prove correctness of the basic forward analysis operations, and finally outline how both streams of work can be combined to show that forward analysis with the common approximation operation correctly decides emptiness for the class of diagonal-free timed automata.

1 Introduction

The foundations of the theory of timed automata are presented in the seminal work of Alur and Dill [1, 2]. They introduced the formalism as a model for systems with real-time constraints and showed how to decide the language emptiness problem via the so-called *region* construction. Unfortunately, the number of regions explored by this algorithm is exponential in the size of the automaton under consideration. Moreover, Alur and Dill also showed that the language emptiness problem for timed automata is PSPACE-hard. Still, the formalism is employed in practical model checking [12, 13, 19] by means of algorithms based on *Difference Bound Matrices* (DBMs). These algorithms (with some more elaborate optimizations) can cope with many interesting real-life model checking problems. The search space examined by the DBM algorithms is potentially infinite. Therefore an approximation is used to obtain a finite search space. The basic idea is to represent every state (called *zone*) by the smallest set of regions which contains the state.

* Supported by DFG project NI 491/16-1

It took nearly a decade after this operation was initially devised, until Patricia Bouyer discovered [5] that the common algorithmic realization of this operation diverges from its intended result: the computed result is always a convex union of regions, whereas the smallest set of regions containing a zone can be non-convex. This left a soundness problem, which fortunately vanishes for the restricted class of so-called diagonal-free timed automata [6] (Section 2.1 precisely characterizes this class). While not as expressive as the full formalism of timed automata, this class is sufficient for modeling most of the problems of practical interest, which explains why the problem was not discovered for many years.

This work aims to solidify the theoretical grounds on which real-time model checking with *diagonal-free* timed automata stands, by formalizing the basic theory and algorithms in Isabelle/HOL, and then going the full length to prove Bouyer's correctness result. Section 2 will present the formalization of the basic notions for diagonal-free timed automata. Then Section 3 will show how we formalized DBMs and obtained soundness and completeness results for their basic algorithms. This includes a formalization of the Floyd-Warshall algorithm. Afterwards (Section 4) we define the notion of regions and prove that they are suitable for deciding the emptiness problem on timed automata. Finally, in Section 5, a refined version of these regions will be used to precisely formalize the approximation operation. To tie the ends of our formalization together, this characterization of approximation will be connected with its algorithmic version. This enables us to reuse the decidability result on the first region construction to prove that DBM-based algorithms together with approximation can decide the language emptiness problem for diagonal-free timed automata. For lack of space, many of our definitions and proofs are shortened or stated informally. We refer the reader to the entry in the Archive of Formal Proofs [15] for the full version (over 18500 lines).

1.1 History and Related Work

As mentioned, the basic theory was devised by Alur and Dill [1, 2]. The use of DBMs was also proposed by Dill [10] and brought to practical model checking by Yi et al. [18]. Bouyer's developments of our main correctness results are spread over two papers. The first one presents a generalization of timed automata to *updatable* timed automata and revisits the basic decidability results for this class [7]. The second one [6] connects these results with DBMs to prove that the combination of DBM-based forward analysis operations and approximation decides the language emptiness problem.

We are aware of one previous proof-assistant formalization of timed automata using PVS [16,17]. This work has the basic decidability result using regions and claims to make some attempt to extend the formalization towards DBMs. Another line of work [11, 14] aims at modeling the class of *p-automata* [3] (which is undecidable in the general case) in Coq and proving properties of concrete p-automata within Coq. A similar approach was pursued with the help of Isabelle/HOL in the *CClair* project [8]. In contrast, the most important contributions of our work are the full formalization of the relevant DBM algorithms,

and particularly the rather intricate developments towards the correctness proof for the approximation operation – both of which pertain to practical real-time model checking.

Unless otherwise stated, our formalizations of the basic notions and DBMs are based on a popular tutorial by Bengtsson and Yi [4], while the developments for the region constructions and the final correctness result follow Bouyer's precise work.

2 Diagonal-Free Timed Automata in Isabelle/HOL

2.1 Syntactic Definition

Compared to standard finite automata, timed automata introduce a notion of clocks. We will fix a type $'c$ for the space of clocks, type $'t$ for time, and a type $'s$ for locations. While most of our formalizations only require $'t$ to belong to a custom type class for totally ordered dense abelian groups, we worked on the concrete type *real* for the region construction for simplicity. Fig. 1 depicts an example of a diagonal-free timed automaton.

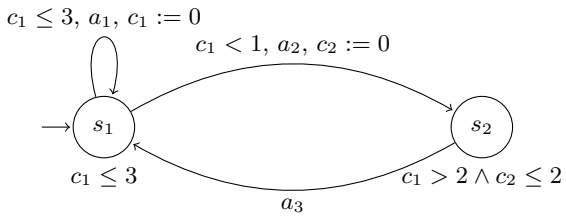

Fig. 1: Example of a diagonal-free timed automaton with two clocks.

Locations and transitions are guarded with *clock constraints*, which have to be fulfilled to stay in a location or to transition between them. The variants of these constraints are modeled by

datatype $('c, 't)$ *cconstraint* =
AND $(('c, 't)$ *cconstraint*$)$ $(('c, 't)$ *cconstraint*$)$ |
LT $'c$ $'t$ | *LE* $'c$ $'t$ | *EQ* $'c$ $'t$ | *GT* $'c$ $'t$ | *GE* $'c$ $'t$

where the atomic constraints in the second line represent the constraint $c \sim d$ for $\sim = <, \leq, =, >, \geq$, respectively. The sole difference to the full class of timed automata is that those would also allow constraints of the form $c_1 - c_2 \sim d$. We define a timed automaton A as a pair $(\mathcal{T}, \mathcal{I})$ where $\mathcal{I} :: 's \Rightarrow ('c, 't)$ *cconstraint* is an assignment of clock invariants to locations; $\mathcal{T}$ is a set of transitions written as $A \vdash l \longrightarrow^{g,a,r} l'$ where

- $l :: 's$ and $l' :: 's$ are start and successor location,

- $g :: ('c, 't)$ *cconstraint* is the guard of the transition,
- $a :: 'a$ is an action label,
- and $r :: 'c$ *list* is a list of clocks that will be reset to zero when the transition is taken.

Standard definitions of timed automata would include a fixed set of locations with a designated start location and a set of end locations. The language emptiness problem usually asks if any number of legal transitions can be taken to reach an end location from the start location. Thus we can confine ourselves to study reachability and implicitly assume the set of locations to be given by the transitions of the automaton. Note that although the definition of clock constraints allows constants from the whole time space, we will later crucially restrict them to the natural numbers in order to obtain decidability.

2.2 Operational Semantics

We want to define an operational semantics for timed automata via an inductive relation. States of timed automata are pairs of a location and a *clock valuation* of type $'c \Rightarrow 't$ assigning time values to clocks. Time lapse is modeled by shifting a clock valuation u by a constant value d: $u \oplus d = (\lambda x.\ u\ x + d)$. Finally, we connect clock valuations and constraints by writing, for instance, $u \vdash AND\ (LT\ c_1\ 1)\ (EQ\ c_2\ 2)$ if $u\ c_1 < 1$ and $u\ c_2 = 2$. The precise definition is standard.

Using these definitions, the operational semantics can be defined as a relation between pairs of locations and clock valuations. More specifically, we define *action steps*

$$\frac{A \vdash l \longrightarrow^{g,a,r} l' \wedge u \vdash g \wedge u' \vdash \textit{inv-of}\ A\ l' \wedge u' = [r \rightarrow 0]u}{A \vdash \langle l, u\rangle \rightarrow_a \langle l', u'\rangle}$$

and *delay steps* via $\dfrac{u \vdash \textit{inv-of}\ A\ l \wedge u \oplus d \vdash \textit{inv-of}\ A\ l \wedge 0 \leq d}{A \vdash \langle l, u\rangle \rightarrow^d \langle l, u \oplus d\rangle}$. Here *inv-of* $(\mathcal{T}, \mathcal{I}) = \mathcal{I}$ and the notation $[r \rightarrow 0]u$ means that we update the clocks in r to 0 in u. We write $A \vdash \langle l, u\rangle \rightarrow \langle l', u'\rangle$ if either $A \vdash \langle l, u\rangle \rightarrow_a \langle l', u'\rangle$ or $A \vdash \langle l, u\rangle \rightarrow^d \langle l', u'\rangle$.

2.3 Zone Semantics

The first conceptual step to get from this abstract operational semantics towards concrete algorithms on DBMs is to consider *zones*. Informally, the concept is simple; a zone is the set of clock valuations fulfilling a clock constraint: $('c, 't)$ *zone* $\equiv ('c \Rightarrow 't)$ *set*. This allows us to abstract from a concrete state $\langle l, u\rangle$ to a pair of location and zone $\langle l, Z\rangle$. We need the following operations on zones:

$$Z^{\uparrow} = \{u \oplus d \mid u \in Z \wedge 0 \leq d\} \text{ and } Z_{r \rightarrow 0} = \{[r \rightarrow 0]u \mid u \in Z\}.$$

Naturally, we define a zone-based semantics by means of another inductive relation:

$$\frac{}{A \vdash \langle l,\, Z\rangle \rightsquigarrow \langle l,\, (Z \cap \{u \mid u \vdash \mathit{inv\text{-}of}\ A\ l\})^{\uparrow} \cap \{u \mid u \vdash \mathit{inv\text{-}of}\ A\ l\}\rangle}$$

$$\frac{A \vdash l \longrightarrow^{g,a,r} l'}{A \vdash \langle l,\, Z\rangle \rightsquigarrow \langle l',\, (Z \cap \{u \mid u \vdash g\})_{r\,\rightarrow\,0} \cap \{u \mid u \vdash \mathit{inv\text{-}of}\ A\ l'\}\rangle}$$

With the help of two easy inductive arguments one can show soundness and completeness of this semantics w.r.t. the original semantics (where * is the *Kleene star* operator):

(Sound) $A \vdash \langle l,\, Z\rangle \rightsquigarrow^* \langle l',\, Z'\rangle \wedge u' \in Z' \Longrightarrow \exists\, u{\in}Z.\ A \vdash \langle l,\, u\rangle \rightarrow^* \langle l',\, u'\rangle$
(Complete) $A \vdash \langle l,\, u\rangle \rightarrow^* \langle l',\, u'\rangle \wedge u \in Z$
$\Longrightarrow \exists\, Z'.\ A \vdash \langle l,\, Z\rangle \rightsquigarrow^* \langle l',\, Z'\rangle \wedge u' \in Z'$

This is an example of where proof assistants really shine. Not only are our Isabelle proofs shorter to write down than for example the proof given in [18] – we have also found that the less general version given there (i.e. where $Z = \{u\}$) yields an induction hypothesis that is not strong enough in the completeness proof. This slight lapse is hard to detect in a human-written proof.

3 Difference Bound Matrices

3.1 Fundamentals

Difference Bound Matrices constrain differences of clocks (or more precisely, the difference of values assigned to individual clocks by a valuation). The possible constraints are given by:

$$\textbf{datatype}\ 't\ \mathit{DBMEntry} = \mathit{Le}\ 't \mid \mathit{Lt}\ 't \mid \infty$$

This yields a simple definition of DBMs: $'t\ \mathit{DBM} \equiv \mathit{nat} \Rightarrow \mathit{nat} \Rightarrow 't\ \mathit{DBMEntry}$. To relate clocks with rows and columns of a DBM, we use a numbering $v :: 'c \Rightarrow \mathit{nat}$ for clocks. DBMs will regularly be accompanied by a natural number n, which designates the number of clocks constrained by the matrix. Although this definition complicates our formalization at times, we hope that it allows us to easily obtain executable code for DBMs while retaining a flexible "interface" for applications. To be able to represent the full set of clock constraints with DBMs, we add an imaginary clock **0**, which shall be assigned to 0 in every valuation. Zero column and row will always be reserved for **0** (i.e. $\forall\, c.\ v\ c > 0$). If necessary, we assume that v is an injection or surjection for indices less or equal to n. Informally, the zone $[M]_{v,n}$ represented by a DBM M is defined as

$$\{u \mid \forall\, c_1,\, c_2,\, d.\ v\ c_1,\, v\ c_2 \leq n \longrightarrow (M\ (v\ c_1)\ (v\ c_2) = \mathit{Lt}\ d \longrightarrow u\ c_1 - u\ c_2 < d) \wedge (M\ (v\ c_1)\ (v\ c_2) = \mathit{Le}\ d \longrightarrow u\ c_1 - u\ c_2 \leq d)\}$$

assuming that $v\ \mathbf{0} = 0$.

Example 1.

$$\begin{array}{c|ccc} & \mathbf{0} & c_1 & c_2 \\ \mathbf{0} & \infty & Lt\ (-3) & Le\ 0 \\ c_1 & \infty & \infty & \infty \\ c_2 & Le\ 4 & \infty & \infty \end{array} \qquad \begin{array}{c|ccc} & \mathbf{0} & c_1 & c_2 \\ \mathbf{0} & Le\ 0 & Lt\ (-3) & Le\ 0 \\ c_1 & \infty & Le\ 0 & \infty \\ c_2 & Le\ 4 & Lt\ 1 & Le\ 0 \end{array} \qquad \begin{array}{c|ccc} & \mathbf{0} & c_1 & c_2 \\ \mathbf{0} & \infty & Le\ 0 & Le\ 0 \\ c_1 & \infty & \infty & Lt\ (-3) \\ c_2 & \infty & Le\ 3 & Le\ 0 \end{array}$$

The left two DBMs both represent the zone described by the constraint $c_1 > 3 \wedge c_2 \leq 4$, while the DBM on the right represents the empty zone. [1]

To simplify the subsequent discussion, we will set $'c = nat$, $v = id$ and assume that the set of clocks of the automaton in question is $\{1..n\}$. We define an ordering relation $\prec$ on $'t\ DBMEntry$ by means of

$$\frac{a < b}{Le\ a \prec Le\ b} \qquad \frac{a < b}{Le\ a \prec Lt\ b} \qquad \frac{a < b}{Lt\ a \prec Lt\ b} \qquad \frac{a \leq b}{Lt\ a \prec Le\ b} \qquad \frac{}{Lt\ \_ \prec \infty} \qquad \frac{}{Le\ \_ \prec \infty}$$

and extend it to $\preceq$ in the obvious way. Observe that $\prec$ and $\preceq$ are total orders. Additionally, we get the following important ordering property of DBMs (by nearly automatic proof):

Lemma 1. $\forall\, i\ j.\ i \leq n \longrightarrow j \leq n \longrightarrow M\ i\ j \preceq M'\ i\ j \Longrightarrow [M]_{v,n} \subseteq [M']_{v,n}$

We can interpret DBMs as a graph with clocks as vertices and difference constraints as edges between them. To give a concrete meaning to this interpretation, we first define addition on DBM entries: $a \boxplus \infty = \infty$; $\infty \boxplus b = \infty$; and $(\sim_1 x) \boxplus (\sim_2 y) = \sim' (x + y)$ where $\sim' = Le$ if $\sim_1 = \sim_2 = Le$ and $\sim' = Lt$ if otherwise. Now the length of a path (of DBM indices representing clocks) defined by [2]

$$len\ M\ s\ t\ [] = M\ s\ t \text{ and } len\ M\ s\ t\ (w \cdot ws) = M\ s\ w \boxplus len\ M\ w\ t\ ws$$

gives the key to reasoning about this interpretation: for any $u \in [M]_{v,n}$ and i, j, xs with $set\ (i \cdot j \cdot xs) \subseteq \{0..n\}$,[3] we get $Lt\ (u\ i - u\ j) \prec len\ M\ i\ j\ xs$ via induction on xs. Setting $i = j$, we can immediately conclude that DBMs with negative cycles are always empty. In the following we will make use of a predicate expressing that a DBM does not contain any negative cycles which only consist of vertices less or equal to k for some k:

$cycle\text{-}free\text{-}up\text{-}to\ M\ k\ n \equiv$
$\forall\, i\ xs.\ i \leq n \wedge set\ xs \subseteq \{0..k\} \longrightarrow Le\ 0 \preceq len\ M\ i\ i\ xs$

We write $cycle\text{-}free\ M\ n$ if $cycle\text{-}free\text{-}up\text{-}to\ M\ n\ n$.

[1] We assume a default clock numbering, mapping c_i to index i, for our examples
[2] [] denotes the empty list and $x \cdot xs$ is a list constructed from head x and tail xs
[3] $set\ xs$ is the set of elements contained in xs

3.2 Operations

We define the necessary operations on DBMs to obtain a basic forward analysis algorithm for reachability.

Floyd-Warshall algorithm From Ex. 1 we can see that to be able to tell if two DBMs represent the same zone, we first need to put them into some *canonical* form. Formally, this canonical form is characterized by the following property:

$$\mathit{canonical}\ M\ n \equiv \forall\, i\ j\ k.\ i \leq n \wedge j \leq n \wedge k \leq n \longrightarrow M\ i\ k \preceq M\ i\ j \boxplus M\ j\ k$$

The key property of non-empty canonical DBMs is that we can find a valuation $u \in [M]_{v,n}$ with $u\ i - u\ j = d$ for any d between $-M\ j\ i$ and $M\ i\ j$, or equivalently:

Lemma 2. *Assume $\mathit{Le}\ d \preceq M\ i\ j$, $\mathit{Le}\ (-d) \preceq M\ j\ i$ for M with cycle-free $M\ n$, canonical $M\ n$, and $i, j \leq n$ with $i \neq j$. We define M' by setting $M'\ i\ j = \mathit{Le}\ d$ and $M'\ j\ i = \mathit{Le}\ (-d)$ and $M'\ i'\ j' = M\ i'\ j'$ for all (i',j') where $(i',j') \neq (i,j)$, (j,i). Then $[M']_{v,n} \subseteq [M]_{v,n}$ and cycle-free $M'\ n$.*

Proof. From Lemma 1, we get $[M']_{v,n} \subseteq [M]_{v,n}$. It remains to show that M' does not contain a negative cycle. Suppose there is one. Then we can also find a *smallest* negative cycle, which, without loss of generality, is of the form $\mathit{len}\ M'\ i\ i\ (j \cdot xs) \prec \mathit{Le}\ 0$ for some xs where $i, j \notin \mathit{set}\ xs$. This proof step is rather intricate in Isabelle. We use a function that explicitly computes smallest negative cycles. An inductive argument yields a result that allows us to rotate cycles. Now, we get $\mathit{Le}\ d \boxplus \mathit{len}\ M'\ j\ i\ xs \prec \mathit{Le}\ 0$. We have $xs \neq []$ as this would directly give us the contradiction $\mathit{Le}\ d \boxplus \mathit{Le}\ (-d) \prec \mathit{Le}\ 0$. This means that $\mathit{Le}\ d \boxplus \mathit{len}\ M\ j\ i\ xs \prec \mathit{Le}\ 0$ (by induction on xs), and because M is canonical, $M\ j\ i \prec \mathit{Le}\ (-d)$, which is a contradiction to our assumption. □

An important consequence is that any canonical DBM without a negative diagonal has at least one valuation, which we can construct by repeatedly applying the theorem. Observe that this also implies that a DBM in canonical form is empty iff there is a negative entry on its diagonal.

The canonical form can be computed by the Floyd-Warshall algorithm for the all-pairs shortest paths problem. A simple HOL formulation of the algorithm is

$$\mathit{fw\text{-}upd}\ M\ k\ i\ j \equiv M(i := (M\ i)(j := \mathit{min}\ (M\ i\ j)\ (M\ i\ k \boxplus M\ k\ j)))$$

$$\begin{aligned}
&\mathit{fw}\ M\ n\ 0\ 0\ 0 = \mathit{fw\text{-}upd}\ M\ 0\ 0\ 0\\
&\mathit{fw}\ M\ n\ (\mathit{Suc}\ k)\ 0\ 0 = \mathit{fw\text{-}upd}\ (\mathit{fw}\ M\ n\ k\ n\ n)\ (\mathit{Suc}\ k)\ 0\ 0\\
&\mathit{fw}\ M\ n\ k\ (\mathit{Suc}\ i)\ 0 = \mathit{fw\text{-}upd}\ (\mathit{fw}\ M\ n\ k\ i\ n)\ k\ (\mathit{Suc}\ i)\ 0\\
&\mathit{fw}\ M\ n\ k\ i\ (\mathit{Suc}\ j) = \mathit{fw\text{-}upd}\ (\mathit{fw}\ M\ n\ k\ i\ j)\ k\ i\ (\mathit{Suc}\ j)
\end{aligned}$$

where $f(a := b) \equiv \lambda x.\ \mathit{if}\ x = a\ \mathit{then}\ b\ \mathit{else}\ f\ x$. We abbreviate $\mathit{fw}\ M\ n\ n\ n\ n$ as $FW\ M\ n$. To prove that this algorithm computes the tightest difference constraint for all pairs of clocks, we claim:

Theorem 1.
cycle-free-up-to $M\ k\ n \wedge i' \leq i \wedge j' \leq j \wedge i \leq n \wedge j \leq n \wedge k \leq n \Longrightarrow$
$Min\ \{len\ M\ i'\ j'\ xs \mid set\ xs \subseteq \{0..k\} \wedge i' \notin set\ xs \wedge j' \notin set\ xs \wedge distinct\ xs\}$
$= fw\ M\ n\ k\ i\ j\ i'\ j'$

The proof is a nested induction, which follows the program structure and uses a standard argument. The theorem implies that FW computes a canonical form:

Corollary 1. *cycle-free* $M\ n \Longrightarrow$ *canonical* $(FW\ M\ n)\ n$

The Floyd-Warshall algorithm also *detects* negative cycles by computing a negative diagonal entry. The key observation is that a matrix of this kind either has a negative diagonal entry to start with, or there is a maximal $k < n$ with *cycle-free-up-to* $M\ k\ n$. The latter means that the algorithm computes a negative diagonal entry in iteration $k + 1$. In either case the negative diagonal entry will be preserved by monotonicity of the algorithm. This yields an emptiness check for DBMs.

Intersection The intersection of two DBMs is trivial to compute. It is simply the point-wise minimum: $And\ A\ B \equiv \lambda i\ j.\ min\ (A\ i\ j)\ (B\ i\ j)$. The operation is correct in the following sense: $[A]_{v,n} \cap [B]_{v,n} = [And\ A\ B]_{v,n}$. The $\subseteq$-direction can directly be proved by Isabelle's simplifier, while $\supseteq$ requires a rather lengthy proof by cases.

Reset We need an operator *reset* such that $u\ c = d$ for all $u \in [reset\ M\ n\ c\ d]_{v,n}$. Thus we define $(reset\ M\ n\ c\ d)\ c\ 0 = Le\ d$ and $(reset\ M\ n\ c\ d)\ 0\ c = Le\ (-d)$. By doing so, all difference constraints involving c are invalidated. Therefore we set the corresponding DBM entries to ∞. However, this alone does not yield a correct operation. Consider clocks c_1, c_2 and c_3 and a DBM represented by the clock constraint $c_1 \geq c_2 + 1 \wedge c_1 \leq c_3$. By setting c_1 to 0, we will lose all constraints on c_2 and c_3. This means that the resulting zone will contain a valuation u with $u\ c_1 = u\ c_2 = u\ c_3 = 0$. There is clearly no way to set c_1 back to a different value such the resulting valuation would satisfy the original constraint. The way to resolve this issue is to encode the information we had about c_2 and c_3 in the original constraint (or DBM) also in the new DBM. This is, we derive $c_2 - c_3 \leq -1$. Concretely, we calculate $(reset\ M\ n\ i\ d)\ j\ k = min\ (M\ j\ i + M\ i\ k)\ (M\ j\ k)$ for all j, $k \leq n$. Note that this computation does nothing if M is already in canonical from, allowing a simpler implementation.

For a list of clocks *cs* and a list of time stamps *ts* ($|cs| = |ts|$), *set-clocks cs ts u* is the valuation for which $(set\text{-}clocks\ cs\ ts\ u)\ cs_i = ts_i$ and the value of $u\ c$ is unchanged for all other clocks $c \notin set\ cs$. We lift *reset* to reset many clocks at once by simply folding it over the list of clocks. We proved correctness of the lifted operation ($reset'$):

(Sound) $(\forall\, c{\in}set\ cs.\ 0 < c \wedge c \leq n) \wedge u \in [reset'\ M\ n\ cs\ v\ d]_{v,n}$
$\Longrightarrow \exists\, ts.\ set\text{-}clocks\ cs\ ts\ u \in [M]_{v,n}$

(Complete) $(\forall\, c{\in}set\ cs.\ 0 < c \wedge c \leq n) \wedge u \in [M]_{v,n}$
$\Longrightarrow [cs{\rightarrow}d]u \in [reset'\ M\ n\ cs\ v\ d]_{v,n}$

The proofs for these results are among the most complex ones in the whole formalization. The reason is that manual case analyses have to be combined with (linear) arithmetic reasoning, which is hard to automate in Isabelle.

Delay We need an operation to compute time lapse, i.e. $([M]_{v,n})^{\uparrow}$. For canonical DBMs, this simply amounts to setting $M\ i\ 0 = \infty$ for all $i \leq n$. In the general case, intuitively we can lose information about the difference of two clocks that was recorded between the upper bound of one of them and the lower bound of the other. Accounting for this, we arrive at the following general operation:

$up\ M \equiv$
$\lambda i\ j.$ *if* $0 < i$ *then if* $j = 0$ *then* ∞ *else* $min\ (M\ i\ 0 \boxplus M\ 0\ j)\ (M\ i\ j)$ *else* $M\ i\ j$

Correctness can be obtained similarly to the reset operation.

Abstraction It is easy to turn an atomic clock constraint into a DBM that represents the same zone. For instance, the zone $\{u \mid u \vdash EQ\ c\ d\}$ is represented by a DBM M where $M\ c\ 0 = Le\ d$ and $M\ 0\ c = Le\ (-d)$, and all other entries are unbounded. Using the already defined intersection operation for constructor *AND*, a function *abstr*, which records entries in this manner while working recursively through a constraint, turns constraints into a DBM-equivalent. Again, we proved correctness (where *collect-clks cc* is the set of all clocks appearing in constraint *cc*):

$\forall\, c{\in}collect\text{-}clks\ cc.\ 0 < c \wedge c \leq n \Longrightarrow [abstr\ cc\ (\lambda i\ j.\ \infty)\ v]_{v,n} = \{u \mid u \vdash cc\}$

3.3 DBM Operational Semantics

In the last section we have elaborated the adequacy of our DBM-equivalents for all zone operations, allowing us to compute the zone semantics with the help of DBMs. Indeed we can define a new operational semantics based on DBMs:

$$\frac{M_i = abstr\ (inv\text{-}of\ A\ l)\ (\lambda i\ j.\ \infty)\ v}{A \vdash \langle l,\ M\rangle \rightsquigarrow_{v,n} \langle l,\ And\ (up\ (And\ M\ M_i))\ M_i\rangle}$$

$$\frac{A \vdash l \longrightarrow^{g,a,r} l' \wedge M_i = abstr\ (inv\text{-}of\ A\ l')\ (\lambda i\ j.\ \infty)\ v}{A \vdash \langle l,\ M\rangle \rightsquigarrow_{v,n} \langle l',\ And\ (reset'\ (And\ M\ (abstr\ g\ (\lambda i\ j.\ \infty))\ v))\ n\ r\ v\ 0)\ M_i\rangle}$$

Using the correctness results for the DBM operations, it is straightforward to show that this semantics is equivalent to the zone semantics:

$$A \vdash \langle l,\ [M]_{v,n}\rangle \rightsquigarrow^* \langle l',\ Z\rangle$$
$$\longleftrightarrow \exists\, M'.\ A \vdash \langle l,\ M\rangle \rightsquigarrow^*{}_{v,n} \langle l',\ M'\rangle \wedge Z = [M']_{v,n}$$

However, we are not done yet: while we can practically compute the semantics of timed automata, the search space could still be infinite. The rest of the paper is concerned with overcoming this problem.

4 From Classic Decidability to a Correct Approximation

4.1 Regions

In their seminal paper, Alur and Dill showed decidability of the emptiness problem for timed automata by giving an adequate finite partitioning of the set of valuations into what they call *regions*. In this section, we will present our formalization of this result and then show how to apply it to obtain a *finite* operational semantics of zones. We use Bouyer's definition of regions as, for one it is more formal and thus easier to formalize, and secondly we will have to use a modified version of it later on.

From now on we will work in a parametric theory (called *locale* in Isabelle), which fixes X as the set of clocks of the automaton. Moreover, a *clock ceiling* k will define an upper bound $k\ c$ for the "relevant" range of any clock $c \in X$ – this ought to correspond to the *maximal* constant appearing for c in any constraint of the timed automaton, e.g., $k\ c_1 = 3$ and $k\ c_2 = 2$ for the automaton of Fig. 1. This is, if $\sim\ c\ m$ is a constraint of the automaton, we postulate that $m \leq k\ c$, $c \in X$, and that m is a natural number.

A single clock value will always fall into one of three types of intervals from

$$\textbf{datatype}\ \mathit{intv} = \mathit{Const}\ \mathit{nat} \mid \mathit{Intv}\ \mathit{nat} \mid \mathit{Greater}\ \mathit{nat}$$

where the set of values they contain is given by the following rules:

$$\frac{u\ x = d}{\mathit{intv\text{-}elem}\ x\ u\ (\mathit{Const}\ d)} \qquad \frac{d < u\ x \wedge u\ x < d + 1}{\mathit{intv\text{-}elem}\ x\ u\ (\mathit{Intv}\ d)}$$

$$\frac{d < u\ x}{\mathit{intv\text{-}elem}\ x\ u\ (\mathit{Greater}\ d)}$$

Let $I :: {'c} \Rightarrow \mathit{intv}$ be assigning intervals to clocks and r be a finite total preorder over $X_0 \equiv \{x \in X \mid \exists\, d.\ I\ x = \mathit{Intv}\ d\}$. Then we define the corresponding region $\mathit{region}\ X\ I\ r$ as the set for which [4]

$$\begin{aligned} u \in \mathit{region}\ X\ I\ r \text{ iff} \quad & \forall\, x{\in}X.\ 0 \leq u\ x \wedge \mathit{intv\text{-}elem}\ x\ u\ (I\ x) \\ & \text{and } \forall\, x{\in}X_0.\ \forall\, y{\in}X_0.\ (x,\ y) \in r \longleftrightarrow \mathit{frac}\ (u\ x) \leq \mathit{frac}\ (u\ y) \end{aligned}$$

We will fix a set of regions $\mathcal{R}_\alpha \equiv \{\mathit{region}\ X\ I\ r \mid \mathit{valid\text{-}region}\ X\ k\ I\ r\}$ where $\mathit{valid\text{-}region}\ X\ k\ I\ r$ holds if X is finite, r is a total preorder on X_0, and $d \leq k\ x$ if $I\ x = \mathit{Const}\ d$, $d < k\ x$ if $I\ x = \mathit{Intv}\ d$, and $k\ x = d$ if $I\ x = \mathit{Greater}\ d$ for all $x \in X$. Observe that this definition remedies the potential overlap of intervals that the definition of *intv-elem* would admit.

It is clear from Fig. 1, and relatively straightforward to prove in Isabelle/HOL, that $\mathcal{R}_\alpha$ is a finite partitioning of

$$V \equiv \{u \mid \forall\, x{\in}X.\ 0 \leq u\ x\}\ ,$$

the set of all positive valuations. What is not so obvious (and not mentioned by Bouyer) but a useful property to work with, is that any valid region is also non-empty. The crux of this proof is to observe that X_0 can be ordered in equivalence

[4] *frac* r denotes the fractional part of any real number r

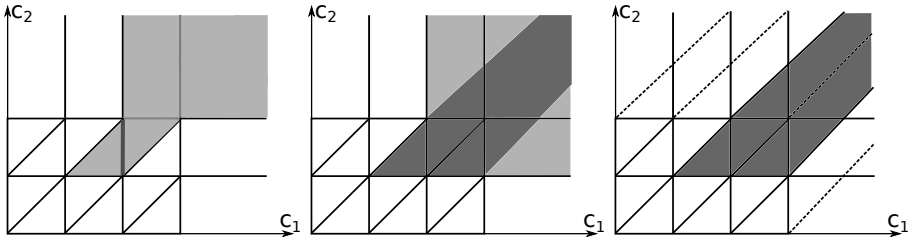

Fig. 2: (1) A region and its time successors in $\mathcal{R}_\alpha$, (2) the α-closure of a zone, and (3) the β-approximation of a zone for $X = \{c_1, c_2\}$ with $k\ c_1 = 3$ and $k\ c_2 = 2$.

classes according to r such that a valuation u can be chosen for which $frac\ (u\ x) \leq frac\ (u\ y)$ iff $(x, y) \in r$. This ordering property of finite total preorders is non-trivial to formalize and makes this step rather technical.

4.2 Decidability with Regions

How are regions and timed automata connected? We will present three key properties that connect regions to time lapse, clock resets, and clock constraints, respectively, allowing us to implement timed automata with the help of regions. Let $[u]_{\mathcal{R}_\alpha} \in \mathcal{R}_\alpha$ be the unique region containing u. We call $[u \oplus t]_{\mathcal{R}\alpha}$ a *time successor* of $[u]_{\mathcal{R}\alpha}$ for $t \geq 0$ and denote by $Succ\ \mathcal{R}_\alpha\ R$ the set of all such time successors of all $u \in R$ (cf. Fig. 2.1). Now the three key properties are in order of decreasing difficulty:

(Set of regions) $R \in \mathcal{R}_\alpha \wedge u \in R \wedge R' \in Succ\ \mathcal{R}_\alpha\ R$
$\Longrightarrow \exists\, t{\geq}0.\ [u \oplus t]_{\mathcal{R}_\alpha} = R'$

(Compatibility with resets) $R \in \mathcal{R}_\alpha \wedge u \in R \wedge 0 \leq d \wedge d \leq k\ x \wedge x \in X$
$\Longrightarrow [u(x := d)]_{\mathcal{R}_\alpha} = \{u(x := d) \mid u \in R\}$

(Compatibility with constraints)
$R \in \mathcal{R}_\alpha \wedge \forall\,(x, m){\in}collect\text{-}clock\text{-}pairs\ cc.\ m \leq k\ x \wedge x \in X \wedge m \in \mathbb{N}$
$\Longrightarrow R \subseteq \{u \mid u \vdash cc\} \vee \{u \mid u \vdash cc\} \cap R = \emptyset$

Proof. We concentrate on the set of regions property as it has the most interesting formalization. Our proof combines elements of the "classic" result as presented e.g., in [9], and Bouyer's approach. Let $R = region\ X\ I\ r \in \mathcal{R}_\alpha$ for some I, r, let $R' = [v \oplus t]_{\mathcal{R}_\alpha}$, and assume $u, v \in R$ and $t \geq 0$. If $I\ x = Greater\ (k\ x)$ for all $x \in X$ ("upper-right region"), we have $Succ\ \mathcal{R}_\alpha\ R = \{R\} = \{R'\}$ and the proposition is obvious.

Otherwise observe that there exists a single *closest* successor R_{succ} of R (depicted as the thick, dark gray line in Fig. 2.1). We refer to Bouyer for a formal construction of this successor. We can show the characteristic property of this closest successor:

$$\forall\, u \in R.\ \forall\, t{\geq}0.\ (u \oplus t) \notin R \longrightarrow (\exists\, t'{\leq}t.\ (u \oplus t') \in R_{succ} \wedge t' \geq 0)$$

At this point Bouyer states that the proposition follows by "immediate induction". However, regarding formalization, this induction is not quite immediate. For instance, we attempted induction on the set of successors. This necessitates a proof that this set is monotone, which we did not find ourselves able to prove without asserting the very property we were about to prove. Instead, we split the argument in two: one for the case where $t < 1$ and the other for the case where t is an integer. For the first case, consider the "critical" set $C = \{x \in X \mid \exists\, d.\ I\ x = Intv\ d \wedge d + 1 \leq u\ x + t\}$, the set of clocks for which $u \oplus t$ is shifted beyond R's interval boundaries. Observe that for the closest successor, the critical set is either the same (if $\{x \in X \mid \exists\, d.\ I\ x = Const\ d\} \neq \emptyset$) or a strict subset (if otherwise). Thus the proposition follows by induction on the cardinality of C. The case where t is an integer follows by direct proof over the structure of regions. Shifting u first by *frac* t and then by $\lfloor t \rfloor$, we arrive at the proposition. □

This allows us to define a region-based operational semantics for timed automata:

$$\frac{R \in \mathcal{R}_\alpha \wedge R' \in Succ\ \mathcal{R}_\alpha\ R \wedge R \cup R' \subseteq \{u \mid u \vdash inv\text{-}of\ A\ l\}}{A,\mathcal{R}_\alpha \vdash \langle l,\ R\rangle \rightsquigarrow \langle l,\ R'\rangle}$$

$$\frac{A \vdash l \longrightarrow^{g,a,r} l' \wedge R \in \mathcal{R}_\alpha}{A,\mathcal{R}_\alpha \vdash \langle l,\ R\rangle \rightsquigarrow \langle l',\ \{[r\rightarrow 0]u \mid u \in R \wedge u \vdash g\} \cap \{u \mid u \vdash inv\text{-}of\ A\ l'\}\rangle}$$

From the aforementioned properties, we proved its adequacy w.r.t. to reachability:

$$\begin{array}{c} A,\mathcal{R}_\alpha \vdash \langle l,[u]_{\mathcal{R}_\alpha}\rangle \rightsquigarrow^* \langle l',R'\rangle \wedge R' \neq \emptyset \\ \longleftrightarrow \exists\, u'.\ A \vdash \langle l,\ u\rangle \rightarrow^* \langle l',\ u'\rangle \wedge [u']_{\mathcal{R}_\alpha} = R' \end{array}$$

Note that it is quite natural that this property is weaker compared to previous ones: (sets of) regions only approximate zones and thus can contain valuations that were never reachable in the concrete semantics.

4.3 Approximating Zone Semantics with Regions

From the pure decidability result on regions, we now move back towards zones by *approximating* zones with the smallest set of regions that covers them. Formally we define the α-*closure* of a zone Z: $Closure_\alpha\ Z = \bigcup\{R \in \mathcal{R} \mid R \cap Z \neq \emptyset\}$. Observe that this set need not be convex (cf. Fig. 2.2). We use the α-closure to define an operational semantics on zones that approximates a zone with its α-closure at the end of each step:

$$A \vdash \langle l,\ Z\rangle \rightsquigarrow \langle l',\ Z'\rangle \Longrightarrow A \vdash \langle l,\ Z\rangle \rightsquigarrow_\alpha \langle l',\ Closure_\alpha\ Z'\rangle$$

Bouyer would now go and prove from the region properties that the α-closure can be "pushed through" each step:

$$Z \subseteq V \wedge A \vdash \langle l, \mathit{Closure}_\alpha\ Z\rangle \rightsquigarrow \langle l', Z'\rangle$$
$$\Longrightarrow \exists\, Z''.\ A \vdash \langle l, Z\rangle \rightsquigarrow_\alpha \langle l', Z''\rangle \wedge Z' \subseteq Z''$$

However, we did not find this property strong enough to prove soundness of ${\rightsquigarrow_\alpha}^*$:

$$A \vdash \langle l, Z\rangle \rightsquigarrow_\alpha{}^* \langle l', Z'\rangle \wedge Z \subseteq V$$
$$\Longrightarrow \exists\, Z''.\ A \vdash \langle l, Z\rangle \rightsquigarrow^* \langle l', Z''\rangle \wedge \mathit{Closure}_\alpha\ Z' \subseteq \mathit{Closure}_\alpha\ Z'' \wedge Z'' \subseteq Z'$$

Note that this property is really what one wants to have since $\mathit{Closure}_\alpha\ Z = \emptyset$ iff $Z = \emptyset$ (assuming that $Z \subseteq V$). We conceived that instead it is sufficient to prove monotonicity of the α-closure w.r.t. to steps in the zone semantics:

$$A \vdash \langle l, Z\rangle \rightsquigarrow \langle l', Z'\rangle \wedge \mathit{Closure}_\alpha\ Z = \mathit{Closure}_\alpha\ W \wedge W \subseteq Z \wedge Z \subseteq V$$
$$\Longrightarrow \exists\, W'.\ A \vdash \langle l,\ W\rangle \rightsquigarrow \langle l',\ W'\rangle \wedge \mathit{Closure}_\alpha\ Z' = \mathit{Closure}_\alpha\ W' \wedge W' \subseteq Z'$$

Combining this with the fact that α-closure is an involution, we proved soundness by induction over ${\rightsquigarrow_\alpha}^*$. Completeness follows easily from monotonicity of $\rightsquigarrow^*$:

$$A \vdash \langle l, Z\rangle \rightsquigarrow^* \langle l', Z'\rangle \wedge Z \subseteq V \wedge Z' \neq \emptyset$$
$$\Longrightarrow \exists\, Z''.\ A \vdash \langle l, Z\rangle \rightsquigarrow_\alpha{}^* \langle l', Z''\rangle \wedge Z' \subseteq Z''$$

While these results are nice from a theoretical standpoint, it is not easier to compute the α-closure than to directly implement timed automata with the region construction presented in the last section. Therefore, the next section will present Bouyer's main insight – that these results can be used to show the correctness of an easily computable approximation operation.

5 Normalization

Consider Fig. 2.3. In addition to $\mathcal{R}_\alpha$ (solid lines), the figure shows a refinement to what we will call $\mathcal{R}_\beta$ (dashed lines). Observe that the smallest set of regions covering the zone painted in dark gray (i.e. its β-closure) is *convex*, whereas its α-closure is not (cf. Fig. 2.2). The idea is to use this β-closure to obtain an effectively computable convex approximation for zones represented by DBMs – DBMs always represent a convex zone and are always covered by a convex β-closure – while inheriting the correctness result from the α-closure as we only refine things.

5.1 β-approximation

Due to a lack of space, we do not present our construction of $\mathcal{R}_\beta$ and only say that it is can be adopted from $\mathcal{R}_\alpha$ with some modifications. Note that we do not need to transfer the (rather intricate) properties connecting $\mathcal{R}_\alpha$ with

transitions of timed automata since we will infer correctness directly from the original construction.

We now want to formalize the notion of a convex approximation of zones with regions from $\mathcal{R}_\beta$. We capture the notion of convexity directly with DBMs. From Example 1, we can see that the types of regions in $\mathcal{R}_\beta$ also induce a specific format for our DBMs: for a DBM entry $M\ i\ j$, we do not need constants outside of $[-\ k\ i;\ k\ j]$ because this is precisely the range to which our regions bound the corresponding values (analogously for constraints involving $\mathbf{0}$). Thus we use the following notion of *normalized* DBMs:

$$\begin{aligned}
&\mathit{normalized}\ M \equiv \\
&(\forall\, i\ j.\ 0 < i \wedge i \leq n \wedge 0 < j \wedge j \leq n \wedge M\ i\ j \neq \infty \longrightarrow \\
&\qquad \mathit{Lt}\ (-\ k\ j) \preceq M\ i\ j \wedge M\ i\ j \preceq \mathit{Le}\ (k\ i)) \wedge \\
&(\forall\, i{\leq}n.\ 0 < i \longrightarrow (M\ i\ 0 \preceq \mathit{Le}\ (k\ i) \vee M\ i\ 0 = \infty) \wedge \mathit{Lt}\ (-\ k\ i) \preceq M\ 0\ i)
\end{aligned}$$

Furthermore, all constraints only need to use integer constants, which we denote by *dbm-int M*. Building from these ideas, we define for any zone Z:

$$\begin{aligned}
\mathit{Approx}_\beta\ Z \equiv \bigcap \{[M]_{v,n} \mid \\
\exists U \subseteq \mathcal{R}_\beta.\ [M]_{v,n} = \bigcup U \wedge Z \subseteq [M]_{v,n} \wedge \mathit{dbm\text{-}int}\ M\ n \wedge \mathit{normalized}\ M\}
\end{aligned}$$

5.2 Connecting Approx_β and $\mathit{Closure}_\alpha$

We already argued that is possible to inherit correctness from $\mathit{Closure}_\alpha$ because we only refine regions. Precisely, Bouyer proposed that for any convex zone Z (i.e. $Z = [M]_{v,n}$ for some DBM M), we have $\mathit{Approx}_\beta\ Z \subseteq \mathit{Closure}_\alpha\ Z$, or equivalently:

Theorem 2. $R \in \mathcal{R}_\alpha \wedge Z \subseteq V \wedge R \cap Z = \emptyset \wedge Z = [M]_{v,n} \wedge \mathit{dbm\text{-}int}\ M\ n \implies R \cap \mathit{Approx}_\beta\ Z = \emptyset$

The formalization of Bouyer's proof for this proposition is one of the most complicated parts of our development. As the prose proof is already sufficiently complicated, we abstain from presenting our formalization of this result.

Analogously to $\rightsquigarrow_\alpha$, we define an approximating semantics $\rightsquigarrow_\beta$ using Approx_β. The main fact we can derive from the Theorem 2 is that $\rightsquigarrow_\alpha$ is an approximation of $\rightsquigarrow_\beta$:

Lemma 3.
$A \vdash \langle l,\ [M]_{v,n}\rangle \rightsquigarrow_\beta \langle l',\ Z'\rangle \wedge \mathit{dbm\text{-}int}\ M\ n \wedge [M]_{v,n} \subseteq W \wedge W \subseteq V$
$\implies \exists\, W'.\ A \vdash \langle l,\ W\rangle \rightsquigarrow_\alpha \langle l',\ W'\rangle \wedge Z' \subseteq W'$

Using this result and some additional work, we could infer soundness and completeness of ${\rightsquigarrow_\beta}^*$ from the corresponding results for ${\rightsquigarrow_\alpha}^*$.

5.3 Computing $Approx_\beta$

So far, we have shown how to obtain a correct approximation operation from $\mathcal{R}_\beta$, which only produces convex sets. The huge gain from that is that this approximation can also be easily computed by *normalizing* DBMs:

$norm\ M\ k\ n \equiv$
$\lambda i\ j.\ \mathsf{let}\ ub = \mathsf{if}\ 0 < i\ \mathsf{then}\ k\ i\ \mathsf{else}\ 0;\ lb = \mathsf{if}\ 0 < j\ \mathsf{then} - k\ j\ \mathsf{else}\ 0$
$\quad \mathsf{in}\ \mathsf{if}\ i \leq n \wedge j \leq n\ \mathsf{then}\ norm\text{-}lower\ (norm\text{-}upper\ (M\ i\ j)\ ub)\ lb$
$\quad\quad \mathsf{else}\ M\ i\ j$
$norm\text{-}upper\ e\ t = (\mathsf{if}\ Le\ t \prec e\ \mathsf{then}\ \infty\ \mathsf{else}\ e)$
$norm\text{-}lower\ e\ t = (\mathsf{if}\ e \prec Lt\ t\ \mathsf{then}\ Lt\ t\ \mathsf{else}\ e)$

Lemma 4. *canonical* $M\ n \wedge [M]_{v,n} \subseteq V \wedge$ *dbm-int* $M\ n \Longrightarrow$
$Approx_\beta\ ([M]_{v,n}) = [norm\ M\ k\ n]_{v,n}$

Again, we abstain from providing a full presentation of our formalization and only mention that the main ideas are: (1) to observe that normalized integral DBMs can always be represented by an equivalent subset of $\mathcal{R}_\beta$, and (2) that $norm\ M\ k\ n$ computes a minimal normalized DBM.

5.4 A Final Semantics

We have assembled all the ingredients to define a semantics for timed automata which captures the essence of what DBM-based model checkers compute:

$$A \vdash \langle l,\ D\rangle \rightsquigarrow_{v,n} \langle l',\ D'\rangle \Longrightarrow A \vdash \langle l,\ D\rangle \rightsquigarrow_{\mathcal{N}} \langle l',\ norm\ (FW\ D'\ n)\ k\ n\rangle$$

Combining the fact that β-approximation is computable and the correctness properties of ${\rightsquigarrow_\beta}^*$ and ${\rightsquigarrow}^*$, we have achieved our main result: a timed automaton can reach a certain location l' iff we can compute a valid run (using the DBM operations and normalization) that ends in l'.

Theorem 3. $Z = [M]_{v,n} \wedge Z \subseteq V \wedge$ *dbm-int* $M\ n \Longrightarrow$
$(\exists\, u{\in}Z.\ \exists\, u'.\ A \vdash \langle l,\ u\rangle \rightarrow^* \langle l',\ u'\rangle)$
$\longleftrightarrow (\exists\, M'.\ A \vdash \langle l,\ M\rangle \rightsquigarrow_{\mathcal{N}}{}^* \langle l',\ M'\rangle \wedge [M']_{v,n} \neq \emptyset)$

6 Conclusion

We have presented a formalization that, beginning with basic definitions and classic results, closes the loop to show correcntess of the basic DBM-based algorithms that are used in forward analysis of timed automata. However, we have not yet harvested potential practical fruits of this development. A self-evident goal is to obtain an executable version for the algorithms above. By combination with a verified version of e.g., depth-first search, this could already yield a verified tool for deciding language emptiness of timed automata, which could in turn be extended to a fully verified model checker. In another direction of development, the author has already started to reuse the presented formalization to formalize first results about decidability of probabilistic timed automata.

Acknowledgement. I would like to thank Tobias Nipkow and the anonymous reviewers for their helpful comments on earlier versions of this paper.

References

1. R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of ICALP'90*, LNCS 443, pages 322–335, 1990.
2. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proc. of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, pages 592–601, 1993.
4. J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lectures on Concurrency and Petri Nets: Advances in Petri Nets*, LNCS 3098, pages 87–124, 2004.
5. P. Bouyer. Untameable timed automata! In *Proc. of STACS'03*, LNCS 2607, pages 620–631, 2003.
6. P. Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.
7. P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *Proc. of CAV'00*, LNCS 1855, pages 464–479, 2000.
8. P. Castéran and D. Rouillard. Towards a generic tool for reasoning about labeled transition systems. In *TPHOLs 2001: Supplemental Proceedings*, 2001. http://www.informatics.ed.ac.uk/publications/report/0046.html.
9. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, 2001.
10. D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, LNCS 407, pages 197–212. 1990.
11. M. Garnacho, J. Bodeveix, and M. Filali-Amine. A mechanized semantic framework for real-time systems. In *Proc. of FORMATS'13*, LNCS 8053, pages 106–120, 2013.
12. T. A. Henzinger, P.-H. Ho, and H. Wong-toi. Hytech: A model checker for hybrid systems. *Software Tools for Technology Transfer*, 1(1):460–463, 1997.
13. G. K. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *Software Tools for Technology Transfer*, 1(1):134–152, 1997.
14. C. Paulin-Mohring. Modelisation of timed automata in Coq. In *Proc. of STACS'01*, LNCS 2215, pages 298–315, 2001.
15. S. Wimmer. Timed automata. *Archive of Formal Proofs*, Mar. 2016. http://isa-afp.org/entries/Timed_Automata.shtml, Formal proof development.
16. Q. Xu and H. Miao. Formal verification framework for safety of real-time system based on timed automata model in PVS. In *Proc. of the IASTED International Conference on Software Engineering, 2006*, pages 107–112, 2006.
17. Q. Xu and H. Miao. Manipulating clocks in timed automata using PVS. In *Proc. of SNPD'09*, pages 555–560, 2009.
18. W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In *Proc. of Formal Description Techniques VII*, pages 243–258, 1994.
19. S. Yovine. KRONOS: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1):123–133, 1997.