Timed Automata for the Development of Real-Time Systems

Md Tawhid Bin Waez
School of Computing, Queen's University, ON
waez@cs.queensu.ca

Juergen Dingel
School of Computing, Queen's University, ON
dingel@cs.queensu.ca

Karen Rudie
Department of Electrical and Computer Engineering, Queen's University, ON
karen.rudie@queensu.ca

September 20, 2011

Technical Report 2011-579
First Version Appeared on August 14, 2011

Abstract

Timed automata are a popular formalism to model real-time systems. They were introduced two decades ago to support formal verification. Since then they have also been used for other purposes and a large has been introduced to be able to deal with the many different kinds of requirements of real-time system.

This paper presents a fairly comprehensive survey, comprised of eighty variants of timed automata. The paper classifies all these eighty variants of timed automata in an effort to determine current developments. It uses analysis techniques, formal properties, and decision problems to draw distinctions between different versions. Moreover, the paper discusses the challenges behind using a timed automata specification to derive an implementation of a working real-time system and presents some solutions. Finally, the paper lists and classifies forty tools supporting timed automata.

The paper does not only discuss many variants and their supporting concepts (e.g., closure properties, decision problems), techniques (e.g., for analysis), and tools, but it also attempts to help the reader navigate the vast literature in the field, to highlight differences and similarities between variants, and to reveal research trends and promising avenues for future exploration.

1 Introduction

Regular languages [236] are the most dominant class of underlying formal languages for current *Model Driven Development (MDD)* [248] techniques because of their appealing closure properties and impressive decidability results. They are used as the underlying formal language in many aspects of MDD: in *temporal logic* [263] and *process algebra* [191] which are used for model design, in *model checking* [115] which is used for model verification, in *controller synthesis* [268] which is used for automated model construction, and in *equivalence and refinement relations* [258] which are used for model conformance testing. Although the use of regular languages is so prevalent in MDD, they are not expressive enough to capture timing properties of *real-time systems* [111]. For the MDD of real-time systems, one has to move to a different formalism which can represent timing information.

There have been many attempts to model real-time systems. Major attempts include *timed Petri nets* [269], *timed transition systems* [175, 252], *timed I/O automata* [240], and *modelcharts* [198]. All these attempts associate lower and upper time bounds with the transitions, but no time constraints to traverse the automaton. None of them developed a theory of timed languages or an algorithm for the verification of timing properties. To overcome these limitations, *timed automata* [20, 21] were introduced by Alur and Dill in the early nineties. Since then, timed automata have become the most dominant formal model to support MDD of real-time systems.

A timed transition system of a timed automaton can be infinitely large due to its ability to express dense time. A timed transition system can be converted into an equivalent finitely large symbolic transition system called *region graph* where reachability is decidable. Decidability of reachability is a core requirement for automated formal verification and this property of timed automata plays a foremost role to establish timed automata as the major real-time formal model. Later on, *zone graphs* were developed and modified continuously to provide better scalability in practice compared to region graph. Rich closure properties and decidability of many important decision problems have contributed to the adaptation of timed automata in many approaches to support MDD of real-time systems. During the first two decades of timed automata, many kinds of generalizations and variants of timed automata have been proposed and studied to address practically all aspects and features of real-time systems. The strong foundation of timed automata has inspired the emergence of a huge number of tools for analysis, verification, controller synthesis, and code synthesis for timed automata. This survey is an attempt to provide a brief and compact description of the development of timed automata and its variants from theory to practice during the first two decades after the birth of timed automata.

The contributions of this survey paper is the listing and grouping of eighty variants of timed automata and the listing and grouping of forty tools which are based on timed automata. This paper describes three kinds of analysis techniques for timed automata based on region, zone, and flattening, respectively. Decision problems and closure properties for timed automata are also enumerated in this survey. It also presents a brief survey on the implementability of timed automata.

The remainder of the paper is organized as follows: Section 2 discusses the syntax of timed automata, while Section 3 explains the operational and symbolic semantics of timed automata. Section 4 presents formal linguistic aspects of timed automata. Section 5 enumerates eighty variants of timed automata and then classifies them into twelve classes. Section 6 discusses implementability challenges and solutions. Section 7 presents some academic tools which are based on timed automata. The paper concludes in Section 8.

2 Syntax

A timed automaton is a finite state automaton with a set of asynchronous (nonnegative real valued) clocks and a set of clock constraints. A *clock valuation* over the set of clocks is a mapping which assigns to each clock a nonnegative real value. An *initial clock valuation* maps each clock of a timed automaton to zero. A vertex in a timed automaton is called a *location*. A location is associated with a clock constraint called the *local invariant*[1] of that location. Control can stay in a location only if the clock valuation satisfies the local invariant of that location. Local invariants are used to ensure the progress of the model [186], that is, control cannot stay in a location forever. Instead of local invariants, *Büchi* or *Muller* acceptance conditions can be used to enforce progress [20, 21]. An edge in a timed automaton is called a *switch*. A switch is associated with a clock constraint, a subset of the clocks, and a *label* (with a symbol). A clock constraint which is associated with a switch is called the *guard* of that switch. A switch can be taken only if the clock valuation satisfies the guard of that switch. Guards are used to restrict the behavior of the automaton. Each associated clock of a switch is reset to 0 when the switch occurs. At any instant, the value of a clock equals the time elapsed since the last time it was reset. While switches are instantaneous, time can elapse in a location. Consider the

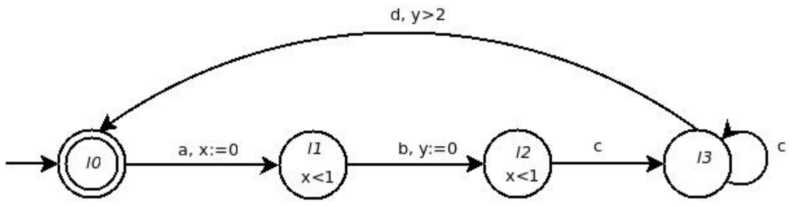


Figure 1: A timed automaton with 2 clocks [12]

example [12] in Figure 1 with two clocks (x and y). The clock x is set to 0 each time the system switches from $l0$ to $l1$ on symbol a. The local invariant ($x < 1$) associated with the locations $l1$ and $l2$ ensures that the c-labeled switch from $l2$ to $l3$ happens within one time unit of the occurrence of a. Resetting clock y together with the b-labeled switch from $l1$ to $l2$ and the guard of the d-labeled switch from $l3$ to $l0$ ensures that the delay between b and the following d is always greater than two time units.

A timed automaton A is a tuple $\langle L, L_0, L_F, \Sigma, \mathcal{C}, E, I\rangle$: where

L is a finite set of locations,

$L_0 \subseteq L$ is the set of initial locations,

$L_F \subseteq L$ is the set of final locations,

Σ is a finite alphabet,

$\mathcal{C}$ is a finite set of nonnegative real valued clocks,

$E \in L \times \Phi(\mathcal{C}) \times \{\Sigma \cup \{\epsilon\}\} \times 2^{\mathcal{C}} \times L$ is the set of switches (edges), and

$I : L \longrightarrow \Phi(\mathcal{C})$ is a mapping that assigns local invariants to locations

and the set $\Phi(\mathcal{C})$ of clock constraints δ is defined inductively by

$$\delta := x \sim q \mid x - y \sim q \mid \neg\delta \mid \delta_1 \wedge \delta_2 \mid true$$

and $q \in \mathbb{Q}$, $\sim \in \{=, <, >, \leq, \geq\}$, elements of the alphabet Σ are observable actions, ϵ represents unobservable actions, and $\mathcal{C}$ is ranged over by x, y etc. The above stated

[1]A timed automaton with local invariants is called *timed safety automaton* [186].

clock constraints only allow one to compare a clock or the difference of two clocks with a rational constant. Clock constraints of the form of $x - y \sim q$ are called *diagonal clock constraints* or *difference clock constraints*. A timed automaton without diagonal clock constraints is called a *diagonal-free timed automaton* [62]. A *flat timed automaton* is a timed automaton which does not have any nested loops: for every location l there is at most one non-empty path from l to itself. Any timed automaton can be emulated by a flat timed automaton [117]. A *k-bounded clock constant* is a clock constraint which involves only constants between $-k$ and k. A switch $e = \langle l, a, \phi, \gamma, l' \rangle \in E$ from location l to l' can occur and reset the set of clocks $\gamma \in 2^{\mathcal{C}}$ on symbol a if the current clock valuation ν satisfies the guard ϕ ($\nu \vDash \phi$). Only the clock constraints which are downwards closed[2] are used as local invariants because a local invariant merely asserts how long control can stay in the associated location of that local invariant.

3 Semantics

3.1 Operational Semantics

A *timed transition system* is a tuple $\langle S, S_0, S_F, \Sigma \cup \mathbb{R}_+, \rightarrow \rangle$ where S is a set of states, $S_0 \subseteq S$ is a set of initial states, $S_F \subseteq S$ is a set of final states, Σ is an alphabet, and $\rightarrow \subseteq S \times \{\Sigma \cup \{\epsilon\} \cup \mathbb{R}_+\} \times S$. The semantics of a timed automaton $A = \langle L, L_0, L_F, \Sigma, \mathcal{C}, E, I \rangle$ is defined by associating a timed transition system $\mathcal{TS}(A)$ of the same alphabet with A [20, 21]: a state in $\mathcal{TS}(A)$ is expressed as a pair (l, ν) such that $l \in L$ and ν is a clock valuation (for $\mathcal{C}$) which satisfies the local invariant $I(l)$. A pair (l, ν) is in S_0 *iff* l is an initial location ($l \in L_0$) and ν is the initial clock valuation ν_0. Similarly, a state (l, ν) is a final state *iff* l is a final location ($l \in L_F$). $\mathcal{TS}(A)$ can have two types of transitions:

Action transition: $(l, \nu) \xrightarrow{a} (l', \nu[\gamma := 0])$ for a switch $\langle l, a, \phi, \gamma, l' \rangle$ if $\nu \vDash \phi$, where $a \in \{\Sigma \cup \{\epsilon\}\}$, $\gamma \in 2^{\mathcal{C}}$ and $\nu[\gamma := 0]$ denotes a clock valuation that differs from ν only in that clocks in γ have been reset to 0.

Time transition: $(l, \nu) \xrightarrow{\tau} (l, \nu + \tau)$ if $(\nu + \tau') \vDash I(l)$ for $\forall \tau' : 0 \leq \tau' \leq \tau$, where $\tau \in \mathbb{R}_+$.

Due to the real-value time transitions, the state-space of the timed transition system of a timed automaton could be infinitely large.

A *timed action* is a pair (t, a), where action $a \in (\Sigma \cup \{\epsilon\})$ is taken by a timed automaton A after $t \in \mathbb{R}_+$ time units since A has been started. The absolute time t is called a *time-stamp* of the action a. A *timed word* is a sequence of timed actions $\xi = (t_1, a_1)(t_2, a_2)...(t_i, a_i)$ where $t_i \leq t_{i+1}$ for $\forall i : i \geq 1$. A *run* of A in $\mathcal{TS}$ with initial state $\langle l_0, \nu_0 \rangle$ over the timed word $\xi = (t_1, a_1)(t_2, a_2)...(t_i, a_i)$ is a sequence of transitions:

$$\langle l_0, \nu_0 \rangle \xrightarrow{t_1} \langle l_0, \nu_0' \rangle \xrightarrow{a_1} \langle l_1, \nu_1 \rangle \xrightarrow{t_2 - t_1} \langle l_1, \nu_1' \rangle \xrightarrow{a_2} \langle l_2, \nu_2 \rangle ... \xrightarrow{a_i} \langle l_i, \nu_i \rangle$$

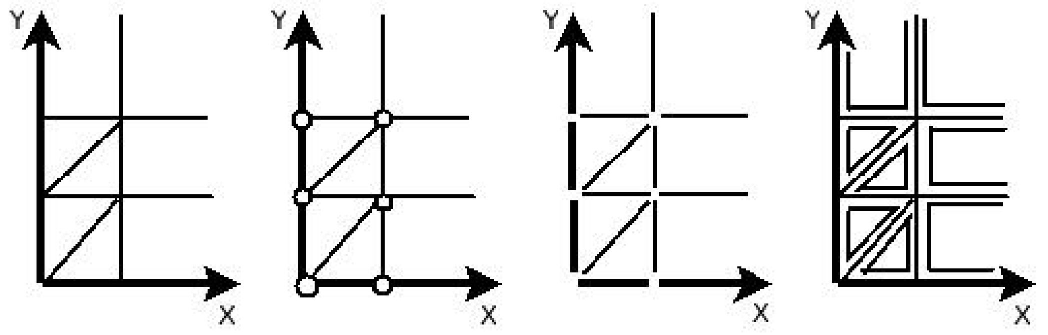
A run is *accepting iff* $\langle l_i, \nu_i \rangle$ is a final state. The *timed language* Σ_t^* over Σ is the set of all timed words over Σ. The *generated timed language* $L_{gt}(A) \subseteq \Sigma_t^*$ is the set of all timed words for which there exists a run of timed automaton A. The set of all timed words with an accepting run of a timed automaton A is the *accepted timed language* $L_t(A) \subseteq L_{gt}(A)$ by A. The *untimed language* L_u is the set of all words in the form $a_1 a_2 a_3 ...$ for which there exists a timed word $\xi = (t_1, a_1)(t_2, a_2)...(t_i, a_i) \in \Sigma_t^*$.

[2] A clock constraint in the form $x \preceq n$ or $x - y \preceq n$ is downwards closed, where $\preceq \in \{<, \leq\}$ and n is a nonnegative integer.

3.2 Symbolic Semantics

Exhaustive verification via state-space exploration is not possible on an infinitely large state-space. In the last two decades, researchers have made many attempts to convert this infinite state-space into an abstract state-space with a finite, tractable number of states such that this coarser state-space preserves all the important properties (for modeling and verification) of the original state-space.

3.2.1 Region Graph



(a) Clock regions (b) 6 intersections (c) 14 lines (d) 8 spaces

Figure 2: All the 28 clock regions in Figure 2(a) for the timed automaton of Figure 1: 6 intersections, 14 lines, and 8 spaces

An infinite state-space of a timed transition system $\mathcal{TS}(A)$ can be converted into an equivalent finite state-space of a symbolic transition system called a *region graph* $\mathcal{R}(A)$ [14, 20, 21]. The decidability results (e.g., reachability analysis, untimed language inclusion, language emptiness, etc.) in timed automata are based on this notion of a symbolic state-space. A *region*, a state of a region graph $\mathcal{R}(A)$, is a pair $\langle l, r\rangle$; where l is a location and r is a set of clock valuations known as *clock region.* Two clock valuations ν and μ are in the same clock region if for any clock x_1 these clock valuations have equal integral part ($\lfloor \nu(x_1)\rfloor = \lfloor \mu(x_1)\rfloor$) and for all clocks these clock valuations preserve the order of the fractional parts (if $fr(\nu(x_1)) \leq fr(\nu(x_i))$ then $fr(\mu(x_1)) \leq fr(\mu(x_i))$, where $fr(c) = c - \lfloor c \rfloor$ and $i \in \mathbb{N}$). The integral part of a clock value is important to decide whether or not a specific clock constraint is satisfied, while the ordering of the fractional parts is needed to decide which clock will change its integral part first. Let k be a function, called a *clock ceiling function*, mapping each clock $x \in \mathcal{C}$ to its ceiling $k(x) \in \mathbb{N}$. Two clock valuations ν and μ are *clock region equivalent* for k, denoted $\nu \approx_k^{\mathcal{R}} \mu$, *iff*

1. $\forall x : x \in \mathcal{C}$, $(\lfloor \nu(x)\rfloor = \lfloor \mu(x)\rfloor) \vee (\nu(x) > k(x) \wedge \mu(x) > k(x))$,

2. $\forall x : x \in \mathcal{C}$, $\nu(x) \leq k(x) \Rightarrow (fr(\nu(x)) = 0) \Leftrightarrow (fr(\mu(x)) = 0))$, and

3. $\forall x, y : x, y \in \mathcal{C}$, $\nu(x) \leq k(x) \wedge \nu(y) \leq k(y) \Rightarrow ((fr(\nu(x)) \leq fr(\nu(y))) \Leftrightarrow (fr(\mu(x)) \leq fr(\mu(y))))$

$\approx^{\mathcal{R}}$ is used instead of $\approx_k^{\mathcal{R}}$, if the clock ceilings are given by the maximal clock constants of the timed automaton under consideration. If the number of clocks $|\mathcal{C}|$ is fixed and each clock $x \in \mathcal{C}$ has a maximal constant m_x, then the number of clock regions is finite: the number of clock regions can be at most $|\mathcal{C}|! \cdot 4^{|\mathcal{C}|} \cdot \prod_{x \in \mathcal{C}}(m_x + 1)$ [21]. All clock regions for the timed automaton of Figure 1 are shown in Figure 2. If $\nu \approx_k^{\mathcal{R}} \mu$ then $\langle l, \nu \rangle$ and $\langle l, \mu \rangle$ are *untimed bisimilar* (or bisimilar w.r.t. $L_u(A)$) for $\forall l : l \in L$. As a consequence, *untimed bisimulation* is used to construct the $\mathcal{R}(A)$.

The first attempt to construct region graphs was made on diagonal-free timed automata. Diagonal clock constraints are necessary to model many applications such as scheduling problems [160]. It was shown that a timed automaton A with difference clock constraints can be converted into an equivalent timed automaton A' which has no difference clock constraints [62]. This conversion is based on a region construction. The size of the transformed model is exponential in the number of diagonal clock constraints.

The number of clock regions in $\mathcal{R}(A)$ grows exponentially with the number of clocks and the size of maximal constants in the clock constraints. Many techniques for the minimization of region automata have been proposed [15, 16, 186, 292]. None of these proposed techniques has been successful in practice. Region automata are not used in practice because the number of regions is often too large to be explored exhaustively.

3.2.2 Zone Graph

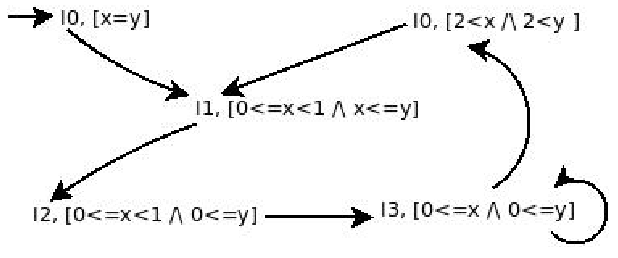
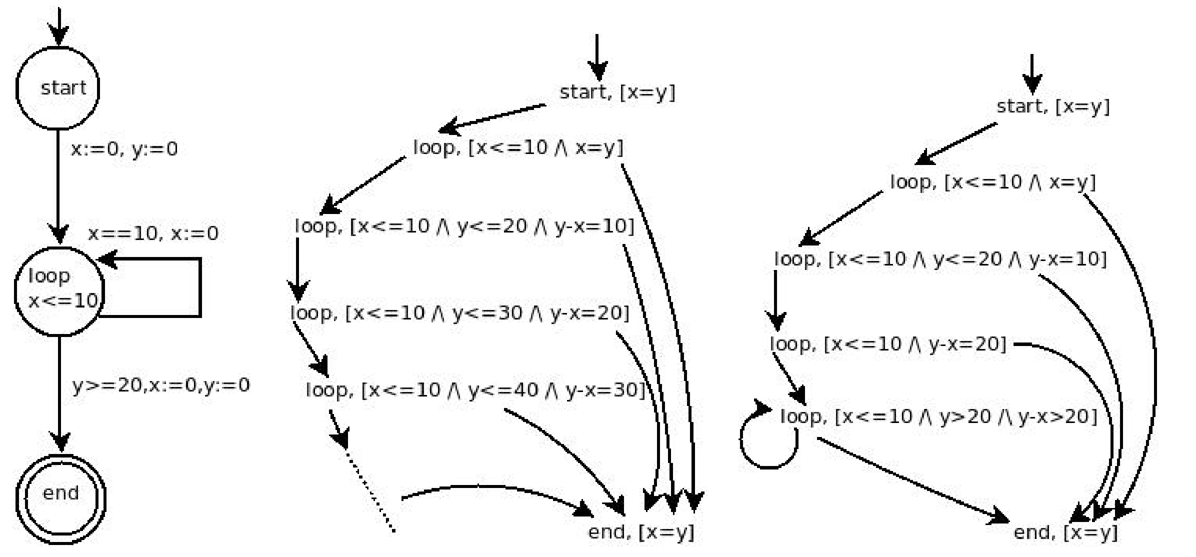


Figure 3: Zone graph for the timed automaton of Figure 1 with only 5 zones

A practically efficient abstract state-space of a timed automaton A is given by its Zone Graph $\mathcal{Z}(A)$ [139, 306, 308]. A zone $\langle l, [\delta] \rangle$ is a pair of a location l and a *clock zone* $[\delta]$. A clock zone $[\delta]$ is the maximal set of clock valuations satisfying $\delta \in \Phi(\mathcal{C})$. If a timed automaton has n clocks, then its clock zones are convex sets in n-dimensional euclidean space. Every clock region is a clock zone [267]. If the addition of two clock regions (or clock zones) is a convex set then the addition is a clock zone [267]. The number of clock zones is the number of convex unions of clock regions [278]; in the worst case, this number is exponential in the number of clock regions. In practice, clock zones are coarser and more compact than clock regions (e.g., the timed automaton of Figure 1 has 28 clock regions as shown in Figure 2, while it has only 5 clock zones as shown in Figure 3). Zones have been used to implement all the major timed automata based tools (e.g., UPPAAL [48], KRONOS [127]).

For a timed automaton $A = \langle L, L_0, L_F, \Sigma, \mathcal{C}, E, I \rangle$, its zone graph $\mathcal{Z}(A)$ is a transition system: states of $\mathcal{Z}(A)$ are zones of A, the zone $\langle l_0, [\mathcal{C} = 0] \rangle$ is the initial state of $\mathcal{Z}(A)$ (where $l_0 \in L_0$ and $\mathcal{C} = 0$ means that the value of any clock in $\mathcal{C}$ is 0), and for every switch $e = \langle l, a, \phi, \gamma, l' \rangle \in E$ and every zone $\langle l, [\delta] \rangle$ there is a transition $\langle \langle l, [\delta] \rangle, a, succ_e(\langle l, [\delta] \rangle) \rangle$; where $succ_e$ is a successor function which returns all the zones which can be reached from the zone $\langle l, [\delta] \rangle$ by first performing the switch e, then letting time pass in the new location, while continuously satisfying the local invariant. The successor function $succ_e$ and reachability analysis in a zone graph are possible because clock zones are closed under the three operations $[\delta_1] \wedge [\delta_2]$, $[\delta]^{\Uparrow,\tau}$, and $[\delta][\gamma := 0]$ where $[\delta_1] \wedge [\delta_2]$ denotes the intersection of $[\delta_1]$ and $[\delta_2]$, $[\delta]^{\Uparrow,\tau}$ denotes the set of interpretations for $\nu + \tau$ for $\nu \in [\delta]$ and $\tau \in \mathbb{R}_+$, and $[\delta][\gamma := 0]$ denotes the set of clock valuations $\nu[\gamma := 0]$ for $\nu \in [\delta]$ and $\gamma \in \mathcal{C}$.

A clock zone $[\delta]$ is *closed under entailment*, if δ cannot be strengthened[3] without reducing the solution set. A *canonical zone graph* $\mathcal{Z}(A)$ means that for every $[\delta] \in \mathcal{Z}(A)$, there is a unique clock zone $[\delta']$ (where $\delta' \in \Phi(\mathbb{C})$) such that $[\delta]$ and $[\delta']$ have exactly same solution set and $[\delta']$ is closed under entailment. Clock zones of a canonical zone graph are represented and manipulated in a data structure called *Difference Bounded Matrices* (DBM) [53, 56, 65, 139, 230]. It is the major structure for the efficient implementation of real-time state-space exploration using symbolic semantics.



(a) A timed automaton (b) An infinite zone graph (c) A k-extrapolated zone graph

Figure 4: A timed automaton with its infinite zone graph and its k-extrapolated (here, $k = 20$) zone graph [58]

Zone graphs are not always finite [58, 128], which makes exhaustive exploration impossible. To remedy this problem, one approach is to construct a *region-closed zone graph* [73, 293, 289]: replace each $[\delta] \in \mathcal{Z}(A)$ by the union of the regions of $\mathcal{R}(A)$ which intersect $[\delta]$. Since the number of regions is finite, there is a finite number of zones after this operation. The region closure of a zone may not be convex. As a result, DBM cannot be used. For this reason, the region-closed zone graph is not used in practice.

Another approach to guarantee finiteness of zone graph is the use of an abstraction operator called the *k-extrapolation* (k is a constant supposed to be greater than the maximal constant occurring in A) [58, 74, 128, 261, 271, 290]. The k-extrapolation operator abstracts $\mathcal{Z}(A)$ into another zone graph $\mathcal{Z}'(A)$ such that all constraints defined in $\mathcal{Z}'(A)$ are k-bounded. The k-extrapolated zone graph is finite, since the number of clock zones with bounded constraints is finite. As an example, a finite k-extrapolated

[3]Let δ_1 be $\delta \wedge x - y \preceq n_1$ ($\delta \wedge x \preceq n_1$) and $|\delta_1| = |\delta_2|$ where δ_2 equals to $\delta \wedge x - y \preceq n_2$ ($\delta \wedge x \preceq n_2$) such that $n_1 > n_2$, then δ_1 can be strengthened by replacing n_1 by n_2 in δ_1.

zone graph of the infinite zone graph of Figure 11(a) is shown in Figure 11(b). A k-extrapolated zone graph is *correct for reachability*[4] only for diagonal-free timed automata [57, 74]. If A has any diagonal constraint, then $\mathcal{Z}(A)$ may have a reachable zone $\langle l, [\delta] \rangle$ where l is not a reachable location in A.

A k-extrapolated zone graph (with diagonal constraints) is correct for reachability [58], if clock valuation ν satisfies a diagonal constraint δ *iff* clock valuation μ satisfies δ where ν and μ are k-extrapolated zone equivalent clock valuations. One method to ensure correctness for reachability of a k-extrapolated zone graph is to check this property [58]. However, checking this property may suffer from an exponential blow-up in the number of zones. The number of zones is multiplied by 2^n, where n is the number of diagonal constraints. To remedy this problem, a new method has been proposed based on counter-example[5] guided abstraction refinement [83] and has been applied [270] in UPPAAL. Not all the diagonal constraints cause incorrectness for reachability. In practice, diagonal constraints produce an incorrect result only very rarely. Since counter-examples are very rare, this refinement method causes very little overhead in practice.

4 Timed Regular Languages and the Decision Problems

A language $L \subseteq \Sigma_t^*$ is a *timed regular language*, if there exists a timed automaton A such that $L = L_t(A)$. An untimed language of a timed regular language is a regular language [21]. *Timed regular expressions* [37, 38] can be used to represent timed regular languages and operations on them. Some variants [89, 90, 141] of timed regular expressions exist in the literature. In MDD, closure properties and decision problems are crucial for modeling, operations on models, and formal verification of a model. For example, the underlying languages need to be closed under *intersection* and *shuffle* to model a concurrent system using an synchronous and interleaving semantics, while *emptiness checking* is used to detect the violation of *safety properties* ("nothing bad will happen") in a model. Timed regular languages are closed under *union* [20, 21], intersection [20, 21], *concatenation* [37, 38], *projection* [20], *renaming* [20], and *Kleene-star* [36, 37, 38]. However, timed regular languages are not closed under *complementation* [20, 21] and shuffle [142, 162]. These results are summarized in Table 1.

Table 1: Closure Properties of Timed Automata

Property	**Closure Under**	**Property**	**Closure Under**
Union	Yes	Kleene-star	Yes
Intersection	Yes	Projection	Yes
Concatenation	Yes	Shuffle	No
Renaming	Yes	Complementation	No

The emptiness checking problem for timed automata is PSPACE-complete and can be solved in time $O(|E| \cdot |\mathcal{C}|! \cdot 4^{|\mathcal{C}|} \cdot (m \cdot m' + 1)^{|\mathcal{C}|})$ where m is the largest numerator in the constants in the clock constraints and m' is the least-common-multiple of the denomi-

[4]We say that a symbolic transition system T' of an original transition system T is correct for reachability *iff* a state s is reachable in T then there is a reachable symbolic state s' in T' which contains s.

[5]A counter-example is a trace where l in A is not reachable, but $\langle l, [\delta] \rangle$ in $\mathcal{Z}(A)$ is reachable.

nators of all the constants in the clock constraints [21, 27]. *Minimum-time reachability*[6] for timed automata is PSPACE-hard [39, 118, 250]. *Timed bisimulation* [18, 109, 233] and *timed simulation* [285] are decidable in EXPTIME. *Universality* [21], *language equivalence* [20, 21], *language inclusion* [20, 21], *determinizability*[7] [163, 288], *computing the clock degree* [288, 305], *minimization of the number of clocks*[8] [163, 288], and *reducing the size of constants*[9] [288] for timed automata are undecidable. Comon and Jurski [117] have shown that the *binary reachability* between any two (set of) states of a timed transition system of a timed automaton is decidable and they left the complexity issue as an open problem. Instead of the conventional region-based (or zone-based) technique, they first convert a timed automaton to an equivalent flat timed automaton and then use the *additive theory of real numbers* to prove the decidability of binary reachability in a timed automaton. We will call their technique flattening technique. The flattening technique allows one to express and verify some important properties that cannot be expressed or verified by region-based (or zone-based) techniques such as "the delay between event $a1$ and event $b1$ is never larger than twice the delay between even $a2$ and event $b2$". On the other hand, their technique is unable to express all the region-based (or zone-based) timing properties, e.g., the flattening technique unable to express unavoidability. These decision problems are summarized in Table 2.

Table 2: Complexity of Decision Problems for Timed Automata

Problem	**Complexity**	**Problem**	**Complexity**
Emptiness checking	PSPACE-complete	Minimum-time reachability	PSPACE-hard
Timed bisimulation	EXPTIME	Computing the clock degree	Undecidable
Timed simulation	EXPTIME	Language equivalence	Undecidable
Universality	Undecidable	Reducing the size of constants	Undecidable
Language inclusion	Undecidable	Minimiza. of the number of clocks	Undecidable
Determinizability	Undecidable	Binary reachability	Decidable

A *deterministic timed automaton* has at most one initial state, no ϵ-transitions, and no pair of switches which have the same action from the same source location with a common clock valuation which can satisfy the guards of both switches. Deterministic timed automata are strictly contained in (nondeterministic) timed automata [20, 21]. A deterministic timed automaton has only one run. Deterministic timed automata are closed under union [20, 21], intersection [20, 21], and complement [20, 21]. Deterministic timed automata are not closed under projection [20, 21] and renaming [24]. These closure properties of deterministic timed automata are summarized in Table 3.

Emptiness checking, universality, language inclusion, languages equivalence problems for deterministic timed automata are PSPACE-complete. These decision problems of deterministic timed automata are summarized in Table 4.

[6]Given a timed automaton A, is there a run of A from some initial location $l_0 \in L_0$ to some final location $l_f \in L_f$? If so, find such a run which consumes minimum-time.

[7]Given an automaton A, does there exists a deterministic automaton B such that $L(A) = L(B)$? If so, construct B.

[8]Given a timed automaton A with n clocks, does there exists a timed automaton B with $n-1$ clocks, such that $L_t(A) = L_t(B)$? If so, construct B.

[9]Given a timed automaton A where constants are not greater than k, does there exist a timed automaton B where constants are not greater than $k-1$, such that $L_t(B) = L_t(A)$? If so, construct B.

Table 3: Closure Properties of Deterministic Timed Automata

Property	Closed Under	Property	Closed Under
Union	Yes	Complementation	Yes
Intersection	Yes	Projection	No
Renaming	No		

Table 4: Complexity of Decision Problems for Deterministic Timed Automata

Problem	Complexity	Problem	Complexity
Emptiness checking	PSPACE-complete	Language inclusion	PSPACE-complete
Universality	PSPACE-complete	Language equivalence	PSPACE-complete

5 Variants of Timed Automata

Many variants [24, 30, 46, 81, 120, 159, 235] of timed automata have been proposed in the literature. There are three major motivations behind this flourish of variants: the major one is to improve existing analysis capabilities of timed automata for the modeling of real-time systems (e.g., to find optimal paths [30], and check schedulability [52, 159], and check memory consumption [30, 50, 159], etc.); the second one is to increase expressiveness by adding features such as probability [46, 217] or recursion [120]; and the last reason is to increase conciseness of the model [81]. There are also some variants of the semantics [43, 133] to make timed automata a more robust and accurate real-time model.

5.1 Timed Automata with Other Clock Constraints

This subsection presents variants of timed automata which add more expressive clock constraints with the existing clock constrains of classical timed automata. Most of them lose many important theoretical properties to facilitate extra expressiveness. In terms of practical applications *parametric timed automata* [25] is the most influential and important class of variants in this subsection.

5.1.1 Timed Automata with Periodic Clock Constraints

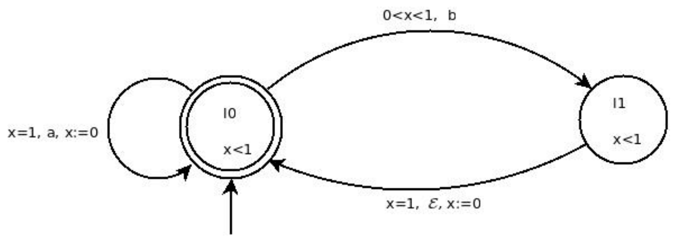


Figure 5: A timed automaton with an ϵ-transition which has no equivalent ϵ-transition-free timed automaton [62]

The class of ϵ-transition-free timed automata is strictly less expressive than the class of timed automata with ϵ-transitions [62]. The timed automaton in Figure 5 accepts a timed language L_ϵ which can be described as follows: in each open time interval $(i, i+1), i \geq 0$ there occurs at most one b; moreover, there is an a at time $i+1$ if and only if there is no b

in $(i, i+1)$. This L_ϵ cannot be accepted by a timed automaton which has no ϵ-transitions. ϵ-transitions without resets can be removed from a timed automaton [60]. Moreover, an ϵ-transition which does not lie in a loop can be eliminated [138]. *Periodic clock constraints* are clock constraints of the form $d+n\cdot\theta \leq x \leq e+n\cdot\theta$ or $d+n\cdot\theta \leq x-y \leq e+n\cdot\theta$ where $n \in \mathbb{N}$, $e \in \mathbb{R}$, and $\theta \in \mathbb{R}_+$. Periodic clock constraints can express properties such as "the value of clock x is odd" or "the value of clock x is of the form $0.7 + 4 \cdot n$ where n is some integer". *Timed automata with periodic clock constraints* in the guards and classical timed automata have the same expressive power [114]. However, the ϵ-transition-free (deterministic) timed automata with periodic clock constraints in the guards are strictly more expressive than the ϵ-transition-free classical (deterministic) timed automata [114]. All ϵ-transitions can be removed from timed automata by using periodic clock constraints and *periodic clock updates*[10] [145].

5.1.2 Additive, Multiplication, and Irrational Clock Constraints

Clock constraints of the form of $x + y \sim q$ are called *additive clock constraints*. The emptiness checking problem is undecidable for *timed automata with additive clock constraints* which have four clocks [59]. Timed automata with additive clock constraints having two clocks are strictly more expressive than classical timed automata with two clocks. The emptiness checking problem is decidable for timed automata with additive clock constraints having two clocks [59]. While the emptiness checking problem is still open for timed automata with additive clock constraints which have three clocks. Introducing clock constraints such as $x = q \cdot y$ in the guards makes the emptiness checking problem for timed automata undecidable [21]. Allowing irrational constants in the clock constraints causes the emptiness checking problem to be undecidable [249]. A summary of all kinds of clock constraints and their effect on the complexity of reachability checking is shown in Table 5.

Table 5: Complexity of Reachability Checking Using Different Clock Constraints

Clock Constraint	**Reachability**	**Clock Constraint**	**Reachability**
$x \sim q$	PSPACE-complete	$d + n \cdot \theta \leq x \leq e + n \cdot \theta$	PSPACE-complete
$x - y \sim q$	PSPACE-complete	$d + n \cdot \theta \leq x - y \leq e + n \cdot \theta$	PSPACE-complete
$x \sim e$	Undecidable	$x \sim q \cdot y$	Undecidable
$x - y \sim e$	Undecidable	$x - y \sim q \cdot z$	Undecidable

5.1.3 Parametric Timed Automata

Timing properties of almost all the real-time protocols are typically not concrete but parametric such as "message delivery within the time it takes to execute two assignment statements" [25]. Concrete timing properties such as "message has to be delivered within 2 time units and an assignment statement has to be executed within 1 time unit" are applicable only for a specific environment. In MDD of real-time systems, parametric timing properties are very appealing for a real-time model of a reusable software module (which is important in MDD of software) or an off-the-shelf real-time hardware (which is gaining popularity in the automotive industry to cope with different *original equipment manufacturers* (OEM) and brands). Moreover, frequently real-time systems are embedded in diverse environments which forces a designer to model the system according to

[10]During a periodic update of a clock that clock is reset to a periodic value instead of 0.

certain parameters. In the early design phase parametric models are usually more convenient for a designer compared to concrete models. Parametric timed automata [25], a generalized form of timed automata, can model parametric timing properties by introducing parametric clock constraints. A parametric timed automaton is a timed automaton which has an accepting run for a parameter valuation of its parametric clock constraints. The emptiness problem for a parametric timed automaton is described as "is there a parameter valuation for which the automaton has an accepting run?". The emptiness checking for parametric timed automata with three or more clocks is undecidable, while it is decidable with only one clock and is an open problem with two clocks [25].

Parametric timed automata can be divided into *linear parametric timed automata* (where all parametric expressions are linear) and *non-linear parametric timed automata.* An important subclass of parametric timed automata is *lower bound automata* [195], in which parameters are only used to calculate the lower bounds in clock constraints. Similarly, the class of *upper bound automata* [195] is a specialization of parametric automata and parameters in upper bound automata are only used to determine the upper bounds in clock constraints. These two classes of automata are together called *lower bound/upper bound automata* or *L/U automata* [195]. Although L/U automata are a restricted form of parametric timed automata, they can be used to model many noteworthy algorithms and protocols such as *Fisher's mutual exclusion algorithm* [222], and *the root contention protocol* [3]. The emptiness checking problem for L/U automata is PSPACE-complete [93, 195]. The universality problem for a parametric timed automaton defined as "does a set of parametric valuations contain all the parametric valuations for which the automaton has an accepting run?". The universality checking for L/U automata is PSPACE-complete [93]. The model checking problem for L/U automata is also decidable [93, 195]. Model checking for parametric timed automata is discussed in [22, 99, 100, 156, 296, 310].

IMITATOR[11] [33, 34] can extract the largest safe[12] subset of parameter values for a parametric timed automaton from a given set of parameter values. HYTECH[13] [178] is also used for the analysis of parametric timed automata such as reachability analysis and operations on states set. Using the open source library REDLIB[14] [301], RED[15] [299] also performs parametric safety analysis, simulation checking, and model checking form parametric timed automata. VerICS[16] [207] and TREX[17] [35] are two other model checkers and analyzers for parametric timed automata.

5.2 Timed Automata with Clock Updates

Variants of timed automata which add more expressive clock updates to the existing clock reset of classical timed automata are discussed in this subsection. Like the variants of Subsection 5.1, variants of this subsection also fail to retain some important theoretical properties of classical timed automata.

[11]For more information on IMITATOR visit http://www.lsv.ens-cachan.fr/˜andre/IMITATOR/.

[12]Safe in a sense that the model is guaranteed not to violate a set of specified safety properties.

[13]For more information visit http://embedded.eecs.berkeley.edu/research/hytech/.

[14]http://sourceforge.net/news/?group_id=226122

[15]RED website: http://cc.ee.ntu.edu.tw/ farn/red/

[16]URL to know more about VerICS is http://verics.ipipan.waw.pl/

[17]TREX website: http://www.liafa.jussieu.fr/˜sighirea/trex/

5.2.1 Updatable Timed Automata

Timed automata with diagonal constraints are *exponentially more concise*[18] than diagonal -free timed automata [78]. Timed automata with diagonal constraints are not more expressive than diagonal-free timed automata [21, 62]. Diagonal constraints may yield different behavior in an extension of timed automata called *updatable timed automata* [79, 80, 81]. Unlike classical timed automata, when a switch is taken, an updatable timed automaton can update a specified subset of clocks to values other than 0. An update u of a clock x is deterministic if u has at most one possible value to assign as $\nu'(x)$ for any clock valuation ν, where $\nu'(x)$ is the value of x after the update u. An example of deterministic update is $x := c$, whereas $x :> c$ is an nondeterministic update which can assign any value as $\nu'(x)$ which is greater than c. The emptiness checking problem for updatable timed automata with updates of the form $x := x - 1$ or $y + c <: x :< z + d$ is undecidable, where $c, d \in \mathbb{Q}_+$ [79, 81]. Only allowing updates of the form $x := c$ or $x := y$ or $x :< c$ keeps the emptiness checking problem PSPACE-complete [79, 81]. Updatable timed automata behave surprisingly for the updates of form $x := x + 1$ or $x := y + c$ or $x :> c$ or $x :\sim y + c$ or $y + c <: x :< y + d$; because these updates make the emptiness checking problem for updatable timed automata with diagonal constraints undecidable, while the emptiness checking problem for diagonal-free updatable timed automata with these updates is PSPACE-complete [79, 81]. Interestingly, updatable timed automata for which the emptiness problem is decidable can be converted into equivalent classical timed automata [80, 81]. These decidable updatable timed automata are more concise than classical timed automata [80, 81]. A summary of different kinds of clock updates and their effect on reachability checking for both diagonal-free timed automata and timed automata with diagonal clock constraints is shown in Table 6.

Table 6: Complexity of Reachability Checking Using Different Clock Updates

Clock Update	**Diagonal-Free**	**With Diagonal**
$\boldsymbol{x := c}$	PSPACE-complete	PSPACE-complete
$\boldsymbol{x := y}$	PSPACE-complete	PSPACE-complete
$\boldsymbol{x := x + 1}$	PSPACE-complete	Undecidable
$\boldsymbol{x := y + c}$	PSPACE-complete	Undecidable
$\boldsymbol{x := x - 1}$	Undecidable	Undecidable
$\boldsymbol{x :< c}$	PSPACE-complete	PSPACE-complete
$\boldsymbol{x :> c}$	PSPACE-complete	Undecidable
$\boldsymbol{x :\sim y + c}$	PSPACE-complete	Undecidable
$\boldsymbol{y + c <: x :< y + d}$	PSPACE-complete	Undecidable
$\boldsymbol{y + c <: x :< z + d}$	Undecidable	Undecidable

5.2.2 Suspension Automata

Suspension automata [247], a variant of timed automata, use *stopwatch*-like clocks and *bounded subtraction clock updates* [159] along with $x := 0$. A bounded subtraction clock update is a clock update of the form $x := x - n$ if $n \leq \nu(x) \leq k(x)$, where $n \in \mathbb{N}_0$ and $k(x)$ is the ceiling for x. The language emptiness checking problem and the language inclusion

[18]The *size of an automaton A*, denoted $|A|$, is the length of its (binary) encoding (states and transitions) on the tape of a *Turing Machine*. Automaton A_1 is exponentially more concise than automaton A_2 if these two automata are language equivalent and $|A_1|$ is polynomial in n, where $|A_2|$ is at least exponential in n.

problem for suspension automata are decidable [247]. However, the language emptiness checking problem for timed automata with *(unbounded) subtraction clock update* in the form $x := x - n$ is undecidable [79].

5.2.3 Integer Reset Timed automata

The class of *integer reset timed automata* [280, 281] is a subclass of classical timed automata since it can reset a clock (to zero) only when it has integer value. Switches without reset can occur at any time (including at fractional times). Integer reset timed automata are less expressive than classical timed automata, e.g., integer reset timed automata cannot distinguish between the time stamps of actions occurring within a unit open interval $(i, i + 1)$. Although language inclusion for classical timed automata is undecidable, it is decidable to check whether a timed regular language contains an integer reset timed regular language [281]. Contrary to classical timed automata, integer reset timed automata are closed under complementation [281].

5.3 Timed Automata with Other Clock Rates

The evolving frequency of a clock is called *clock rate* of that clock. All clocks of a classical timed automaton have the same monotone clock rate. Adding different kinds of clock rates with classical timed automata gives birth to a very expressive, challenging, and popular arena of formal methods called formal methods for hybrid systems. Many researchers consider these variants a completely separate class from timed automata called *hybrid automata* [17, 19].

5.3.1 Rectangular Automata & Controlled Timed Automata

Each clock may have a different clock rate in *rectangular automata* [183, 185] which is an interesting extension of timed automata. Each clock rate is bounded by upper and lower bound constants. For each (initialized) rectangular automaton[19] there is an equivalent timed automaton [185], thus the reachability problem is decidable for this variant. Relaxing either the clock rate boundedness or the initialization assumption leads to undecidability of the reachability problem. Like rectangular automata, clocks in *controlled timed automata* [135] have variable clock rates. Controlled timed automata also allow periodic clock constraints and stopwatch-like clocks. *Stopwatches automata* [106], *interrupt timed automata* [61], and *distributed time-asynchronous automata* [144] are other two general variants of timed automata which use stopwatch to increase the expressive power of timed automata. *Distributed timed automata with independently evolving clocks* [7] are inspired by distributed time-asynchronous automata and execute in a network of timed automata each of which may have different clock rates.

Perturbed timed automata [29] is a special kind of (initialized) rectangular timed automata which considers *timing perturbation.* Another well-known variant of timed automata which considers perturbation are *robust timed automata* [168]. Section 6 presents more discussion on timing perturbation of timed automata.

[19]The initialization property states that whenever the rate of a clock changes it must be reset.

5.3.2 Hybrid Automata

Hybrid systems (e.g., *biological cell networks* [167]) are described by the combination of analog and digital inputs and outputs. Hybrid automata [19], (probably) the most famous and most expressive generalization of timed automata, can model hybrid systems. Hybrid automata thus model discrete controllers embedded within an analog environment (e.g., a digitally controlled drone flies in a continuously changing environment). A hybrid automaton is a finite automaton associated with real-valued variables whose trajectories obey general dynamic laws described by differential equations. Under specified conditions a hybrid automaton can change to different dynamic laws. There are many subclasses of hybrid automata which are not timed automata such as *(non-initialized) rectangular automata* [183], *affine hybrid automata* [167], *polynomial hybrid automata* [164]. The area of hybrid automata is exceedingly large (for example timed automata can be seen as a subclass of hybrid automata) and out of the scope of this survey. Interested readers can read surveys on hybrid automata [104, 125, 176, 221, 287, 291].

5.4 Timed Automata with Resources

This group of variants has been introduced almost a decade after the introduction of classical timed automata. This group of variants has quickly received a lot of attention because of the significance of resources in real-time systems. Now there are 18 variants which can be classified as timed automata with resources. No other class of timed automata has so many variants.

5.4.1 Weighted Timed Automata or Priced Timed Automata

In MDD, a timed automaton serves as a superior model for a real-time system over a finite state automaton because timed automata can explicitly assert time constraints. However, a classical timed automaton is unable to inform the designer how many resources (bandwidth, power, development time, money, etc.) its implementation will consume. This resource consumption information (especially optimal resource consumption) may play a crucial role in MDD. A designer can extract the total resource consumption information of the implementation from a model if the designer attaches a resource consumption function to each state and to each transition of that model. If the resource consumption is proportional to the units of time the implementation stays in a state, then a timed automaton with resource consumption functions is more desirable than a finite state automaton with resource consumption functions. After recognizing the absence of timed automata with resource consumption functions, Alur et al. and Larsen et al. independently introduced timed automata with resource consumption functions in 2001, and called them *weighted timed automata* [30] and *priced timed automata* [50], respectively.

A weighted/priced timed automaton consists of a timed automaton A and a price/cost function $\mathcal{P}$ that maps every location $l \in L$ and every switch $e \in E$ to a nonnegative rational number: $\mathcal{P}(l)$ is the cost for staying in l per unit of time and $\mathcal{P}(e)$ is the cost for performing the switch e. Thus, every run in a weighted/priced automaton has its own accumulated cost and an automaton may have many runs. As a result, to reach a location from a source location with optimal cost (minimum or maximum cost) is an important decision problem for weighted/priced automata. This problem is called *optimal reachability* and is decidable [30, 50]. Optimal reachability is a general form of minimum-time reachability. The optimal-reachability problem for weighted/priced timed automata is PSPACE-complete [30, 50, 75]. Optimal reachability and *optimal scheduling* using weighted/priced automata are well studied and supported in UPPAAL CORA[20], a variant of UPPAAL, which is a specialized tool for optimal reachability and optimal scheduling [51, 52, 228]. REMES-IDE can transform REMES[21] [274] (REsource Model for Embeded Systems) models into behaviorally equivalent weighted/priced timed automata [197]. REMES-IDE provides a graphical editor for the resulting priced automata, as a tool to visually inspect transformation results. Model files for both UPPAAL (timed automata) and UPPAAL CORA (weighted/priced timed automata) can be exported to REMS-IDE for verification and analysis.

A timed-cost-extended version of CTL (branching-time temporal logic) is WCTL, while WMTL is a timed-cost-extended version of LTL (linear-time temporal logic). Model-checking with respect to WCTL is undecidable for weighted/priced timed automata with three clocks or more [97]. Model checking of weighted/priced timed automata with one clock with respect to WCTL is PSPACE-complete [84], while the decidability of WCTL model checking for weighted/priced automata with two clocks is still open [228]. However, model checking with respect to WMTL is decidable only for one clock weighted/priced timed automata using a single stopwatch-cost[22] variable [86].

Because of the great applied significance of weighted/priced automata, they have been studied extensively and many variants have been proposed such as *uniformly-priced timed automata* [49], *dual-priced timed automata* [231], *multi-priced timed automata* [76, 231, 232], *concavely-priced timed automata* [206], *priced timed game automata* [77], *concavely-priced probabilistic timed automata* [204], *weighted integer reset timed automata* [245], and *priced probabilistic timed automata* [63, 64]. More interesting information about weighted/priced automata may be found in a recent article [82] by Bouyer et al..

5.4.2 Task Automata or Timed Automata Extended With Real-Time Tasks

Finite automata can only describe the *arrival sequence* among the actions, while classical timed automata can describe both the arrival sequence among the actions and the *arrival time* of an action. Like finite automata, classical timed automata also describe every action as an instantaneous instance. It is not clear, however, how every action of every real-time system can be instantaneous. This assumption makes classical timed automata a restrictive or very abstract model for real-time systems. Norström et al. [251] have extended timed automata by adding real-time tasks with actions. Later on their work evolved into *task automata* or timed automata extended with real time tasks with locations [159, 160, 212]. A task is an executable program. A task can be described by its *task type* (or *task name*)[23], *best case computational time*, *worst case computational time*, *relative deadline*[24], (optionally) *priority for scheduling*, and (occasionally) resource consumption information. Task automata may model a real-time system which is composed of both *periodic tasks* and *sporadic tasks.* Task automata are at least as expressive as classical timed automata [159, 251].

A task automaton is a restricted updatable timed automaton with an extra clock x_{done} and a partial function $\mathcal{F}$. Like suspension automata, task automata only allow bounded

[20]http://www.cs.aau.dk/~behrmann/cora/

[21]A REMES model is a *state-machine* based behavioral language with support for hierarchical modeling, resource annotations, continuous time, and notions of explicit entry and exit points that make it suitable for component-based modeling of embedded systems. For more information visit http://www.fer.hr/dices/remes-ide.

[22]A cost is stopwatch if it behaves like a clock that can be stopped and restarted.

[23]There can be more than one task with the same name or type.

[24]Relative deadline depends on a task's release time from the location.

subtraction updates along with classical reset to zero. Clock x_{done} is reset whenever a task finishes. The partial function $\mathcal{F}(l)$[25] may annotate a location l with a task. An incoming switch triggers an instance of each annotated task of a location; thus the associated guard of that incoming switch specifies the possible trigger time of the annotated task(s) of a location. A task is entered into an task queue (i.e., the ready queue in an operating system) whenever it is triggered. In an execution queue, tasks are executed according to a given scheduling algorithm, e.g., *fixed priority scheduling* or *earliest deadline first*. In a *fixed task automaton* all the tasks have a *constant computational time*[26], while the computational time of a task may vary in a *flexible task automaton*. The control behavior of a *feedback task automaton* can be influenced by the reset of a clock x_{done} (or the actual finishing time of a task); on the other hand, the actual finishing time cannot influence the control of a *non-feedback task automaton*. A non-feedback task automaton does not have any clock constraint which includes x_{done}.

In MDD of real-time systems, one of the most significant concerns is *schedulability analysis* prior to implementation. Classical timed automata do not have any support for specifying resource consumption information and computational time information; although classical weighted/priced timed automata can specify resource consumption information, they are unable to specify different computational time information such as best case computational time, worst case computational time, etc. A task automaton is schedulable if there exists a scheduling strategy such that all possible sequences of actions generated by the automaton are schedulable in the sense that all associated tasks can be computed within their deadlines. Scheduling algorithms can be divided into *preemptive scheduling algorithms* and *non-preemptive scheduling algorithms*: a currently executed task can be pre-empted by another task in a preemptive scheduling algorithm, whereas, a currently executed task cannot be pre-empted by another task in a non-preemptive scheduling algorithm. The *non-preemptive schedulability checking problem* of a task automaton can be converted into the reachability problem of a classical timed automaton and thus it is PSPACE-complete [159, 251]. The *preemptive schedulability checking problem* of a task automaton can also be converted into the reachability problem of a classical timed automaton if that automaton is not both a flexible task automaton and a feedback task automaton [159]. The preemptive schedulability checking problem of a task automaton is undecidable if it is both a flexible task automaton and a feedback task automaton [159]. Table 7 shows the schedulability problem is undecidable only for a small class of task automata.

Table 7: Complexity of Preemptive and Non-Preemptive Scheduling of Task Automata

Task Automata	**Preemptive Scheduling**	**Non-Preemptive Scheduling**
Fixed & Feedback	PSPACE-complete	PSPACE-complete
Fixed & Non-Feedback	PSPACE-complete	PSPACE-complete
Flexible & Feedback	Undecidable	PSPACE-complete
Flexible & Non-Feedback	PSPACE-complete	PSPACE-complete

Boundedness checking [27] is a useful analysis for a model in MDD of a real-time system, because it can be used for the estimation of the memory consumption by the implemen-

[25] $\mathcal{F}(l)$ is a partial function because some locations may not have any annotated task.

[26] If task has a constant computational time then the best case computational time is equal to the worst case computational time.

[27] The boundedness checking problem is to check whether the size of the task queue for all reachable states is bounded.

tation of that model. If every task has finite size then schedulability implies boundedness but not the other way around, as a bounded ready queue may not be schedulable. As a result, boundedness checking is decidable for a task automaton if that automaton is schedulable [31]. The memory allocated for a bounded task queue can be fixed at compile-time and no exception handling for a queue overflow is needed at run time. A real-time model can exhibit *zeno* behavior, i.e., infinite sequence of action transitions occur within a finite time unit. Such a model is not implementable and also non-schedulable. It is possible to impose a deterministic semantics of a task automaton and also preserve the safety properties satisfied by the non-deterministic semantics [31]. Non-determinism among action transitions is resolved by assigning unique priorities to the action transitions and non-determinism among time transitions is resolved by implementing the maximal-progress assumption [307]. The expressive power, available analyses (schedulability, boundedness checking, non-zenoness checking, resource consumption computation) and deterministic semantics make task automata a suitable model supporting for the MDD of real-time systems. In particular, it is a good model for code synthesis if the target platform ensures the *synchrony hypothesis*, i.e., the run-times of related system functions are negligible compared to the different execution times of the associated tasks of the model. TIMES[28] [31, 32, 42], based on task automata, is a popular tool in the research community for real-time code synthesis and scheduling. Schedulability analysis problems of task automata for multi-processor platforms have been studied in [211].

5.4.3 Timed P Automata

A biologically inspired (specifically, the structure and the functioning of living cells) model called *P systems* [29] [265, 266] has received huge attention[30] in the area of theoretical computer science for its impressive computational and modeling power. Membrane computing naturally models mobility, distributed parallel computing, biomolecular systems, and ecological systems. A P system comprises a hierarchy of membranes; each of these membranes contains a multiset of reactant objects and (possibly) other membranes. An evaluation rule describes reactants and the resulting product. An evaluation rule can be applied only to objects of that membrane.

In *timed P systems* [108], a variant of P systems, each evolution rule is associated with an integer which represents the number of time units needed by the rule to be entirely executed. Recently proposed *timed P automaton* [44] is a timed automaton with a discrete time domain where every location is a timed P system. Timed P automata are useful to study a population which dynamically changes with time (e.g., the population of a place whose dynamics changes with seasons).

5.5 Timed Automata with Probability

Any real-time property can be either a *hard real-time property* (e.g., "the car stops within 800 time unit after the break is applied") or a *soft real-time property* (e.g., "at most 3% of all the messages will not be delivered within 5 unit of times"). While hard real-time properties are essential in many safety critical real-time systems (e.g., robotic surgery),

[28]For more information visit http://www.timestool.com.

[29]P Systems was introduced in 1998 by Gheorghe Păun, whose last name is the cause of the letter P in 'P Systems'. For more information on P systems please visit http://ppage.psystems.eu/.

[30]On 3rd October 2003, Membrane Computing has been selected by Thomson Institute for Scientific Information (ISI) as a "Fast Emerging Research Front in Computer Science".

soft real-time properties are required for many commonly used real-time systems (e.g., video streaming). Unfortunately, classical timed automata and all its variants discussed above cannot support soft real-time properties. To serve as a complete model for the MDD of real-time systems, timed automata have to have support for the specification and analysis of soft real-time properties along with hard real-time properties. Soft real-time properties are frequently used in fault tolerant real-time systems (e.g., communication protocols, multimedia protocols) where hard real-time properties are too restrictive: violating a hard deadline does not affect the functionality of the protocol. Soft real-time properties are supported by a popular extension of timed automata called *discrete probabilistic timed automata* [13, 46, 202, 217]. Recently, an expressive generalization of discrete probabilistic timed automata has been proposed called *first-order probabilistic timed automata* [161]. Hierarchic first-order superposition-based theorem proving and probabilistic model checking both are useful for models based on first-order probabilistic timed automata based models.

Clocks in a *continuous probabilistic timed automaton* [218] can be reset according to *continuous probability distributions.* On top of soft deadline properties, continuous probabilistic timed automata also enable *stochastic timing*, that is, soft deadlines must be satisfied under the assumption that some set of events is influenced by a certain continuous time probability distribution. An example [216] of stochastic timing properties is "the arrival rate of video frames is normal with mean of 40ms and variance of 5ms, and service is exponential with rate 45ms". Thus stochastic timing properties can estimate some important *performance parameters* such as *throughput* and *mean service time.*

Every switch of a probabilistic timed automaton encodes its likelihood to occur. This likelihood is calculated from the execution of certain actions by the system. Hence, probabilistic timed automata can be used to evaluate *quality of service* which is the quantitative estimation of the probability of achieving some target (e.g., perform a certain task in a time bound). Timed probabilistic properties can be expressed by *probabilistic timed branching-time temporal logic* (PTCTL) [217]. Model checking for PTCTL can be performed by converting it into model checking for *probabilistic branching-time temporal logic* (PTCL) [217]. Zone-based forward and backwards PTCTL symbolic model checking also has been studied in the literature [217, 219, 220]. Among tools, UPPAAL PRO[31] and Fortuna [64] can analyse *maximum probabilistic reachability properties* of probabilistic timed automata. The latest version of PRISM[32] (PRISM 4.0) [173, 215] provides more general support for the verification and analysis of both discrete and continuous probabilistic timed automata. mcpta [1] is another model checker for probabilistic timed automata.

5.6 Timed Automata with Communication

Concurrent and communicating models are ideal to model mobile systems, cloud computing, and concurrent embedded systems. Untimed concurrent and communicating models widely use FIFO channels (queues) to communicate among them; channels are also common in real-time concurrent and communicating models such as *communicating real-time state machines* [276], π_{klt}*-calculus* [264], and UPPAAL-models [48]. In 2006, Krcál and Yi developed *communicating timed automata* [208]. A communicating timed automaton is a

[31]http://www.cs.aau.dk/˜arild/uppaal-probabilistic/

[32]PRISM is a well established and popular open source verification tool for probabilistic models such as Markov chains, Markov decision processes, probabilistic automata. For more information visit http://www.prismmodelchecker.org/.

network of timed automata extended with (unbounded) channels. Untimed communicating finite state models are not more for expressive than (classical) finite state automata. A communicating timed automaton with only one channel and no sharing states has the power of a one-counter machine. In contrast, a communicating timed automaton with only two channels and no sharing states has the power of two-counter machines (or Turing machines), thus channels make the verification of communicating timed automata more difficult [48]. Other timed automata variants which also use channels to communicate are *multi-queue discrete timed automata* [262], *omega deterministic timed alternating finite automata* [158], synchronized concurrent timed automata [298], and queue-connected discrete timed automata [196]. An interesting timed automata variant with communication is *phase event automata* [192, 193] which combines both state-base (e.g., Kripke structure) and event-based (e.g., finite state automata) structures. The benefit is one can combine the benefits of both process algebra (which depends on event-base structure) and model-checking (which depends on state-base structure). The trade-off for this kind of structure in practice is that the chance of manual errors during modifying is increased.

A state in a *hierarchical state machine* can be either a normal state or a superstate (which contains some other states). Although hierarchical state machines (e.g., STATECHARTS [172], UML[33] [70]) are a widespread model in MDD, very little research has been done to understand their theoretical aspects such as expressiveness, decision problems, concurrency complexity, and formal (unambiguous) semantics. Alur et al. [26] published one of the first publications on the topic of the decision problems and succinctness of (untimed) *communicating hierarchical state machine.* Inspired by their work another group came up with *communicating hierarchical timed automata* [223, 224] to study theoretical aspects of real-time hierarchical state machines. Like (untimed) communicating hierarchical state machines, the reachability problem for communicating hierarchical timed automata is EXPSPACE-Complete. Beyer and Rust developed a hierarchical variant of timed automata for modular specification. The name of their variant is *Cottbus timed automata* [68, 69] which are developed in Cottbus, Germany. Rabbit[34] [67] is a model checker (reachability and refinement-checker) for Cottbus timed automata. Another hierarchical timed automata variant is *timed cooperating automata* [225, 226] which is a real-time variant of *cooperating automata* [148].

5.7 Timed Automata with Determinizability

Alur, Fix, and Henzinger [23, 24] proposed a determinizable subclass of timed automata named *event-clock automata* after determining that the major obstacle to achieving determinizability of classical timed automata is nondeterministic clock resets. All the clocks in an event-clock automaton are divided into two disjoint sets: one set contains only *event-recording clocks* and another set has only *event-predicting clocks.* Every action (or event) in event-clock automata has a one-to-one relation with an event-recording clock and with an event-predicting clock. All the clocks in an event-recording automaton are associated with actions and the number of actions are fixed, thus the number of clocks is fixed. An event-recording clock records when the associated action occurred the last time, and an event-predicting clock shows when the associated action will occur next time. Event-clock automata do not have any ϵ-transitions. Removing all the event-predicting clocks from an event-clock automaton will convert it into an *event-recording automaton.* Similarly,

[33]For more information on UML, please visit http://www.omg.org/spec/UML/2.1.2/
[34]Rabbit's website: http://www.sosy-lab.org/˜dbeyer/Rabbit/.

eliminating all the event-recording clocks from an event-clock automaton will transform it into an *event-predicting automaton.*

Event-clock (or event-recording or event-predicting) automata are determinizable thus they are closed under complement. Event-clock (or event-recording or event-predicting) timed automata are closed under all the Boolean operations. The language-inclusion problem for event-clock automata is PSPACE-complete. Dima defined a class of regular expressions equivalent to event-clock automata [140]. D'Souza discussed the logical characterization of event-clock automata and event-recording automata [149, 150]. *Event-clock visibly pushdown automata* [284] and *recursive event-clock automata* [187] have also been proposed for determinizable self-embedded recursive timed automata. *Product interval timed automata* [153] are a subclass of event-recording automata that can be used to model the timed behavior of asynchronous digital circuits. Other related timed automata variants are *timed automata with input-determined guards* [152], *eventual timed automata* [151], *counter-free input-determined timed automata* [113], and *continuous timed automata with input-determined guards* [112]. TEMPO [277] is a model checker for event-recording automata and was first released in 2001.

5.8 Timed Automata with Self-Embedded Recursion

Self-embedded recursion[35] can model naturally the control flow of sequential computation in typical programming languages with nested and recursive invocations of program modules. A *pushdown timed automaton* [119, 120, 123] is a variant of classical timed automaton which can express real-time self-embedded recursive properties by augmenting a timed automaton with a stack. Many real-time non-regular properties are required for real-time software verification. Unfortunately, introducing self-embedded recursion destroys many important closure properties (e.g., intersection) for modeling and verification. Therefore, these kind of properties are usually handled by less expressive but practically efficient *finite indexing* techniques such as bounded real-time model-checking [260]. The binary reachability of a pushdown timed automaton is decidable [119, 120, 123]. The binary reachability of *past pushdown timed automata* [121, 122], a parametric variant of *discrete pushdown timed automata* where the past-formulas[36] can be used as clock constraints, is also decidable. The universality problem and language inclusion problem for *timed visibly pushdown automata* [157] (nondeterministic timed version of *visibly pushdown automata* [28]) even with a single clock is undecidable. A deterministic timed automata version of visibly pushdown automata called event-clock visibly pushdown automata [284] is closed under Boolean operations. It is decidable to check whether a timed visibly pushdown language is included in an event-clock visibly pushdown language [284]. In 2010, *recursive timed automata* [294] and *timed recursive state machines* [55] were proposed.

5.9 Timed Automata with Succinctness

The main motivation behind the creation of this group of timed automata variants is to improve modeling rather than to achieve better analyses.

[35]Balanced parentheses languages are well known examples for self-embedded recursion.

[36]A past formula is a formula which includes the past parametric values.

5.9.1 Alternating Timed Automata

An (untimed) *alternating finite automaton* [110] is a nondeterministic finite automaton whose transitions are divided into existential and universal transitions. For example, let A be an alternating automaton. For an existential transition $(s_1, a, s_2 \vee s_3)$, A nondeterministically chooses to switch the state to either s_2 or s_3 after reading a (like a nondeterministic finite automaton). For a universal transition $(s_1, a, s_2 \wedge s_3)$, A moves to s_2 and s_3 after reading a (which simulates the behavior of a parallel machine). An alternating finite automaton accepts a word if there exists a run tree on that word such that every path ends in an accepting state. Due to the universal quantification, a run is represented by a run tree. Any alternating finite automaton is equivalent to a nondeterministic finite automaton. Alternating models are useful to express clauses which are combined by Booleans.

Alternating (tree) timed automata [136], a real-time extension of alternating automata, are closed under all Boolean operations [234, 235, 254, 255]. Emptiness checking for alternating timed automata is decidable only for one clock over finite timed words; any extension (infinite timed words, more than one clock, silent transitions) leads to undecidability [235, 255]. Undecidability proofs of the emptiness checking problem for alternating timed automata with one clock over infinite words rely on the ability to express "infinitely often" properties. *Weak alternating timed automata* [259] do not permit one to express "infinitely often" properties, thus the emptiness checking problem for weak alternating timed automata over infinite words is decidable. Interestingly, bounded time model checking of alternating timed automata (over finite or infinite words) is decidable as in bounded time the emptiness checking is decidable [201]. TCTL model checking for alternating timed automata has also been discussed [136].

5.9.2 Timed Automata with Deadlines

Urgency (*urgent transitions* and *urgent locations*) is a common and important concept in real-time models (such as in timed Petri nets [203] or in timed I/O automata [165]) because they allow more succinct representation and resolution of non-determinism in real-time concurrent models. When an urgent transition (switch) is enabled the control of the timed automaton has to perform the transition instantaneously without spending any time at that location. All the transitions originating from an urgent location are urgent transitions. To our knowledge, urgency has been first introduced by Bornot et al. with timed automata as *timed automata with deadlines* [71]. Later on many others generalized timed automata with deadlines. Among them Brabuti and Tesei proposed a model which is called *timed automata with urgent transitions* [45]. In Brabuti's model, an urgent transition must be performed within a fixed time interval from its enabling time and a urgent transition has higher priority than other non-urgent transitions enabled in the same state. Although from a language point of view timed automata with urgent transitions are not more expressive than classical timed automata, from a specification point of view the use of urgent transitions allows shorter and clearer specifications of urgent and periodic behaviors. *Variable-driven timed automata* [286] and *prioritized timed automata* [237] are two additional timed automata variants which mainly focus on urgency issues. UPPAAL [48] and many other tools use urgency for the specification of their models.

5.10 Timed Automata with Games

A classical timed automaton models only closed real-time systems (where every thing is controlled) while there exist many open real-time systems which interact with uncontrolled environments (or other systems) and these uncontrolled environments influence the behavior of those systems. A good example of real-time open systems is a pacemaker (an open system) which continuously interacts with a heart (an uncontrolled environment). Pacemaker's performance crucially depends on the exact timing of an action performed either by the system or by the environment. *Timed game automata* [41, 244] along with their controller synthesis strategies have been introduced to model such open real-time systems. The *game reachability* problem is whether the system has a strategy to reach a target state regardless of how the environment behaves. The *game minimum-time reachability* problem in timed game automata is finding the minimum time required by the system to reach a target state regardless of how the environment behaves. Both the game reachability and the game minimum-time reachability problems for timed game automata are EXPTIME-complete [98, 184, 205]. Bouyer et al. have discussed optimal strategies in priced timed game automata [77] which is a combination of timed game automata and priced timed automata.

UPPAAL TIGA [47] is a well-known tool for solving games based on timed game automata with respect to reachability and safety properties. Synthia [154, 155], a recently developed tool in Saarland University, Germany, performs verification and controller synthesis for timed game automata.

5.11 Concluding Remarks on Variants

All the variants of timed automata that have been mentioned in this survey are listed in Table 8 and Table 9, while Table 10 show the classification of these variants. The fourth columns of Table 8 and Table 9 list the year in which the variant was first proposed in the literature is given (corresponding papers are cited in the second columns). These years may not be exact proposed years as these are not confirmed by the related authors. These years have been collected according to the first associated published paper. We have listed almost eighty variants of timed automata and there may be many more. The number is impressive if one considers that the first variant was just proposed only two decades ago. Interestingly, every variant was proposed to answer a new problem (e.g., parametric timed automata for real-time protocols, probabilistic timed automata for soft-deadline, priced timed automata for resource-consumption, communicating hierarchical timed automata for the formal analyses of UML-RT/Statechart). Thus, the class of timed automata is becoming a one-stop formal solution for the model-driven development of real-time systems.

All 80 variants were first proposed between 1990 and 2010. As we don't have complete data for the current year (2011), we analyse the data of the first twenty first years (1990-2010), only. Figure 6 shows the number of variants that were proposed in each of the three 7-year time periods between 1990 and 2010 and shows increasing rate with which new variants have been developed. This increasing rate may suggest that timed automata yet to be applied into all potential kinds of real-time models.

Figure 7 presents number of variants for each class from 1990 to 2010. The column chart of Figure 7 shows the class of timed automata with resources has the highest number of variants and the class of timed automata with determinizability has the second highest number of variants. The main motivation behind the flourish of the class of timed

Table 8: Variants of Timed Automata: Part 1

Variant	**Citation**	**Authors**	**Year**
Timed Büchi Automata	[20]	Alur, Dill	1990
Timed Muller Automata	[20]	Alur, Dill	1990
Diagonal-Free Timed Automata	[20]	Alur, Dill	1990
Timed Automata with ϵ-Transitions	[20]	Alur, Dill	1990
Timed Automata with Diagonal Constraints	[20]	Alur, Dill	1990
Timed Automata without ϵ-Transitions	[20]	Alur, Dill	1990
Timed Automata with Multiplication Clock Constraints	[20]	Alur, Dill	1990
Discrete Probabilistic Timed Automata	[13]	Alur, Courcoubetis, Dill	1991
Parametric Timed Automata	[25]	Alur, Henzinger, Vardi	1993
Hybrid Automata	[19]	Alur, Courcoubetis, Henzinger, Ho	1993
Timed Safety Automata	[186]	Henzinger, Nicollin, Sifakis, Yovine	1994
Event-Clock Automata	[23]	Alur, Fix, Henzinger	1994
Event-Recording Automata	[23]	Alur, Fix, Henzinger	1994
Event-Predicting Automata	[23]	Alur, Fix, Henzinger	1994
Suspension Automata	[247]	McManis, Varaiya	1994
Rectangular Automata	[185]	Henzinger, Kopke, Puri, Varaiya	1995
Timed Game Automata	[244]	Maler, Pnueli, Sifakis	1995
Robust Timed Automata	[168]	Gupta, Henzinger, Jagadeesan	1997
Timed Automata with Deadlines	[71]	Bornot, Sifakis, Tripakis	1997
Controlled Timed Automata	[135]	Demichelis, Zielonka	1998
Recursive Event-Clock Automata	[187]	Henzinger, Raskin, Schobbens	1998
Cottbus Timed Automata	[68]	Beyer, Rust	1998
Product Interval Timed Automata	[153]	D'Souza, Thiagarajan	1999
Extended Timed Automata with Tasks	[251]	Norström, Wall, Yi	1999
Flat Timed Automata	[117]	Comon, Jurski	1999
Stopwatch Automata	[106]	Cassez, Larsen	2000
Updatable Timed Automata	[79]	Bouyer, Dufourd, Fleury, Petit	2000
Discrete Pushdown Timed Automata	[123]	Dang, Ibarra, Bultan, Kemmerer, Su	2000
Timed Automata with Periodic Clock Constraints	[114]	Choffrut, Goldwurm	2000
Timed Automata with Additive Clock Constraints	[59]	Bérard, Dufourd	2000
Timed Automata with Irrational Clock Constraints	[249]	Miller	2000
Timed Cooperating Automata	[225]	Lanotte, Maggiolo-Schettini, Peron	2000
Continuous Probabilistic Timed Automata	[218]	Kwiatkowska, Norman, Segala, Sproston	2000
Weighted Timed Automata	[30]	Alur, Torre, Pappas	2001
Priced Timed Automata	[50]	Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn	2001
Pushdown Timed Automata	[119]	Dang	2001
Past Pushdown Timed Automata	[121]	Dang, Bultan, Ibarra, Kemmerer	2001
L/U Automata	[195]	Hune, Romijn, Stoelinga, Vaandrager	2001
Uniformly-Priced Timed Automata	[49]	Behrmann, Fehnker, Hune, Larsen, Pettersson, Romijn	2001
Synchronized Concurrent Timed Automata	[298]	Wang	2001
Extended Timed Automata with Asynchronous Processes	[160]	Fersman, Pettersson, Yi	2002

Table 9: Variants of Timed Automata: Part 2

Variant	**Citation**	**Authors**	**Year**
Multi-Queue Discrete Timed Automata	[262]	Pietro, Dang	2003
Queue-Connected Discrete Timed Automata	[196]	Ibarra, Dang, Pietro	2003
Timed Automata with Input-Determined Guards	[152]	D'Souza, Tabareau	2004
Timed Automata with Urgent Transitions	[45]	Barbuti, Tesei	2004
Timed Automata with ASAP Semantics	[132]	De Wulf, Doyen, Raskin	2004
Priced Timed Game Automata	[77]	Bouyer, Cassez, Fleury, Larsen	2004
Eventual Timed Automata	[151]	D'Souza, Mohan	2005
Alternating Timed Automata	[234]	Lasota, Walukiewicz	2005
Prioritized Timed Automata	[237]	Lin, Hsiung, Huang, Chen	2005
Perturbed Timed Automata	[29]	Alur, La Torre, Madhusudan	2005
Dual-Priced Timed Automata	[231]	Larsen, Rasmussen	2005
Multi-Priced Timed Automata	[231]	Larsen, Rasmussen	2005
Phase Event Automata	[193]	Jochen Hoenicke, Patrick Maier	2005
Omega Deterministic Timed Alternating Finite Automata	[158]	Fellah, Noureddine	2006
Communicating Timed Automata	[208]	Krcál, Yi	2006
Communicating Hierarchical Timed Automata	[223]	Lanotte, Maggiolo-schettini, Milazzo, Troina	2006
Priced Probabilistic Timed Automata	[63]	Berendsen, Jasper, Jansen, Katoen	2006
Continuous Timed Automata with Input-Determined Guards	[112]	Chevalier, D'Souza, Prabhakar	2006
Timed Visibly Pushdown Automata	[157]	Emmi, Majumdar	2006
Task Automata	[159]	Fersman, Krcal, Pettersson, Yi	2007
Distributed Time-Asynchronous Automata	[144]	Dima, Lanotte	2007
Fixed Task Automata	[159]	Fersman, Krcal, Pettersson, Yi	2007
Flexible Task Automata	[159]	Fersman, Krcal, Pettersson, Yi	2007
Feedback Task Automata	[159]	Fersman, Krcal, Pettersson, Yi	2007
Non-Feedback Task Automata	[159]	Fersman, Krcal, Pettersson, Yi	2007
Counter-Free Input-Determined Timed Automata	[113]	Chevalier, D'Souza, Prabhakar	2007
Integer Reset Timed Automata	[281]	Suman, Pandya, Krishna, Manasa	2008
Distributed Timed Automata with Independently Evolving Clocks	[7]	Akshay, Bollig, Gastin, Mukund, Kumar	2008
Concavely-Priced Timed Automata	[206]	Jurdziński, Trivedi	2008
Concavely-Priced Probabilistic Timed Automata	[204]	Jurdziński, Kwiatkowska, Norman,Trivedi	2009
Interrupt Timed Automata	[61]	Bérard, Haddad	2009
Weak Alternating Timed Automata	[259]	Parys, Walukiewicz	2009
Timed P Automata	[44]	Barbuti, Maggiolo-Schettini, Milazzo, Tesei	2009
Event-Clock Visibly Pushdown Automata	[284]	Tang, Ogawa	2009
Variable Driven Timed Automata	[286]	Timo, Rollet	2010
Recursive Timed Automata	[294]	Trivedi, Wojtczak	2010
Timed Recursive State Machines	[55]	Benerecetti, Minopoli, Adriano	2010
First-Order Probabilistic Timed Automata	[161]	Fietzke,Hermanns, Weidenbach	2010
Weighted Integer Reset Timed Automata	[245]	Manasa, Krishna, Jain	2011

Table 10: Classification of the Variants of Timed Automata

Class	**Variants**
Classical Timed Automata	Timed Büchi Automata, Timed Muller Automata, Diagonal-Free Timed Automata, Timed Automata with Diagonal Constraints, Timed Automata with ϵ-Transitions, Timed Automata without ϵ-Transitions, Timed Safety Automata, Flat Timed Automata
Timed Automata with Other Clock Constraints	Timed Automata with Multiplication Clock Constraints, Timed Automata with Periodic Clock Constraints, Timed Automata with Additive Clock Constraints, Timed Automata with Irrational Clock Constraints, Parametric Timed Automata, L/U Automata
Timed Automata with Clock Updates	Updatable Timed Automata, Suspension Automata, Integer Reset Timed Automata, Weighted Integer Reset Timed Automata
Timed Automata with Other Clock Rates	Hybrid Automata, Rectangular Automata, Controlled Timed Automata, Stopwatch Automata, Distributed Time-Asynchronous Automata, Distributed Timed Automata with Independently Evolving Clocks, Interrupt Timed Automata
Timed Automata with Resources	Weighted Timed Automata, Priced Timed Automata, Uniformly-Priced Timed Automata, Dual-Priced Timed Automata, Multi-Priced Timed Automata, Priced Probabilistic Timed Automata, Extended Timed Automata with Tasks, Extended Timed Automata with Asynchronous Processes, Task Automata, Fixed Task Automata, Flexible Task Automata, Feedback Task Automata, Non-Feedback Task Automata, Concavely-Priced Timed Automata, Concavely-Priced Probabilistic Timed Automata, Timed P Automata, Weighted Integer Reset Timed Automata, Priced Timed Game Automata
Timed Automata with Probability	Discrete Probabilistic Timed Automata, Continuous Probabilistic Timed Automata, Concavely-Priced Probabilistic Timed Automata, First-Order Probabilistic Timed Automata
Timed Automata with Determinizability	Event-Clock Automata, Event-Recording Automata, Event-Predicting Automata, Eventual Timed Automata, Recursive Event-Clock Automata, Product Interval Timed Automata, Timed Automata with Input-Determined Guards, Continuous Timed Automata with Input-Determined Guards, Counter-Free Input-Determined Timed Automata, Event-Clock Visibly Pushdown Automata
Timed Automata with Self-Embedded Recursion	Recursive Event-Clock Automata, Discrete Pushdown Timed Automata, Pushdown Timed Automata, Past Pushdown Timed Automata, Timed Visibly Pushdown Automata, Event-Clock Visibly Pushdown Automata, Recursive Timed Automata, Timed Recursive State Machines
Timed Automata with Communication	Communicating Timed Automata, Communicating Hierarchical Timed Automata, Multi-Queue Discrete Timed Automata, Omega Deterministic Timed Alternating Finite Automata, Synchronized Concurrent Timed Automata, Queue-Connected Discrete Timed Automata, Phase Event Automata, Timed Cooperating Automata, Cottbus Timed Automata
Timed Automata with Succinctness	Timed Automata with Deadlines, Prioritized Timed Automata, Variable Driven Timed Automata, Timed Automata with Urgent Transitions, Alternating Timed Automata, Weak Alternating Timed Automata
Timed Automata with Robustness	Robust Timed Automata, Timed Automata with ASAP Semantics, Perturbed Timed Automata
Timed Automata with Games	Timed Game Automata, Priced Timed Game Automata

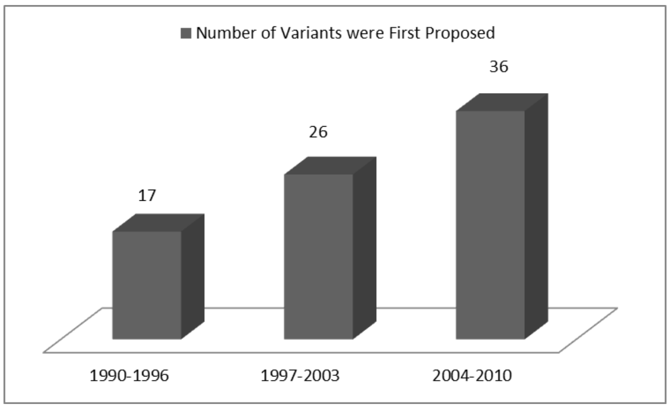


Figure 6: Numbers of Variants were First Proposed in Different Time Periods

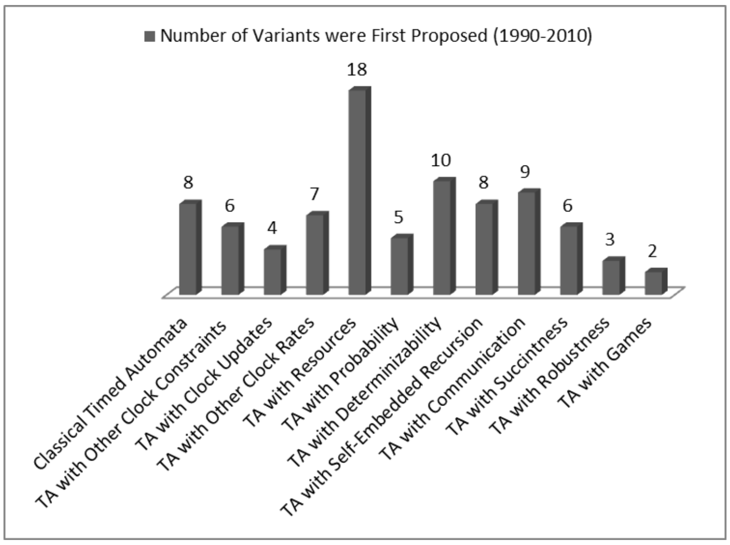


Figure 7: Number of Variants were First Proposed For Each Class During 1990-2010

automata with resources is to improve expressiveness and analysis capabilities. On the other hand, the goal of the research on timed automata with determinizability is to improve the complexity of key decision problems and to achieve more closure properties. Typically, an increase in expressive power and analysis capabilities comes at the expense of increased complexity and fewer closure properties. Figure 7 points out both of these conflicting goals are being researched most. Figure 8 shows how many variants were proposed for each category during between 1990 and 1996. We can see that the research was in a foundational stage then, because Classical Timed Automata still received most of the attention.

Figure 9 shows which classes of variants were first proposed between 1997 and 2003; this period appears to have been more exploratory, because there is no focus on a single category.

On the other hand, Figure 10 shows that between 2004 and 2010 timed automata with resources were of most interest.

According to these column charts, all the classes of timed automata other than the class of classical timed automata are still being actively explored. Researchers are still trying to develop suitable variants for different purposes.

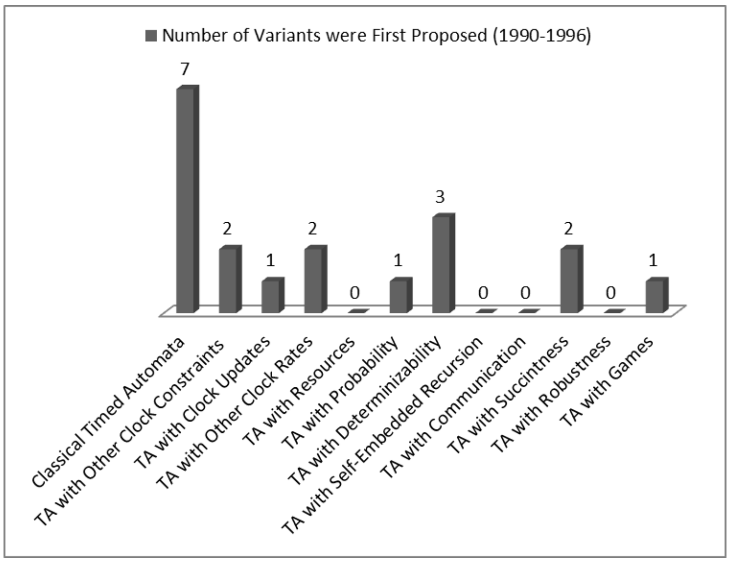


Figure 8: Numbers of Variants of Different Classes were First Proposed During 1990-1996

6 Implementation

Formal verification, control theory, and other model-based analyses lose a lot of their value when the implementation of the model is not accurate. An accurate implementation of a model conveys and contains all the verified, controlled, and analysed properties of its archetypical model. Moreover, the correspondence between the model and the accurate implementation is precisely understood.

6.1 Challenges

The semantic mismatch between the continuous-time of timed automata and the discrete-time of implementation platforms (e.g., operating system and related hardware are the implementation platform for software) makes the implementation of timed automata a hard problem. The notion of instantaneous action (which is performed in precisely zero time units) is another barrier to implement timed automata accurately as in practice no action can be executed in precisely zero time units. This semantic mismatch and practically non-instantaneous actions create (usually tiny amounts of) clock drift and violations of the guards. Furthermore, a timed automaton with zeno behavior or a behavior where actions have to be executed in less and less time (even without causing zeno behavior) cannot be implementable on a finite-speed platform [105].

MDD advocates generating code automatically from models. Automatic accurate code generation from timed automata has the potential to improve reliability and reduce costs:

i. Automatic accurate code generation from a formal model should be an essential part of safety-critical software development because code synthesis is (probably) the best way to avoid error-prone manual programming. The group of safety-critical software includes safety-critical software for nuclear plants, life-critical software for medical devices, control software for space shuttles, etc. Along with logical time, concrete time is a major concern for almost all safety-critical software. Therefore,

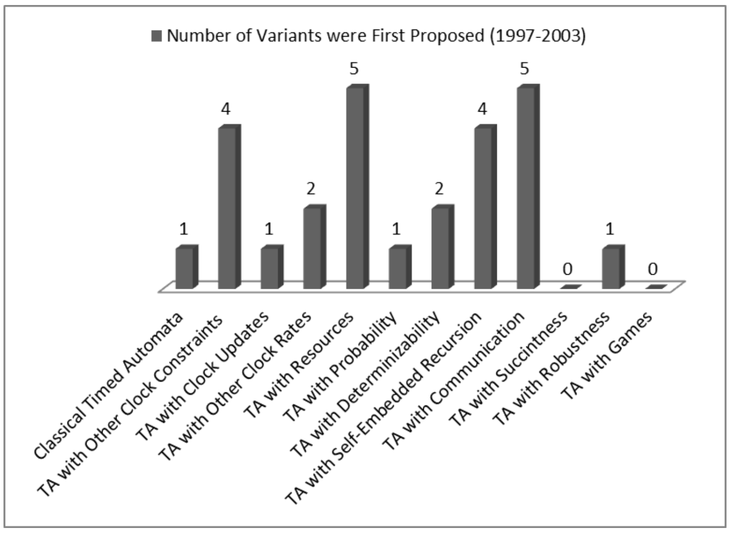


Figure 9: Numbers of Variants of Different Classes were First Proposed During 1997-2003

accurate code synthesis for timed automata has vital significance for safety-critical software development as timed automata are a prominent real-time formal model.

ii. The wages of competent programmers to write accurate real-time code is high. Moreover, accurate manual programming takes a huge amount of time compared to automatic code generation.

6.2 Solutions

Puri [267] has shown that timed automata are not *robustly safe*[37]. Puri used the *progress cycle assumption*[38] for a region-based search of the strongly connected components to compute the robust reachable set of states of a timed automaton with closed clock constraints. He showed that in a non-robustly safe timed automaton an infinitesimally small amount of timing perturbation can make bad or unsafe states reachable which are unreachable with no perturbation.

To tackle this implementation problem, a platform-dependent parametric semantics of timed automata called *almost ASAP* [129, 130, 131, 132, 133] has been proposed by Wulf et al.. Instead of instantaneous actions as in classical semantics, in the almost ASAP semantics actions have to be performed within a parametric timing tolerance. Depending on this timing tolerance the clock constraints are enlarged. This parametric timing tolerance changes according to the speed of the implementation platform. Even though the almost ASAP semantics based approach is an ad hoc approach, if a parametric timing tolerance is good for a platform then that parametric time unit is also good for any faster platform. They have also shown how to determine a suitable parametric timing tolerance for a platform by using an algorithm which is inspired by Puri's robust reachable state set finding algorithm. Both these algorithms are based on region graphs thus they are not practically efficient. Zone-based pragmatically efficient algorithms have been

[37]Robust reachability computes the set of reachable states if there exist timing perturbations up to a certain amount. In a robustly safe timed automaton no unsafe state is robustly reachable.

[38]In the progress cycle assumption it is assumed that each clock is reset at least once in every cycle of a timed automaton.

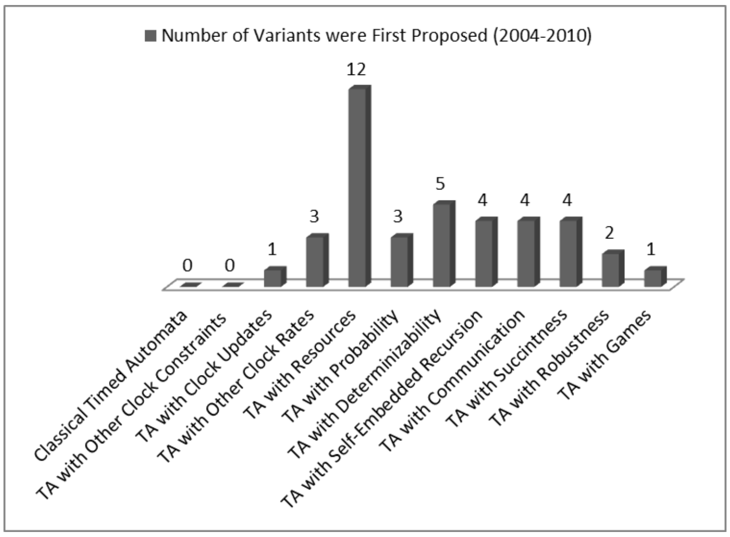


Figure 10: Numbers of Variants of Different Classes were First Proposed During 2004-2010

offered in different papers [126, 282] including one [143] by Dima where he proposed a zone-based algorithm which can also be applied to timed automata which have open clock constraints. A discussion on the decision problems and formal verifications of different kinds of robust timed automata exists in the literature [29, 88, 168, 199, 253, 283, 131].

Another group of researchers has advocated a modeling-based solution [11] to ensure implementability of timed automata with classical semantics. Their approach utilizes a network of timed automata to model the behavior of the platform. Their approach is practically inefficient because of the enormous state-space created by that network of timed automata. Unlike methods based on almost ASAP semantics, this modeling-based approach is not guaranteed to also ensure correctness on a faster platform, i.e., on a faster platform a timed automaton (which is correct on a slower platform) may perform incorrectly by producing more behavior or higher sampling rates.

An alternative way to check implementability of timed automata is to check its *samplability*. A timed automaton is samplable if its semantics is preserved under a discretization (sampling rate) of time. Sampled semantics of a timed automaton is a finite approximation of its classical semantics. The *sampling problem* of a timed automaton is to decide whether there is a sampling rate such that any untimed behaviors accepted by it with classical semantics can be also accepted by it with sampled semantics. Recently a group from Uppsala University has shown the sampling problem for timed automata is decidable [5]. In an accepted but not yet published conference paper, Bouyer et al. have addressed both the robustness and samplability of timed automata [85]. They have shown that any timed automaton can be converted into a new timed automaton with same behavior such that this new timed automaton is both robust and samplable. Sampling rates of timed automata have been discussed by a few other groups [40, 105, 210].

Massive industrial demands for accurate code synthesis from high level models (especially real-time models) are currently attracting attention from researchers both from industry and academia. Accurate code synthesis from high level models is (still) challenging because of the wide gap between the high level model and the low level code. One of the earliest successful code synthesis for timed automata (including its variants) is the code synthesis for task automata which is discussed in Section 5.4.2. Wulf et al. [129, 134] synthesized code for a *Philips audio control protocol* [72] on the LEGO MINDSTORMS platform using their proposed almost ASAP semantics. Another group used *real-time specification for Java (RTSJ)* [304] to generate Java code from timed automata [171]. They developed a prototype tool called TART[39] [170] which implements their method. Unfortunately, their technique does not perform code validation and their synthesized code does not preserve timing properties of its archetype timed automaton because of the synchrony hypothesis.

In 2010, a group of researchers from the University of Pennsylvania proposed an iterative process to generate accurate code for timed automata [200]. Instead of code synthesis, their main focus was to ensure that the implementation has the exact same timing properties as its archetypical model. This iterative process consists of a cycle of modeling, model checking, code generation, testing, and reverse engineering. In their approach, reverse engineering and testing steps use empirical (rather than formal) methods and therefore their technique cannot ensure total safety. In the same year, another group independently proposed a similar approach [214]. Readers interested in this subarea can also check some related works [179, 180, 181, 182, 194, 295, 303] which are not specific to timed automata.

7 Tools

At the beginning, timed automata were used only by verifiers and analyzers of real-time formal models. Since then, the use of timed automata based tools has been spreading to almost every aspect of real-time MDD such as controller synthesis [10, 47, 155], code synthesis [31, 170], scheduling [32], probabilistic analyses [1, 2, 64, 215], (optimal) resource analyses [52, 54, 64], parametric analyses [34, 35, 178, 207, 302], analyses for higher level models [91, 101, 137, 146, 166, 188, 207], code synthesis for higher level models [272, 273], real-time web service analyses [102, 147, 190], component-based development [4, 169, 241, 275], performance evaluation [174], test suites generation [189], black-box testing [229], multicore software analyses [239], distributed systems analyses [242, 302], mixed-reality systems analyses [137], quality assurance [279], and many more. The rich and strong theoretical foundation of timed automata makes them a good candidate to use as the underlying formal model for real-time MDD. Region-based approaches are not suitable for practical purposes because of the exponential size of the region automata. Most of the tools use zone-based approaches as these approaches are practically more efficient and scalable. Section 3.2.2 describes how zone-based techniques have changed a lot during last two decades to overcome many deficiencies. These large number of changes have made it challenging to keep these tools up to date. Only a few number of tools such as UPPAAL [48, 124], RED [297, 299, 302], and VerICS [207] have been actively maintained and have evolved for a long period of time. As zone-based techniques are now well-established, there is a very bright future waiting for timed automata based tools.

Table 11 and Table 12 list some of the timed automata based or closely related tools available in the web and literature. All these listed tools are developed and maintained mainly for research and academic purposes. The first column of these tables displays the names of the listed tools; the second column shows group names of these tools; the third column presents the description of the functionality of these tools; the fourth column exhibits citations to any publication describing these tools; at the end, the fifth and sixth

[39]To download TART please visit http://itee.uq.edu.au/~niusha/TART/

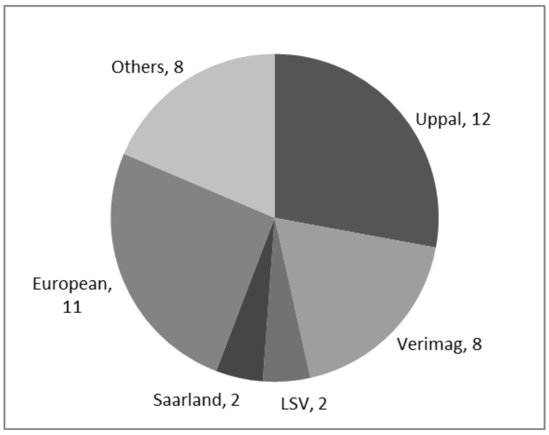
Table 11: Timed Automata Based or Related Tools: Part 1

Name	**G.**	**Description**	**Cita.**	**First Rele.**	**Latest Rele.**
UPPAAL	U	An integrated tool environment for modeling, validation and verification of real-time systems modeled as networks of timed automata, extended with data types.	[48, 124]	1995	2011
UPPAAL PRO	U	A tool for probabilistic reachability analysis for probabilistic timed automata.	[2]	2008	2009
UPPAAL PORT	U	A tool for component-based modelling, simulation, and verification of real-time and embedded systems modelled as timed automata.	[169]	2006	2008
UPPAAL CoVer	U	A tool for creating test suites from timed automata with coverage specified by coverage observers.	[189]	2005	2009
UPPAAL TIGA	U	A tool for solving games based on timed game automata with respect to reachability and safety properties.	[47]	2005	2011
UPPAAL CORA	U	A tool for cost optimal reachability analyses for priced timed automata.	[52]	2002	2006
UPPAAL TRON	U	A black-box conformance testing tool, based on timed automata, for embedded real-time software.	[229]	2004	2009
TIMES	U	A tool set for modelling, schedulability analysis, synthesis of (optimal) schedules and executable code.	[31, 32]	2002	2005
CATS	U	A tool for compositional timing and performance analysis of real-time systems modeled using timed automata and the real-time calculus.	[209]	2007	2007
SAVE IDE	U, E	A tool for design, analysis and implementation of component-based embedded real-time systems using timed automata.	[275]	2008	2009
McAiT	U, O	A timing analyzer for multicore real-time software using timed automata.	[238]	2010	2010
TASM	U, O	A tool for specification, simulation, and verification of real-time systems using timed automata.	[257]	2007	2008
Kronos	V	A tool for checking whether a timed automaton satisfies a TCTL formula.	[127]	1992	2002
SynthKro	V	A tool for controller synthesis of timed automata.	[10]	2002	2002
Open-Kronos	V	A model-checker for timed Büchi automata.	[293]	1997	2005
TAXYS	V	A timed automata based tool for the development and verification of real-time embedded systems.	[66, 116]	2000	2001
minim	V	A tool for minimization of timed automata with respect to time-abstracting bisimulation.	[292]	1996	2001
RTSpin	V	A verification tool which extends Spin with quantitative dense time features using timed Büchi automata.	[290]	1993	2004
IF	V	A validation platform which uses a specification language based on timed automata extended with discrete data variables, various communication primitives, dynamic process creation and destruction. This language is expressive enough to represent major modeling and programming languages for distributed systems such as real-time SDL and UML.	[91, 92]	1998	2004
TReX	V	A tool for reachability analysis of complex systems modelled as parametric timed automata.	[35]	2000	2006

Table 12: Timed Automata Based or Related Tools: Part 2

Name	**G.**	**Description**	**Cita.**	**First Rele.**	**Latest Rele.**
IMITATOR	L	A tool for extracting the largest safe subset of parameter values for a parametric timed automaton from a given set of parameter values.	[33, 34]	2009	2011
CMC	L	A tool for compositional model-checking of real-time systems.	[227]	1995	2004
Synthia	S	A tool for verification and controller synthesis for timed automata.	[154, 155]	2011	2011
mcpta	S	A model checker for probabilistic timed automata.	[1]	2009	2011
MCTA	E	A model checker for real-time specifications modelled as timed automata. It can find shortest error trace.	[213]	2008	2009
Rabbit	E	A model checker for Cottbus timed automata.	[67]	1999	2002
MIRELA Framework	E	A framework which uses mixed reality language MIRELA. MIRELA's compiler generates timed automata for simulation and verification of time constraints in this framework.	[137]	2007	2008
XAL	E	A web oriented programming language based on timed automata.	[103]	2008	2008
WST	E	A tool for design, validation and verification of composite Web Services with timed restrictions using timed automata.	[102]	2007	2011
VerICS	E	A (bounded, unbounded, parametric, and non-parametric) model checker for timed and multi-agent systems modeled by networks of communicating automata such as timed automata, time Petri nets. It supports model checking for model specified in high level languages such as Promela, UML, Java, and Estelle.	[207]	2003	2010
PRISM 4:0	E	A verification tool for probabilistic models including probabilistic timed automata.	[173, 215]	2010	2011
AITARTOS	E	A tool for automatic implementation of timed automata model in a real-time operating system.	[214]	2010	2011
Fortuna	E	A model checker priced probabilistic timed automata.	[64]	2010	2010
Priced-Timed Maude	E	An analyzer for priced timed automata.	[54]	2008	2008
RED	O	A model checker and simulation checker for timed automata and parametric analyzer for parametric timed automata.	[297, 299, 302]	2000	2011
HyTech	O	A model checking and analyses tool for linear hybrid automata including parametric timed automata.	[177, 178]	1995	2003
TEMPO	O	A model-checker for event recording automata.	[277]	2001	2001
DREAM	O	A distributed real-time embedded systems analyzer based on timed automata.	[242]	2005	2007
TART	O	A prototype tool to generate Java code from timed automata using RTSJ.	[170]	2010	2010
VInTiMe	O	VInTiMe is a suite of timed automata based tools (Lapsus [96], VTS [9], ObsSlice [94], and Zeus [95]) that combines high-level expressive power, unassisted property-preserving model-reduction and low-level distributed model checking power to describe and verify complex real-time systems.	[8]	2003	2009

columns show the first release year and the latest release year of these tools. All these release years have been confirmed by the respective developers other than TASM [257], TEMPO [277], HyTech [177, 178], RED [297, 299, 302], TART [170], Fortuna [64], PRISM 4:0 [173, 215], XAL [103], and McAiT [239]. Depending on origins, all these tools can be assigned to one of six different research groups: Uppaal (U), Verimag (V), LSV (L), Saarland (S), European (E), and Others (O). Uppaal group is formally a collaboration between two research groups of Uppsala University, Sweden and Aalborg University, Denmark. Tools like McAiT [239], SAVE IDE [275], TASM [257] have been put into the Uppaal group because apart from the strong influence of UPPAAL tool, some of the major developers (e.g., Wang Yi, Paul Pettersson) of these tools have strong past or present ties with the Uppaal group. All the tools of the Verimag group originated from Verimag, France research center. Tools created in Laboratory Specification and Verification (LSV), ENS Cachan, France have been put into the LSV group. The Saarland group represents two recently developed tools (mcpta [1] and Synthia [154, 155]) at Saarland University, Germany. The European group combines all the other tools which are developed by different European research groups. Tools originating from the rest of the world have been put into the Others group. From Figure 11 indicates that almost all the timed automata based research tools are developed in Europe. It is also obvious from this



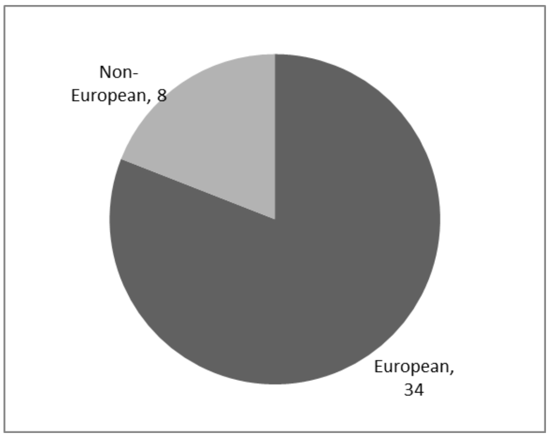


Figure 11: Numbers of Tools Created by Different Research Groups

grouping that Uppaal and Verimag are the main two driving forces of timed automata based tool research. This grouping and release dates indicate the Verimag group has not been very active in this research arena recently. A large number of tools, the diversity in tools functionality, and the long maintenance period suggest that the UPPAAL group is the most established group in this arena.

Table 13 lists timed automata based or related tools together with their major functionality, while Figure 12 and Figure 13 show how many tools have a particular purpose or functionality. The column charts of Figure 12 and Figure 13 clearly indicate that majority of timed automata based tools is used for real-time analysis and verification purposes and that tools are also beginning to be used for other purposes.

8 Conclusion

In only two decades the theory of timed automata has established itself as an integral part of real-time systems development. The beauty of its evolution is that after people

Table 13: Major Purposes of Timed Automata Based or Related Tools

Purposes	**Tools**
Black-Box Testing & Related	UPPAAL TRON, UPPAAL CoVer
Code Synthesis	TIMES, SAVE IDE, AITARTOS, TART
Controller Synthesis	UPPAAL TIGA, SynthKro, Synthia
Component-Based Development	UPPAAL PORT, SAVE IDE
Model Minimization	minim, VInTiMe
Mixed Reality Language Development	MIRELA Framework
Web Related Development	XAL, WST
Parametric Analysis and Verification	TReX, IMITATOR, VerICS, HyTech, RED
Probabilistic Analysis and Verification	UPPAAL PRO, mcpta, PRISM 4:0, Fortuna
Resource Analysis and Verification	UPPAAL CORA, TIMES, CATS, Fortuna, Priced-Timed Maude
Other Analyses and Verification	UPPAAL, TASM, McAiT, Kronos, Open-Kronos, TAXYS, RTSpin, IF, CMC, MCTA, Rabbit, VerICS, RED, TEMPO, DREAM, VInTiMe

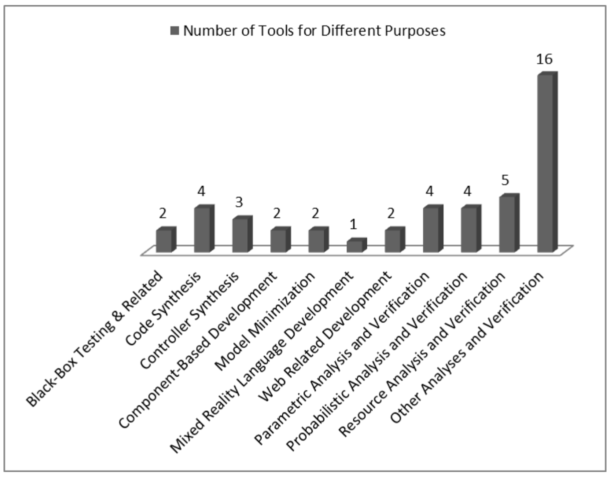


Figure 12: Number of Tools for Different Purposes

have found shortcomings and a solution is developed shortly after that. By observing the origin dates of the variants and tools, it is clear that this area is becoming more active day by day. This survey is only a snapshot of this area. It is impossible to cover the whole area in a single short article. The main motivation of this survey was to sort out this scattered arena instead of giving full detailed information about a few decision problems or variants or tools related to timed automata. To our knowledge no survey on timed automata exists which covers at least twenty variants or tools. Hopefully, this survey will be handy for a researcher who is interested in real-time model driven development.

This survey did not discuss real-time temporal logics, real-time formal verification, and real-time controller synthesis because these topics are mostly related to real-time formal models in general instead of being specific to timed automata. There are many surveys [6, 107, 87, 243, 246, 256, 291, 300, 309] which may be attractive for readers who are interested in these real-time formal methods.

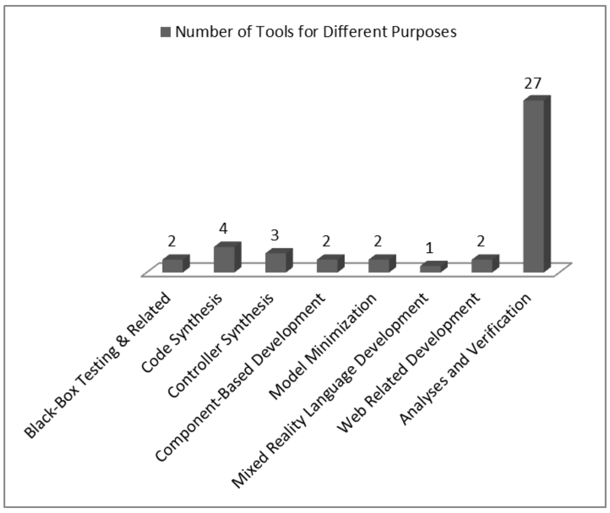


Figure 13: Number of Tools for Different Purposes

Acknowledgements

We want to thank Wang Yi, Patricia Bouyer, Hubert Comon, Catalin Dima, Stavros Tripakis, Sergio Yovine, Paul Pettersson, Holger Hermanns, Alexandre David, Jochen Hoenicke, Didier Lime, Andreas Podelski, Dirk Beyer, Fernando P. Schapachnik, Peter Csaba Ölveczky, Mihaela Sighireanu, Anders Hessel, Marius Mikucionis, Maciej Szreter, Maria Emilia Cambronero Piqueras, Francois Laroussinie, Gregorio Diaz, Marius Bozga, Bachir Djafri, Enrique Martínez López, Pavel Kučera, Hans-Jörg Peter, Séverine Sentilles, Martin Wehrle, and Franck Cassez to help us regarding their works. We also want to thank James R. Cordy and Ahmed E. Hassan for their administrative help. At the end, we want to thank Stefan D. Bruda for his help.

References

[1] mcpta. URL `http://www.modestchecker.net/`.

[2] UPPAAL PRO. URL `http://www.cs.aau.dk/~arild/uppaal-probabilistic/`.

[3] IEEE Standard for a High-Performance Serial Bus. *IEEE Std 1394-1995*, Aug. 1996.

[4] T. Abdellatif, J. Combaz, and J. Sifakis. Model-based implementation of real-time applications. In *Proceedings of the tenth ACM international conference on Embedded software*, EMSOFT '10, pages 229–238. ACM, New York, NY, USA, 2010. ISBN 978-1-60558-904-6. URL `http://doi.acm.org/10.1145/1879021.1879052`.

[5] P. A. Abdulla, P. Krcál, and W. Yi. Sampled semantics of timed automata. *Computing Research Repository*, 2010.

[6] L. Aceto and F. Laroussinie. Is your model checker on time? On the complexity of model checking for timed modal logics. *J. Log. Algebr. Program.*, 52-53:7–51, 2002.

[7] S. Akshay, B. Bollig, P. Gastin, M. Mukund, and K. Narayan Kumar. Distributed timed automata with independently evolving clocks. In *Proceedings of the 19th international conference on Concurrency Theory*, CONCUR '08, pages 82–97. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-85360-2. URL `http://dx.doi.org/10.1007/978-3-540-85361-9_10`.

[8] A. Alfonso, V. Braberman, D. Garbervetsky, N. Kicillof, A. Olivero, and F. Schapachnik. VInTiMe: Combining high-level finesse with low-level muscle to verify real-time systems. In *PRISE 2004: First International Conference on Principles of Software Engineering*. Buenos Aires, Argentina, oct 2004. URL `http://publicaciones.dc.uba.ar/Publications/2004/ABGKOS04`.

[9] A. Alfonso, V. Braberman, N. Kicillof, and A. Olivero. Visual timed event scenarios. In *Proceedings of the 26th International Conference on Software Engineering*, ICSE '04, pages 168–177. IEEE Computer Society, Washington, DC, USA, 2004. ISBN 0-7695-2163-0. URL `http://portal.acm.org/citation.cfm?id=998675.999423`.

[10] K. Altisen and S. Tripakis. Tools for controller synthesis of timed systems. In *2nd Workshop on Real-Time Tools (RT-TOOLS'2002)*, july 2002.

[11] K. Altisen and S. Tripakis. Implementation of timed automata: An issue of semantics or modeling? In *FORMATS*, pages 273–288, 2005.

[12] R. Alur. Timed automata. *Theoretical Computer Science*, 126:183–235, 1999.

[13] R. Alur, C. Courcoubetis, and D. Dill. Model-checking for probabilistic real-time systems (extended abstract). In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, pages 115–126. Springer-Verlag New York, Inc., New York, NY, USA, 1991. ISBN 0-387-54233-7. URL `http://portal.acm.org/citation.cfm?id=111713.111721`.

[14] R. Alur, C. Courcoubetis, and D. Dill. Model-checking in dense real-time. *Inf. Comput.*, 104:2–34, May 1993. ISSN 0890-5401. URL `http://portal.acm.org/citation.cfm?id=178164.178166`.

[15] R. Alur, C. Courcoubetis, D. L. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verification based on automata emptiness. In *IEEE Real-Time Systems Symposium*, pages 157–166, 1992.

[16] R. Alur, C. Courcoubetis, N. Halbwachs, D. L. Dill, and H. Wong-Toi. Minimization of timed transition systems. In *Proceedings of the Third International Conference on Concurrency Theory*, CONCUR '92, pages 340–354. Springer-Verlag, London, UK, 1992. ISBN 3-540-55822-5. URL `http://portal.acm.org/citation.cfm?id=646727.703209`.

[17] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theor. Comput. Sci.*, 138:3–34, February 1995. ISSN 0304-3975. URL `http://dl.acm.org/citation.cfm?id=202379.202381`.

[18] R. Alur, C. Courcoubetis, and T. A. Henzinger. The observational power of clocks. In *Lecture Notes in Computer Science*, volume 836/1994, pages 162–177. Springer-Verlag, 1994.

[19] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229. Springer-Verlag, London, UK, 1993. ISBN 3-540-57318-6. URL `http://portal.acm.org/citation.cfm?id=646874.709849`.

[20] R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Proceedings of the Seventeenth International Colloquium on Automata, Languages and Programming*, pages 322–335. Springer-Verlag New York, Inc., New York, NY, USA, 1990. ISBN 0-387-52826-1. URL `http://portal.acm.org/citation.cfm?id=90397.90438`.

[21] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126: 183–235, April 1994. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=180782.180519`.

[22] R. Alur, K. Etessami, S. La Torre, and D. Peled. Parametric temporal logic for "model measuring". *ACM Trans. Comput. Logic*, 2:388–407, July 2001. ISSN 1529-3785. URL `http://doi.acm.org/10.1145/377978.377990`.

[23] R. Alur, L. Fix, and T. A. Henzinger. A determinizable class of timed automata. In *Proceedings of the 6th International Conference on Computer Aided Verification*, CAV '94, pages 1–13. Springer-Verlag, London, UK, 1994. ISBN 3-540-58179-0. URL `http://portal.acm.org/citation.cfm?id=647763.735669`.

[24] R. Alur, L. Fix, and T. A. Henzinger. Event-clock automata: a determinizable class of timed automata. *Theor. Comput. Sci.*, 211:253–273, January 1999. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=293573.297329`.

[25] R. Alur, T. A. Henzinger, and M. Y. Vardi. Parametric real-time reasoning. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 592–601. ACM, New York, NY, USA, 1993. ISBN 0-89791-591-7. URL `http://doi.acm.org/10.1145/167088.167242`.

[26] R. Alur, S. Kannan, and M. Yannakakis. Communicating hierarchical state machines. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, ICAL '99, pages 169–178. Springer-Verlag, London, UK, 1999. ISBN 3-540-66224-3. URL `http://portal.acm.org/citation.cfm?id=646229.681725`.

[27] R. Alur and P. Madhusudan. Decision problems for timed automata: A survey. In M. Bernardo and F. Corradini, editors, *Formal Methods for the Design of Real-Time Systems — Revised Lectures of the International School on Formal Methods for the Design of Computer, Communication and Software Systems (SFM-RT'04)*, volume 3185 of *Lecture Notes in Computer Science*, pages 1–24. Springer-Verlag, Sept. 2004.

[28] R. Alur and P. Madhusudan. Visibly pushdown languages. In *Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC 04)*, pages 202–211. ACM Press, 2004.

[29] R. Alur, S. L. Torre, and P. Madhusudan. Perturbed timed automata. In *Hybrid Systems*, pages 70–85, 2005.

[30] R. Alur, S. L. Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control*, HSCC '01, pages 49–62. Springer-Verlag, London, UK, 2001. ISBN 3-540-41866-0. URL `http://portal.acm.org/citation.cfm?id=646881.710623`.

[31] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. TIMES - a tool for modelling and implementation of embedded systems. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS '02, pages 460–464. Springer-Verlag, London, UK, 2002. ISBN 3-540-43419-4. URL `http://portal.acm.org/citation.cfm?id=646486.694613`.

[32] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi. TIMES: a tool for schedulability analysis and code generation of real-time systems. In P. Niebert and K. G. Larsen, editors, *Proc. of FORMATS'03*, number 2791 in Lecture Notes in Computer Science, pages 60–72. Springer–Verlag, 2004.

[33] E. André. IMITATOR: A tool for synthesizing constraints on timing bounds of timed automata. In *Proceedings of the 6th International Colloquium on Theoretical Aspects of Computing*, ICTAC '09, pages 336–342. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-03465-7. URL `http://dx.doi.org/10.1007/978-3-642-03466-4_22`.

[34] É. André. IMITATOR II: A tool for solving the good parameters problem in timed automata. In *INFINITY*, pages 91–99, 2010.

[35] A. Annichini, A. Bouajjani, and M. Sighireanu. TReX: A tool for reachability analysis of complex systems. In *Proceedings of the 13th International Conference on Computer Aided Verification*, CAV '01, pages 368–372. Springer-Verlag, London, UK, 2001. ISBN 3-540-42345-1. URL `http://portal.acm.org/citation.cfm?id=647770.734247`.

[36] E. Asarin. Equations on timed languages. In *Proceedings of the First International Workshop on Hybrid Systems: Computation and Control*, pages 1–12. Springer-Verlag, London, UK, 1998. ISBN 3-540-64358-3. URL `http://portal.acm.org/citation.cfm?id=646878.710297`.

[37] E. Asarin, P. Caspi, and O. Maler. A Kleene theorem for timed automata. In *Proceedings of the 12th Annual IEEE Symposium on Logic in Computer Science*, LICS '97, pages 160–171. IEEE Computer Society, Washington, DC, USA, 1997. ISBN 0-8186-7925-5. URL `http://portal.acm.org/citation.cfm?id=788019.788856`.

[38] E. Asarin, P. Caspi, and O. Maler. Timed regular expressions. *J. ACM*, 49(2): 172–206, 2002.

[39] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Proceedings of the Second International Workshop on Hybrid Systems: Computation and Control*, HSCC '99, pages 19–30. Springer-Verlag, London, UK, 1999. ISBN 3-540-65734-7. URL `http://portal.acm.org/citation.cfm?id=646879.710314`.

[40] E. Asarin, O. Maler, and A. Pnueli. On discretization of delays in timed automata and digital circuits. In *Proceedings of the 9th International Conference on Concurrency Theory*, CONCUR '98, pages 470–484. Springer-Verlag, London, UK, 1998. ISBN 3-540-64896-8. URL `http://portal.acm.org/citation.cfm?id=646733.701304`.

[41] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proceedings of the 5th IFAC Cconference on System Structure and Control (SSC'98)*, pages 469–474. Elsevier Science, july 1998.

[42] M. Åsberg, T. Nolte, and P. Pettersson. Prototyping and code synthesis of hierarchically scheduled systems using TIMES. *Journal of Convergence (Consumer Electronics)*, 1(1):77–86, December 2010. URL `http://www.mrtc.mdh.se/index.php?choice=publications&id=2376`.

[43] C. Baier, N. Bertrand, P. Bouyer, T. Brihaye, and M. Größer. Probabilistic and topological semantics for timed automata. In *Proceedings of the 27th International Conference on Foundations of Software Technology and Theoretical Computer Science*, FSTTCS'07, pages 179–191. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-77049-6, 978-3-540-77049-7. URL `http://portal.acm.org/citation.cfm?id=1781794.1781811`.

[44] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and L. Tesei. Timed P automata. *Electronic Notes Theoretical Computer Science*, 227:21–36, January 2009. ISSN 1571-0661. URL `http://portal.acm.org/citation.cfm?id=1486279.1486497`.

[45] R. Barbuti and L. Tesei. Timed automata with urgent transitions. *Acta Inf.*, 40: 317–347, March 2004. ISSN 0001-5903. URL `http://portal.acm.org/citation.cfm?id=1006191.1006194`.

[46] D. Beauquier. On probabilistic timed automata. *Theor. Comput. Sci.*, 292:65–84, January 2003. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=642013.642018`.

[47] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and L. Didier. UPPAAL-Tiga: time for playing games! In *Proceedings of the 19th International Conference on Computer Aided Verification*, CAV'07, pages 121–125. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-73367-6. URL `http://portal.acm.org/citation.cfm?id=1770351.1770370`.

[48] G. Behrmann, A. David, and K. G. Larsen. A tutorial on UPPAAL. In M. Bernardo and F. Corradini, editors, *International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004. Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–237. Springer Verlag, 2004. URL `http://doc.utwente.nl/51010/`.

[49] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, and J. Romijn. Efficient guiding towards cost-optimality in UPPAAL. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2001, pages 174–188. Springer-Verlag, London, UK, 2001. ISBN 3-540-41865-2. URL `http://portal.acm.org/citation.cfm?id=646485.694458`.

[50] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. W. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proceedings of the 4th International Workshop on Hybrid Systems: Computation and Control*, HSCC '01, pages 147–161. Springer-Verlag, London, UK, 2001. ISBN 3-540-41866-0. URL `http://portal.acm.org/citation.cfm?id=646881.710599`.

[51] G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Priced timed automata: Algorithms and applications. In *FMCO*, pages 162–182, 2004.

[52] G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Perform. Eval. Rev.*, 32:34–40, March 2005. ISSN 0163-5999. URL `http://doi.acm.org/10.1145/1059816.1059823`.

[53] R. Bellman. *Dynamic Programming.* Princeton University Press, 1957.

[54] L. Bendiksen and P. C. Ölveczky. The priced-timed maude tool. In *Proceedings of the 3rd international conference on Algebra and coalgebra in computer science*, CALCO'09, pages 443–448. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 3-642-03740-2, 978-3-642-03740-5. URL `http://portal.acm.org/citation.cfm?id=1812941.1812981`.

[55] M. Benerecetti, S. Minopoli, and A. Peron. Analysis of timed recursive state machines. In *Proceedings of the 17th International Symposium on Temporal Representation and Reasoning*, pages 61–68, 2010.

[56] J. Bengtsson. *Clocks, DBMs and States in Timed Systems.* PhD thesis, Department of Information Technology, Uppsala University,, Uppsala, Sweeden, 2002.

[57] J. Bengtsson and W. Yi. On clock difference constraints and termination in reachability analysis of timed automata. In J. S. Dong and J. Woodcock, editors, *Proc. of ICFEM'03*, number 2885 in Lecture Notes in Computer Science. Springer–Verlag, 2003.

[58] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In W. Reisig and G. Rozenberg, editors, *In Lecture Notes on Concurrency and Petri Nets*, Lecture Notes in Computer Science vol 3098. Springer–Verlag, 2004.

[59] B. Bérard and C. Dufourd. Timed automata and additive clock constraints. *Inf. Process. Lett.*, 75:1–7, July 2000. ISSN 0020-0190. URL `http://portal.acm.org/citation.cfm?id=359506.359509`.

[60] B. Bérard, P. Gastin, and A. Petit. On the power of non-observable actions in timed automata. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science*, pages 257–268. Springer-Verlag, London, UK, 1996. ISBN 3-540-60922-9. URL `http://portal.acm.org/citation.cfm?id=646511.759229`.

[61] B. Bérard and S. Haddad. Interrupt timed automata. In *Proceedings of the 12th International Conference on Foundations of Software Science and Computational Structures: Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, FOSSACS '09, pages 197–211. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-00595-4. URL `http://dx.doi.org/10.1007/978-3-642-00596-1_15`.

[62] B. Bérard, A. Petit, V. Diekert, and P. Gastin. Characterization of the expressive power of silent transitions in timed automata. *Fundam. Inf.*, 36:145–182, November 1998. ISSN 0169-2968. URL `http://portal.acm.org/citation.cfm?id=305052.305055`.

[63] J. Berendsen, D. N. Jansen, and J.-P. Katoen. Probably on time and within budget: On reachability in priced probabilistic timed automata. In *Proceedings of the 3rd international conference on the Quantitative Evaluation of Systems*, pages 311–322. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 0-7695-2665-9. URL `http://portal.acm.org/citation.cfm?id=1173695.1173993`.

[64] J. Berendsen, D. N. Jansen, and F. W. Vaandrager. Fortuna: Model checking priced probabilistic timed automata. In *QEST*, pages 273–281, 2010.

[65] B. Berthomieu and M. Menasche. An enumerative approach for analyzing time Petri nets. In *Proceedings IFIP*, pages 41–46. Elsevier Science Publishers, 1983.

[66] V. Bertin, E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venier, D. Weil, and S. Yovine. TAXYS = Esterel + Kronos - a tool for verifying real-time properties of embedded systems. In *Conference on Decision and Control*, 2001.

[67] D. Beyer, C. Lewerentz, and A. Noack. Rabbit: A tool for BDD-based verification of real-time systems. In *CAV*, pages 122–125, 2003.

[68] D. Beyer and H. Rust. Modeling a production cell as a distributed real-time system with Cottbus timed automata. In *FBT*, pages 148–159, 1998.

[69] D. Beyer and H. Rust. Cottbus timed automata: Formal definition and semantics. In *Proceedings of the 2nd IEEE/IFIP Joint Workshop on Formal Specifications of Computer-Based Systems (FSCBS 2001*, pages 75–87, 2001.

[70] G. Booch, R. Maksimchuk, M. Engle, B. Young, J. Conallen, and K. Houston. *Object-Oriented Analysis and Design with Applications, Third Edition.* Addison-Wesley Professional, third edition, 2007. ISBN 9780201895513.

[71] S. Bornot, J. Sifakis, and S. Tripakis. Modeling urgency in timed systems. In *Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, COMPOS'97, pages 103–129. Springer-Verlag, London, UK, 1998. ISBN 3-540-65493-3. URL `http://portal.acm.org/citation.cfm?id=646738.702093`.

[72] D. Bosscher, I. Polak, and F. W. Vaandrager. Verification of an audio control protocol. In *Proceedings of the Third International Symposium Organized Jointly with the Working Group Provably Correct Systems on Formal Techniques in Real-Time and Fault-Tolerant Systems*, pages 170–192. Springer-Verlag, London, UK, 1994. ISBN 3-540-58468-4. URL `http://portal.acm.org/citation.cfm?id=646843.706645`.

[73] A. Bouajjani, S. Tripakis, and S. Yovine. On-the-fly symbolic model checking for real-time systems. In *Proceedings of the 18th IEEE Real-Time Systems Symposium*, RTSS '97, pages 232–243. IEEE Computer Society, Washington, DC, USA, 1997. ISBN 0-8186-8268-X. URL `http://portal.acm.org/citation.cfm?id=827269.828996`.

[74] P. Bouyer. Forward analysis of updatable timed automata. *Form. Methods Syst. Des.*, 24:281–320, May 2004. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=987276.987303`.

[75] P. Bouyer, T. Brihaye, V. Bruyère, and J.-F. Raskin. On the optimal reachability problem of weighted timed automata. *Form. Methods Syst. Des.*, 31:135–175, October 2007. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=1288667.1288679`.

[76] P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Form. Methods Syst. Des.*, 32:3–23, February 2008. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=1331427.1331460`.

[77] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal Strategies in Priced Timed Game Automata. In K. Lodaya and M. Mahajan, editors, *Proceedings of the 24th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160. Springer, Chennai, India, December 2004.

[78] P. Bouyer and F. Chevalier. On conciseness of extensions of timed automata. *J. Autom. Lang. Comb.*, 10:393–405, April 2005. ISSN 1430-189X. URL `http://portal.acm.org/citation.cfm?id=1375928.1375929`.

[79] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Are timed automata updatable? In *CAV*, pages 464–479, 2000.

[80] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Expressiveness of updatable timed automata. In *MFCS*, pages 232–242, 2000.

[81] P. Bouyer, C. Dufourd, E. Fleury, and A. Petit. Updatable timed automata. *Theor. Comput. Sci.*, 321:291–345, August 2004. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=1040971.1040979`.

[82] P. Bouyer, U. Fahrenberg, K. G. Larsen, and N. Markey. Quantitative analysis of real-time systems using priced timed automata. *Communications of the ACM*, 54: 78–87, Sept. 2011. URL `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BFLM-cacm11.pdf`.

[83] P. Bouyer, F. Laroussinie, and P.-A. Reynier. Diagonal constraints in timed automata: Forward analysis of timed systems. In P. Pettersson and W. Yi, editors, *Proceedings of the 3rd International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 112–126. Springer, Uppsala, Sweden, Nov. 2005. URL `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/BLR05-DBM.pdf`.

[84] P. Bouyer, K. G. Larsen, and N. Markey. Model-checking one-clock priced timed automata. In *Proceedings of the 10th International Conference on Foundations of Software Science and Computational Structures*, FOSSACS'07, pages 108–122. Springer-Verlag, Berlin, Heidelberg, 2007. URL `http://portal.acm.org/citation.cfm?id=1760037.1760048`.

[85] P. Bouyer, K. G. Larsen, N. Markey, O. Sankur, and C. Thrane. Timed automata can always be made implementable. In J.-P. Katoen and B. König, editors, *Proceedings of the 22nd International Conference on Concurrency Theory (CONCUR'11)*, volume 6901 of *Lecture Notes in Computer Science*. Springer, Aachen, Germany, Sept. 2011. To appear.

[86] P. Bouyer and N. Markey. Costs are expensive! In *FORMATS*, pages 53–68, 2007.

[87] P. Bouyer, N. Markey, J. Ouaknine, and J. Worrell. On expressiveness and complexity in real-time model checking. In *Proceedings of the 35th international colloquium on Automata, Languages and Programming, Part II*, ICALP '08, pages 124–135. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-70582-6. URL `http://dx.doi.org/10.1007/978-3-540-70583-3_11`.

[88] P. Bouyer, N. Markey, and P. Reynier. Robust model-checking of linear-time properties in timed automata. In *Latin American Theoretical INformatics*, pages 238–249, 2006.

[89] P. Bouyer and A. Petit. Decomposition and composition of timed automata. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, ICAL '99, pages 210–219. Springer-Verlag, London, UK, 1999. ISBN 3-540-66224-3. URL `http://portal.acm.org/citation.cfm?id=646229.681713`.

[90] P. Bouyer and A. Petit. A Kleene/Büchi-like theorem for clock languages. *Journal of Automata, Languages and Combinatorics*, 7(2):167–186, 2002.

[91] M. Bozga, S. Graf, L. Mounier, and I. Ober. Modeling and Verification of Real-Time Systems Using the IF Toolbox. In N. Navet, S. Merz, , and , editors, *Modeling and Verification of Real-time Systems*, chapter 10, pages 319–352. Wiley, http://www.wiley.com/, janvier 2008. URL `http://www.iste.co.uk/index.php?p=a&ACTION=View&id=195`.

[92] M. Bozga, S. Graf, I. Ober, I. Ober, and J. Sifakis. The IF toolset. In *SFM-04:RT 4th Int. School on Formal Methods for the Design of Computer, Communication and Software Systems: Real Time*, number 3185 in LNCS, June 2004.

[93] L. Bozzelli and S. La Torre. Decision problems for lower/upper bound parametric timed automata. *Form. Methods Syst. Des.*, 35:121–151, October 2009. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=1644391.1644396`.

[94] V. Braberman, D. Garbervetsky, and A. Olivero. ObsSlice: A timed automata slicer based on observers. In *CAV2004: 16th International Conference on Computer Aided Verification*, volume 3114 of *LNCS*, pages 470–474. Springer, jul 2004. URL `http://publicaciones.dc.uba.ar/Publications/2004/BGO04`.

[95] V. Braberman, A. Olivero, and F. Schapachnik. Issues in distributed timed model checking: Building Zeus. *Int. J. Softw. Tools Technol. Transf.*, 7:4–18, February 2005. ISSN 1433-2779. URL `http://portal.acm.org/citation.cfm?id=1045581.1045585`.

[96] V. A. Braberman and M. Felder. Verification of real-time designs: Combining scheduling theory with automatic formal verification. In *Proceedings of the 7th*

European software engineering conference held jointly with the 7th ACM SIGSOFT international symposium on Foundations of software engineering, ESEC/FSE-7, pages 494–510. Springer-Verlag, London, UK, 1999. ISBN 3-540-66538-2. URL `http://dx.doi.org/10.1145/318773.319266`.

[97] T. Brihaye, V. Bruyère, and J.-F. Raskin. Model-checking for weighted timed automata. In *FORMATS/FTRTFT*, pages 277–292, 2004.

[98] T. Brihaye, T. A. Henzinger, V. S. Prabhu, and J. françois Raskin. Minimum-time reachability in timed games. In *International Congress of Mathematicans*, pages 825–837, 2007.

[99] V. Bruyère, E. Dall'Olio, and J.-F. Raskin. Durations, parametric model-checking in timed automata with presburger arithmetic. In *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '03, pages 687–698. Springer-Verlag, London, UK, UK, 2003. ISBN 3-540-00623-0. URL `http://portal.acm.org/citation.cfm?id=646517.696334`.

[100] V. Bruyère and J.-F. Raskin. Real-time model-checking: Parameters everywhere. *Logical Methods in Computer Science*, 3(1), 2007.

[101] M.-E. Cambronero, G. Díaz, E. Martínez, V. Valero, and L. Tobarra. WST: a tool supporting timed composite web services model transformation. *Simulation: Transactions of the Society for Modeling and Simulation International*, 2010.

[102] M.-E. Cambronero, G. Díaz, V. Valero, and E. Martínez. Validation and verification of web services choreographies by using timed automata. *J. Log. Algebr. Program.*, 80(1):25–49, 2011.

[103] S. Campana, L. Spalazzi, and F. Spegni. XAL: A web oriented programming language based on timed-automata. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pages 862–868. IEEE Computer Society, Washington, DC, USA, 2008. ISBN 978-0-7695-3496-1. URL `http://portal.acm.org/citation.cfm?id=1486927.1487051`.

[104] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli. Languages and tools for hybrid systems design. *Found. Trends Electron. Des. Autom.*, 1:1–193, January 2006. ISSN 1551-3076. URL `http://portal.acm.org/citation.cfm?id=1166403.1166404`.

[105] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Proceedings of the 5th International Workshop on Hybrid Systems: Computation and Control*, HSCC '02, pages 134–148. Springer-Verlag, London, UK, 2002. ISBN 3-540-43321-X. URL `http://portal.acm.org/citation.cfm?id=646882.710764`.

[106] F. Cassez and K. G. Larsen. The impressive power of stopwatches. In *Proceedings of the 11th International Conference on Concurrency Theory*, CONCUR '00, pages 138–152. Springer-Verlag, London, UK, 2000. ISBN 3-540-67897-2. URL `http://portal.acm.org/citation.cfm?id=646735.701625`.

[107] F. Cassez and N. Markey. *Communicating Embedded Systems – Software and Design*, chapter Control of Timed Systems, pages 83–120. Oct. 2009. URL `http://www.iste.co.uk/index.php?f=a&ACTION=View&id=288`.

[108] M. Cavaliere and D. Sburlan. Time-independent P systems. In *Workshop on Membrane Computing*, pages 239–258, 2004.

[109] K. Cerans. Decidability of bisimulation equivalences for parallel timer processes. In *Proceedings of the Fourth International Workshop on Computer Aided Verification*, pages 302–315. Springer-Verlag, London, UK, 1993. ISBN 3-540-56496-9. URL `http://portal.acm.org/citation.cfm?id=647761.735349`.

[110] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28: 114–133, January 1981. ISSN 0004-5411. URL `http://doi.acm.org/10.1145/322234.322243`.

[111] A. M. K. Cheng. *Real-Time Systems: Scheduling, Analysis, and Verification.* John Wiley & Sons, Inc., New York, NY, USA, 1 edition, 2002. ISBN 0471184063.

[112] F. Chevalier, D. D'Souza, and P. Prabhakar. On continuous timed automata with input-determined guards. In *FSTTCS*, pages 369–380, 2006.

[113] F. Chevalier, D. D'Souza, and P. Prabhakar. Counter-free input-determined timed automata. In *Proceedings of the 5th international conference on Formal modeling and analysis of timed systems*, FORMATS'07, pages 82–97. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-75453-9, 978-3-540-75453-4. URL `http://portal.acm.org/citation.cfm?id=1779879.1779887`.

[114] C. Choffrut and M. Goldwurm. Timed automata with periodic clock constraints. *J. Autom. Lang. Comb.*, 5:371–403, October 2000. ISSN 1430-189X. URL `http://portal.acm.org/citation.cfm?id=360852.360853`.

[115] E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model checking.* MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-03270-8.

[116] E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venter, D. Weil, and S. Yovine. TAXYS: A tool for the development and verification of real-time embedded systems. In *Proceedings of the 13th International Conference on Computer Aided Verification*, CAV '01, pages 391–395. Springer-Verlag, London, UK, 2001. ISBN 3-540-42345-1. URL `http://portal.acm.org/citation.cfm?id=647770.734253`.

[117] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proceedings of the 10th International Conference on Concurrency Theory*, CONCUR '99, pages 242–257. Springer-Verlag, London, UK, 1999. ISBN 3-540-66425-4. URL `http://portal.acm.org/citation.cfm?id=646734.701462`.

[118] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Form. Methods Syst. Des.*, 1:385–415, December 1992. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=175770.175774`.

[119] Z. Dang. Binary reachability analysis of pushdown timed automata with dense clocks. In *Proceedings of the 13th International Conference on Computer Aided Verification*, CAV '01, pages 506–518. Springer-Verlag, London, UK, 2001. ISBN 3-540-42345-1. URL `http://portal.acm.org/citation.cfm?id=647770.734112`.

[120] Z. Dang. Pushdown timed automata: a binary reachability characterization and safety verification. *Theor. Comput. Sci.*, 302:93–121, June 2003. ISSN 0304-3975. URL `http://dx.doi.org/10.1016/S0304-3975(02)00743-0`.

[121] Z. Dang, T. Bultan, O. H. Ibarra, and R. A. Kemmerer. Past pushdown timed automata. In *Revised Papers from the 6th International Conference on Implementation and Application of Automata*, CIAA '01, pages 74–86. Springer-Verlag, London, UK, 2001. ISBN 3-540-00400-9. URL `http://portal.acm.org/citation.cfm?id=647268.721711`.

[122] Z. Dang, T. Bultan, O. H. Ibarra, and R. A. Kemmerer. Past pushdown timed automata and safety verification. *Theor. Comput. Sci.*, 313:57–71, February 2004. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=985565.985571`.

[123] Z. Dang, O. H. Ibarra, T. Bultan, R. A. Kemmerer, and J. Su. Binary reachability analysis of discrete pushdown timed automata. In *Proceedings of the 12th International Conference on Computer Aided Verification*, CAV '00, pages 69–84. Springer-Verlag, London, UK, 2000. ISBN 3-540-67770-4. URL `http://portal.acm.org/citation.cfm?id=647769.733961`.

[124] A. David, K. Larsen, P. Pettersson, W. Yi, and G. Behrmann. Developing UPPAAL over 15 years. *Software: Practice and Experience*, 2010.

[125] J. M. Davoren and A. NERODE. Logics for hybrid systems. *Proceedings of The IEEE*, 88:985–1010, 2000.

[126] C. Daws and P. Kordy. Symbolic robustness analysis of timed automata. In E. Asarin and P. Bouyer, editors, *Proceedings of the 4th International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'06)*, volume 4202 of *Lecture Notes in Computer Science*, pages 143–155. Springer-Verlag, Sept. 2006.

[127] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In *Proceedings of the DIMACS/SYCON Workshop on Hybrid Systems III: Verification and Control*, pages 208–219. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 3-540-61155-X. URL `http://portal.acm.org/citation.cfm?id=239587.239607`.

[128] C. Daws and S. Tripakis. Model checking of real-time reachability properties using abstractions. In *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pages 313–329. Springer-Verlag, London, UK, 1998. ISBN 3-540-64356-7. URL `http://portal.acm.org/citation.cfm?id=646482.691457`.

[129] M. De Wulf. *From Timed Models to Timed Implementations.* Thèse de doctorat, Département d'Informatique, Université Libre de Bruxelles, Belgium, December 2006.

[130] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robustness and implementability of timed automata. In Y. Lakhnech and S. Yovine, editors, *Proceedings of the*

Joint International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'04) and Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'04), volume 3253 of *Lecture Notes in Computer Science*, pages 118–133. Springer-Verlag, Sept. 2004.

[131] M. De Wulf, L. Doyen, N. Markey, and J.-F. Raskin. Robust safety of timed automata. *Form. Methods Syst. Des.*, 33:45–84, December 2008. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=1454351.1454357`.

[132] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: From timed models to timed implementations. In R. Alur and G. J. Pappas, editors, *Proceedings of the 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 296–310. Springer-Verlag, Mar.-Apr. 2004.

[133] M. De Wulf, L. Doyen, and J.-F. Raskin. Almost ASAP semantics: from timed models to timed implementations. *Formal Aspects of Computing*, 17:319–341, October 2005. ISSN 0934-5043. URL `http://portal.acm.org/citation.cfm?id=1095527.1095530`.

[134] M. De Wulf, L. Doyen, and J.-F. Raskin. Systematic implementation of real-time models. In *FM*, pages 139–156, 2005.

[135] F. Demichelis and W. Zielonka. Controlled timed automata. In *Proceedings of the 9th International Conference on Concurrency Theory*, CONCUR '98, pages 455–469. Springer-Verlag, London, UK, 1998. ISBN 3-540-64896-8. URL `http://portal.acm.org/citation.cfm?id=646733.701317`.

[136] M. Dickhöfer and T. Wilke. Timed alternating tree automata: The automata-theoretic solution to the TCTL model checking problem. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, ICAL '99, pages 281–290. Springer-Verlag, London, UK, 1999. ISBN 3-540-66224-3. URL `http://portal.acm.org/citation.cfm?id=646229.681548`.

[137] J.-Y. Didier, B. Djafri, and H. Klaudel. The MIRELA framework: modeling and analyzing mixed reality applications using timed automata. *Journal of Virtual Reality and Broadcasting*, 6(1), Feb. 2009. `urn:nbn:de:0009-6-17423,`, ISSN 1860-2037.

[138] V. Diekert, P. Gastin, and A. Petit. Removing epsilon-transitions in timed automata. In *Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science*, STACS '97, pages 583–594. Springer-Verlag, London, UK, 1997. ISBN 3-540-62616-6. URL `http://portal.acm.org/citation.cfm?id=646512.695332`.

[139] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proceedings of the International Workshop on Automatic Verification Methods for Finite State Systems*, pages 197–212. Springer-Verlag New York, Inc., New York, NY, USA, 1990. ISBN 0-387-52148-8. URL `http://portal.acm.org/citation.cfm?id=88032.88140`.

[140] C. Dima. Kleene theorems for event-clock automata. In *Proceedings of the 12th International Symposium on Fundamentals of Computation Theory*, FCT '99, pages 215–225. Springer-Verlag, London, UK, 1999. ISBN 3-540-66412-2. URL `http://portal.acm.org/citation.cfm?id=647899.740957`.

[141] C. Dima. Regular expressions with timed dominoes. In *Proceedings of the 4th International Conference on Discrete Mathematics and Theoretical Computer Science*, DMTCS'03, pages 141–154. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 3-540-40505-4. URL `http://portal.acm.org/citation.cfm?id=1783712.1783725`.

[142] C. Dima. *Timed shuffle expressions*, pages 95–109. Springer-Verlag, London, UK, 2005. ISBN 3-540-28309-9. URL `http://portal.acm.org/citation.cfm?id=1099332.1099344`.

[143] C. Dima. Dynamical properties of timed automata revisited. In *Proceedings of the 5th international conference on Formal modeling and analysis of timed systems*, FORMATS'07, pages 130–146. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-75453-9, 978-3-540-75453-4. URL `http://portal.acm.org/citation.cfm?id=1779879.1779890`.

[144] C. Dima and R. Lanotte. Distributed time-asynchronous automata. In *Proceedings of the 4th international conference on Theoretical aspects of computing*, ICTAC'07, pages 185–200. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-75290-0, 978-3-540-75290-5. URL `http://portal.acm.org/citation.cfm?id=1777259.1777272`.

[145] C. Dima and R. Lanotte. Removing all silent transitions from timed automata. In *Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS '09, pages 118–132. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-04367-3. URL `http://dx.doi.org/10.1007/978-3-642-04368-0_11`.

[146] J. S. Dong, P. Hao, S. C. Qin, J. Sun, and W. Yi. Timed automata patterns. *IEEE Transactions on Software Engineering*, 52(1), 2008. ISSN 0098-5589.

[147] J. S. Dong, Y. Liu, J. Sun, and X. Zhang. Verification of computation orchestration via timed automata. In *IEEE International Conference on Formal Engineering Methods*, pages 226–245, 2006.

[148] D. Drusinsky and D. Harel. On the power of bounded concurrency i: Finite automata. *J. ACM*, 41:517–539, May 1994. ISSN 0004-5411. URL `http://doi.acm.org/10.1145/176584.176587`.

[149] D. D'Souza. A logical characterisation of event recording automata. In *Proceedings of the 6th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, FTRTFT '00, pages 240–251. Springer-Verlag, London, UK, 2000. ISBN 3-540-41055-4. URL `http://portal.acm.org/citation.cfm?id=646846.706968`.

[150] D. D'Souza. A logical characterisation of event clock automata. *International Journal Foundations Computer Science*, 14(4):625–640, 2003.

[151] D. D'Souza and M. R. Mohan. Eventual timed automata. In *FSTTCS*, pages 322–334, 2005.

[152] D. D'Souza and N. Tabareau. On timed automata with input-determined guards. In *FORMATS/FTRTFT*, pages 68–83, 2004.

[153] D. D'Souza and P. S. Thiagarajan. Product interval automata: A subclass of timed automata. In *Proceedings of the 19th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 60–71. Springer-Verlag, London, UK, 1999. ISBN 3-540-66836-5. URL `http://portal.acm.org/citation.cfm?id=646837.708502`.

[154] R. Ehlers, R. Mattmüller, and H.-J. Peter. Combining symbolic representations for solving timed games. In *FORMATS*, pages 107–121, 2010.

[155] R. Ehlers, R. Mattmüller, and H.-J. Peter. Synthia: Verification and synthesis for timed automata. In *23rd International Conference on Computer Aided Verification, to appear*. Cliff Lodge, Snowbird, Utah, USA, July 2011.

[156] E. A. Emerson and R. J. Trefler. Parametric quantitative temporal reasoning. In *Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science*, LICS '99, pages 336–. IEEE Computer Society, Washington, DC, USA, 1999. ISBN 0-7695-0158-3. URL `http://portal.acm.org/citation.cfm?id=788021.788942`.

[157] M. Emmi and R. Majumdar. Decision problems for the verification of real-time software. In *HSCC*, pages 200–211, 2006.

[158] A. Fellah and S. Noureddine. Some succinctness properties of ω-DTAFA. In *Proceedings of the 5th WSEAS International Conference on Software Engineering, Parallel and Distributed Systems*, pages 97–103. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 2006. ISBN 960-8457-41-6. URL `http://portal.acm.org/citation.cfm?id=1365739.1365755`.

[159] E. Fersman, P. Krčál, P. Pettersson, and W. Yi. Task automata: Schedulability, decidability and undecidability. *Inf. Comput.*, 205:1149–1172, August 2007. ISSN 0890-5401. URL `http://portal.acm.org/citation.cfm?id=1274187.1274249`.

[160] E. Fersman, P. Pettersson, and W. Yi. Timed automata with asynchronous processes: Schedulability and decidability. In *Proceedings of the 8th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS '02, pages 67–82. Springer-Verlag, London, UK, UK, 2002. ISBN 3-540-43419-4. URL `http://portal.acm.org/citation.cfm?id=646486.694626`.

[161] A. Fietzke, H. Hermanns, and C. Weidenbach. Superposition-based analysis of first-order probabilistic timed automata. In *Proceedings of the 17th international conference on Logic for programming, artificial intelligence, and reasoning*, LPAR'10, pages 302–316. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-16241-X, 978-3-642-16241-1. URL `http://portal.acm.org/citation.cfm?id=1928380.1928402`.

[162] O. Finkel. On the shuffle of timed regular languages. *Bulletin of the European Association for Theoretical Computer Science*, Volume 88:182–184, February 2006.

[163] O. Finkel. Undecidable problems about timed automata. In *FORMATS*, pages 187–199, 2006.

[164] M. Fränzle. What will be eventually true of polynomial hybrid automata? In *Proceedings of the 4th International Symposium on Theoretical Aspects of Computer Software*, TACS '01, pages 340–359. Springer-Verlag, London, UK, 2001. ISBN 3-540-42736-8. URL `http://portal.acm.org/citation.cfm?id=645870.668686`.

[165] B. Gebremichael and F. Vaandrager. Specifying urgency in timed I/O automata. In *Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods*, pages 64–74. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7695-2435-4. URL `http://portal.acm.org/citation.cfm?id=1109722.1110561`.

[166] A. Gherbi and F. Khendek. Timed-automata semantics and analysis of UML/SPT models with concurrency. In *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, ISORC '07, pages 412–419. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2765-5. URL `http://dx.doi.org/10.1109/ISORC.2007.57`.

[167] R. Ghosh and C. Tomlin. Symbolic reachable set computation of piecewise affine hybrid automata and its application to biological modelling: Delta-notch protein signalling. *Systems Biology*, 1:170–183, 2004.

[168] V. Gupta, T. A. Henzinger, and R. Jagadeesan. Robust timed automata. In *Proceedings of the International Workshop on Hybrid and Real-Time Systems*, pages 331–345. Springer-Verlag, London, UK, 1997. ISBN 3-540-62600-X. URL `http://portal.acm.org/citation.cfm?id=646883.710916`.

[169] J. Håkansson, J. Carlson, A. Monot, P. Pettersson, and D. Slutej. Component-based design and analysis of embedded systems with UPPAAL PORT. In *Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis*, ATVA '08, pages 252–257. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-88386-9. URL `http://dx.doi.org/10.1007/978-3-540-88387-6_23`.

[170] N. Hakimipour, P. Strooper, and A. Wellings. TART: Timed-automata to real-time java tool. In *Proceedings of the 2010 8th IEEE International Conference on Software Engineering and Formal Methods*, SEFM '10, pages 299–309. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-4153-2. URL `http://dx.doi.org/10.1109/SEFM.2010.39`.

[171] N. Hakimipour, P. A. Strooper, and R. Duke. Exploring model-based development for the verification of real-time Java code. In *VERIFY*, 2008.

[172] D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987. URL `http://linkinghub.elsevier.com/retrieve/pii/0167642387900359`.

[173] A. Hartmanns and H. Hermanns. A modest approach to checking probabilistic timed automata. In *Proceedings of the 2009 Sixth International Conference on the Quantitative Evaluation of Systems*, QEST '09, pages 187–196. IEEE Computer

Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3808-2. URL `http://dx.doi.org/10.1109/QEST.2009.41`.

[174] M. Hendriks and M. Verhoef. Timed automata based analysis of embedded system architectures. In *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, page 179. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 1-4244-0054-6. URL `http://portal.acm.org/citation.cfm?id=1898953.1899117`.

[175] T. Henzinger, Z. Manna, and A. Pnueli. Temporal proof methodologies for real-time systems. In *Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '91, pages 353–366. ACM, New York, NY, USA, 1991. ISBN 0-89791-419-8. URL `http://doi.acm.org/10.1145/99583.99629`.

[176] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–. IEEE Computer Society, Washington, DC, USA, 1996. ISBN 0-8186-7463-6. URL `http://portal.acm.org/citation.cfm?id=788018.788803`.

[177] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to hytech. In *Proceedings of the First International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 41–71. Springer-Verlag, London, UK, 1995. ISBN 3-540-60630-0. URL `http://portal.acm.org/citation.cfm?id=646479.693768`.

[178] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model checker for hybrid systems. In *Proceedings of the 9th International Conference on Computer Aided Verification*, CAV '97, pages 460–463. Springer-Verlag, London, UK, 1997. ISBN 3-540-63166-6. URL `http://portal.acm.org/citation.cfm?id=647766.760724`.

[179] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Embedded control systems development with Giotto. *SIGPLAN Not.*, 36:64–72, August 2001. ISSN 0362-1340. URL `http://doi.acm.org/10.1145/384196.384208`.

[180] T. A. Henzinger, B. Horowitz, and C. M. Kirsch. Giotto: a time-triggered language for embedded programming. Technical report, Berkeley, CA, USA, 2001.

[181] T. A. Henzinger and C. M. Kirsch. The embedded machine: Predictable, portable real-time code. In *Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation*, PLDI '02, pages 315–326. ACM, New York, NY, USA, 2002. ISBN 1-58113-463-0. URL `http://doi.acm.org/10.1145/512529.512567`.

[182] T. A. Henzinger, C. M. Kirsch, M. A. A. Sanvido, and W. Pree. From control models to real-time code using Giotto. *IEEE Control Systems Magazine*, 23:50–64, 2003.

[183] T. A. Henzinger and P. W. Kopke. State equivalences for rectangular hybrid automata. In *Proceedings of the 7th International Conference on Concurrency Theory*, CONCUR '96, pages 530–545. Springer-Verlag, London, UK, 1996. ISBN 3-540-61604-7. URL `http://portal.acm.org/citation.cfm?id=646731.703830`.

[184] T. A. Henzinger and P. W. Kopke. Discrete-time control for rectangular hybrid automata. *Theor. Comput. Sci.*, 221:369–392, June 1999. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=309476.309523`.

[185] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? In *Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing*, STOC '95, pages 373–382. ACM, New York, NY, USA, 1995. ISBN 0-89791-718-9. URL `http://doi.acm.org.proxy.queensu.ca/10.1145/225058.225162`.

[186] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111:394–406, 1994.

[187] T. A. Henzinger, J.-F. Raskin, and P.-Y. Schobbens. The regular real-time languages. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, ICALP '98, pages 580–591. Springer-Verlag, London, UK, 1998. ISBN 3-540-64781-3. URL `http://portal.acm.org/citation.cfm?id=646252.686189`.

[188] P. Herber, J. Fellmuth, and S. Glesner. Model checking SystemC designs using timed automata. In *Proceedings of the 6th IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, CODES+ISSS '08, pages 131–136. ACM, New York, NY, USA, 2008. ISBN 978-1-60558-470-6. URL `http://doi.acm.org/10.1145/1450135.1450166`.

[189] A. Hessel and P. Pettersson. CoVer - a real-time test case generation tool. In *19th IFIP International Conference on Testing of Communicating Systems and 7th International Workshop on Formal Approaches to Testing of Software*, 2007.

[190] Y. Hirshfeld and A. Rabinovich. Logics for real time: Decidability and complexity. *Fundam. Inf.*, 62:1–28, January 2004. ISSN 0169-2968. URL `http://portal.acm.org/citation.cfm?id=1227039.1227041`.

[191] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21:666–677, August 1978. ISSN 0001-0782. URL `http://doi.acm.org/10.1145/359576.359585`.

[192] J. Hoenicke. *Combination of Processes, Data, and Time.* PhD thesis, University of Oldenburg, July 2006. URL `http://csd.Informatik.Uni-Oldenburg.DE/~skript/pub/Papers/csp-oz-dc.pd% f`.

[193] J. Hoenicke and P. Maier. Model-checking of specifications integrating processes, data and time. In *FM*, pages 465–480, 2005.

[194] X. Hu, M. Lawford, and A. Wassyng. *Formal Verification of the Implementability of Timing Requirements*, pages 119–134. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-03239-4. URL `http://portal.acm.org/citation.cfm?id=1614485.1614500`.

[195] T. Hune, J. Romijn, M. Stoelinga, and F. W. Vaandrager. Linear parametric model checking of timed automata. In *Proceedings of the 7th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, TACAS 2001,

pages 189–203. Springer-Verlag, London, UK, 2001. ISBN 3-540-41865-2. URL http://portal.acm.org/citation.cfm?id=646485.694456.

[196] O. H. Ibarra, Z. Dang, and P. S. Pietro. Verification in loosely synchronous queue-connected discrete timed automata. *Theor. Comput. Sci.*, 290:1713–1735, January 2003. ISSN 0304-3975. URL http://portal.acm.org/citation.cfm?id=781861.781881.

[197] D. Ivanov, M. Orlić, C. Seceleanu, and A. Vulgarakis. REMES tool-chain: A set of integrated tools for behavioral modeling and analysis of embedded systems. In *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ASE '10, pages 361–362. ACM, New York, NY, USA, 2010. ISBN 978-1-4503-0116-9. URL http://doi.acm.org.proxy.queensu.ca/10.1145/1858996.1859076.

[198] F. Jahanian and A. K.-L. Mok. A graph-theoretic approach for timing analysis and its implementation. *IEEE Trans. Comput.*, 36:961–975, August 1987. ISSN 0018-9340. URL http://portal.acm.org/citation.cfm?id=32415.32421.

[199] R. Jaubert and P.-A. Reynier. Quantitative robustness analysis of flat timed automata. In *FOSSACS*, pages 229–244, 2011.

[200] E. Jee, S. Wang, J. K. Kim, J. Lee, O. Sokolsky, and I. Lee. A safety-assured development approach for real-time software. 2010. URL http://repository.upenn.edu/cis_papers/430.

[201] M. Jenkins, J. Ouaknine, A. Rabinovich, and J. Worrell. Alternating timed automata over bounded time. In *Proceedings of the 2010 25th Annual IEEE Symposium on Logic in Computer Science*, LICS '10, pages 60–69. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-4114-3. URL http://dx.doi.org/10.1109/LICS.2010.45.

[202] H. E. Jensen. Model checking probabilistic real time systems. In *Proceedings of the 7th Nordic Workshop on Programming Theory*, pages 247–261. Chalmers Institute of Technology, 1996.

[203] N. D. Jones, L. H. Landweber, and Y. E. Lien. Complexity of some problems in Petri nets. *Theoretical Computer Science*, 4:277–299, 1977.

[204] M. Jurdziński, M. Kwiatkowska, G. Norman, and A. Trivedi. Concavely-priced probabilistic timed automata. In *Proceedings of the 20th International Conference on Concurrency Theory*, CONCUR 2009, pages 415–430. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-04080-1. URL http://dx.doi.org/10.1007/978-3-642-04081-8_28.

[205] M. Jurdzinski and A. Trivedi. Reachability-time games on timed automata. In *ICALP*, pages 838–849, 2007.

[206] M. Jurdziński and A. Trivedi. Concavely-priced timed automata. In *Proceedings of the 6th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS '08, pages 48–62. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-85777-8. URL http://dx.doi.org/10.1007/978-3-540-85778-5_5.

[207] M. Knapik, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, and A. Zbrzezny. Parametric model checking with VerICS. *T. Petri Nets and Other Models of Concurrency*, 4:98–120, 2010.

[208] P. Krcál and W. Yi. Communicating timed automata: The more synchronous, the more difficult to verify. In *CAV*, pages 249–262, 2006.

[209] P. Krčál, L. Mokrushin, and W. Yi. A tool for compositional analysis of timed systems by abstraction (extended abstract). In E. B. Johnsen, O. Owe, and G. Schneider, editors, *Proc. of NWPT'07, the 19th Nordic Workshop on Programming Theory, Oslo, Oct. 10-12*, 2007.

[210] P. Krčál and R. Pelánek. On sampled semantics of timed systems. In R. Ramanujam and S. Sen, editors, *Proceedings of FSTTCS'05, Hyderabad, India.*, volume 3821 of *Lecture Notes in Computer Science*, pages 310–321. Springer-Verlag, 2005.

[211] P. Krčál, M. Stigge, and W. Yi. Multi-processor schedulability analysis of preemptive real-time tasks with variable execution times. In *Proceedings of the 5th International Conference on Formal Modeling and Analysis of timed Systems*, FORMATS'07, pages 274–289. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 3-540-75453-9, 978-3-540-75453-4. URL `http://portal.acm.org/citation.cfm?id=1779879.1779899`.

[212] P. Krčál and W. Yi. Decidable and undecidable problems in schedulability analysis using timed automata. In K. Jensen and A. Podelski, editors, *Proceedings of TACAS'04, Barcelona, Spain.*, volume 2988 of *Lecture Notes in Computer Science*, pages 236–250. Springer-Verlag, 2004.

[213] S. Kupferschmid, M. Wehrle, B. Nebel, and A. Podelski. Faster than UPPAAL? In *Proceedings of the 20th international conference on Computer Aided Verification*, CAV '08, pages 552–555. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-70543-7. URL `http://dx.doi.org/10.1007/978-3-540-70545-1_53`.

[214] P. Kučera, O. Hynčica, and P. Honzík. Implementation of timed automata in a real-time operating system. In *Proceedings of World Congress on Engineering and Computer Science*, volume I, pages 56–60, October 2010. ISBN 978-988-17012-0-6.

[215] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, LNCS. Springer, 2011.

[216] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying soft deadlines with probabilistic timed automata. In *Proc. Workshop on Advances in Verification (Wave'2000)*, July 2000.

[217] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282: 101–150, June 2002. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=568395.568399`.

[218] M. Z. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Verifying quantitative properties of continuous probabilistic timed automata. In *Proceedings of the 11th*

International Conference on Concurrency Theory, CONCUR '00, pages 123–137. Springer-Verlag, London, UK, 2000. ISBN 3-540-67897-2. URL `http://portal.acm.org/citation.cfm?id=646735.701637`.

[219] M. Z. Kwiatkowska, G. Norman, and J. Sproston. Symbolic computation of maximal probabilistic reachability. In *Proceedings of the 12th International Conference on Concurrency Theory*, CONCUR '01, pages 169–183. Springer-Verlag, London, UK, 2001. ISBN 3-540-42497-0. URL `http://portal.acm.org/citation.cfm?id=646736.701779`.

[220] M. Z. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Inf. Comput.*, 205(7):1027–1077, 2007.

[221] G. Labinaz, M. M. Bayoumi, and K. Rudie. A survey of modeling and control of hybrid systems. *Annual Reviews in Control*, 21:79–92, 1997.

[222] L. Lamport. A fast mutual exclusion algorithm. *ACM Trans. Comput. Syst.*, 5:1–11, January 1987. ISSN 0734-2071. URL `http://doi.acm.org/10.1145/7351.7352`.

[223] R. Lanotte, A. Maggiolo-schettini, P. Milazzo, and A. Troina. Modeling long-running transactions with communicating hierarchical timed automata. In *Proc. of Formal Methods for Open Object-Based Distributed Systems (FMOODS'06), LNCS 4037*. Springer, 2006.

[224] R. Lanotte, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. Design and verification of long-running transactions in a timed framework. *Sci. Comput. Program.*, 73: 76–94, October 2008. ISSN 0167-6423. URL `http://portal.acm.org/citation.cfm?id=1435014.1435307`.

[225] R. Lanotte, A. Maggiolo-Schettini, and A. Peron. Timed cooperating automata. *Fundam. Inf.*, 43:153–173, August 2000. ISSN 0169-2968. URL `http://portal.acm.org/citation.cfm?id=353327.358771`.

[226] R. Lanotte, A. Maggiolo-Schettini, S. Tini, and A. Peron. Transformations of timed cooperating automata. *Fundam. Inf.*, 47:271–282, October 2001. ISSN 0169-2968. URL `http://portal.acm.org/citation.cfm?id=1220035.1220043`.

[227] F. Laroussinie and K. G. Larsen. CMC: A tool for compositional model-checking of real-time systems. In *Proceedings of the FIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XI) and Protocol Specification, Testing and Verification (PSTV XVIII)*, FORTE XI / PSTV XVIII '98, pages 439–456. Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 1998. ISBN 0-412-84760-4. URL `http://portal.acm.org/citation.cfm?id=646216.681836`.

[228] K. G. Larsen. Priced timed automata: Theory and tools. In *FSTTCS*, pages 417–425, 2009.

[229] K. G. Larsen, M. Mikucionis, B. Nielsen, and A. Skou. Testing real-time embedded software using UPPAAL-TRON: an industrial case study. In *Proceedings of the 5th ACM international conference on Embedded software*, EMSOFT '05, pages 299–306. ACM, New York, NY, USA, 2005. ISBN 1-59593-091-4. URL `http://doi.acm.org/10.1145/1086228.1086283`.

[230] K. G. Larsen, J. Pearson, C. Weise, and W. Yi. Clock difference diagrams. *Nordic J. of Computing*, 6:271–298, September 1999. ISSN 1236-6064. URL `http://portal.acm.org/citation.cfm?id=774455.774459`.

[231] K. G. Larsen and J. I. Rasmussen. Optimal conditional reachability for multi-priced timed automata. In *FoSSaCS*, pages 234–249, 2005.

[232] K. G. Larsen and J. I. Rasmussen. Optimal reachability for multi-priced timed automata. *Theor. Comput. Sci.*, 390:197–213, January 2008. ISSN 0304-3975. URL `http://portal.acm.org/citation.cfm?id=1330765.1330861`.

[233] K. G. Larsen and Y. Wang. Time-abstracted bisimulation: implicit specifications and decidability. *Inf. Comput.*, 134:75–101, May 1997. ISSN 0890-5401. URL `http://portal.acm.org/citation.cfm?id=255487.255491`.

[234] S. Lasota and I. Walukiewicz. Alternating timed automata. In V. Sassone, editor, *Proceedings of the 8th International Conference on Foundations of Software Science and Computation Structure (FoSSaCS'05)*, volume 3441 of *Lecture Notes in Computer Science*, pages 250–265. Springer-Verlag, Apr. 2005.

[235] S. Lasota and I. Walukiewicz. Alternating timed automata. *ACM Trans. Comput. Logic*, 9:10:1–10:27, April 2008. ISSN 1529-3785. URL `http://doi.acm.org/10.1145/1342991.1342994`.

[236] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1997. ISBN 0132624788.

[237] S.-W. Lin, P.-A. Hsiung, C.-H. Huang, and Y.-R. Chen. Model checking prioritized timed automata. In *ATVA*, pages 370–384, 2005.

[238] M. Lv, N. Guan, W. Yi, and G. Yu. McAiT - a timing analyzer for multicore real-time software. In *Proceedings of The 19th International Symposium on Automated Technology for Verification and Analysis (ATVA 2011)*. Taipei, Taiwan, October 2011. Tool paper, To appear.

[239] M. Lv1, N. Guan1, W. Yi1, and G. Yu. *McAiT (Rev 1.0) User Manual*, 2011. URL `http://www.neu-rtes.org:81/mcait/`.

[240] N. Lynch and H. Attiya. Using mappings to prove timing properties. In *Proceedings of the Ninth Annual ACM Symposium on Principles of Distributed Computing*, PODC '90, pages 265–280. ACM, New York, NY, USA, 1990. ISBN 0-89791-404-X. URL `http://doi.acm.org/10.1145/93385.93428`.

[241] G. Macariu and V. Cretu. Timed automata model for component-based real-time systems. In *Proceedings of the 2010 17th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, ECBS '10, pages 121–130. IEEE Computer Society, Washington, DC, USA, 2010. ISBN 978-0-7695-4005-4. URL `http://dx.doi.org/10.1109/ECBS.2010.20`.

[242] G. Madl and N. Dutt. Tutorial for the Open-source DREAM Tool. In *CECS Technical Report*, 2006.

[243] O. Maler, D. Nickovic, and A. Pnueli. Pillars of computer science. chapter Checking Temporal Properties of Discrete, Timed and Continuous Behaviors, pages 475–505. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 3-540-78126-9, 978-3-540-78126-4. URL `http://portal.acm.org/citation.cfm?id=1805839.1805865`.

[244] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *Symposium on Theoretical Aspects of Computer Science*, pages 229–242, 1995.

[245] L. Manasa, S. N. Krishna, and C. Jain. Model checking weighted integer reset timed automata. *Theory Comput. Syst.*, 48(3):648–679, 2011.

[246] N. Markey. *Verification of Embedded Systems – Algorithms and Complexity*. Mémoire d'habilitation, École Normale Supérieure de Cachan, France, Apr. 2011. URL `http://www.lsv.ens-cachan.fr/Publis/PAPERS/PDF/hdr-nm.pdf`.

[247] J. McManis and P. Varaiya. Suspension automata: A decidable class of hybrid automata. In *Proceedings of the 6th International Conference on Computer Aided Verification*, CAV '94, pages 105–117. Springer-Verlag, London, UK, 1994. ISBN 3-540-58179-0. URL `http://portal.acm.org/citation.cfm?id=647763.735660`.

[248] S. J. Mellor, A. N. Clark, and T. Futagami. Guest editors' introduction: model-driven development. *IEEE Software*, 20:14–18, 2003. ISSN 0740-7459.

[249] J. S. Miller. Decidability and complexity results for timed automata and semi-linear hybrid automata. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control*, HSCC '00, pages 296–309. Springer-Verlag, London, UK, 2000. ISBN 3-540-67259-1. URL `http://portal.acm.org/citation.cfm?id=646880.710453`.

[250] P. Niebert, S. Tripakis, and S. Yovine. Minimum-time reachability for timed automata. In *Proceedings of the 8th Mediteranean Conference on Control and Automation*, MED'2000.

[251] C. Norström, A. Wall, and W. Yi. Timed automata as task models for event-driven systems. In *Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications*, RTCSA '99, pages 182–. IEEE Computer Society, Washington, DC, USA, 1999. ISBN 0-7695-0306-3. URL `http://portal.acm.org/citation.cfm?id=519167.828781`.

[252] J. S. Ostroff. *Temporal Logic for Real Time Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1989. ISBN 0-471-92402-4.

[253] J. Ouaknine and J. Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *Proceedings of the 18th Annual IEEE Symposium on Logic in Computer Science*, pages 198–. IEEE Computer Society, Washington, DC, USA, 2003. ISBN 0-7695-1884-2. URL `http://portal.acm.org/citation.cfm?id=788023.789050`.

[254] J. Ouaknine and J. Worrell. On the decidability of metric temporal logic. In *Proceedings of the 20th Annual Symposium on Logic in Computer Science (LICS'05)*, pages 188–197. IEEE Comp. Soc. Press, July 2005.

[255] J. Ouaknine and J. Worrell. On the decidability and complexity of metric temporal logic over finite words. *Logicical Methods in Computer Science*, 3(1), Mar. 2007.

[256] J. Ouaknine and J. Worrell. Some recent results in metric temporal logic. In *Proceedings of the 6th international conference on Formal Modeling and Analysis of Timed Systems*, FORMATS '08, pages 1–13. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-85777-8. URL `http://dx.doi.org/10.1007/978-3-540-85778-5_1`.

[257] M. Ouimet and K. Lundqvist. The TASM toolset: Specification, simulation, and formal verification of real-time systems. In *Proceedings of the 19th international conference on Computer aided verification*, CAV'07, pages 126–130. Springer-Verlag, Berlin, Heidelberg, 2007. ISBN 978-3-540-73367-6. URL `http://portal.acm.org/citation.cfm?id=1770351.1770371`.

[258] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183. Springer-Verlag, London, UK, 1981. ISBN 3-540-10576-X. URL `http://portal.acm.org/citation.cfm?id=647210.720030`.

[259] P. Parys and I. Walukiewicz. Weak alternating timed automata. In *Proceedings of the 36th Internatilonal Collogquium on Automata, Languages and Programming: Part II*, ICALP '09, pages 273–284. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-02929-5. URL `http://dx.doi.org/10.1007/978-3-642-02930-1_23`.

[260] W. Penczek and B. Woźna. Towards bounded model checking for Timed Automata. In L. Czaja, editor, *Proceedings of the International Workshop on Concurrency, Specification and Programming (CS&P'01)*, pages 195–209. Warsaw University, 2001.

[261] P. Pettersson. *Modelling and Verification of Real-Time Systems Using Timed Automata: Theory and Practice.* PhD thesis, Department of Computer Systems, Uppsala University, 1999.

[262] P. S. Pietro and Z. Dang. Automatic verification of multi-queue discrete timed automata. In *Proceedings of the 9th Annual International Conference on Computing and Combinatorics*, COCOON'03, pages 159–171. Springer-Verlag, Berlin, Heidelberg, 2003. ISBN 3-540-40534-8. URL `http://portal.acm.org/citation.cfm?id=1756869.1756893`.

[263] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, Washington, DC, USA, 1977. URL `http://portal.acm.org/citation.cfm?id=1398506.1382534`.

[264] E. Posse and J. Dingel. Theory and implementation of a real-time extension to the π-calculus. In *FMOODS/FORTE*, pages 125–139, 2010.

[265] G. Păun. Computing with membranes. Technical report, 1998.

[266] G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61:108–143, August 2000. ISSN 0022-0000. URL `http://portal.acm.org/citation.cfm?id=353298.353304`.

[267] A. Puri. Dynamical properties of timed automata. In *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, FTRTFT '98, pages 210–227. Springer-Verlag, London, UK, 1998. ISBN 3-540-65003-2. URL `http://portal.acm.org/citation.cfm?id=646845.706811`.

[268] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM J. Control Optim.*, 25(1):206–230, 1987. ISSN 0363-0129.

[269] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets*. PhD thesis, Massachusetts Institute of Technology. Department of Electrical Engineering, 1973.

[270] P.-A. Reynier. Diagonal constraints handled efficiently in UPPAAL. Technical Report LSV-07-02, Laboratoire Spécification et Vérification, ENS Cachan, France, 2007.

[271] T. G. Rokicki. *Representing and modeling digital circuits*. PhD thesis, Department of Computer Science, Stanford University, Stanford, CA, USA, 1993.

[272] K. Sacha. Verification and implementation of dependable controllers. In *Proceedings of the 2008 Third International Conference on Dependability of Computer Systems DepCoS-RELCOMEX*, pages 143–151. IEEE Computer Society, Washington, DC, USA, 2008. ISBN 978-0-7695-3179-3. URL `http://portal.acm.org/citation.cfm?id=1440469.1441305`.

[273] K. Sacha. Verification and implementation of software for dependable controllers. *Int. J. Crit. Comput.-Based Syst.*, 1:238–254, February 2010. ISSN 1757-8779. URL `http://dx.doi.org/10.1504/IJCCBS.2010.031717`.

[274] C. Seceleanu, A. Vulgarakis, and P. Pettersson. REMES: A resource model for embedded systems. In *Proceedings of the 2009 14th IEEE International Conference on Engineering of Complex Computer Systems*, ICECCS '09, pages 84–94. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-0-7695-3702-3. URL `http://dx.doi.org/10.1109/ICECCS.2009.49`.

[275] S. Sentilles, A. Pettersson, D. Nystrom, T. Nolte, P. Pettersson, and I. Crnkovic. Save-IDE - a tool for design, analysis and implementation of component-based embedded systems. In *Proceedings of the 31st International Conference on Software Engineering*, ICSE '09, pages 607–610. IEEE Computer Society, Washington, DC, USA, 2009. ISBN 978-1-4244-3453-4. URL `http://dx.doi.org/10.1109/ICSE.2009.5070567`.

[276] A. C. Shaw. Communicating real-time state machines. *IEEE Trans. Softw. Eng.*, 18:805–816, September 1992. ISSN 0098-5589. URL `http://dx.doi.org/10.1109/32.159840`.

[277] M. Sorea. Tempo: A model-checker for event-recording automata. In *Proceedings of RT-TOOLS'01*. Aalborg, Denmark, August 2001. Also available as Technical Report SRI-CSL-01-04, Computer Science Laboratory, SRI International, Menlo Park, CA, 2001, http://www.csl.sri.com/papers/csl-01-04/.

[278] M. Sorea. *Verification of Real-Time Systems through Lazy Approximations*. PhD thesis, Universität Ulm, Germany, 2003. URL `http://www.informatik.uni-ulm.de/ki/Papers/sorea04-diss.pdf`.

[279] J. Steiner, K. Diethers, M. Hagner, and U. Goltz. Model based quality assurance for a robotic software architecture. In D. Schütz and F. Wahl, editors, *Robotic Systems for Handling and Assembly*, volume 67 of *Springer Tracts in Advanced Robotics*, pages 373–389. Springer Berlin / Heidelberg, 2011.

[280] P. V. Suman and P. K. Pandya. Determinization and expressiveness of integer reset timed automata with silent transitions. In *Proceedings of the 3rd International Conference on Language and Automata Theory and Applications*, LATA '09, pages 728–739. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-642-00981-5. URL `http://dx.doi.org/10.1007/978-3-642-00982-2_62`.

[281] P. V. Suman, P. K. Pandya, S. N. Krishna, and L. Manasa. Timed automata with integer resets: Language inclusion and expressiveness. In *FORMATS*, pages 78–92, 2008.

[282] M. Swaminathan and M. Franzle. A symbolic decision procedure for robust safety of timed systems. In *Proceedings of the 14th International Symposium on Temporal Representation and Reasoning*, pages 192–. IEEE Computer Society, Washington, DC, USA, 2007. ISBN 0-7695-2836-8. URL `http://portal.acm.org/citation.cfm?id=1270404.1271915`.

[283] M. Swaminathan, M. Fränzle, and J.-P. Katoen. The surprising robustness of (closed) timed automata against clock-drift. In *IFIP TCS*, pages 537–553, 2008.

[284] N. Tang and M. Ogawa. Event-clock visibly pushdown automata. In *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*, SOFSEM '09, pages 558–569. Springer-Verlag, Berlin, Heidelberg, 2009. ISBN 978-3-540-95890-1. URL `http://dx.doi.org/10.1007/978-3-540-95891-8_50`.

[285] S. Tasiran, R. Alur, R. P. Kurshan, and R. K. Brayton. Verifying abstractions of timed systems. In *Proceedings of the 7th International Conference on Concurrency Theory*, CONCUR '96, pages 546–562. Springer-Verlag, London, UK, 1996. ISBN 3-540-61604-7. URL `http://portal.acm.org/citation.cfm?id=646731.703837`.

[286] O. N. Timo and A. Rollet. Conformance testing of variable driven automata. In *Proceedings of the 8th IEEE International Workshop on Factory Communication Systems Communication in Automation*, 2010.

[287] C. Tomlin, I. Mitchell, A. M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of The IEEE*, 91:986–1001, 2003.

[288] S. Tripakis. Folk theorems on the determinization and minimization of timed automata. *Inf. Process. Lett.*, 99(6):222–226, 2006.

[289] S. Tripakis. Checking timed Büchi automata emptiness on simulation graphs. *ACM Trans. Comput. Logic*, 10:15:1–15:19, April 2009. ISSN 1529-3785. URL `http://doi.acm.org/10.1145/1507244.1507245`.

[290] S. Tripakis and C. Courcoubetis. Extending Promela and Spin for real time. In *Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 329–348. Springer-Verlag, London, UK, 1996. ISBN 3-540-61042-1. URL `http://portal.acm.org/citation.cfm?id=646480.693791`.

[291] S. Tripakis and T. Dang. *Model-based Design of Heterogeneous Systems*, chapter Modeling, Verification and Testing using Timed and Hybrid Automata. CRC Press, 2009.

[292] S. Tripakis and S. Yovine. Analysis of timed systems using time-abstracting bisimulations. *Form. Methods Syst. Des.*, 18:25–68, January 2001. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=371031.371045`.

[293] S. Tripakis, S. Yovine, and A. Bouajjani. Checking timed Büchi automata emptiness efficiently. *Form. Methods Syst. Des.*, 26:267–292, May 2005. ISSN 0925-9856. URL `http://portal.acm.org/citation.cfm?id=1084737.1084741`.

[294] A. Trivedi and D. Wojtczak. Recursive timed automata. In *Proceedings of the 8th international conference on Automated technology for verification and analysis*, ATVA'10, pages 306–324. Springer-Verlag, Berlin, Heidelberg, 2010. ISBN 3-642-15642-8, 978-3-642-15642-7. URL `http://portal.acm.org/citation.cfm?id=1927331.1927356`.

[295] J. Voeten, O. Florescu, J. Huang, and H. Corporaal. Error computation for predictable real-time software synthesis. *Simulation*, 87:334–350, April 2011. ISSN 0037-5497. URL `http://dx.doi.org/10.1177/0037549710364204`.

[296] F. Wang. Parametric timing analysis for real-time systems. *Inf. Comput.*, 130:131–150, November 1996. ISSN 0890-5401. URL `http://portal.acm.org/citation.cfm?id=246778.246780`.

[297] F. Wang. Efficient data structure for fully symbolic verification of real-time software systems. In *Proceedings of the 6th International Conference on Tools and Algorithms for Construction and Analysis of Systems: Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS 2000*, TACAS '00, pages 157–171. Springer-Verlag, London, UK, 2000. ISBN 3-540-67282-6. URL `http://portal.acm.org/citation.cfm?id=646484.691753`.

[298] F. Wang. Symbolic verification of complex real-time systems with clock-restriction diagram. In *Proceedings of the IFIP TC6/WG6.1 - 21st International Conference on Formal Techniques for Networked and Distributed Systems*, FORTE '01, pages 235–250. Kluwer, B.V., Deventer, The Netherlands, The Netherlands, 2001. ISBN 0-7923-7470-3. URL `http://portal.acm.org/citation.cfm?id=646219.682167`.

[299] F. Wang. Efficient verification of timed automata with BDD-like data structures. *Int. J. Softw. Tools Technol. Transf.*, 6:77–97, July 2004. ISSN 1433-2779. URL `http://portal.acm.org/citation.cfm?id=1014530.1014533`.

[300] F. Wang. Formal verification of timed systems: A survey and perspective. volume 92, pages 1283–1305. IEEE, August 2004.

[301] F. Wang. REDLIB for the formal verification of embedded systems. In *Proceedings of the Second International Symposium on Leveraging Applications of Formal Methods, Verification and Validation*, pages 341–346. IEEE Computer Society, Washington, DC, USA, 2006. ISBN 978-0-7695-3071-0. URL `http://portal.acm.org/citation.cfm?id=1396805.1397092`.

[302] F. Wang, L.-W. Yao, and Y.-L. Yang. Efficient verification of distributed real-time systems with broadcasting behaviors. *Real-Time Systems*, 47(4):285–318, 2011. URL `http://dx.doi.org/10.1007/s11241-011-9122-0`.

[303] A. Wassyng, M. Lawford, and X. Hu. Timing tolerances in safety-critical software. In J. Fitzgerald, I. Hayes, and A. Tarlecki, editors, *FM 2005: Formal Methods: International Symposium of Formal Methods Europe Proceedings*, volume 3582 of *LNCS*, pages 157 – 172. Springer-Verlag, Newcastle, UK, July 2005.

[304] A. Wellings. *Concurrent and Real-Time Programming in Java.* John Wiley & Sons, 2004. ISBN 047084437X.

[305] T. Wilke. *Automaten und Logiken für zeitabhängige Systeme.* Dissertation, Christian-Albrechts-Universität zu Kiel, 1994.

[306] M. Yannakakis and D. Lee. An efficient algorithm for minimizing real-time transition systems. In *Proceedings of the 5th International Conference on Computer Aided Verification*, CAV '93, pages 210–224. Springer-Verlag, London, UK, 1993. ISBN 3-540-56922-7. URL `http://portal.acm.org/citation.cfm?id=647762.735497`.

[307] W. Yi. *A Calculus of Real Time Systems.* PhD thesis, Department of Computer Science, Chalmers University of Technolog y, 1991.

[308] W. Yi, P. Pettersson, and M. Daniels. Automatic verification of real-time communicating systems by constraint-solving. In *Proceedings of the 7th IFIP WG6.1 International Conference on Formal Description Techniques VII*, pages 243–258. Chapman & Hall, Ltd., London, UK, 1995. ISBN 0-412-64450-9. URL `http://portal.acm.org/citation.cfm?id=646213.681364`.

[309] S. Yovine. Model checking timed automata. In *European Educational Forum: School on Embedded Systems*, pages 114–152, 1996.

[310] D. Zhang and R. Cleaveland. Fast on-the-fly parametric real-time model checking. In *Proceedings of the 26th IEEE International Real-Time Systems Symposium*, pages 157–166. IEEE Computer Society, Washington, DC, USA, 2005. ISBN 0-7695-2490-7. URL `http://portal.acm.org/citation.cfm?id=1106608.1106649`.