Extending Timed Automata with Clock Derivatives

David Cortés[1], Jean Leneutre[1], Vadim Malvone[1], James Ortiz[1,2], and Pierre-Yves Schobbens[3]

[1] LTCI, Institut Polytechnique de Paris, Télécom Paris, France
{david.cortes, jean.leneutre, vadim.malvone, james.ortizvega}@telecom-paris.fr
[2] Université Paris Est Creteil, LACL, F-94010 Creteil, France
{james.ortiz-vega}@u-pec.fr
[3] University of Namur, NADI, Namur, Belgium
pierre-yves.schobbens@unamur.be

Abstract. The increasing complexity of safety-critical systems in domains like aerospace, robotics, and industrial control demands precise modeling and verification methods. While Timed Automata (TA) and Distributed Timed Automata (DTA) are standard formalisms for real-time systems, they assume synchronized clocks or lack the expressiveness to capture clock drift and indirect timing dependencies. To overcome these limits, we propose Timed Automata with Clock Derivatives (idTA), extending TA with rate constraints to model independent clock evolution. We also introduce DL_ν, a temporal logic over Multi-Timed Labeled Transition Systems (MLTS), capturing properties of systems with unsynchronized clocks. We show that model checking for DL_ν is EXPTIME-complete. Finally, we present MIMETIC, a model checking tool supporting idTA and DL_ν, providing a platform for analyzing clock interactions and verification of Distributed Real-time Systems (DRTS).

Keywords: Timed Automata, Reachability, Distributed Timed Systems.

1 Introduction

Distributed Real-Time Systems (DRTS) are foundational to many critical applications, including traffic light coordination, avionics, telecommunication infrastructures, and medical monitoring devices. These systems consist of spatially distributed components that operate concurrently under strict timing requirements. An essential aspect of DRTS is how time is managed across components. Two primary approaches exist: *synchronous semantics* and *asynchronous semantics*. In synchronous semantics, all clocks in the system evolve at the same uniform rate, assuming perfect synchronization akin to a global Newtonian time. In contrast, asynchronous semantics allow clocks to evolve at different rates, enabling components to operate independently and adaptively. Numerous real-time formalisms have been proposed to model and verify DRTS.While Timed

Automata (TA) [4] have proven effective in modeling real-time systems, they assume perfectly synchronized clocks, which is often unrealistic in distributed settings. Extensions such as Distributed Timed Automata (DTA) [19] and Timed Automata with Independent Clocks (icTA) [1], and Distributed Event Clock Automata (DECA)[24] address this by allowing clocks to evolve independently. Hybrid Automata (HA) extend TA by incorporating both discrete transitions and continuous dynamics [27]. There are also several temporal logic formalisms, such as Timed μ-calculus [15], L_ν [20], and TCTL [29]. These logics are used to specify sequential timed properties governed by timing constraints. Other logics, such as DRTL [22], APTL [31], DECTL [24], and ML_ν [23], have been developed to capture the timing properties of distributed components in DRTS.

However, these models still fall short in expressing relative rate constraints or indirect clock dependencies, a common requirement in many real-world DRTS scenarios [18][28]. To illustrate, consider two sensors, A and B, each with its own local clock x and y. Sensor A samples every 2 seconds (w.r.t. x), and B must calibrate exactly after every two samples, using only y. With no direct communication, synchrony must be maintained by enforcing $\dot{y} = \dot{x}$. This dependency is rate-based, whereas standard TA are event-driven: clocks have fixed rate 1 and guards test only their values. Hence TA cannot express state-dependent rates or integral constraints, one needs priced/weighted TA [2] or (hybrid) stopwatch automata [9] instead. This type of inter-clock rate constraint cannot be captured by either DTA, icTA or DECA. To overcome these shortcomings, we propose Timed Automata with Clock Derivatives (independent-derivative timed automata (idTA)) an expressive yet decidable extension of TA. The key innovation in idTA lies in its ability to impose rate constraints over the derivatives of clocks, allowing modelers to express relative timing requirements between components. These constraints define how quickly each clock is allowed to evolve, either in absolute terms or relative to others. Crucially, while idTA enables richer timing behavior, it does not generalize to arbitrary differential equations as in Hybrid Automata (HA). Instead, it preserves decidability by limiting rate expressions to derivative comparisons, avoiding the complexity and undecidability issues that plague full hybrid models. This balance of expressiveness and tractability allows idTA remaining compatible with symbolic verification methods such as zone-based reachability analysis. Symbolic compatibility means our operators and constraints admit finite symbolic representations and effective fixpoint computation.

To reason about properties expressed in idTA, we introduce a novel temporal logic: DLν, an extension of the timed modal logic L_ν.It adds time-bounded modalities and simple clock constraints to basic modal logic. Our DLν captures the semantics of idTA, allowing specification of timing relationships influenced by clock derivatives. We also present MIMETIC, a prototype tool that supports symbolic verification for idTA and L_ν, allowing efficient model checking to verifying distributed systems with dynamic timing behavior.

Structure of the paper. Section 2 covers preliminaries. Section 3 defines idTA, extending TA and DTA with rate constraints. Section 4 proves decidability of

reachability. Section 5 introduces the logic DL_ν and shows its model checking is EXPTIME-complete. Section 6 discusses expressiveness, and Section 7 presents the MIMETIC tool. Sections 8 and 9 cover related work and conclude.

2 Background

Let $\mathbb{N}_{>0}$ and $\mathbb{N}_{\geq 0}$ denote the sets of positive and non-negative natural, $\mathbb{R}_{\geq 0}$ the non-negative reals and $\mathbb{Z}$ the integers. For sets X and Y, we use standard set operations: intersection ($X \cap Y$), union ($X \cup Y$), disjoint union ($X \uplus Y$), complement ($\overline{X}$), difference ($X \backslash Y$), empty set ($\emptyset$), and (strict) inclusion ($X \subseteq Y$, $X \subset Y$). A timed word over an alphabet Σ is a finite sequence $\theta = ((\sigma_0, t_0)(\sigma_1, t_1) \dots (\sigma_n, t_n)) \in (\Sigma \times \mathbb{R}_{\geq 0})^*$, where timestamps are non-decreasing: $t_i \leq t_{i+1}$. A timed language $\mathcal{L}$ is any subset of $(\Sigma \times \mathbb{R}_{\geq 0})^*$.

Clock Constraints. Let X be a finite set of variables (clocks) ranging over $\mathbb{R}_{\geq 0}$. The set $\Phi^+(X)$ of clock constraints over X is given by the following grammar:

$$\phi := true \mid x \sim c \mid x - y \sim c \mid \phi_1 \ \wedge \ \phi_2$$

where $x, y \in X$, $c \in \mathbb{N}$, and $\sim \in \ \{<, \ >, \ \leq, \ \geq, \ =\}$. The clock constraints of the form $true$, $x \sim c$ are called *non-diagonal constraints* and those of the form $x - y \sim c$ are called *diagonal constraints*. The set of non-diagonal constraints over X is denoted by $\Phi(X)$.

Clock Invariants. Let X be a finite set of clocks. Let $\Delta(X)$ be a set of clock invariants. Clock invariants are clock constraints as follows:

$$\varphi := true \mid x < c \mid x \leq c \mid \varphi_1 \ \wedge \ \varphi_2$$

where $x, y \in X$, $c \in \mathbb{N}$. Clock invariants restrict the amount of time that can be spent in a given state without changing to the next state.

Clock Valuations. A clock valuation $\nu \in \mathbb{R}^X_{\geq 0}$ over X is a mapping $\nu : X \to \mathbb{R}_{\geq 0}$. We note ν_0 the mapping that associates 0 to each clock. For a time value $t \in \mathbb{R}_{\geq 0}$, we denote by $\nu + t$ the valuation defined by $(\nu + t)(x) = \nu(x) + t$. Given a clock subset $Y \subseteq X$, we note $\nu[Y \leftarrow 0]$ the valuation defined as follows: $\nu[Y \leftarrow 0](x) = 0$ if $x \in Y$ and $\nu[Y \leftarrow 0](x) = \nu(x)$ otherwise. Given a clock constraint $\phi \in \Phi(X)$ and a clock valuation ν, we say that ν satisfies ϕ, denoted by $\nu \models \phi$ and formally defined as follows:

$$\nu \models x \sim c \iff \nu(x) \sim c$$
$$\nu \models \phi_1 \ \wedge \ \phi_2 \iff \nu(x) \models \phi_1 \ \wedge \ \nu(x) \models \phi_2$$
$$\nu \models true \iff true$$

Similarly, for a clock invariant $\delta \in \Delta(X)$ and a clock valuation ν, ν satisfies δ, denoted by $\nu \models \delta$.

Timed Automata (TA) are an extension of finite automata with a set of clocks, that evolve synchronously with time and enable measuring delays [4].

Definition 1 (Timed Automata). *A TA is a tuple* $\mathcal{A} = (S, s_0, \Sigma, X, \rightarrow_{ta}, I, F)$ *where:* S *is a finite set of locations,* $s_0 \in S$ *is the initial location,* Σ *is a finite alphabet,* X *is a finite set of clock names,* $\rightarrow_{ta} \subseteq S \times \Sigma \times \Phi(X) \times 2^X \times S$ *is the finite transition relation,* $I : S \rightarrow \Delta(X)$ *associates to each location a clock invariant,* $F \subseteq S$ *is a finite set of final locations.*

For a transition $(s, a, \phi, Y, s') \in \rightarrow_{ta}$, we write $s \xrightarrow{a,\phi,Y} s'$ and call s and s' the source and target locations, ϕ the guard, a the action, Y the reset set of clocks.

Definition 2 (Semantics of TA). *The semantics of a TA* $\mathcal{A}$ *is given by a Timed Labeled Transition Systems* $\mathsf{TLTS}(\mathcal{A}) = (Q, Q_F, q_0, \Sigma \uplus \mathbb{R}_{\geq 0}, \rightarrow_{tlts})$ *where* $Q = \{(s, \nu) \in S \times \mathbb{R}^X_{\geq 0} \mid \nu \models I(s)\}$ *is the set of states over* $\mathcal{A}$*, with initial state* $q_0 = (s_0, \nu_0)$*,* $Q_F = \{(s_f, \nu) \in F \times \mathbb{R}^X_{\geq 0} \mid \nu \models I(s_f)\}$ *is the set of final states over* $\mathcal{A}$ *and* $\rightarrow_{tlts} \subseteq Q \times (\Sigma \uplus \mathbb{R}_{\geq 0}) \times Q$ *is the transition relation defined by:*

1. *Discrete transition:* $(s, \nu) \xrightarrow{a}_{tlts} (s', \nu')$*, such that* $s \xrightarrow{a,\phi,Y} s'$*,* $\nu \models \phi$*,* $\nu' = \nu[Y \leftarrow 0]$ *and* $\nu' \models I(s')$*,*
2. *Delay transition:* $(s, \nu) \xrightarrow{t}_{tlts} (s, \nu + t)$ *for any* $t \in \mathbb{R}_{\geq 0}$*, such that* $\nu + t \models I(s)$*.*

Distributed Timed Automata (DTA), icTA, and Reachability. In [1], each process in a TA has its own clocks (DTA, icTA) that evolve synchronously within, but independently across, processes, all driven by a shared hardware clock. Clocks are globally readable but locally resettable. In contrast, [19] models interleaved execution, yielding a subclass of stopwatch automata [9]. While reachability in TA is decidable [4], region-based methods are often intractable. Convex unions of regions, or zones [7], enable more scalable analysis, as implemented in tools like UPPAAL and KRONOS.

Zone Graphs and Difference Bound Matrices (DBMs). In the analysis of TA, clock zones symbolically represent sets of valuations as convex sets defined by diagonal constraints $\phi \in \Phi^+(X)$, where a zone is given by $\mathcal{Z} = \{\nu \mid \nu \models \phi\}$. To normalize this representation, a reference clock $x_0 = 0$ is added to the clock set X. A symbolic state is a pair $(s, \mathcal{Z})$, where s is a location and $\mathcal{Z}$ a zone. The set of symbolic states and transitions forms the zone graph of a TA $\mathcal{A}$, written $\mathsf{ZG}(\mathcal{A}) = (Q, q_0, \Sigma, T)$, where Q is the set of symbolic states, q_0 the initial one, and T the transition relation. Transition come directly from the operational semantics of TA, lifted to zones via symbolic post operators. Operations such as time elapse, resets, and intersections preserve zones [7, 4]. Zones are efficiently represented using *Difference Bound Matrices* (DBMs) [7]. A DBM is a square matrix indexed by $X := X \cup \{x_0\}$, where each entry $\mathbf{d}_{i,j} = (d_{i,j}, \preceq)$ encodes the constraint $x_i - x_j \preceq d_{i,j}$ with $\preceq \in \{<, \leq\}$ and $d_{i,j} \in \mathbb{Z} \cup \{\infty, -\infty\}$. The semantics of a DBM $\mathcal{D}$ is the clock zone: $\mathcal{Z} = (x_0 = 0) \ \wedge \bigwedge_{0 \leq i \neq j \leq n} x_i - x_j \preceq d_{i,j}$. To improve efficiency, DBMs are canonicalized via the Floyd-Warshall algorithm [11]. Zone abstraction using $Extra_{LU}$ and $Extra^+_{LU}$ [8] keeps only bounds relevant to guards and invariants, yielding a finite zone graph and better reachability performance.

Timed Logic L_ν and Time Language. L_ν is a modal logic used to specify properties of states in a TLTS over the set of actions [20]. Formulas combine

simple state predicates (e.g., constraints on clocks) with modalities over steps (time elapse and labeled actions), and a greatest fixpoint operator to express invariants/safety. We denote by $\mathcal{L}(\mathcal{A})$ the timed language of a TA $\mathcal{A}$, i.e., the set of timed words generated by accepting runs: $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathsf{TLTS}(\mathcal{A}))$

3 Timed Automata with Clock Derivatives

In this section, we present a derivative-based timed semantics for TA, called idTA, which incorporates clock derivatives to capture the rates of change of independent clocks. This extension enables more accurate modeling of timing behavior in distributed systems, where clocks may evolve at different rates. Our semantics offer two main advantages. First, expressiveness is significantly enhanced: while previous frameworks like [23] allowed for independent clocks, they lacked explicit rate constraints. In contrast, idTA introduces precise rate bounds (e.g., $\dot{x} \leq \dot{y}$), allowing direct and indirect timing dependencies between components. This subsumes the multi-timed semantics in [23] and enables the modeling of richer distributed behaviors. Second, idTA supports modular verification by associating rate constraints with locations, allowing local analysis of each component under its own timing constraints, simplifying reasoning in large-scale systems.

Definition 3 (Rate Constraints). *Let X be a finite set of clocks. Let $\dot{X}$ be a finite set of time clock derivatives. The set $\Psi(X)$ of rate constraints over the set of clocks X is given by the following grammar:*

$$\psi := true \mid \dot{x} \sim 1 \mid \dot{x} \sim \dot{y} \mid \psi_1 \ \wedge \ \psi_2$$

where $\dot{x}, \dot{y} \in \dot{X}$, $x, y \in X$ and $\sim \in \{<, >, \leq, \geq, =\}$. A rate constraint ψ is a conjunction of comparing two clock derivative values or a clock derivative value with a natural constant 1. A classical clock (as in TA) will be described by $\dot{x} = 1$.

The constant 1 serves as a standard reference rate, representing normal time progression as in traditional TA. By focusing on $\dot{x} = 1$, we ensure the framework remains expressive enough for key use cases like DRTS, while avoiding unnecessary complexity. Allowing general comparisons with arbitrary constants would increase the complexity of the semantics and model-checking process without necessarily providing meaningful added expressiveness for the intended applications. As previously mentioned, TA assumes perfectly synchronous clocks, which is not always feasible in DRTS, where clocks may evolve differently due to factors like temperature, humidity, pressure, or aging. To formalize the independent evolution of these local clocks, we slightly adapt the definition of rates from [1].

Definition 4 (Rates). *Let X be a finite set of clocks. A rate is a tuple $\tau = (\tau_x)_{x \in X}$ of local time functions. Each local time function τ_x maps the reference time (i.e., a global time) to the time of the clock x, i.e, $\tau_x : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$. The functions τ_x must be continuous, strictly increasing, divergent, and satisfy $\tau_x(0) = 0$. The set of all these tuples τ is denoted by Rates. For all $x \in X$ and $t \in \mathbb{R}_{\geq 0}$, $\tau(t) = (\tau_x(t))_{x \in X}$ where $\tau_x(t)$ is a member of the tuple $\tau(t)$.*

Definition 5 (Addition to a valuation). *Let X be a finite set of clocks, a clock valuation $\nu : X \to \mathbb{R}_{\geq 0}$ and $\boldsymbol{d} \in \mathbb{R}^X_{\geq 0}$: the valuation $\nu + \boldsymbol{d}$ is defined by $(\nu + \boldsymbol{d})(x) = \nu(x) + \boldsymbol{d}_x$ for all $x \in X$.*

Definition 6 (Semantics of Rate Constraints). *Let X be a finite set of clocks. Given a rate constraint $\psi \in \Psi(X)$ and a tuple of functions $\tau \in Rates$, we denote that τ satisfies ψ at time t with $(\tau, t) \models \psi$. In particular, the formal definition is as follows:*

$$(\tau, t) \models true \iff true$$

$$(\tau, t) \models \dot{x} \sim 1 \iff \tau_x \text{ is differentiable at } t \text{ and } d\tau_x/dt(t) \sim 1$$

$$(\tau, t) \models \dot{x} \sim \dot{y} \iff \tau_x \text{ is differentiable at } t \text{ and } \tau_y \text{ is differentiable at } t$$
$$\text{and } d\tau_x/dt(t) \sim d\tau_y/dt(t)$$

$$(\tau, t) \models \psi_1 \ \wedge \ \psi_2 \iff (\tau, t) \models \psi_1 \text{ and } (\tau, t) \models \psi_2$$

The semantics of our idTA is defined via an extension of the TLTS framework called MLTS [23], where each run is a sequence of action–timestamp tuples, capturing the local times of independent clocks. This enables precise modeling of distributed timing across components.

Definition 7 (Timed Automata with Clock Derivatives (idTA)). *A Timed automaton with Clock Derivatives (idTA) is a tuple $\mathcal{A} = (S, s_0, \Sigma, X, \to_{idTA}, I, R, F)$ where : S is a finite set of locations, $s_0 \in S$ is the initial location, Σ is a finite alphabet, X is a finite set of clock names, $\to_{idTA} \subseteq S \times \Sigma \times \Phi(X) \times 2^X \times S$ is the finite transition relation, $I : S \to \Delta(X)$ associates to each location a clock invariant, $R : S \to \Psi(X)$ associates to each location a rate constraint, $F \subseteq S$ is a finite set of final locations.*

Definition 8 (Semantics of idTA). *Given an idTA $\mathcal{A} = (S, s_0, \Sigma, X, \to_{idTA}, I, R, F)$ and $\tau \in Rates$, the timed semantics of $\mathcal{A}$ is given by a MLTS over X, denoted by MLTS($\mathcal{A}$, τ) = (Q, q_0, Σ, Q_F, $\to_{mlts}$), where the set of states Q consists of triples composed of a location, a clock valuation and lastly the reference time: $Q = \{(s, \nu, t) \in S \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}_{\geq 0} \mid \nu \models I(s) \text{ and } (\tau, t) \models R(s)\}$, the starting state is $q_0 = (s_0, \nu_0, 0)$, where ν_0 is the valuation that initializes all the clocks to zero, Σ is the alphabet of $\mathcal{A}$, the set of final states Q_F consists of triples $\{(s_f, \nu, t) \in F \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}_{\geq 0} \mid \nu \models I(s_f) \text{ and } (\tau, t) \models R(s_f)\}$, and the transition relation $\to_{mlts}$ is defined by:*

1. Discrete transition: A transition (q_i, a, q_{i+1}) is denoted $q_i \xrightarrow{a}_{mlts} q_{i+1}$ where $q_i = (s_i, \nu_i, t_i)$, $q_{i+1} = (s_{i+1}, \nu_{i+1}, t_{i+1})$, $a \in \Sigma$, there exists a transition $(s_i, a, \phi, Y, s_{i+1}) \in \to_{idTA}$, such that $\nu_i \models \phi$, $\nu_{i+1} = \nu_i[Y \leftarrow 0]$, $\nu_{i+1} \models I(s_{i+1})$, $(\tau, t_{i+1}) \models R(s_{i+1})$, $t_i = t_{i+1}$ and,
2. Delay transition: A transition $(q_i, \boldsymbol{d}, q_i')$ is denoted $q_i \xrightarrow{\boldsymbol{d}}_{mlts} q_i'$ where $q_i = (s_i, \nu_i, t_i)$, $q_i' = (s_i, \nu_i + \boldsymbol{d}, t_{i+1})$, $\boldsymbol{d} = \tau(t_{i+1}) - \tau(t_i)$ and $\forall t \in [t_i, t_{i+1}]$: $\nu_i + (\tau(t) - \tau(t_i)) \models I(s_i)$ and $(\tau, t) \models R(s_i)$.

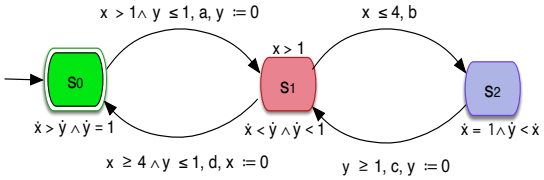

Fig. 1. A TA with Clock Derivatives (idTA) $\mathcal{A}$

Example 1. Figure 1 shows an idTA $\mathcal{M}$ with alphabet $\Sigma = \{a, b, c, d\}$, clocks $X = \{x, y\}$, and rate constraints in locations s_0, s_1 and s_2. The language of the automaton $\mathcal{M}$ includes all timed words where action a occurs first, followed by b, c and d with delays constrained by the rate conditions imposed in each location ($s_0 = \dot{x} > \dot{y}$ and $\dot{y} = 1$), ($s_1 = \dot{x} < \dot{y}$ and $\dot{y} < 1$) and ($s_2 = \dot{y} < \dot{x}$ and $\dot{x} = 1$).

4 Decidability

In this section, we use zone-based abstraction [5] to define timed zone graphs that account for clock derivatives and to present a reachability algorithm. The algorithm is a symbolic breadth-first search (BFS) over the zone graph: from a symbolic state $(s, \mathcal{Z}$ we take all admissible time elapses at once (respecting invariants), intersect with guards, apply resets, canonicalize/abstract the resulting zones, and enqueue any new successors; the search stops when some zone intersects the target. We also show that reachability for idTA is decidable.

4.1 Clock Zones and Operations with Clock Derivatives

A symbolic state $q = (s, \mathcal{Z})$ represents all concrete states $(s', \nu) \in q$ such that $s = s'$ and $\nu \in \mathcal{Z}$, i.e., all valuations ν in zone $\mathcal{Z}$ at location s. Similarly, we write $(s, \mathcal{Z}) \subseteq (s', \mathcal{Z}')$ if $s = s'$ and $\mathcal{Z} \subseteq \mathcal{Z}'$. To define clock zones [20] over a clock set X, we use $\Phi^+(X)$, the set of diagonal constraints. In standard TA, synchronized clocks make differences meaningful, but idTA requires adapting zone operations to handle independent rates. We extend time and discrete successor/predecessor operations accordingly, while intersection, restricted projection, and resets retain their standard semantics [7, 29].

Definition 9 (Semantic Operations on clock zones). *Let $\mathcal{Z}$ be a clock zone. The semantics of the time successor and time predecessor on a clock zone can be defined as:*

1. *Time successor:* $\mathcal{Z}\uparrow_\psi = \{\nu + \boldsymbol{d} \mid \nu \in \mathcal{Z},\ \boldsymbol{d} \in \mathbb{R}^X_{\geq 0}, \exists t, t' > 0, t \leq t',\ \text{and } \exists \tau \in \mathit{Rates}, \boldsymbol{d} = \tau(t') - \tau(t) \text{ and } \exists \psi \in \Psi(X),\ (\tau, t) \models \psi\}$,
2. *Time predecessor:* $\mathcal{Z}\downarrow_\psi = \{\nu - \boldsymbol{d} \mid \nu \in \mathcal{Z},\ \boldsymbol{d} \in \mathbb{R}^X_{\geq 0}, \exists t, t' > 0, t \leq t',\ \text{and } \exists \tau \in \mathit{Rates}, \boldsymbol{d} = \tau(t') - \tau(t) \text{ and } \exists \psi \in \Psi(X),\ (\tau, t) \models \psi\}$.

Intuitively, $\mathcal{Z}\uparrow_\psi$ collects all valuations reachable by letting time flow from $\mathcal{Z}$ along allowed rates while ψ holds, and $\mathcal{Z}\downarrow_\psi$ the valuations that could have flowed

into $\mathcal{Z}$, for example, in standard TA with $\dot{x} = \dot{y} = 1$, if $\mathcal{Z}$ contains $(x = 2, y = 0)$ then $\mathcal{Z}\uparrow$ includes $(x = 5, y = 3)$ after 3s, and conversely from $(x = 5, y = 3)$ we have $(x = 2, y = 0) \in \mathcal{Z}\downarrow$.

Proposition 1. *Let $\mathcal{Z}$ be a clock zone. Then $\mathcal{Z}\uparrow_\psi$, and $\mathcal{Z}\downarrow_\psi$ are also clock zones.*

Definition 10 (Discrete Successor). *Let $q = (s, \mathcal{Z})$ be a zone and e=$(s, a, \phi, Y, s') \in \rightarrow_{idTA}$ be a transition of $\mathcal{A}$, then* post$(\mathcal{Z}, e)$=$\{\nu' \mid \exists \nu \in \mathcal{Z}, \exists \tau \in Rates, (s, \nu) \xrightarrow{e}_{mlts(\mathcal{A},\tau)} (s', \nu')\}$ *is the set of valuations q can reach by transition e.*

The zone $(s', \mathsf{post}(\mathcal{Z}, e))$ describes the discrete successor of the zone $(s, \mathcal{Z})$ under the transition e. The definition is similar for the discrete predecessor.

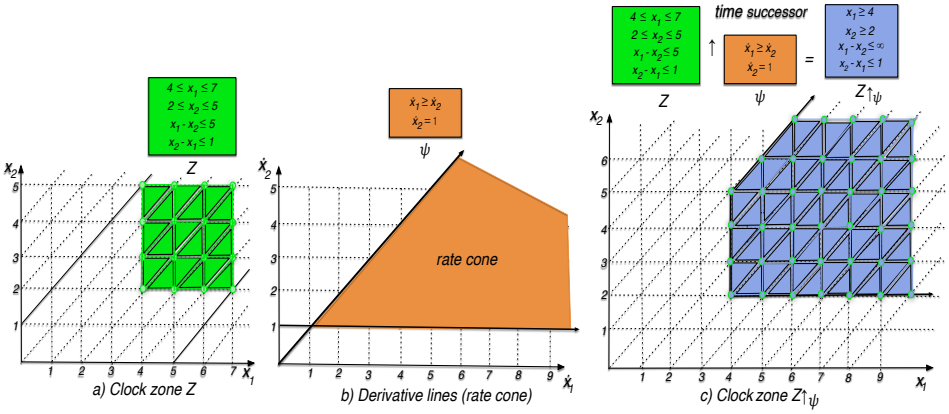

Fig. 2. a) The clock zone $\mathcal{Z}$, b) Derivative lines, c) The time successor of $\mathcal{Z}$ ($\mathcal{Z}\uparrow_\psi$)

Example 2. Consider a clock zone $\mathcal{Z} = \{(4 \le x_1 \le 7) \wedge (2 \le x_2 \le 5)\}$, rate constraint $\psi = (\dot{x_2} = 1) \wedge (\dot{x_1} \ge \dot{x_2})$, and clock invariant $I = \{(x_1 \le 6) \wedge (x_2 \ge 4)\}$. Figure 2 presents an example of *a*) a clock zone $\mathcal{Z}$, *b*) the derivative lines (rate cone), *c*) the time successor of the clock zone $\mathcal{Z}$ ($\mathcal{Z}\uparrow_\psi$). The time successor of $\mathcal{Z}$ is: $\mathcal{Z}\uparrow_\psi = \{(x_1 \ge 4) \wedge (1 \le x_2 \le 5) \wedge (x_2 - x_1 \le 1)\}$. Figure 3 presents an example of the intersection of the clock zone $\mathcal{Z}\uparrow_\psi$ of Figure 2.*c* with the invariant $I = \{(x_1 \le 6) \wedge (x_2 \ge 4)\}$.

4.2 Difference Bound Matrices with Clock Derivatives

We introduce Difference Bound Matrices with Clock Derivatives (DBMCs), an extension of classical DBMs [10][7], designed to capture the independent evolution of clocks by tracking differences between clock derivatives rather than clock values. Traditional DBMs assume synchronous clocks, where differences remain

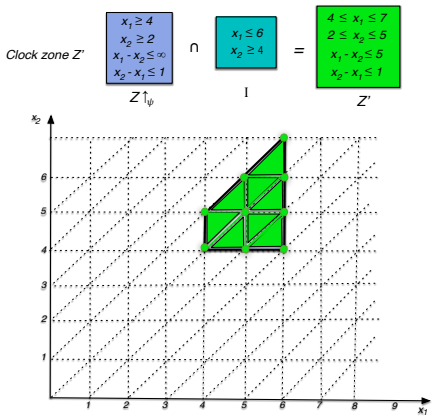

Fig. 3. The clock zone $\mathcal{Z}'$ after intersection with the invariant $I = \{x_1 \leq 6 \wedge x_2 \geq 4\}$

stable over time, but DBMCs allow for clocks advancing at variable rates, making them suitable for distributed systems. Each DBMC encodes a set of rate constraints: comparisons between two derivatives or between a derivative and the constant 1. They serve two purposes: (1) checking the consistency of rate constraints, and (2) adjusting clock differences during time successor and predecessor computations. Essentially, DBMCs extend DBMs by explicitly managing rate constraints in our idTA.

Definition 11 (Difference Bound Matrix with Clock Derivatives). *A Difference Bound Matrix (DBMC) over the set of n clock derivatives* $\{\dot{x_1}, \dot{x_2}, \ldots, \dot{x_n}\}$ *is a* $(n+1)\times(n+1)$ *square matrix (arrowhead matrix) of* $(\{-1,0,1\}\times\{<,\leq\}) \cup \{(\infty,<)\}$ *with rows and columns indexed by* $\{\dot{x_0}, \dot{x_1}, \dot{x_2}, \ldots, \dot{x_n}\}$. *The DBMC can be represented as follows:*

$$\mathcal{E} = (\boldsymbol{e}_{i,j})_{0\leq i,j\leq n}$$

where each $\boldsymbol{e}_{i,j}$ *is of the form* $(e_{i,j}, \preceq)$, $\preceq \in \{<, \leq\}$ *and* $e_{i,j} \in \{-1,0,1\}$ *is called a bound. Formally, the semantics of DBMC* $\mathcal{E}$ *is the rate constraints:*

$$\psi = (\dot{x_0} = 0) \ \wedge \bigwedge_{0\leq i\neq j\leq n}^{n} \dot{x_i} - \dot{x_j} \preceq e_{i,j}$$

where $\dot{x_i}$ *is a clock derivative of the row* i, $\dot{x_j}$ *is a clock derivative of the column* j *and the clock* $\dot{x_0}$ *is always equal to 0.*

Since the variable x_0 is always equal to 0, it can be used to express rate constraints that involve only a single variable (clock derivatives). The language of rate constraints only allows comparison with 1. Thus, $\mathbf{e}_{i,0} = (e_{i,0}, \preceq)$ means $\dot{x_i} \preceq e_{i,0}$ and $e_{i,0} = 1$ (or $e_{i,0} = \infty$). Similarly, $\mathbf{e}_{0,j} = (e_{0,j}, \preceq)$ means $-\dot{x_j} \preceq e_{0,j}$. Also, for each clock difference $\dot{x_i} - \dot{x_i}$, let $\mathbf{e}_{i,i} = (0, \leq)$ or for each unbounded clock difference $\dot{x_i} - \dot{x_j}$ with $i \neq j$ and $i, j \geq 1$, let $\mathbf{e}_{i,j} = (0, \leq)$ (or $e_{i,j} = \infty$). A DBMC $\mathcal{E}$ that satisfied the definition of rates, implies that $\dot{x_i} > 0$ for $0 \leq i \leq n$, thus $\mathcal{E}_{0,i}$ is at least $(0, <)$. Our DBMCs and their canonical forms are defined analogously to standard DBMs.

Example 3. Consider the following clock zone with independent local clocks:

$$\mathcal{Z} = [\![3 \leq x_1 \leq 7 \wedge x_2 \geq 0 \wedge 3 \leq x_1 - x_2 \leq 5]\!]$$

The clock zone $\mathcal{Z}$ can be represented by the matrix $\mathcal{D}$ and by canonizing it, we can get $\mathcal{D}'$:

$$\mathcal{D} = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} (0,\leq) & (-3,\leq) & (0,\leq) \\ (7,\leq) & (0,\leq) & (5,\leq) \\ (\infty,<) & (-3,\leq) & (0,\leq) \end{pmatrix} \end{array}, \quad \mathcal{D}' = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} (0,\leq) & (-3,\leq) & (0,\leq) \\ (7,\leq) & (0,\leq) & (5,\leq) \\ (4,\leq) & (-3,\leq) & (0,\leq) \end{pmatrix} \end{array}$$

We can see that $\mathcal{D}$ does not capture many implicit rate constraints. Therefore, the conjunction of the atomic rate constraints $\psi = \{\dot{x_2} \geq 1 \wedge \dot{x_2} \geq \dot{x_1}\}$ is represented by the matrix $\mathcal{E}$:

$$\mathcal{E} = \begin{array}{c} \\ \dot{x_0} \\ \dot{x_1} \\ \dot{x_2} \end{array} \begin{array}{c} \begin{array}{ccc} \dot{x_0} & \dot{x_1} & \dot{x_2} \end{array} \\ \begin{pmatrix} (0,\leq) & (\infty,<) & (-1,\leq) \\ (\infty,<) & (0,\leq) & (0,\leq) \\ (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix} \end{array}$$

The canonical form of DBMCs follows the same principles as standard DBMs, using the Floyd-Warshall shortest-path algorithm [11] to ensure consistency. In addition to computing the canonical form, we verify the satisfiability of atomic rate constraints by checking the emptiness of the corresponding derivative cone (rate line) and ensuring compatibility with the encoded constraints. Zone transformation operations, such as time successor and predecessor, must also be adapted to respect rate constraints. For example, consider a DBM $\mathcal{D}'$ representing the zone $\mathcal{Z}' = [\![3 \leq x_1 \leq 7 \wedge 0 \leq x_2 \leq 4 \wedge 3 \leq x_1 - x_2 \leq 5]\!]$ and a DBMC $\mathcal{E}$ encoding rate constraints $\psi = \{\dot{x_2} \geq 1 \wedge \dot{x_2} \geq \dot{x_1}\}$. The time successor operation computes $\mathcal{Z} \uparrow_\psi$, representing all reachable valuations by delays satisfying ψ. This is achieved by removing individual clock upper bounds and relaxing constraints on clock differences. Specifically, all first-column entries $\mathbf{d}_{i,0}$ $(1 \leq i \leq n)$ are set to $(\infty, <)$, and diagonal entries $\mathbf{d}_{i,j}$ $(i \neq j)$ are set to $(\infty, \leq)$ when $\mathbf{e}_{i,j} = (\infty, <)$. Figure 4 illustrates the representation of a clock zone $\mathcal{Z}'$ in conjunction with the rate constraints ψ for the clock zone $\mathcal{Z}' = [\![3 \leq x_1 \leq 7 \wedge 0 \leq x_2 \leq 4 \wedge 3 \leq x_1 - x_2 \leq 5]\!]$ and rate constraints $\psi = \{\dot{x_2} \geq 1 \wedge \dot{x_2} \geq \dot{x_1}\}$. Using matrices $\mathcal{D}'$ and $\mathcal{E}$, we obtain the following matrix:

$$\mathcal{D}'' = \begin{array}{c} \\ x_0 \\ x_1 \\ x_2 \end{array} \begin{array}{c} \begin{array}{ccc} x_0 & x_1 & x_2 \end{array} \\ \begin{pmatrix} (0,\leq) & (-3,\leq) & (0,\leq) \\ (\infty,<) & (0,\leq) & (5,\leq) \\ (\infty,<) & (\infty,<) & (0,\leq) \end{pmatrix} \end{array}$$

We now implement two essential operations on our DBMC-based representation of clock zones, required for reachability analysis: time successor and time predecessor [7][29]. These operations extend the classical definitions from [7], but are adapted to incorporate clock derivatives, enabling precise handling of independent clock rates and derivative-based constraints.

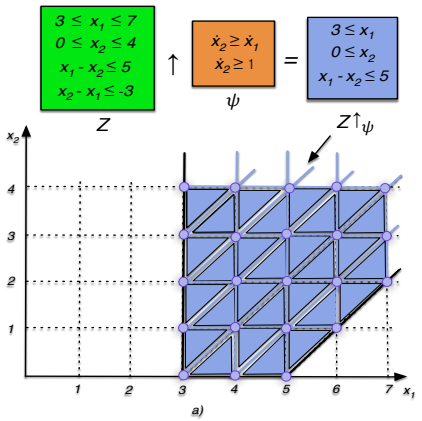

Fig. 4. A clock zone $\mathcal{Z}$

Time Successor: A time successor operation can be computed using a canonical DBM $\mathcal{D} = (\mathbf{d}_{i,j})_{0 \leq i,j \leq n}$ and a canonical DBMC $\mathcal{E} = (\mathbf{e}_{i,j})_{0 \leq i,j \leq n}$ where the computation of the time successor operation of a DBM $\mathcal{D}'$ where $\mathcal{D}' = (\mathbf{d}'_{i,j})_{0 \leq i,j < n}$, consists first in removing from $\mathcal{D} = (\mathbf{d}_{i,j})_{0 \leq i,j \leq n}$ all the upper bounds on the values of clocks, that is, for each $0 \leq i \leq n$, $\mathbf{d}'_{i,0} = (\infty, <)$. Second, the upper bounds of the constraints on the differences between clocks are removed, which is done by replacing the entries in the diagonal clock constraints of $\mathcal{D}'$ ($\mathbf{d}'_{i,j}$ and with $i \neq j$ and $i, j \geq 1$) by $(\infty, <)$, when the entries $\mathbf{e}_{i,j} = (\infty, <)$ with $i \neq j$ and $i, j \geq 1$. The successor algorithm works as follows: it repeatedly removes the upper bounds of all individual clocks, which is done by replacing all elements in the first column of $\mathcal{D}'$ by $(\infty, <)$ and replacing the diagonal elements of $\mathcal{D}'$ ($\mathbf{d}'_{i,j}$ and with $i \neq j$ and $i, j \geq 1$) by $(\infty, <)$ when the entry $\mathbf{e}_{i,j} = (\infty, <)$ with $i \neq j$ and $i, j \geq 1$.

Time Predecessor: A time predecessor operation can be computed using a canonical DBM $\mathcal{D} = \mathcal{D} = (\mathbf{d}_{i,j})_{0 \leq i,j \leq n}$ and a canonical DBMC $\mathcal{E} = (\mathbf{e}_{i,j})_{0 \leq i,j \leq n}$ where the computation of the time predecessor operation of a DBM $\mathcal{D}'$ where $\mathcal{D}' = (\mathbf{d}'_{i,j})_{0 \leq i,j < n}$, consists first in removing from $\mathcal{D} = (\mathbf{d}_{i,j})_{0 \leq i,j \leq n}$ all the lower bounds on the values of clocks, that is, for each $1 \leq i \leq n$, $\mathbf{d}'_{0,i} = (0, \leq)$. Second, the lower bounds of the constraints on the differences between clocks are removed, which is done by replacing the entries in the diagonal clock constraints of $\mathcal{D}'$ ($\mathbf{d}'_{i,j}$ and with $i \neq j$ and $i, j \geq 1$) by $(0, <)$, when the entries $\mathbf{e}_{i,j} = (0, <)$ with $i \neq j$ and $i, j \geq 1$. The predecessor algorithm works as follows: it repeatedly removes the lower bounds of all individual clocks, which is done by replacing all elements in the first column of $\mathcal{D}'$ by $(0, <)$ and replacing the diagonal elements $\mathbf{d}'_{i,j}$ and with $i \neq j$ and $i, j \geq 1$ by $(0, <)$ when the entry $\mathbf{e}_{i,j} = (0, <)$ with $i \neq j$ and $i, j \geq 1$.

Reachability Algorithm: The algorithm 1 constructs a finite symbolic zone graph ($\mathsf{ZG}_{Extra^+_{LU}}(\mathcal{A})$) for a given idTA $\mathcal{A}$ using a depth-first search approach. Algorithm 1 starts with the pair $q_0 = (s_0, Extra^+_{LU}(\mathcal{Z}_0))$ where s_0 is the initial

location of the automaton $\mathcal{A}$, $\mathcal{Z}_0 \leftarrow [\![\bigwedge_{x \in X} x = 0]\!]$ represents the initial zone, where all clocks are set to 0 and $Extra^+_{LU}$ [8][6] (line 5) is the extrapolation abstraction technique. D and Q represents the waiting set of pairs and set of visited pairs Q such that $\mathsf{D} \subseteq \mathsf{Q}$ and a set of transitions T_{ZG} in line 6. The algorithm consists of a loop that iterates over D in line 7. At each iteration, the algorithm takes a pair $\mathcal{Z}_1$ from D and removes it from D in line 8. The algorithm enters an inner loop (lines 9-18) and for each discrete transition $e = (s, a, \phi, Y, s')$ with $\mathcal{Z} \wedge \phi \neq \emptyset$ the algorithm computes the successors of s and associates each successor with the zones $(s', Extra^+_{LU}(\mathsf{post}(\mathcal{Z}, e))) = \mathcal{Z}_2$ (lines 11-12). This result is a set of new pairs $(s, \mathcal{Z}) \in S \times \Phi^+(X)$. The transitions e with successors are stored in the set of labels E_{ZG} in line 12. In line 13, it is checked if there exists an already visited pair $(s', \mathcal{Z}_3) \in \mathsf{Q}$ such that $\mathcal{Z}_2) \subseteq \mathcal{Z}_3$. If true, the discrete transition between $(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s', \mathcal{Z}_3)$ is stored in the set of transitions T_{ZG} (line 14). Otherwise, the discrete transition between $(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s', \mathcal{Z}_2)$ is stored in the set of transitions T_{ZG} and the successor pair $(s', \mathcal{Z}_2)$ is stored in Q and D (lines 16-18). During the search, the algorithm also computes the delay transition by the conjunction between the time successors of the current construction zone $\mathcal{Z}_1$ $(s, Extra^+_{LU}(\mathcal{Z}_1)\uparrow)$, the invariant constraint and rate constraint of the current location s (i.e., $Extra^+_{LU}(\mathcal{Z}_1 \uparrow_{R(s)} \wedge I(s))$) in line 19. In line 20, it is checked if there exists an already visited pair $(s, \mathcal{Z}_3) \in \mathsf{Q}$ such that $\mathcal{Z}_2 \subseteq \mathcal{Z}_3$. If true, the delay transition between $(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s, \mathcal{Z}_3)$ is stored in the set of transitions T_{ZG} (line 21). Otherwise, the delay transition between $(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s, \mathcal{Z}_2)$ is stored in the set of transitions T_{ZG} and the successor pair $(s, \mathcal{Z}_2)$ is stored in Q and D (lines 23-26). Finally, in line 26, the algorithm returns a zone graph.

Proposition 2 (Soundness). *Let $\mathcal{A}$ be an idTA, and let $ZG(\mathcal{A})$ be the zone graph constructed by Algorithm 1. Then for every transition $(s, \mathcal{Z}_1) \xrightarrow{e}_{ZG} (s', \mathcal{Z}_2)$ in the graph, and every clock valuation $\nu \in \mathcal{Z}_1$, there exists an execution in the concrete semantics of $\mathcal{A}$ starting from (s, ν) that leads to some (s', ν') with $\nu' \in \mathcal{Z}_2$.*

Proposition 3 (Completeness). *Let $\mathcal{A}$ be an idTA, and let (s', ν') be a state reachable in the concrete semantics of $\mathcal{A}$ from the initial state (s_0, ν_0). Then there exists a path in the zone graph $ZG(\mathcal{A})$ from $(s_0, \mathcal{Z}_0)$ to some $(s', \mathcal{Z}')$ such that $\nu' \in \mathcal{Z}'$.*

Proposition 4 (Termination). *Let $\mathcal{A}$ be a finite idTA with clocks X. For each $x \in X$, let the maximal constant be $M(x) := \max\{c \in \mathbb{N} \mid$ a constraint of the form $x - x_0 \bowtie c$ or $x_0 - x \bowtie c$ occurs in a guard or invariant of $\mathcal{A}\}$, where x_0 is the reference clock ($x_0 = 0$) and $\bowtie \in \{<, \leq\}$. Then Algorithm 1 terminates and computes a finite zone graph $ZG(\mathcal{A})$.*

Proposition 5 (Complexity). *Let $\mathcal{A}$ be an idTA. Then the reachability problem for $\mathcal{A}$ is PSPACE-complete.*

Algorithm 1 Reachable Zone Graph with Subsumption.
Input: An idTA $\mathcal{A}$=$(S, s_0, \Sigma, X, \rightarrow_{idTA}, I, R, F)$
Output: A reachable zone graph $\mathsf{ZG}(\mathcal{A}) = (Q, q_0, (\Sigma \cup \{\epsilon\}), T_{\mathsf{ZG}})$

1: // $s \in S$ ▷ is a location of $\mathcal{A}$, $\mathcal{Z}_{1 \leq i \leq 3}$ are DBM
2: // T_{ZG} ▷ is a set of transitions (i.e. $\rightarrow_{\mathsf{ZG}} = T_{\mathsf{ZG}}$)
3: //D and Q ▷ are sets of pairs in $S \times \Phi^+(X)$
4: **function** ZG BuildZoneGraph(idTA $\mathcal{A}$)
5: $q_0 = (s_0, Extra^+_{LU}(\mathcal{Z}_0))$ ▷ s.t for all $x \in X$ and $\nu \in \mathcal{Z}_0$, $\nu(x) = 0$
6: Q, D $= \{q_0\}$, $T_{\mathsf{ZG}} = \emptyset$
7: **while** D != $\emptyset$ **do**
8: Choose and Remove $(s, \mathcal{Z}_1)$ from D
9: **for all** transition $e = (s, a, \phi, Y, s')$ s.t $\mathcal{Z}_1 \wedge \phi \neq \emptyset$ **do**
10: //$\mathcal{Z}_2$ is the successor
11: $\mathcal{Z}_2 = Extra^+_{LU}(\mathsf{post}(\mathcal{Z}_1, e))$
12: $E_{\mathsf{ZG}} = E_{\mathsf{ZG}} \cup \{e\}$
13: **if** $\exists (s', \mathcal{Z}_3) \in$ Q $s.t$ $\mathcal{Z}_2 \subseteq \mathcal{Z}_3$ **then**
14: $T_{\mathsf{ZG}} = T_{\mathsf{ZG}} \cup \{(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s', \mathcal{Z}_3)\}$
15: **else**
16: $T_{\mathsf{ZG}} = T_{\mathsf{ZG}} \cup \{(s, \mathcal{Z}_1) \xrightarrow{e}_{\mathsf{ZG}} (s', \mathcal{Z}_2)\}$
17: Q $=$ Q $\cup \{(s', \mathcal{Z}_2)\}$
18: D $=$ D $\cup \{(s', \mathcal{Z}_2)\}$
19: $\mathcal{Z}_2 = Extra^+_{LU}((\mathcal{Z}_1 \uparrow_{R(s)} \wedge I(s)))$
20: **if** $\exists (s, \mathcal{Z}_3) \in$ Q $s.t$ $\mathcal{Z}_2 \subseteq \mathcal{Z}_3$ **then**
21: $T_{\mathsf{ZG}} = T_{\mathsf{ZG}} \cup \{(s, \mathcal{Z}_1) \xrightarrow{\epsilon}_{\mathsf{ZG}} (s, \mathcal{Z}_3)\}$
22: **else**
23: $T_{\mathsf{ZG}} = T_{\mathsf{ZG}} \cup \{(s, \mathcal{Z}_1) \xrightarrow{\epsilon}_{\mathsf{ZG}} (s', \mathcal{Z}_2)\}$
24: Q $=$ Q $\cup \{(s, \mathcal{Z}_2)\}$
25: D $=$ D $\cup \{(s, \mathcal{Z}_2)\}$
26: **return** $\mathsf{ZG}(Q, q_0, (\Sigma \cup \{\epsilon\}), T_{\mathsf{ZG}})$

5 Timed Modal Logic with Clock Derivatives

In this section, we introduce DL_ν, an extension of L_ν [20] that incorporates clock derivatives. We present its syntax and semantics over executions of MLTS, and establish that the model checking problem for DL_ν is EXPTIME-complete.

Definition 12. *Let Σ be a finite alphabet, X be a finite set of clocks and Id the set of proposition identifiers. The formulas of DL_ν over Σ, X, and Id are defined by the grammar:*

$$\varphi ::= true \mid false \mid \varphi \wedge \varphi \mid \phi \mid \psi \mid [a]\varphi \mid \langle a \rangle \varphi \mid x \ \underline{in} \ \varphi \mid \exists \varphi \mid \forall \varphi \mid x - y \sim k \mid D$$

where $a \in \Sigma$, $x, y \in X$, $k = d - c$ and c, d are non-negative integers and , $\sim \in \{=, >, \geq, <, \leq\}$, $D \in$ Id, $\phi \in \Phi(X)$ a clock constraint, $\psi \in \Psi(X)$ a rate constraint, $[a]\varphi$, $\langle a \rangle \varphi$ are two modalities of the logic, and $\exists \varphi$ and $\forall \varphi$ are the two timed modalities.

The identifiers Id are specified within a declaration environment E, which assigns a DL_ν formula to each identifier, enabling the definition of properties with greatest fixpoints. A declaration is denoted as $D \stackrel{\text{def}}{=} \varphi$, meaning that $E(D) = \varphi$. We now formally define the semantics of DL_ν formulas. Let $\mathcal{A}$ be an idTA over $\tau \in \mathsf{Rates}$, with its semantics given by $\mathsf{MLTS}(\mathcal{A}, \tau) = (Q, q_0, \Sigma, Q_F, \rightarrow_{mlts})$. We interpret DL_ν formulas over extended states, where an extended state is a pair (q, μ), with $q \in Q$ representing an MLTS state and μ a valuation for the formula clocks in X. An extended state satisfies an identifier D if it belongs to the maximal fixpoint of the equation $D = E(D)$. The formal semantics of DL_ν formulas, interpreted over $\mathsf{MLTS}(\mathcal{A}, \tau)$, is defined by the satisfaction relation $\models$, which corresponds to the largest relation satisfying the implications in Definition 13.

Definition 13. *Let Σ be a finite alphabet, X be a finite set of clocks. The semantics of formulas in DL_ν is implicitly given with respect to a given MLTS inductively as follows:*

$$
\begin{array}{lcl}
(q,\mu) \models & true & \Leftrightarrow true \\
(q,\mu) \models & false & \Leftrightarrow false \\
(q,\mu) \models & \varphi_1 \wedge \varphi_2 & \Leftrightarrow (q,\mu) \models \varphi_1 \text{ and } (q,\mu) \models \varphi_2 \\
(q,\mu) \models & \phi & \Leftrightarrow \mu \models \phi \text{ for } \phi \in \Phi(X) \\
(q,\mu) \models & \psi & \Leftrightarrow \mu \models \psi \text{ for } \psi \in \Psi(X) \\
(q,\mu) \models & [a]\varphi & \Leftrightarrow \forall q \xrightarrow{a}_{mlts} q', (q',\mu) \models \varphi \\
(q,\mu) \models & \langle a \rangle \varphi & \Leftrightarrow \exists q \xrightarrow{a}_{mlts} q', (q',\mu) \models \varphi \\
(q,\mu) \models & x \ \underline{in} \ \varphi & \Leftrightarrow (q, \mu[x \to 0]) \models \varphi \\
(q,\mu) \models & \exists \varphi & \Leftrightarrow \exists \boldsymbol{d} \in \mathbb{R}^X_{\geq 0}, \exists q' \in Q, \text{ such that } q \xrightarrow{\boldsymbol{d}}_{mlts} q', \\
 & & (q, \mu + \boldsymbol{d}) \models \varphi \\
(q,\mu) \models & \forall \varphi & \Leftrightarrow \forall \boldsymbol{d} \in \mathbb{R}^X_{\geq 0}, \forall q' \in Q, \text{such that } q \xrightarrow{\boldsymbol{d}}_{mlts} q', \\
 & & (q, \mu + \boldsymbol{d}) \models \varphi \\
(q,\mu) \models & x \ + \ c \ \sim \ y \ + \ d & \Leftrightarrow \mu(x) \ + \ c \sim \mu(y) \ + \ d \\
(q,\mu) \models & D & \text{the maximal fixpoint in } \ E(D)
\end{array}
$$

Two formulas are equivalent (i.e., $\equiv$) iff they are satisfied by the same set of extended states in every MLTS.

Definition 14. *A state q in a MLTS satisfies a formula φ, iff $(q, \mu_0) \models \varphi$ where μ_0 is the clock valuation that maps each formula clock to zero.*

Let $\mathcal{A}$ be an idTA and $\varphi \in \mathsf{DL}_\nu$, then $\mathcal{A} \models \varphi$ iff $\forall \tau \in \mathsf{Rates}$, $\mathsf{MLTS}(\mathcal{A}, \tau) \models \varphi$.

Theorem 1. *Let X be a set of clocks. Let $\mathcal{M}$ be a MLTS and q_1, q_2 be equivalent states in Q. Let μ be a clock valuation for the formula clocks in X, then the extended states (q_1, μ) and (q_2, μ) satisfy exactly the same formula in DL_ν.*

Example 4. Consider the idTA $\mathcal{M}$ described in Figure 1. Let X be a set of clocks and $\dot{X}$ be a set of clock derivatives, where $x, y \in X$ and $\dot{x}$, $\dot{y} \in \dot{X}$. The initial state (q_0, μ_0) (i.e., $q_0 = (S_0, \nu_0)$) satisfies the following DL_ν formula φ:

$$\varphi \;=\; y \;\underline{in}\; \exists((\; 2 \geq y \geq 0 \;\wedge\; \dot{x} > \dot{y} \;\wedge \langle a \rangle\; S_1) \wedge \;\; (x \geq 1 \;\wedge\; \dot{y} = 1) \;\wedge\; \langle a \rangle\; S_1)$$

Intuitively, this formula means that the action a can be performed after a delay between 0 and 2, and satisfying the rate constraint $\dot{x} > \dot{y}$, for instance, 1 and the action a can be performed after a delay 1 time units, and satisfying the rate constraint $\dot{y} = 1$.

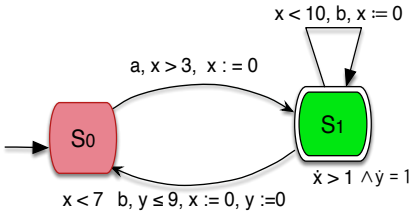

Fig. 5. A TA with clock derivatives (idTA) $\mathcal{M}$

Example 5. Consider the idTA $\mathcal{M}$ described in Figure 5. The state (q_1, μ_1) (i.e., $q_1 = (S_1, \nu_1)$) satisfies the following DL_ν formula φ:

$$D_{S_1} \;=\; x \;\underline{in}\; \exists(x \leq 10 \;\wedge\; \dot{x} > 1 \;\wedge\; \dot{y} = 1 \;\wedge\; \langle b \rangle\; D_{S_1}) \wedge [b]\; (x \leq 10 \;\wedge \dot{x} > 1 \;\wedge \dot{y} = 1 \;\wedge\; x \;\underline{in}\; D_{S_1}) \wedge \forall\; D_{S_1} \vee \;\; y \;\underline{in}\; \exists(y \leq 9 \;\wedge\; \langle b \rangle\; S_0) \wedge x \;\underline{in}\; (\langle b \rangle\; S_0)$$

Intuitively, this formula means that the action b can be performed before a delay 10 time units (self-loop) or the action b can be performed before a delay 9 time units.

Now, we consider the model checking problem of DL_ν sentences on idTA models, that is given a DL_ν formula φ and an idTA $\mathcal{A}$, in deciding whether $\mathcal{A} \models \varphi$ holds.

Theorem 2. *The model checking problem of DL_ν on idTA is EXPTIME-complete.*

6 Expressiveness of idTA

Here, we compare the expressiveness of idTA with existing formalisms such as TA, DTA, and subclasses of Hybrid Automata (HA) [27] including Initialized Rectangular Hybrid Automata (IRHA) [14] and Linear Hybrid Automata (LHA) [3]. We define idTA as a strict extension of these models by proving language containment and separation results.

Theorem 3 (TA vs idTA and idTA vs TA).

1. *Let $\mathcal{A} = (S, s_0, \Sigma, X, \rightarrow_{TA}, I, F)$ be a TA, where all clocks evolve synchronously at a uniform rate. There exists an idTA $\mathcal{A}' = (S, s_0, \Sigma, X, \rightarrow_{idTA}, I, R, F)$ such that $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, where $R = \{\dot{x} = 1 \mid x \in X\}$.*
2. *There exists an idTA $\mathcal{A} = (S, s_0, \Sigma, X, \rightarrow_{idTA}, I,\; R, F)$ such that no TA $\mathcal{B}$ exists with $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$. Specifically, let $\mathcal{A}$ have two locations s_1 and s_2 with rate constraints $\dot{x} = 1$ in s_1 and $\dot{x} > 1$ in s_2. No TA with globally synchronous clocks can simulate this behavior.*

Example 6. In an asynchronous sensor network, each sensor runs on an independent local clock that may drift due to factors like battery level or signal strength. idTA captures this behavior by allowing clocks to evolve at variable rates (e.g., $\dot{x}_i > 1$) and by supporting constraints such as $\dot{x}_i \geq \dot{x}_j$ to express partial synchronization. In contrast, standard TA assumes all clocks advance uniformly, making it unsuitable for modeling such timing variability.

Theorem 4 (DTA vs idTA and idTA vs DTA).

1. *Let Proc be a set of processes. Let* $\mathcal{D} = (S, s_0, \Sigma, \{X_i\}_{i \in Proc}, \rightarrow_{DTA}, I, F)$ *be a DTA where each process* i *has independent clocks* X_i*. There exists an idTA* $\mathcal{D} = (S, s_0, \Sigma, X, \rightarrow_{idTA}, I, R, F)$ *such that* $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$*, where* R *enforces the same independent evolution as in* $\mathcal{D}$*.*
2. *There exists an idTA* $\mathcal{A} = (S, s_0, \Sigma, X, \rightarrow_{idTA}, I, R, F)$ *such that no DTA* $\mathcal{B}$ *exists with* $\mathcal{L}(\mathcal{B}) = \mathcal{L}(\mathcal{A})$*. Specifically, let* $\mathcal{A}$ *have rate constraints* $\dot{x}_1 > \dot{x}_2$*. Since DTA assumes independent clocks, no DTA can enforce this dependency.*

Theorem 5 (idTA vs HA). *Let* $\mathcal{H} = (S, s_0, \Sigma, X, \rightarrow_{HA}, I, F, F_X)$ *be a HA where* F_X *defines arbitrary differential equations* $\dot{x} = f(x, u)$*. Then, there exists an HA* $\mathcal{H}$ *such that no idTA* $\mathcal{A}$ *can satisfy* $\mathcal{L}(\mathcal{H}) = \mathcal{L}(\mathcal{A})$*, since idTA is restricted to constraints of the form* $a \leq \dot{x} \leq b$ *with constant bounds.*

Proof. Since idTA only allows rate constraints of the form $\dot{x} \sim 1$ or $\dot{x} \sim \dot{y}$ with $\sim \in \{<, >, \leq, \geq, =\}$, it cannot model systems where clock derivatives depend on continuous variables. Consider a hybrid automaton where $\dot{x} = x^2$ (quadratic growth). Any equivalent idTA must encode this with a finite set of constant derivatives, which cannot fully capture the smooth, non-linear evolution of HA. This proves that idTA is strictly weaker than HA.

Theorem 6 (idTA vs IRHA). *Let* $\mathcal{A}$ *be an idTA, and* $\mathcal{I}$ *an IRHA. Then,* (1) *for every idTA* $\mathcal{A}$*, there exists an IRHA* $\mathcal{I}$ *such that* $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{I})$*.* (2) *There exists an IRHA* $\mathcal{I}'$ *such that for every idTA* $\mathcal{A}$*,* $\mathcal{L}(\mathcal{A}) \neq \mathcal{L}(\mathcal{I}')$*.*

Theorem 7 (idTA vs LHA). *Let* $\mathcal{A}$ *be an idTA and* $\mathcal{H}$ *an LHA. Then* $\mathcal{L}(\mathcal{A}) \subset \mathcal{L}(\mathcal{H})$*.*

Theorem 8 (IRHA vs LHA). *Let* $\mathcal{I}$ *be an IRHA and* $\mathcal{L}$ *a LHA. Then,* (1) *for every IRHA* $\mathcal{I}$*, there exists an LHA* $\mathcal{L}$ *such that* $\mathcal{L}(\mathcal{I}) = \mathcal{L}(\mathcal{L})$*.* (2) *there exists a LHA* $\mathcal{L}'$ *such that for every IRHA* $\mathcal{I}$*,* $\mathcal{L}(\mathcal{I}) \neq \mathcal{L}(\mathcal{L}')$*.*

Theorem 9. *The expressiveness hierarchy among the models satisfies the following strict inclusions:* $\mathcal{L}(TA) \subset \mathcal{L}(idTA) \subset \mathcal{L}(IRHA) \subset \mathcal{L}(LHA)$*.*

7 Model Checker MIMETIC

We have developed a model checking prototype called MIMETIC[4], which implements the symbolic zone-based reachability algorithm presented in this paper.

[4] MIMETIC source code and jar are available at: `https://github.com/jortizve/MIMETIC/`

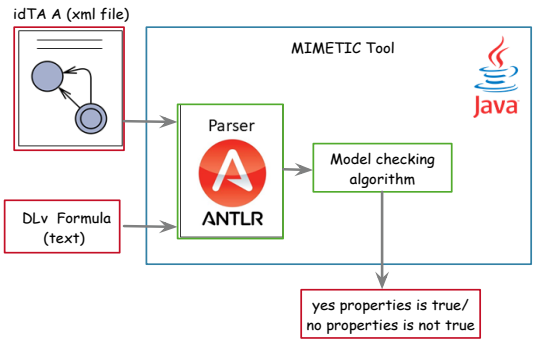

Fig. 6. MIMETIC Tool

The tool supports the verification of properties expressed in our extended temporal logic, DL_ν. Its architecture is modular and extensible, integrating components for parsing, symbolic analysis, and interactive property evaluation. As illustrated in Figure 6, MIMETIC comprises several core components. The tool reads system models described in a XML-based format inspired by UPPAAL, but extended to support rate constraints over clock derivatives. Parsing is handled by an ANTLR-based engine [25], which processes both the system description and the corresponding DL_ν property to be verified. The user interface allows the user to load an XML model file, write or paste a DL_ν formula into a dedicated text box, and launch the verification process. Once complete, the tool displays the result in a human-readable format, indicating whether the property holds along with reachability statistics. MIMETIC is implemented in Java 11 and supports both command-line and graphical execution. The graphical interface provides capabilities for loading models, editing temporal formulas, visualizing the generated transition system, and viewing execution logs. For users working with UPPAAL, models can be easily adapted by introducing derivative constraints directly in the XML used by MIMETIC. As an illustrative example, the user might enter the following formula:

```
1  $$y in EE(y <= 3 &&  y >= 1 &\& <a>tt) && (x' >= 1 && <a>tt)$$
```

This formula specifies a drift-aware temporal condition, combining reachability with rate-based constraints.

8 Related Work

Several formalisms focus on handling distributed timing behaviors in DRTS. Asynchronous Distributed Timed Automata (ADTA), introduced by Krishnan [19], model independently evolving clocks for each component, reflecting the decentralized nature of distributed systems. Akshay et al. [1] examined the untimed language of DTA, while extensions such as Distributed Event Clock Automata (DECA) [24] introduced independent clocks in event-driven systems. Timed Input/Output Automata (TIOA) [17] further enhanced the modeling of distributed

algorithms, particularly for applications like clock synchronization [16]. Robust extensions address uncertainties in real-time systems, such as clock drift. Puri [26] developed robust timed automata where clocks drift within bounded intervals, accommodating imprecise timing in physical systems. These extensions improve the reliability of models when precise synchronization is unattainable, making them suitable for fault-tolerant designs. Hybrid Automata (HA) [13] extend TA by incorporating both discrete transitions and continuous dynamics. Hybrid Input/Output Automata (HIOA) [21] model systems with complex behaviors, including trajectory-based state evolution. Similarly, Stopwatch Timed Automata (SWA) [3] allow clocks to be paused and resumed, enhancing expressiveness. Two important subclasses of HA are Initialized Rectangular Hybrid Automata (IRHA) [14] and Linear Hybrid Automata (LHA) [3]. IRHA restricts the dynamics of continuous variables to piecewise-constant bounds and requires variables to be reset upon entering new modes. LHA, on the other hand, allows linear dynamics and constraints over both discrete and continuous variables, with some restrictions to preserve decidability. We find that idTA subsumes many behaviors that can be expressed by IRHA, especially in scenarios where clock derivatives are constant or reset upon transitions. However, unlike IRHA, idTA do not require initialization of all variables upon mode changes, allowing for more flexible modeling of persistent timing constraints. Compared to LHA, idTA provide a subset of linear behaviors specifically, those that can be encoded by clock derivatives and guarded transitions, but without the full generality of arbitrary differential equations or affine constraints. However, several hybrid modal extensions face significant undecidability challenges, with reachability and simulation problems often being undecidable [9][16]. Despite their advancements, many existing tools, such as UPPAAL [30], TEMPO [12], and HyTech [3], face limitations. Tools like SWA and HIOA suffer from undecidability issues, while others, such as UPPAAL, primarily support forward reachability analysis, lacking adaptability for complex distributed interactions.

9 Conclusion

In this paper, we have introduced idTA, an extension of both DTA and TA, featuring new semantics that incorporate independent clocks and rate constraints. In addition, we extended the logic Lν to support these rate constraints, resulting in the more expressive logic DL$_\nu$. We demonstrated that the model checking problem for this logic is EXPTIME-complete, preserving decidability while increasing expressiveness. We also provided several examples illustrating how distributed real-time behaviors can be modeled using idTA and DL$_\nu$, and we described in detail the implementation and functionalities of our model-checking algorithms. There are several promising directions for future work. One avenue is to further examine the connections between idTA and subclasses of Hybrid Automata (HA) [27], such as Linear HA or Initialized Rectangular HA. Another direction is the application of idTA and DL$_\nu$ to model and verify realistic industrial systems. We also aim to enhance our verification tool, MIMETIC, by incorporating least and greatest fixpoint operators for richer temporal reasoning.

Acknowledgments. This work has been partially supported by the French National Research Agency under the France 2030 label (Superviz ANR-22- PECY-0008). The views reflected herein do not necessarily reflect the opinion of the French government.

References

1. Akshay, S., Bollig, B., Gastin, P., Mukund, M., Kumar, K.N.: Distributed timed automata with independently evolving clocks. In: International Conference on Concurrency Theory. Lecture Notes in Computer Science, vol. 5201, pp. 82–97 (2008)
2. Alur, R., La Torre, S., Pappas, G.J.: Optimal paths in weighted timed automata. In: Computation and Control. pp. 49–62 (2001)
3. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. (1995)
4. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. **126**(2), 183–235 (1994)
5. Behrmann, G., Bouyer, P., Fleury, E., Larsen, K.G.: Static guard analysis in timed automata verification. In: Garavel, H., Hatcliff, J. (eds.) TACAS. Lecture Notes in Computer Science, vol. 2619. Springer (2003)
6. Behrmann, G., Bouyer, P., Larsen, K.G., Pelánek, R.: Lower and upper bounds in zone-based abstractions of timed automata. International Journal on Software Tools for Technology Transfer **8**(3), 204–215 (2006)
7. Bengtsson, J., Yi, W.: Timed automata: Semantics, algorithms and tools. In: Lecture Notes on Concurrency and Petri Nets (2004)
8. Bouyer, P.: Forward analysis of updatable timed automata. Formal Methods in System Design **24**(3), 281–320 (2004)
9. Cassez, F., Larsen, K.G.: The impressive power of stopwatches. In: Palamidessi, C. (ed.) CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings. Lecture Notes in Computer Science, vol. 1877, pp. 138–152. Springer (2000)
10. Dill, D.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) Proc. Automatic Verification Methods for Finite State Systems. LNCS, vol. 407, pp. 197–212. Springer (1990)
11. Dill, D.L.: Timing assumptions and verification of finite-state concurrent systems. In: Sifakis, J. (ed.) Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems (AVMFSS'89). Lecture Notes in Computer Science, vol. 407, pp. 197–212. Springer-Verlag (1990)
12. Georgiou, C., Musial, P.M., Ploutarchou, C.: Tempo-toolkit: Tempo to java translation module. In: 2013 IEEE 12th International Symposium on Network Computing and Applications, Cambridge, MA, USA, August 22-24, 2013. IEEE Computer Society (2013)
13. Henzinger, T.: The theory of hybrid automata. In: Proceedings 11th Annual IEEE Symposium on Logic in Computer Science. pp. 278–292 (1996). https://doi.org/10.1109/LICS.1996.561342
14. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing. p. 373–382. STOC '95, Association for Computing Machinery, New York, NY, USA (1995). https://doi.org/10.1145/225058.225162, https://doi.org/10.1145/225058.225162

15. Henzinger, T.A., Nicollin, X., Sifakis, J., Yovine, S.: Symbolic model checking for real-time systems. Inf. Comput. **111**(2), 193–244 (Jun 1994)
16. Kaynar, D., Lynch, N., Segala, R., Vaandrager, F.: The Theory of Timed I/O Automata, Second Edition. Morgan & Claypool Publishers, 2nd edn. (2010)
17. Kaynar, D.K., Lynch, N., Segala, R., Vaandrager, F.: Timed i/o automata: A mathematical framework for modeling and analyzing real-time systems. In: Proceedings of the 24th IEEE International Real-Time Systems Symposium. pp. 166–. RTSS '03, IEEE Computer Society, Washington, DC, USA (2003)
18. Kopetz, H.: Real-Time Systems: Design Principles for Distributed Embedded Applications. Springer, 3rd edn. (2022), published on September 24, 2022
19. Krishnan, P.: Distributed timed automata. In Workshop on Distr. Systems (1999)
20. Laroussinie, F., Larsen, K.G., Weise, C.: From timed automata to logic - and back. In: International Symposium on Mathematical Foundations of Computer Science. Lecture Notes in Computer Science, vol. 969, pp. 529–539 (1995)
21. Lynch, N., Segala, R., Vaandrager, F.: Hybrid i/o automata. Inf. Comput. pp. 105–157 (2003)
22. Mall, R., Patnaik, L.: Specification and verification of timing properties of distributed real-time systems (1990)
23. Ortiz, J.J., Amrani, M., Schobbens, P.: Multi-timed bisimulation for distributed timed automata. In: NASA Formal Methods - 9th International Symposium, NFM 2017, Moffett Field, CA, USA, May 16-18, 2017, Proceedings (2017)
24. Ortiz, J.J., Legay, A., Schobbens, P.Y.: Distributed event clock automata. In: Implementation and Application of Automata. Lecture Notes in Computer Science, vol. 6807, pp. 250–263 (2011)
25. Parr, T.: The Definitive ANTLR 4 Reference. Pragmatic Bookshelf, Raleigh, NC, 2 edn. (2013)
26. Puri, A.: Dynamical properties of timed automata. Discrete Event Dynamic Systems **10**(1), 87–113 (2000)
27. Raskin, J.: An introduction to hybrid automata. In: Handbook of Networked and Embedded Control Systems, pp. 491–518. Birkhäuser (2005)
28. Ruh, J., Steiner, W., Fohler, G.: Clock synchronization in virtualized distributed real-time systems using ieee 802.1as and acrn (2021), `https://arxiv.org/abs/2105.03374`
29. Tripakis, S., Yovine, S.: Analysis of timed systems using time-abstracting bisimulations. Formal Methods in System Design **18**(1), 25–68 (2001)
30. The UPPAAL tool, available at `http://www.uppaal.com/`
31. Wang, F., Mok, A.K., Emerson, E.A.: Distributed real-time system specification and verification in aptl. ACM Trans. Softw. Eng. Methodol. **volume 2** (1993)