Overview of Robustness in Timed Systems

S. Akshay[1⋆], B. Bérard[2⋆], P. Bouyer[3⋆], S. Haar[3⋆], S. Haddad[3⋆], C. Jard[1⋆], D. Lime[4⋆], N. Markey[3⋆], P.-A. Reynier[5⋆], O. Sankur[3⋆], Y. Thierry-Mieg[2⋆]

[1] ENS Cachan, IRISA, Rennes, France.
E-mail: akshay.sundararaman@irisa.fr, Claude.Jard@bretagne.ens-cachan.fr

[2] Université Pierre & Marie Curie, LIP6/MoVe, CNRS UMR 7606, Paris, France.
E-mail: name@lip6.fr

[3] ENS Cachan, LSV, CNRS UMR 8643 & INRIA, Cachan, France.
E-mail: name@lsv.ens-cachan.fr

[4] École Centrale de Nantes, IRCCyN CNRS UMR 6597, Nantes, France.
E-mail: Didier.Lime@irccyn.ec-nantes.fr

[5] Aix-Marseille Université, LIF, CNRS UMR 7279, Marseille, France.
E-mail: Pierre-Alain.Reynier@lif.univ-mrs.fr

1 Introduction

Robustness requirements for timed systems. A large amount of research has been devoted to the specification, verification and synthesis of timed and hybrid systems which has led to:

– classification of problems w.r.t. decidability and complexity;
– design of efficient algorithms and appropriate data structures for verification and synthesis;
– implementation of tool suites;
– analysis of industrial case studies.

However the theoretical developments have assumed unrealistic perfect behaviors of the quantitative variables resulting in technical difficulties for the implementation stage of such systems.

Robustness aims at capturing such features, ensuring taht the a system is able to resist to small perturbations related to implementation. More precisely, in the context of timed systems, robustness addresses the problems of clock measurement inaccuracies, relative clock drifts, non instantaneous controller reactions, etc.

Challenges in robustness (section 2). The current practices adopted for the modelling and specification process do not include robustness. So The first challenge consists in integrating robustness as a key element in this stage. A second challenge should be to evaluate robustness of a system as as a function taking as input a perturbation and returning its quantitative effect on the system. This would lead to potential certification of systems. In order to reach these goals different aproaches have been proposed and are presented in the next paragraphs.

⋆ supported by project IMPRO (ANR-2010-BLAN-0317)

Languages for robustness specification (section 3). A first way for introducing a formal semantics which takes into account time perturbations consists in specifying behaviors in terms of "tubes" instead of single trajectories and study timed automata w.r.t. this semantics. There are different way to consider tube: either w.r.t. a global trajectory or at the level of transition firings.

Robust guard-based models (section 4). This approach consists in adding a quantitative parameter reflecting the precision of the clock measurement performed by the implementation. Then the semantics is appropriately adapted. In this context, given some property one asks whether there exists a precision such that the model fulfills the property. This problem is solved for the expression of the property in different temporal logics. One can also search for (near) optimal value for the required precision. In view of synthesis of robust systems, searchers have designed transformation algorithms that given some timed model produces another equivalent one which is in addition robust. Only authorizing strict guards in the transitions of the model is an alternative way to express robustness. So this approach has been also investigated.

Robust drift-based models (section 5). Guards enlargement has been combined with another type of perturbations: clock drifts where the evolving rate of clocks is prone to small variations. The same kind of problems as above are tackled but are (obviously) more difficult to solve. Thus the studies are often limited to reachability problems.

Distributed models (section 6). A more radical approach than the drift of clocks consists in considering asynchronous clocks. This is a typical feature of distributed systems which are thus good candidates for studying robustness.

Discrete-time clocks and discretization approaches (section 7). Many systems actually include "discrete-time" clocks sampling continuous-time processes. This raises specific robustness problems. For instance, given some property to be satisfied by the system, does there exist a sampling interval for the digital clocks of a controller that yields the required property. On the other hand, the analysis of robustness can be conducted through the study of equivalence of the same model (or between two different models) w.r.t. a discrete-time or a continuous-time semantics. This research has been also carried on, in the context of the tube languages.

Platform execution approaches (section 8). Instead of considering robustness at the semantical level, both the system and the execution platform are specified in the model. This is an ad hoc but practical approach that has two advantages. First the plaform can be modelled in full detail. Second standard analysis can be directly applied. In particular this approach when the platform is a standard as in the case of PLCs.

Outline of the paper. In the next sections, we successively summarize the most relevant papers related to the above approaches. Then in section 9, we propose some directions toward an unifying formal framework for robustness problems.

2 Challenges in robustness

2.1 Summary of [HS06]

Th. A. Henzinger and J. Sifakis. The Embedded Systems Design Challenge.

This theorem-free position paper starts by examining , in the context of Embedded Systems research and design, the current scientific foundations (analytical vs computational approaches) on the one hand, and the current engineering practices, with regard to their limitations. In both contexts, the articles identifies widening gaps between the existing approaches, and goes on to call for an integrated vision and practice. Then, "two demands on a solution" are discussed, namely *heterogeneity* and *constructivity.*

Citing the definition in the paper, "*Heterogeneity* is the property of embedded systems to be built from components with different characteristics." The emphasis is here on the fact that (i) the composed systems can have different execution and interaction semantics, and (ii) the abstractions underlying the models that are used in the design of composed systems have to be heterogeneous themselves. The article calls for a *metamodeling* that "preserves properties during model composition and supports meaningful analyses and transformations across heterogenerous model boundaries."

This connects to the second demand, that of *constructivity.* On the one hand, the authors ask for construction methods that are adapted to specific contexts and their constraints (hardware synthesis, architectures,protocols,...), such that component construction be computationally efficient and "guarantee correctness more or less by construction". Secondly, theories are needed for combining such results in a methodology for constructing heterogeneous composite systems. Two kinds of rules are (probably) needed according to the authors , namely compositionality and noninterference rules.

The article concludes by suggesting the following research action plan :

- Focus must shift from purely functional properties to extra-functional ones, in particular *performance* and *robustness.*
- *Performance* should be addressed by constructing dedicated components (schedulers) for resource management, taking into account performance requirements and uncertainties concerning the execution and the environment.
- *Robustness* needs to be satisfied by each component, in the sense that even under changed circumstances, required properties continue to be satisfied. Robustness requirements are transversal and variegated, including resistance to failures or attacks. The paper calls for development of "theories, methods, and tools that support the construction of robust embedded systems without resorting to [e.g. redundancy or other] ... massive, expensive overengineering.

2.2 Summary of [Hen08]

This is a *position paper* that aims at establishing embedded systems design as *the* key research domain for computer scientists, by highlighting two system characteristics hitherto mostly neglected in software design. These two challenges from the title, *predictability* and *robustness*, are discussed each in one separate section. In the first, the author advocates to ensure predictability by making systems deterministic. For the sake of clarification, he presents four possible sources of non-determinism : input nondeterminism, unobservable implementation non-determinism, don't care non-determinism and observable implementation non-determinism; only the fourth of these needs to be avoided to design predictable systems. This non-determinism amounts to omitting "implementation details that do influence the observable behaviour of the system", e.g. in cases where the output depends on the order in which several tasks are scheduled. The challenge identified as a conclusion reads as follows:

"... to build, on top of nondeterministic system implementations , system abstractions that are deterministic with regard to nonfunctional properties such as time and resource consumption."

It should be noted that probabilistic behaviour is explicitly qualified as a "general case of deterministic behaviour", and that the two challenges set forth by the paper are *not* concerned with statistical properties.

The *robustness* challenge is associated to the quest for *continuity*, in the following sense:

"A system is continuous if continuous changes in the environment or platform cannot cause discontinuous changes in the system behaviour."

The paper thus links robustness of a system with its sensitivity to perturbations, in the spirit of the robustness theory in mathematical statistics or system and control theory. As with the first challenge, the need to focus on robustness is used by the author as an argument to conclude that "embedded systems design.. offers a prime opportunity to reinvigorate Computer Science".

3 Languages for robustness specification

3.1 Summary of [GHJ97]

The paper studies the results obtained by relaxing the acceptance conditions on timed automata to accept tubes instead of standard trajectories.

A trajectory is a finite word over $\Sigma \times \mathbb{R}_{>0}$, where the positive real value d in a pair (a, d) represents the (non null) delay before the action a. The set **Traj** of trajectories can be equipped with several metrics involving only the delay or absolute time sequences of trajectories. For a metric d, the d-tube around trajectory τ with diameter ε is defined by $T_d(\tau, \varepsilon) = \{\tau' \mid d(\tau, \tau') < \varepsilon\}$. The induced topology is defined in a syandard way: a set O is open if for any trajectory $\tau \in O$, there exists $\varepsilon > 0$ such that $T_d(\tau, \varepsilon) \subseteq O$. Open sets are called tubes.

The first result establishes that all standard metrics define the same set **Tube** of tubes.

For a subset $L \subseteq \mathbf{Traj}$, the set of tubes in L is defined by: $\widetilde{L} = \{O \in \mathbf{Tube} \mid O \subseteq \overline{L}\}$, where $\overline{L}$ is the topological closure of L. Thus, $\widetilde{L}$ can be identified with the maximal tube in L which is $\overline{L}^{int}$, the interior of $\overline{L}$.

The authors use a variant of the classical timed automata, with the same expressive power. For such an automaton $\mathcal{A}$, the tube language of $\mathcal{A}$ is the set $\widetilde{[\![\mathcal{A}]\!]}$. They obtain the following results :

- The tube emptiness problem is PSPACE-complete.
- There is a timed automaton $\mathcal{A}$ which is not tube-determinizable : there is no Alur-Dill timed automaton which accepts the set $\widetilde{[\![\mathcal{A}]\!]}$.

3.2 Abstract of [HR00]

After introducing a notion of robust acceptance of a path by a timed system, the authors prove that w.r.t. this definition two classical decision problems remain undecidable: the universality problem for timed automata and the reachability problem for rectangular automata.

Let us detail this notion of robust acceptance. First the authors define a distance d between two trajectories τ, τ':

- The distance is infinite if their untimed words do not coincide.
- Otherwise their distance is the maximal time difference between the occurrences of corresponding events.

Equipped with this distance, the set of trajectories is a topologic space where the open sets are called d-tubes (in order to emphasize the distance). The standard notion of language is the based on *trajectory languages*; here a *tube language* is a subset of tubes. Given a trajectory language L, one defines its corresponding tube language $[L]$ as the set of tubes included in the closure of L (usually denoted by $\overline{L}$). Observe that the tube language $[L]$ can be identified with the maximal tube that it contains which is simply the interior of the closure of L, $(\overline{L})^{\circ}$.

Timed automata (TA) are called *open* (resp. *closed*) if the intervals defined by their initial condition, preconditions and postconditions are open (resp. closed). Then, some results from [GHJ97] are recalled

- Given a TA A, one can built a closed TA $\overline{A}$ such that $L(\overline{A}) = \overline{L(A)}$ and an open TA A° such that $[L(A^{\circ})] = [L(\overline{A})] = [L(A)]$.
- In a open TA, given an accepted trajectory τ by some path σ, there exists $\varepsilon > 0$ such that the open ball of center τ and radius ε is accepted along σ.
- Given any TA A, there is no tube O such that $L(A)$ and its complement $L(A)^c$ are dense in O.
- The *complement* of a tube language $\mathcal{L}$ is the set of tubes disjoint from the maximal tube of $\mathcal{L}$ (i.e. $\bigcup \mathcal{L}$). If L is a trajectory language such that there is no tube O with $L(A)$ and its complement $L(A)^c$ dense in O then $[L(C)]^c = [L(C)^c]$.

In order to prove the undecidability of the universability problem for tube languages of TA, one encodes the behaviour of a two-counter machine (counters C and D) as follows. Disregarding time, a configuration where the current instruction is I, the current value of counter C (resp. D) is vc (resp. vd) is encoded by the word:

$$B^{cf}B^{ins}E^{ins}IB^{C}(B^{c}E^{c})^{vc}E^{C}B^{D}(B^{d}E^{d})^{vd}E^{D}E^{cf}$$

A (possibly illegal) execution is a sequence of configurations.

We now explain the role of time. Given an interval $[a,b]$ we say that $(a+1,b+1)$ is *generated* by $[a,b]$. First one requires that the time interval of a configuration takes place inside the interval generated by the time interval of the previous configuration. Similarly one requires that the intervals corresponding to the specification of the instruction, counters C and D take place in the interval generated by the corresponding interval in the previous configuration. When a counter, say C is unchanged, the same requirements apply to the intervals corresponding to $B^{c}E^{c}$

3.3 Summary of [ATM05]

The article deals with *imperfect clocks:* With a fixed perturbation parameter $\varepsilon \in (0,1)$, a *perturbed time step* $(q,\nu) \xrightarrow{d}_{\varepsilon} (q',\nu')$ of A satisfies, for all clocks x, $d(1-\varepsilon) \leq \nu'(x) \leq \nu(x)+d(1+\varepsilon)$. Defining perturbed steps etc. in the analogous way, one obtains t the ε-perturbed language $L_{\varepsilon}(A)$ of a TA A as the collection of all timed words of A corresponding to an ε-perturbed run leading from an initial to a final state. From results by Henzinger et al. on rectangular (hybrid) automata, the authors deduce (without detailed proof) that $L_{\varepsilon}(A)$ is timed regular. For *one-clock* timed automata, a determinization for the perturbed language is given; the construction is shown to be tight for one-clock TAs but fails in the presence of 2 or more clocks: more precisely, for any $\varepsilon \in [0,1)$, an A_{ε} can be found for which the complement of A_{ε} ε-perturbed language cannot be accepted by any timed automaton.

Another result given in the paper is the undecidability of the language inclusion problem for perturbed automata . More precisely, for any pair (A,B) of TAs and fixed $\varepsilon > 0$, the problem of deciding whether $L(B) \subseteq L_{\varepsilon}(A)$ is *decidable* for 1-clock automata, yet *undecidable* in the presence of two or more clocks. The proof of the latter is sketched as follows : "Using that property that using two perturbed clocks one can require two events to be some precise distance apart, we can encode computations of Turing machines". The proof of decidability for 1-clock machines uses the determinization technique developped in the paper. It allows an application to refinement checking: Let TAs A_I (Implementation) and A_S (Specification) be given; one wishes to check whether $L(A_I) \subseteq L(A_S)$. If A_S is the product of 1-clock automata $A_1,\ldots,A_k$, check whether $L(A_I) \subseteq L_{\varepsilon}(A_1) \| L_{\varepsilon}(A_2) \ldots 1 \| L_{\varepsilon}(A_k)$, which is decidable by the previous result.

4 Robust guard-based models

4.1 Summary of [DDR05]

In this work, the author consider the problem of the implementation of timed automata. The setting considered by the authors is that of a controller reacting to input events produced by an environment.

Therefore, they first propose the so-called Almost ASAP semantics. Intuitively, it consists in firing the transitions as soon as possible (ASAP), relaxed by the time needed by the digital platform that implements the timed automaton to process the input sensors. The AASAP (standing for Almost ASAP) semantics of a controller is formally introduced, using an additional parameter Δ that represents the relaxation of the instantaneous reactivity. This semantics, quite complex, precisely describes how input events are stored, and then treated by the automaton.

The author exhibit an important property of their semantics, called *Faster is Better*, and defined as follows: if for some value Δ_1 the controller prevents the environment to enter a set of bad states, then it will also be the case for any smaller parameter value, that is for any $\Delta_2 < \Delta_1$. This property is important in the sense that it respects the intuition that if a platform correctly implements a system, a more powerful platform will also do.

To assess the realism of the AASAP semantics, the authors also define formally an implementation semantics, based on the following realistic modelization of the platform of implementation. This procedure repeatedly executes what they call an execution round:

- read current time in the clock register of the CPU
- update the list of input events (read sensors)
- evaluate guards of transitions leaving the current locations. If at least one evaluates to true, choose non deterministically one of the enabled transitions.
- start next round.

The semantics associated with this modelization involves two parameters, denoted Δ_L and Δ_P, representing respectively the maximal duration of an execution round, and the precision of the clock register of the CPU.

Then, the authors prove a simulation theorem between the two previous semantics. Given Δ, Δ_L, Δ_P such that $3\Delta_L + 4\Delta_P < \Delta$, the AASAP semantics w.r.t. Δ simulates the implementation semantics w.r.t. Δ_L and Δ_P.

The previous result entails that if one can find a positive value of Δ such that the AASAP semantics of the controller w.r.t. Δ prevents the environment to enter a set of bad states, then for any values of Δ_L and Δ_P such that $3\Delta_L + 4\Delta_P < \Delta$, the implementation semantics of the controller w.r.t. Δ_L and Δ_P also prevents the environment to enter this set of bad states. This constitutes a strong theorem of implementability.

However, it remains to be able to analyze the AASAP semantics. Therefore the authors present a last construction, which turns the AASAP semantics of an automaton $\mathcal{A}$ w.r.t. Δ into a syntactic construction of $\mathcal{A}$. More precisely, they build a new timed automaton $\mathcal{A}'$ in which guards and invariants are parameterized by Δ. They prove that the new timed automaton (equipped with its standard semantics) simulates the AASAP semantics of $\mathcal{A}$, and conversely.

In addition, in the last construction, the parameter Δ is only used to enlarge guards. This construction can thus be seen as the guard enlargement of another timed automaton. This last construction thus builds a link between the AASAP semantics and the guard enlargement operation introduced by Puri in [Pur98,Pur00].

Finally, they present how to analyze this last parameterized timed automaton using the tool HyTech, without any guarantee of termination however.

4.2 Summary of [BMR06]

This paper studies the robust model-checking problem for LTL (or more generally ω-regular) properties. The model of robustness is the one studied in [DDMR08], that is, guards are enlarged by some parameter Δ, and the problem is to decide whether there exists a bound Δ_0 such that the system with guards extended by any $\Delta \leq \Delta_0$ satisfies the property.

To solve this problem for LTL and ω-regular properties and a given TA A, we proceed as follows:

- first we construct an extended region automaton $\Gamma(A)$, which contains all standard transitions of the region automaton, and has extra transitions, called γ-transitions which roughly represents cycles that can be used to add extra behaviours in the automaton. There is a transition $(\ell, r) \xrightarrow{\gamma} (\ell, r')$ whenever $\overline{r} \cap \overline{r}' \neq \emptyset$ ($\overline{\cdot}$ is the topological closure) and furthermore (ℓ, r') is in an SCC. This extended region automaton represents the behaviours of A_Δ for $\Delta > 0$ small enough.
- then we check the property on $\Gamma(A)$. Properties that can be checked are roughly all universal ('all behaviours should satisfy the property') untimed properties, this is in particular the case of LTL. There is an equivalence between A robustly satisfies ϕ (in the sense A_Δ satisfies ϕ for Δ small enough) and $\Gamma(A)$ satisfies ϕ.

This procedure yields optimal complexity bounds for the robust model-checking problem (PSPACE for LTL).

4.3 Summary of [BMR08]

This paper is also interested in the robust model-checking problem, as proposed in [DDMR08]. In terms of decidability, this paper proves that the robust model-checking problem is decidable for Bounded-MTL and coFlat-MTL, fragments of MTL that have been considered recently. Furthermore the complexity is the same as the standard model-checking problem.

The method to prove this result is new, and relies on the construction of a CAROT (a class of channel automata) C which will behave like A_Δ. The idea is the following: we will add n clocks to the system, whose values will be separated by $\frac{1}{n}$ (clock x_i will have value $\alpha + \frac{i}{n} \mod 1$). They will not be tested by the automaton A. However for a guard $x \leq c + \Delta$ to be satisfied, it will be sufficient to check that either $x \leq c$ or $c < x < c + 1$ but at most one clock x_i will have fractional part between 0 and the fractional part of x. Of course this kind of constraints cannot be used in timed automata, but we will use channel automata to take advantage of this remark.

We can encode regions by words as follows. The region $0 < x < y < 1 \wedge 1 \leq z \leq 2 \wedge z = y + 1$ can be encoded by the word $\{x\}\{y,z\}$ and the information $\mathsf{int}(x) = \mathsf{int}(y) = 0$ and $\mathsf{int}(z) = 1$ ($\mathsf{int}(\ \cdot\ )$ denotes the integral part). When time elapses the next region to be reached is $0 < x < 1 \wedge y = 1 \wedge z = 2$, which is encoded by the word $\{x\}$ and the information $\mathsf{int}(x) = 0$, $y = 1$ and $z = 2$. Then the next region is encoded by the word $\{y,z\}\{x\}$ with the information $\mathsf{int}(x) = 0$, $\mathsf{int}(y) = 1$ and $\mathsf{int}(z) = 2$. We see here that time elapsing can be simulated by the behaviour of a channel, where we read from the head the clocks with largest fractional part and write clocks with smallest fractional part at the tail of the channel. Now the extra clocks x_i that we have mentioned above will be represented by a special symbol, say Δ on the channel. For instance, we may have the word $\Delta^2\{x\}\Delta^3\{y,z\}\Delta$ on the channel, with extra information on the integral part. An enlarged constraint $y \geq 2 - \Delta$ can now be checked on this encoding as follows: either $y \geq 2$, or $y < 2$ but $\mathsf{int}(y) = 1$ and there is at most one Δ on the right of y on the channel.

It is proven that this channel automaton simulates the enlarged automata (the more Δ's on the channel the smallest is the parameter in the enlarged guards). Using this encoding the robust model-checking problem is solved for fragments of MTL (timed properties).

4.4 Summary of [DK06]

The paper considers the follwowing problem, first introduced and solved in [Pur98] and [Pur00]: given a timed automaton $\mathcal{A}$ and a safety objective given as a set Bad of states that should be avoided, does there exist a positive value Δ_0 such that, when all guards are enlarged by Δ_0, the set Bad is not reachable.

While this problem has been solved by Puri (see also [DDMR04],[DDMR08]) using a region-based algorithm, the objective of the authors is to propose a zone-based algorithm. This question is relevant as it is well-known that for timed systems, while region-based approaches are useful to prove decidability results, they do not offer algorithms that are suitable for implementation. One thus often aims at developing an alternative algorithm based on zones which, though it may have a higher theoretical complexity, will be much more efficient in practice.

In this work, the authors consider the restricted setting of Puri (closed guards, so-called progress cycles, bounded clocks) and in addition, though it is not made explicit in their paper, they only handle the case of flat timed automata, *i.e.* such that each location of the automaton lies on at most one cycle. We will detail below why this restriction is important.

The approach of Puri is based on a precise study of cycles. Indeed, an infinitesimal guard enlargement may only have an effect through the repetitive firing of a cycle. The resulting algorithm is based on the cycles of the region graph: considering a forward reachability analysis in the region graph, every cycle in the region graph that is around a neighbour region (of what has been computed as reachable so far) is added to the set of reachable regions.

To extend this approach to zones, the authors introduce the notion of *stable zone of a cycle* ρ of the automaton. It is defined as the set of valuations that have both infinitely many successors and predecessors through the cycle ρ. It can be expressed as the intersection of the two greatest fixpoints $\nu X.\mathsf{Post}_\rho(X)$ and $\nu X.\mathsf{Pre}_\sigma(X)$, where Post_σ (resp. Pre_σ) denotes the operator computing the successors by ρ (resp. the predecessors by ρ).

They prove that the stable zone of the cycle ρ contains all the cycles in the region graph whose support is ρ^i for some $i \geq 1$. This yields the following procedure:

– compute the stable zone for each cycle ρ in the timed automaton $\mathcal{A}$
– perform a symbolic forward analysis of $\mathcal{A}$ using zones. Each time a stable zone is "touched" (*i.e.* the intersection is non empty), add the full stable zone to the set of reachable zones.

Considering this algorithm, it appears that the first step requires to enumerate all the cycles of the automaton, which can terminate only if the automaton is flat.

4.5 Summary of [JR11]

This work suscribes to the setting og guard enlargement by a parameter Δ. Most of the works in this setting aim at deciding whether there exists a positive value Δ_0 such that some property is verified.

This guard enlargement enjoys the following monotonicity property: the set of runs increases with the value of Δ. This property implies that for different kind of properties (safety, LTL), as soon as a property is verified for some parameter value Δ_0, it also holds for any parameter value $\Delta \leq \Delta_0$.

It is thus natural to consider the optimization problem raised by this monotonicity property: is it possible to compute the largest of value of Δ for which the property is verified?

In this work, the authors consider this problem for the case of safety properties. The approach proposed is based on the use of parametric zones to effectively compute the parametric reachability set of a timed automaton. Parametric zones have been introduced in different works. The parametric zones used in this work only have a single parameter (Δ), and are monotone in this parameter.

The results presented in this work do not apply to the full class of timed automata, but only to the restricted setting of flat timed automata. This restriction is due to the fact that the algorithm proposed can be seen as a parametric extension of the one of [DK06], which already holds only for flat timed automata.

More precisely, a pumping lemma is introduced, which allows to accelerate the computation of the greatest fixpoints defining the stable zone (see the summary of [DK06]). This gives an algorithm for the parametric computation of the stable zone. This is used to derive an algorithm computing the parametric reachability set, that is the reachability set of $\mathcal{A}$ enlarged by Δ, for all values of Δ. This set can be represented as finite union of parametric zones.

This computation allows to derive the optimal value for Δ (for any safety objective). In addition, it is shown that this value is a rational number.

4.6 Summary of [BMS11]

In this paper the approach using channel machines for robust model-checking is investigated further (see Section 4.3). A pumping lemma is proven for channel automata obtained via the reduction of [BMR08]: there are many Δ's on the channel, and what we pump are Δ's. This allows to obtain for instance a tighter bound for Δ_0 (mentioned in Section 4.2) than what was computed for instance in [DDMR08]. In particular, it shows that the robust model-checking for A reduces to the standard model-checking of A_{Δ_0}, and this yields an optimal PSPACE bound.

Furthermore the pumping lemma is extended to general timed automata, including those with non-progress cycles (it is worth recalling here that all previous works on robust model-checking assumed progress cycle in the automaton, which means that every clock was reset on every cycle). This implies the decidability of the robust model-checking problem for general timed automata (in PSPACE), for LTL or ω-regular properties.

4.7 Summary of [San11]

This paper is interested in checking the robustness of timed automata against guard enlargement in the sense of [DDMR08]. Rather than robust model-checking a given property, the problem solved in this paper is checking untimed language equivalence between a given timed automaton $\mathcal{A}$ and its enlargement $\mathcal{A}_\Delta$ for some $\Delta > 0$.

The main result is the decidability of this problem for timed automata with progress cycles and bounded clocks, in exponential space, and in polynomial space for a deterministic class of timed automata. The paper shows that a given timed automaton $\mathcal{A}$ preserves its untimed language for *some* enlargement parameter $\Delta > 0$ if, and only if it does so for Δ_0, which only depends on the size of the automaton. The algorithm then consists in checking untimed language equivalence between $\mathcal{A}$ and $\mathcal{A}_{\Delta_0}$.

4.8 Summary of [BLM$^+$11]

In this paper the robust implementation problem is investigated. The main result is that for any timed automaton A, we can construct another timed automaton B

which is a robust implementation of A. Robust implementation can take several meanings in this context. However one requirement is that A and B are strongly timed bisimilar. Another requirement is that behaviours in B_Δ and B are in relation: it can be that B_Δ is safe if B is, or that B and B_Δ are ϵ-bisimilar (that is, delays should be matched up to ϵ). Another result is that we can always construct sampled implementations from a timed automaton.

These constructions are based on the region automaton, with an appropriate granularity.

4.9 Summary of [SBM11]

This paper is interested in the robust implementation problem: how to modify a given timed automaton so that its semantics is preserved (in some sense) under guard enlargements? The main idea is to *shrink* the guards of the timed automaton, that is, enlarge by a negative amount, so that for small enough guard enlargement, the semantics is strictly included in that of the original automaton. Given appropriate shrinking and enlarging parameters, this always ensures that the behaviours of the implementation (new automaton) are included in the behaviours of the specification (original automaton). However, the new automaton may become blocking, or it may not contain desired behaviours. The main result is that one can decide (and compute) shrinking parameters for all guards, so that the resulting timed automaton (a) is non-blocking, or (b) it can time-abstract simulate the original automaton, or both. Problem (a) is decidable in PSPACE and in NP when the number of edges per location is bounded by a constant, while Problem (b) is decidable in EXPTIME. One can also decide in EXPTIME whether both properties can be satisfied by a set of parameters.

The algorithm is based on *parameterized shrunk zones*, that is, zones whose facets are shrunk by parameters. The new data structure introduced in order to represent these zones is difference bound matrices (DBM) with parameterized expressions using the operations max and plus. It turns out that when one applies an operation on a parameterized DBM (such as, time predecessors, intersection, normalization...), one always obtains a parameterized DBM where new parameters can always be expressed combining the old parameters using only maximization a sum. Based on this observation, the paper shows that the synthesis of shrinking parameters satisfying the desired properties can be reduced to solving equations in the max-plus algebra.

The paper also revisits the *program semantics* of [WDR04], and studies a simpler variant. A second result shown here is that when a timed automaton is *shrinkable*, that is, when non-blockingness and/or time-abstract simulation hold for a given set of shrinking parameters, then these properties are preserved by the program semantics of the shrunk timed automaton. Thus, the existence of shrinking parameters satisfying the above properties is a sufficient condition of the implementability in a detailed specific platform.

4.10 Summary of [Doy07]

Laurent Doyen. Robust parametric reachability for timed automata. Information Processing Letters 102(5):208-213, Elsevier, 2007

A parametric timed automaton or PTA consists of the ingredients of a TA plus a finite set of parameters from a common domain $\mathbb{P}$; the parameters are allowed to appear in invariants or guards of the PTA. The parametric reachability problem (PRP) is to determine whether or not for a given location l_f the set of parameter valuations κ which make l_f reachable is empty. Previous results from the literature had shown that the PRP is decidable under certain constraints, in particular if only one clock is compared to parameters, but is undecidable in general for dense time. The starting point of the present article is the question whether this undecidability is owed only to the fact that *closed* guards and invariants are admitted, i.e. equality constraints like $x = c$ or weak inequalities like $x \leq c$. To this end, the author defines *open PTAs* as PTAs in which, on the contrary, all guards and invariants are formed by conjunctions of formulas $x > c$ and $x < c$, with c either a parameter or a nonnegative integer constant. Yet the main result (theorem 2) of the paper states that PRP in dense time is undecidable even for open PTAs. In other words, PTA are "robustly indecidable" wrt the PRP: even the fact of making the system model robust by removing equality based constraints does not lift the undecidability of PRP.

The proof of theorem 2 proceeds by constructing, from a given 2-counter machine M, an open PTA A_M that simulates M. Five clocks and two parameters are sufficient for this, and A_M has as many locations as there are states in M: To every state q_i, $i = 0, \ldots, m$ of M corresponds a location l_i of A_M, such that l_m is reachable iff M halts (i.e. reaches q_m). The theorem thus covers the class of open PTA with ≥ 2 parameters and ≥ 5 clocks; whether or not undecidability still holds with fewer clocks and parameters is left open.

4.11 Summary of [CHP08]

This paper is related to two-player concurrent timed games with parity objectives. First recall that in this setting, at each turn, the two players independentaly propose a time delay and an action, and the action with the action with the shorter delay is chosen.

The paper contains a first part in which are presented results on the complexity of these games. These results are based on an original reduction of these games to turn-based, finite-state parity games (this reduction is based on the region graph construction). As this is not related to robustness, we do not detail more this part.

In a second part, the authors are interested in the robustness of the winning strategies they synthesize. The definitions they consider perturb the strategy in two different manners:

- a *jitter* ε_j: if player 1 proposes to play action a after a delay $d \geq 0$, then player 2 can pick another delay $d' \geq 0$ such that $|d - d'| \leq \varepsilon_j$

– a *response time* ε_r: player 1 must propose delays $d \geq \max(0, \varepsilon_r - \delta_{env})$, where δ_{env} is the delay since the last action of the environment

Intuitively, the jitter corresponds to the precision of the controller in the delays it chooses, and the response time represents the delay that must exist between an action of the environment and the response of the controller.

The authors introduce two different notions of robust strategy:

– a strategy is limit-robust if each play can be perturbed by a positive jitter (but it may be a different jitter for each play)
– a strategy is bounded-robust for a jitter ε_j and a response-time ε_r if its moves satisfy the response-time condition of ε_r, and if they can be perturbed by the jitter ε_j

Then the authors present two constructions to reduce the existence of limit-robust (resp. ε_j-ε_r bounded-robust) strategies to that of standard strategies. More precisely, given a timed game $\mathcal{G}$:

– in the first construction, they build a new timed game $\mathcal{G}'$ such that player 1 has a winning strategy in $\mathcal{G}'$ if, anf only if, he has a limit-robust winning strategy in $\mathcal{G}$,
– in the second construction, given two values ε_j, ε_r, they build a new timed game $\mathcal{G}''_{\varepsilon_j,\varepsilon_r}$ such that player 1 has a winning strategy in $\mathcal{G}''_{\varepsilon_j,\varepsilon_r}$ if, and only if, he has an ε_j-ε_r bounded-robust winning strategy in $\mathcal{G}$.

Finally, note, as mentioned by the authors in their paper, that "the question of the existence of a lower bound on the jitter for which a game can be won with a bounded-robust strategy remains open". The results presented in the paper do not give clues on how to solve this problem, as the jitter ε_j and the response-time ε_r are explicitely used in the second construction.

5 Robust drift-based models

5.1 Summary of [DDMR04] and [DDMR08]

The second paper is a long version of the first one, with full (and improved) proofs. Both papers build on the earlier papers [Pur98] and [Pur00], in which the following *parametric-robustness* question is raised and solved: given a timed automaton $\mathcal{A}$, does there exist positive values for ε_0 and Δ_0 such that $\mathcal{A}$, under an enlarged and drifting semantics where guards are enlarged by $\Delta < \Delta_0$ and clocks drift by $\varepsilon < \varepsilon_0$, remains safe.

While the paper [Pur00] first focuses on clock drifts only, and then extends the results to the general case, the paper [DDMR08] focuses on guard enlargement, and then extend to the general case. The techniques used in both papers are very similar (and are sketched below), but the proofs in [Pur00] are somewhat sketchy (see the proof of Lemma 6.6) and sometimes wrong (Lemma 6.4, corrected in [DDMR08, Lemma 16]).

The main result of the papers are the following:

– under the enlarged semantics (guard enlargement, drifts, or both), and under some technical restrictions (especially to timed automata for which any cycle in the region automaton resets all the clocks—this is the so-called *progress-cycle* assumption), the set of configurations that are reachable *for any positive values of the parameters* is region-definable. Moreover, this set is the same if considering only guard enlargement, or only clock drifts, or both.
– that set can be computed in polynomial space, by adding new transitions to the region automaton (the presentation as an extension of the region automaton was only made explicit in [BMR06], but it directly follows from the results in these papers). We add a new transition from a region (l, r) to a region (l, r') whenever there is a cycle (in the region graph) on (l, r'), and r and r' are neighbour regions.
– When the set of configurations that are reachable for any positive values of the parameters does not intersect a (zone-)set of bad configurations, then we can effectively build positive values of the parameters for which this is the case. In other terms,

$$\bigcap_{\Delta>0} \mathsf{Reach}(\mathcal{A}_\Delta) \cap B = \varnothing \quad \Leftrightarrow \quad \exists \Delta > 0.\ \mathsf{Reach}(\mathcal{A}_\Delta) \cap B = \varnothing.$$

The techniques used to prove these results are as follows:

– the analysis of *limit cycles* in the timed automaton, which are cycles in the timed automaton (hence returning back to the exact initial configuration) cycling (possibly several times) along a cycle of the region automaton. Given a cycle p in the region automaton, L_p is the set of points in the initial region p_0 of p which have a path (following p, possibly several times) back to themselves. From any point in p_0, L_p is reachable and co-reachable (in the classical semantics).
– under the enlarged semantics (for any positive enlargement), for any two points a and b in L_p, there is a path from a to b: this is proved "by hand" (*i.e.*, by explicitly building a path) that for any point a and b in p_0 that are close enough, it is possible to modify any cycle from a to a in order to get a path from a to b in the enlarged timed automaton.
  Combining this with the previous result, we get that under the enlarged semantics, the set of reachable configurations for any positive enlargement is included in the set of reachable region in the extended automaton.
– Completeness is obtained by showing that any finite-length trajectory in the enlarged automaton can be mimicked by a trajectory in the original automaton, both trajectories visiting the same sequence of transitions and always remaining at a distance of at most ϵ from each other, provided that the enlargement is small enough (compared to ϵ and the size of the automaton). This is proved using parametric DBMs.

5.2 Summary of [SF07] and [SFK08]

We only report on the second paper: the first paper is a one-page poster paper that appeared in the proceedings of TIME'07. The corresponding 11-page research report is fully included in the second paper.

The authors follow the approach of [Pur98], by considering drifting clocks in timed automata. The difference is that they restrict to finite runs (and then generalize to infinite runs in resynchronizing systems).

For finite run drifting clocks, robust safety is defined as follows: given an integer n (corresponding to the maximal length of the runs to be considered), there exists ε_0 such that no bad state is reachable under the ε-drifting semantics, for $0 < \varepsilon < \varepsilon_0$. Not very surprisingly (contrary to what the title indicates), given a bound on the length of the runs, it is always possible to find a positive ϵ_0 for which the accumulated drift will not be sufficient to really reach new sets of states. The only newly reachable configurations are in the neighbouring regions, so that when the timed automaton only has closed gards, safety is equivalent to finite-run robust safety (Theorem 1).

In a second part (which is the main difference between [SFK08] and [SF07]), the authors introduce a model of timed automata in which clocks are regularly (at least once every μ time units) resynchronized (with imprecision Δ). Applying the previous results, provided that μ and Δ are small enough, safety robustness is again shown equivalent to plain robustness.

6 Distributed models

6.1 Summary of [Kri99]

Distributed Timed Automata (DTA) have been introduced by P. Krishnan in 1999. He considers networks of timed automata with the particular assumption that time is local to each automaton. Each automaton owns a set of clocks which only the owner can reset, but everyone may check the value of everyone's clocks (local read/write, distant read). At any given instant the global time is a vector of real values (i.e. the collection of the time at each agent).

6.2 Summary of [DL07]

Here, the semantics is given as usual by a Timed Transition System (TTS). In time passage transitions, all components evolve independently one of the other, the local times being incremented independently, which will increment the global time with the sum of local increments. The paper considers the timed languages defined by accepting paths of such TTS. It is proved that the class of timed languages defined by DTA is strictly larger that the timed languages of timed automata (TA). This class is proved to be equivalent to the languages defined by a particular class of stopwatch automata, called *partitioned stopwatch automata*. We can thus retain that *DTA are more expressive that TA*. It is also proved that the class of DTA is not closed under intersection.

6.3 Summary of [ABG$^+$08]

This paper provides a framework for distributed systems with independently evolving local clocks (DTA). Each component of the distributed system is modeled by a timed automaton. Each clock is said to belong to exactly one of the components. All clocks belonging to the same timed automaton evolve at the same rate. However clocks belonging to different components are allowed to evolve at rates that are independent of each other. Now, clocks belonging to a component can be read/checked by another component but a clock can only be reset by the component it belongs to. Thus, the different components interact with each other only through reading of each other clocks.

Since time values on different components are completely unrelated, the paper focuses on studying the underlying untimed behaviors of these distributed timed automata rather than their timed behaviors. Thus, clocks (and time itself) are synchronization tools rather than being a part of the observation. This is a crucial point where this work departs from previous works, such as [Pur00,DDMR04]. This also explains why the DTA differ from hybrid automata.

Different semantics are considered for (1) negative specifications, i.e., if some *bad* system configuration be reached under *some* choice of local time rates, and (2) positive specifications, i.e., checking if a system exhibits some *good* behavior under *all* relative clock speeds. The *universal* semantics (used to guarantee that the system always exhibits a positive specification) describes the behaviors exhibited by the system no matter how time evolves in the individual components. While the *existential* semantics (used to check if system avoids a negative specification), is the set of behaviors that the system might exhibit under some (bad) choice of local time rates. The main results are:

- Model checking DTA against regular negative specifications is decidable. This is proved by defining a finite equivalence relation over the set of configurations of a distributed timed automaton, using which it is proved that the existential semantics always yields a regular set of untimed behaviors.
- Checking emptiness and universality for the universal semantics are undecidable. This is shown by a reduction from Post's correspondence problem. This result is further strengthened to a bounded case, where there are restrictions on the relative time rates.
- Finally, to be able to synthesize and verify positive specifications, a more intuitive *reactive* semantics is introduced. Here, the behaviour of the system is controlled in a step-by-step manner, depending on how time progresses on each local component. Thus, the resulting behaviours always satisfy a positive specification. By defining an equivalent alternating automaton it is shown that the reactive semantics always yields a regular set of behaviours.

6.4 Summary of [OLS11]

Considering distributed clocks rises the question of the subclass of distributed event clock automata (EDTA) that was addressed recently. Interestingly, it appears that EDTA have the same expressive power than event clock TA, contrary to what it happens for DTA. The distributed nature of clocks do not impact the behaviors of event clock automata, confirming the effect of forgetting induced by the ECA.

7 Discretization approaches

7.1 Abstract of [HMP92]

The main goal of the paper is to establish conditions ensuring that the satisfaction of a property by a timed system is equivalent in dense and discrete time settings.

Given a set of real timed sequences Π, the authors define two "integral" operators:

- $Z(\Pi) \subseteq \Pi$ the subset of sequences with integer dates (i.e. considering only the sequences compatible with a discrete-time semantic).
- $[\Pi] = \{[\rho]_\varepsilon \mid \rho \in \Pi \wedge 0 \leq \varepsilon < 1\}$ where $[\cdot]_\varepsilon$ is defined as follows. Let $\rho = (e_1, \tau_1) \ldots (e_n, \tau_n)$, then $[\rho]_\varepsilon = (e_1, \tau'_1) \ldots (e_n, \tau'_n)$ with $\tau'_i = \min(n \mid n \in \mathbb{N} \wedge \tau_i \leq n + \varepsilon)$. This operator associates with every sequence of Π a set of sequences indexed by $0 \leq \varepsilon < 1$ with integer dates stamped by a clock whose is incremented as soon as the absolute date is strictly greater than $n + \varepsilon$ for all $n \in \mathbb{N}$. By definition (choosing any ε), $Z(\Pi) \subseteq [\Pi]$.

Given a timed system $\mathcal{S}$ (resp. a formula φ) $[\![\mathcal{S}]\!]_\mathbb{R}$ (resp. a formula $[\![\varphi]\!]_\mathbb{R}$) denotes the real sequences generated by $\mathcal{S}$ (satisfying φ). The discrete-time semantics is obtained by applying the operator Z.

So more formally, the goal of the paper is to find a sufficient condition to obtain the equivalence:

$$[\![\mathcal{S}]\!]_\mathbb{R} \subseteq [\![\varphi]\!]_\mathbb{R} \quad \text{iff} \quad Z([\![\mathcal{S}]\!]_\mathbb{R}) \subseteq Z([\![\varphi]\!]_\mathbb{R})$$

The authors introduce two useful definitions.

Definition 1 (Digitization and inverse digitization).
Π is closed under digitization *iff $[\Pi] \subseteq Z(\Pi)$ (implying the equality) which is in fact equivalent to $[\Pi] \subseteq \Pi$.*

Π is closed under inverse digitization *iff $\forall \rho$ timed sequence $[\rho] \subseteq \Pi \Rightarrow \rho \in \Pi$.*

The main theorem is the following one.

Theorem 1. *If $[\![\mathcal{S}]\!]_\mathbb{R}$ is closed under digitization and $[\![\varphi]\!]_\mathbb{R}$ is closed under inverse digitization then the equivalence above holds.*

Proof.
By monotonicity of Z operator, $[\![\mathcal{S}]\!]_\mathbb{R} \subseteq [\![\varphi]\!]_\mathbb{R}$ implies $Z([\![\mathcal{S}]\!]_\mathbb{R}) \subseteq Z([\![\varphi]\!]_\mathbb{R})$.

Assume $Z(\llbracket\mathcal{S}\rrbracket_{\mathbb{R}}) \subseteq Z(\llbracket\varphi\rrbracket_{\mathbb{R}})$.
By digitization property of $\mathcal{S}$:

$$[\llbracket\mathcal{S}\rrbracket_{\mathbb{R}}] = Z(\llbracket\mathcal{S}\rrbracket_{\mathbb{R}}) \subseteq Z(\llbracket\varphi\rrbracket_{\mathbb{R}}) \subseteq \llbracket\varphi\rrbracket_{\mathbb{R}}$$

Let any $\rho \in \llbracket\mathcal{S}\rrbracket_{\mathbb{R}}$, using the previous line we have

$$[\rho] \subseteq \llbracket\varphi\rrbracket_{\mathbb{R}}$$

By inverse digitization property of φ:

$$\rho \in \llbracket\varphi\rrbracket_{\mathbb{R}}$$

$\square$

Then the authors identify a class of timed transition systems which is closed under digization and a subset of formulas of MTL which is closed under inverse digitization (including relevant properties like some cases of bounded invariance and bounded response properties).

Finally they propose for every formula φ of MTL a strenghthened and a weakened version of φ both closed under inverse digitization thus enabling "approximate" model-checking of φ in the discrete-time setting.

7.2 Summary of [OW03]

The paper studies links between digitization (and inverse digitization) as introduced in [HMP92] and robust acceptance as proposed in [GHJ97] for timed automata.

The authors consider the set **Traj** of (weakly monotonic) finite timed traces over some alphabet Σ. For a subset L of **Traj**, $Z(L)$ denotes the subset of sequences in L with integer dates. A classical observation is that integral timed traces over Σ are in a one-to-one correspondance with untimed traces over the alphabet $\Sigma \cup \{\surd\}$.

For a (mixed) timed automaton $\mathcal{A}$, the subset of **Traj** of sequences accepted by $\mathcal{A}$ is denoted by $\llbracket\mathcal{A}\rrbracket$, with $Z(\llbracket\mathcal{A}\rrbracket)$ for the subset of sequences with integer dates and $\widetilde{\llbracket\mathcal{A}\rrbracket}$ for the set $\overline{\llbracket\mathcal{A}\rrbracket}^{int}$ of robust traces obtained by tube acceptance.

An *open timed automaton* is a timed automaton in which all clock constraints in guards are open (*i.e.* use only $<$ or $>$) and a *closed timed automaton* is a timed automaton in which all clock constraints in guards are closed (*i.e.* use only $\leq$ or $\geq$).

For a timed automaton $\mathcal{A}$, it is easy to build from the region automaton an untimed automaton $\mathcal{A}^{\surd}$ accepting $Z(\llbracket\mathcal{A}\rrbracket)^{\surd}$.

The language which is not regular (taking into account the fact that no ε-transitions are permitted) is $\{(a,k_1)...(a,k_n) \mid n \in \mathbb{N}, k_i \in \mathbb{N} \text{ for } i = 1,2,\ldots,n\}$.

The results are presented in the tables below, borrowed from the paper. The emptiness and robust emptiness problem are decidable.

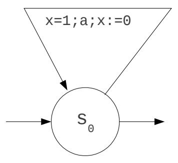

Fig. 1. The counter-example

$\mathcal{A}$	$[\![\mathcal{A}]\!] = \emptyset$?	$\widetilde{[\![\mathcal{A}]\!]} = \emptyset$?
Mixed	Decidable	Decidable

The integral emptiness problem is decidable but whether the integral robust emptiness problem is decidable is open.

$\mathcal{A}$	$Z([\![\mathcal{A}]\!]) = \emptyset$?	$Z(\widetilde{[\![\mathcal{A}]\!]}) = \emptyset$?
Mixed	Decidable	?

The universality problem is only decidable for open timed automata under the standard semantics.

$\mathcal{A}$	$[\![\mathcal{A}]\!] = \mathbf{Traj}$?	$\widetilde{[\![\mathcal{A}]\!]} = \mathbf{Traj}$?
Mixed	Undecidable	Undecidable
Open	Decidable	Undecidable
Closed	Undecidable	Undecidable

The integral universality problem is only decidable under the standard semantics.

$\mathcal{A}$	$Z([\![\mathcal{A}]\!]) = Z(\mathbf{Traj})$?	$Z(\widetilde{[\![\mathcal{A}]\!]}) = Z(\mathbf{Traj})$?
Mixed	Decidable	Undecidable

Denoting by CUD (resp. CUID) the closure under digitization (resp. inverse digitization), closure under digitization is only decidable under the standard semantics.

$\mathcal{A}$	$[\![\mathcal{A}]\!]$ *CUD* ?	$\widetilde{[\![\mathcal{A}]\!]}$ *CUD* ?
Mixed	Decidable	Undecidable
Open	Decidable	Undecidable
Closed	Yes	Undecidable

Closure under inverse digitization is undecidable for mixed and closed timed automata under the standard semantics and otherwise holds.

$\mathcal{A}$	$[\![\mathcal{A}]\!]$ *CUID* ?	$\widetilde{[\![\mathcal{A}]\!]}$ *CUID* ?
Mixed	Undecidable	Yes
Open	Yes	Yes
Closed	Undecidable	Yes

Finally, the status of the tick languages is investigated:

Property	$Z([\![\mathcal{A}]\!])^{\surd}$	$Z(\widetilde{[\![\mathcal{A}]\!]})^{\surd}$
Regular ?	Yes	No
Recursive ?	Yes	Yes

7.3 Summary of [Bey01a,Bey01b,BLN03]

These papers present the work leading to the development of the tool Rabbit, for the verification of timed systems using a discretization approach and symbolic data structures (i.e. BDD) to cope with the large state spaces.

The FME 2001 paper [Bey01a] defines the particular automata which are used by the tool (called Cottbus timed automata, which are classical TA with syntactic support to define a system compositionally). The paper then addresses the problem of discretization for *closed timed automaton*, i.e. a timed automaton in which all clock constraints in guards are closed and use only $\leq$ or $\geq$ and $\wedge$.

The goal is to prove location equivalence of the automata under continuous or discrete semantics. To this end, the authors define a relation associating to any continuous clock assignment its discrete integer representative. v' is an integer assignment representative of v in the continuous interpretation if there exists a $\gamma \in \mathbb{R}$ such that either $v' - 1 + \gamma < v < v' + \gamma$ (basic case) or $v' - 1 + \gamma < v$ and v' is the greatest constant used in a guard of the automaton plus one. The second case allows to abstract away divergent cases by a representative that is one above the highest clock constant. The proof is then quite straightforward, and proves location equivalence, i.e. a location is reachable under discrete semantics iff. it is reachable under continuous semantics.

The authors mention several other similar proofs of equivalence between discrete and continuous semantics, in particular Henzinger, Manna and Pnueli's seminal work on discrete clocks [HMP92] and Popova's work on time Petri nets [PZ91].

They also cite a CONCUR'98 paper by Asarin, Maler and Pnueli [AMP98], which addresses the following problem (excerpts from abstract): "given a digital circuit composed of gates whose real-valued delays are in an integer bounded interval, is there a way to discretize time while preserving the qualitative behavior of the circuit?"

This problem is described as open in J.A. Brzozowski and C-J.H. Seger's book "Asynchronous Circuits" (1994). When preservation of qualitative behavior is interpreted in a strict sense, as having all original sequences of events with their original ordering we obtain the following two results: 1) For acyclic (combinatorial) circuits whose inputs change only once, the answer is positive: there is a constant N, depending on the maximal number of possible events in the circuit, such that if we restrict all events to take place at multiples of N, we still preserve qualitative behaviors. 2) For cyclic circuits the answer is negative: a simple circuit with three gates can demonstrate a qualitative behavior which cannot be captured by any discretization. Nevertheless [AMP98] shows that a weaker notion of preservation, similar to that of [HMP92], allows in many cases to verify discretized circuits with $N = 1$ such that the verification results are valid in dense time.

Given this equivalence between discrete and continuous analysis for these models, the paper [Bey01b] presents the tool Rabbit, that includes verification of reachability properties using a BDD based approach. Clocks are simply integer variables with a domain ranging from 0 to the highest constant the clock is compared against in a transition guard. The paper compares favorably compares a BDD encoding of the discrete system to dedicated encodings of the continuous semantic. This paper also points to other references describing a heuristic for variable ordering in the BDD, and a CEGAR style loop for incremental refinement of a specification expressed in Cottbus automata. This refinement relation includes support for time constrained systems.

The CAV'03 paper [BLN03] is a shorter and perhaps more mature presentation of the same information as in [Bey01b], including more comparisons to tools such as Uppaal or RED.

7.4 Summary of [MLR08]

This paper addresses the class of bounded Petri nets with stopwatches (SwPNs), which is an extension of T-time Petri nets (TPNs) where time is associated with transitions. Contrary to TPNs, SwPNs encompass the notion of actions that can be reset, stopped and started. Models can be defined either with discrete-time or dense-time semantics. Unlike dense-time, discrete-time leads to combinatorial explo- sion (state space is computed by an exhaustive enumeration of states). We can however take advantage from discrete-time, especially when it comes to SwPNs: state and marking reachability problems, undecidable even for bounded nets, become decidable once discrete-time is considered. Thus, to mitigate the issue of combinatorial explosion, we now aim to extend the well-known symbolic handling of time (using convex polyhedra) to the discrete-time setting. This is basically done by computing the state space of discrete-time nets as the discretization of the state space of the corresponding dense-time model. First, we prove that this technique is correct for TPNs but not for SwPNs in general: in fact, for the latter, it may add behaviors that do not really belong to the evolution of the discrete-time net. To overcome this problem, we propose a splitting of the general polyhedron that encompasses the temporal in- formation of the net into an union of simpler polyhedra which are safe with respect to the symbolic successor computation. We then give an algorithm that computes symbolically the state space of discrete-time SwPNs and finally exhibit a way to perform TCTL model-checking on this model.

The paper contains proofs that the decomposition is valid and preserves properties of interest, in this case TCTL. A way to understand the paper is that DBM cannot encode arbitrary polyhedra (i.e. time zones), but splitting the polyhedra for a time zone into smaller convex polyhedra allows a) to preserve the semantics b) to use existing verification techniques. The proof material shows that it is sufficient to preserve the semantics that we consider polyhedra that cover one time unit. Figure 2 shows how a time zone is split into smaller convex polyhedra.

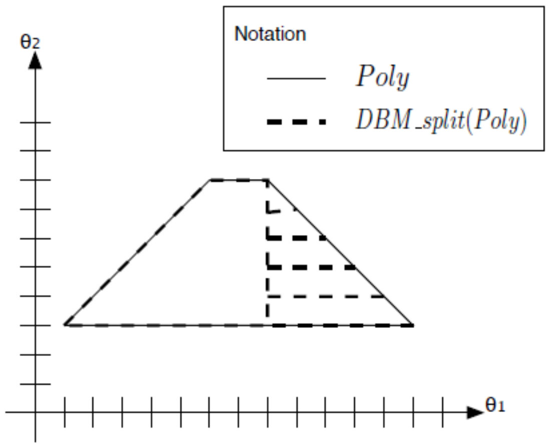

Fig. 2. Illustration of the effects of a splitting procedure. Poly represents the temporal domain associated to a symbolic class of a discrete-time ITPN. DBM split(Poly) corresponds to a potential decomposition of this polyhedron into a union of DBMs such that $Disc(Poly) = Disc(DBMsplit(Poly))$ (i.e. the discrete semantics are preserved).

7.5 Summary of [CHR02]

A comparison of control problems for Timed and Hybrid Systems (Cassez, Henzinger and Raskin): In the first half, the authors review and compare several known formalisms and results for the control of Timed and Hybrid systems. In the second half, some new results regarding the comparisons and one undecidability result are proved. The models considered are tractable subclasses of hybrid automata, namely rectangular automata (RA for short), initialized rectangular automata (iRA) and timed automata (TA), in decreasing order of expressive power. In TA all real-valued variables have constant derivative 1; in RA the derivatives of all variables are bounded by constants from above and below and in iRA these bounds are allowed to change only when the variable is reset.

For the model of control, two broad distinctions are made between dense and discrete time. Dense time control semantics is defined by splitting here each move into a continuous, time-elapse step (made by the plant) and a discrete-action, instantaneous step (made by the controller). The discrete-time or sampling semantics is obtained by fixing a number $\beta > 0$ and requiring that each continuous step take exactly β units of time; thus, the plant evolves continuously while the controller observes the system every β time units and makes a discrete action only then. Within both the above models, two further distinctions are made: in the discrete time case, between known and unknown sampling rate depending on knowledge β and in the case of dense time between known and unknown switch conditions, depending on knowledge of a set of predicates based on which the controller makes its discrete steps.

A control relation κ of a transition system H (a TA/iRA/RA) assigns every state to a non-empty subset of actions such that, from every state there is a transition with some action in *this* set. Thus the semantics of H under κ is the set of states reachable in H under the control relation κ. Fixing an RA H, a (polyhedral) set of its states R, the problems considered in the paper are:

1. The known sampling rate control problem (KSR for short) for discrete semantics, asks if given a sampling rate β, there exists a control relation κ such that R is *not reachable* in β-sampled semantics of H under κ.
2. The unknown sampling rate control problem (USR for short) for discrete semantics, asks if there exists a $\beta > 0$ such that the above statement holds.
3. The known switch conditions dense-time control problem (KSC for short) asks if there exists a control relation κ such that R is non-reachable in the dense-time semantics of H under κ. That is, given a set of predicates P on the plant state at each controller state, can an unsafe plant state be prevented by a controller choosing switch conditions from P?
4. The unknown switch conditions dense-control problem (USC for short) asks if there exists such a set of predicates P on the plant state such that the above holds (formalizing this is a bit more complicated).

Theorems 1, 2, state that the "unknown" versions (2,4) are more general than the "known" counterparts (1,3). Decidability results are known, except for USR:

System	KSC	USC	KSR
TA	dec [MPS95]	dec [MPS95]	dec [HWT92]
iRA	dec [HHM99]	undec [HKPV98]	dec [HK99]
RA	undec [HHM99]	undec [HHM99]	dec [HK99]

The new results in this paper are:

– Unknown sampling rate control (USR) is not as powerful as dense-time control(KSC/USC): shown by constructing a TA that can be controlled by known or unknown switch conditions dense-time controllers to avoid a Bad location, but not by any discrete-time controller, no matter how fast it is.

– Dense-time control with known switch conditions (KSC) are not as powerful as known sampling rate control (KSR): shown by a simple TA construction.
– It is stated that the USR problem is undecidable for TA (and hence for iRA and RA) by showing the undecidability of the following problem: does there exist $\beta > 0$ such that, given a TA H and a set of states R, R is not reachable in the β-sampled semantics of H. But, in fact what is *proved* to be undecidable is the following related problem (let us call it USR'):
  Does there exist a $\beta > 0$ such that given a TA H and a set of states R, R is reachable in the β-sampled semantics of H.

This problem is significant in its own right but its not clear how it implies USR. In any case, proving this undecidability forms the main proof of the paper and is done by a reduction from the reachability problem for 2-counter systems. The proof encodes unbounded values of counters c using the difference between two clocks x, y. First part is to design a TA sampled at rate $\frac{1}{\beta}$ that simulates runs of a 2-counter machine with counter values less than $\lfloor\beta\rfloor$, as follows: if $x \geq y$, then $c = (x-y)\beta$ and if $x < y$, then $c = \left(\frac{\lfloor\beta+1\rfloor}{\beta} - (y-x)\right)\beta$. The maximal value that can be encoded with $x, y \in [0,1]$ for counter c is then exactly $\lfloor\beta\rfloor$. Thus, by increasing β arbitrarily large counter values can be encoded, which is the crucial property. By resetting x, y at every sampling point it is ensured that $x - y$ is always a multiple of $\frac{1}{\beta}$ and when overflow occurs it goes into a deadlock non-final state. In the second part, state reachability problem for 2-counter machines is reduced to the USR' problem for TA, completing the proof.

8 Platform execution approaches

8.1 Summary of [AT05]

This paper questions whether implementability issues in timed automata should really be handled at the semantical level (as proposed in [GHJ97,Pur98,DDMR04]). Instead, it proposes a modelling approach, where the execution platform is fully described[6] (as timed automata), with digital clocks, positive response time. This platform runs in parallel with a discretized version of the timed automaton under study, sending tick events when the modelled CPU is ready to apply a new transition in the automaton.

This approach allows for a very precise modelling of the execution platform: for instance, the digital clock can be exactly periodic (incremented by Δ every Δ time units), drifting (incremented by Δ after some delay in $[\Delta-\varepsilon, \Delta+\varepsilon]$), ... Similarly, the CPU can trigger transitions in the automaton more-or-less periodically and regularly.

[6] Notice that in particular, the problem is not the parameterized problem of deciding the existence of an execution platform, but the problem of verifying that a given platform is fine.

Obviously, the composition of all these automata can be checked using UpPAal of Kronos, as it is made of plain timed automata. Controller synthesis techniques can also be applied quite directly. The authors do not report on their experiments, and it is not clear whether this will be efficient in practice: first, time in the original timed automaton is discretized, leading to a (possibly huge) timed automaton with discrete variables. This automaton is then composed with several (admittedly small) timed automata modelling the platform, resulting in a possibly enormous automaton.

Finally, the authors study the question whether "faster is better" in their approach: this property states that if a timed automaton is "correct" (safe, say) on some execution platform, it will remain correct on a faster platform. This property holds of the semantical approaches of [Pur98,DDMR04]. It is not the case here: actually, already for the sampled semantics, refining the sampling time may introduce new (and possibly incorrect) behaviours. The same holds here, as this modelling approach is able to encode the sampling semantics.

8.2 Summary of [Die01]

This paper introduces a formal modelisation for simple programs implemented on programmable logic controllers (PLC). This configuration is often used in industry for the implementation of real-time systems.

A PLC is a hardware device with a minimalist operating system that basically and repeatedly reads input signals, compute new outputs and produce output signals accordingly. The duration of sych a cycle can vary with the program, the number of inputs and outputs and the different PLCs available.

An important feature is that inputs are received asynchronously and periodically polled for by the OS before computing the outputs.

The combination of the program and hardware is modeled as PLC-automata, which are syntactically Moore machines equipped with:

– an upper bound ϵ for the duration of cycles;
– a function S_e giving for each state q a set of input signals to ignore (for some time);
– a function S_t giving for each state q the duration during which signals in $S_e(q)$ will be ignored.

So the programs modeled by PLC-automata are very simple: they have a state which can be updated according to the inputs and they can ignore some inputs for some time upon changing states.

The semantics of PLC-automata is given using duration calculus (DC) [CHR91] and also using timed automata, thus defining a subclass of TA that is implementable with PLCs, if ϵ is big enough.

The paper also shows, in terms of DC, how PLC-automata can be composed in terms of synchronisation (input sharing on the same PLC), pipelining (chaining) and transmission (communication through some medium).

Finally the modelisation is illustrated on a leaking gas burner case-study.

9 Towards a formalization of robustness problems

9.1 The models and their semantics

We first need to distinguish between two types of models: those used for the implementation systems and those intended for the specifications. In both cases, we also have to describe the semantics for these models. The different semantics are given by a type and relevant parameters.

The models considered here include explicit time constraints, with the possible addition of other quantitative features. Therefore, they can be timed automata (TA) or networks of timed automata (NTA), time or timed Petri nets (TPN or TdPN), and possibly hybrid automata (HA) and weighted timed automata (WTA).

For such a model $\mathcal{S}$, we denote the semantics by $[\![\mathcal{S}]\!]_x^{type}$ where *type* is the type of the semantics and $x \in X$ is the chosen value of the parameters in a set X.

Implementation models. We list some possible types of semantics with the associated parameters.

- **A discrete-time semantics** which means that the clocks are observed every δ time units. Here the main parameter of this semantics is δ.
- **A monitoring semantics** which means that the system is executed by a monitor whose capabilities constitute the parameters of the semantics: the maximal cycle lengths, the reaction delay, etc.
- **A perturbed semantics** where the parameters could be the precision and the drift of the clocks.
- **A distributed semantics** where the parameters describe the kind of synchronization between processes.

Specification models. Similar semantics also exist at the specification level. However here the parameters correspond to the required features (precision, delay, etc.). Generally these models are more abstract and are more *powerful*, for instance the mechanisms of time control could be infinitely precise.

9.2 Expression of the properties of the models

9.3 Relation between models

We also need an approximation relation $\mathcal{R}$ between models, given for instance by ε-(bi)simulation or some metrics (like trace distance), and a property specification language $\mathcal{P}$ like for instance a set of safety properties, a timed or untimed temporal logics, game logics or (bi)-simulations.

9.4 Formalization of the robustness problems

With these notations, two main problems can be stated: the first one concerns the verification of an implementation (possibly w.r.t. a given property), while the more difficult second one addresses the question of implementation synthesis (also w.r.t. some property).

Given an implementation semantics $[\![\cdot]\!]_x^{imp}$, a specification semantics $[\![\cdot]\!]_y^{abs}$, and a relation $\mathcal{R}$, the verification problem is, for some suitable definition of $\preceq$:

$$\forall x\ \exists y\ \forall \mathcal{S}\ \exists \mathcal{S}' \text{ such that } ([\![\mathcal{S}]\!]_x^{imp} \preceq [\![\mathcal{S}]\!]_y^{abs}) \wedge \mathcal{R}(\mathcal{S},\mathcal{S}')$$

while the synthesis problem is

$$\forall y\ \exists x\ \forall \mathcal{S}'\ \exists \mathcal{S} \text{ such that } ([\![\mathcal{S}]\!]_x^{imp} \preceq [\![\mathcal{S}]\!]_y^{abs}) \wedge \mathcal{R}(\mathcal{S},\mathcal{S}')$$

Variants of the problem can be related to some $\mathcal{P}$:

$$\forall x\ \exists y\ \forall \mathcal{S}\ \forall \varphi \exists \mathcal{S}' \text{ such that } ([\![\mathcal{S}']\!]_x^{imp} \models_y^{abs} \varphi \Rightarrow [\![\mathcal{S}]\!]_y^{abs} \models_x^{imp} \varphi) \wedge \mathcal{R}(\mathcal{S},\mathcal{S}')$$

References

[AB01] E. Asarin and A. Bouajjani. Perturbed Turing machines and hybrid systems. In *In Proc 16th annual symposium on logic in computer science (LICS)*, pages 269–278. IEEE Comput Soc, 2001.

[ABG$^+$08] S. Akshay, Benedikt Bollig, Paul Gastin, Madhavan Mukund, and K. Narayan Kumar. Distributed timed automata with independently evolving clocks. In *CONCUR*, pages 82–97, 2008.

[ADM05] Parosh Aziz Abdulla, Johann Deneux, and Pritha Mahata. Closed, open, and robust timed networks. *Electr. Notes Theor. Comput. Sci.*, 138(3):117–151, 2005.

[AMP98] Eugene Asarin, Oded Maler, and Amir Pnueli. On discretization of delays in timed automata and digital circuits. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR 98*, Lecture Notes in Computer Science, pages 470–484. Springer, 1998.

[AMRT05] Karine Altisen, Nicolas Markey, Pierre-Alain Reynier, and Stavros Tripakis. Implémentabilité des automates temporisés. In Hassane Alla and Éric Rutten, editors, *Actes du 5ème Colloque sur la Modélisation des Systèmes Réactifs (MSR'05)*, pages 395–406, Autrans, France, October 2005. Hermès. Invited paper.

[AT05] Karine Altisen and Stavros Tripakis. Implementation of timed automata: An issue of semantics or modeling? In Paul Pettersson and Wang Yi, editors, *Proceedings of the 3rd International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 273–288. Springer-Verlag, September 2005.

[ATM05] Rajeev Alur, Salvatore La Torre, and P. Madhusudan. Perturbed timed automata. In *Hybrid Systems: Computation and Control, 8th International Workshop, HSCC 2005*, volume 3414 of *Lecture Notes in Computer Science*, pages 70–85, Zurich, Switzerland, March 2005. Springer.

[Bey01a] Dirk Beyer. Improvements in BDD-based reachability analysis of timed automata. In J. N. Oliveira and P. Zave, editors, *Proceedings of the Tenth International Symposium of Formal Methods Europe (FME 2001, Berlin, March 12-16): Formal Methods for Increasing Software Productivity*, LNCS 2021, pages 318–343. Springer-Verlag, Berlin, 2001.

[Bey01b] Dirk Beyer. Rabbit: Verification of real-time systems. In P. Pettersson and S. Yovine, editors, *Proceedings of the Workshop on Real-Time Tools (RT-TOOLS 2001, Aalborg, August 20)*, pages 13–21, Uppsala, 2001.

[BLM$^+$11] Patricia Bouyer, Kim G. Larsen, Nicolas Markey, Ocan Sankur, and Claus Thrane. Timed automata can always be made implementable. In *Proc. of CONCUR'11*, Lecture Notes in Computer Science. Springer, 2011.

[BLN03] Dirk Beyer, Claus Lewerentz, and Andreas Noack. Rabbit: A tool for BDD-based verification of real-time systems. In W. A. Hunt and F. Somenzi, editors, *Proceedings of the 15th International Conference on Computer Aided Verification (CAV 2003, Boulder, CO, July 8-12)*, LNCS 2725, pages 122–125. Springer-Verlag, Berlin, 2003.

[BM99] Jos C. M. Baeten and Sjouke Mauw, editors. *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings*, volume 1664 of *Lecture Notes in Computer Science*. Springer, 1999.

[BMR06] Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust model-checking of linear-time properties in timed automata. In Jose R. Correa, Alejandro Hevia, and Marcos Kiwi, editors, *Proceedings of the 7th Latin American Symposium on Theoretical Informatics (LATIN'06)*, volume 3887 of *Lecture Notes in Computer Science*, pages 238–249, Valdivia, Chile, March 2006. Springer.

[BMR08] Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata *via* channel machines. In Roberto Amadio, editor, *Proceedings of the 11th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'08)*, volume 4962 of *Lecture Notes in Computer Science*, pages 157–171, Budapest, Hungary, March-April 2008. Springer.

[BMS11] Patricia Bouyer, Nicolas Markey, and Ocan Sankur. Robust model-checking of timed automata via pumping in channel machines. In *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*, volume 6919 of *Lecture Notes in Computer Science*, pages 97–112. Springer, 2011.

[CHP08] Krishnendu Chatterjee, Thomas A. Henzinger, and Vinayak S. Prabhu. Timed parity games: Complexity and robustness. In Franck Cassez and Claude Jard, editors, *Proc. 6th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2008)*, volume 5215 of *Lecture Notes in Computer Science*, pages 124–140. Springer, 2008.

[CHR91] Zhou Chaochen, C.A.R. Hoare, and Anders P. Ravn. A calculus of durations. *Information Processing Letters*, 40(5):269 – 276, 1991.

[CHR02] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A comparison of control problems for timed and hybrid systems. In *Hybrid Systems: Computation and Control, 5th International Workshop, HSCC 2002, Stanford, CA, USA, March 25-27, 2002, Proceedings*, volume 2289 of *Lecture Notes in Computer Science*, pages 134–148. Springer, 2002.

[DDMR04] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robustness and implementability of timed automata. In Yassine Lakhnech and Sergio Yovine, editors, *Proceedings of the Joint Conferences 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, Grenoble, France, September 2004. Springer.

[DDMR08] Martin De Wulf, Laurent Doyen, Nicolas Markey, and Jean-François Raskin. Robust safety of timed automata. *Formal Methods in System Design*, 33(1-3):45–84, December 2008.

[DDR05] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost asap semantics: from timed models to timed implementations. *Formal Aspects of Computing*, 17(3):319–341, 2005.

[Die01] H. Dierks. Plc-automata: a new class of implementable real-time automata. *Theoretical Computer Science*, 253(1):61–93, 2001.

[DK06] Conrado Daws and Piotr Kordy. Symbolic robustness analysis of timed automata. In *FORMATS 2006*, volume 4202 of *Lecture Notes in Computer Science*, pages 143–155. Springer, 2006.

[DL07] Catalin Dima and Ruggero Lanotte. Distributed time-asynchronous automata. In *ICTAC*, pages 185–200, 2007.

[Doy07] Laurent Doyen. Robust parametric reachability for timed automata. *Information Processing Letters*, 102(5):208–213, 2007. Elsevier.

[GHJ97] Vineet Gupta, Thomas A. Henzinger, and Radha Jagadeesan. Robust timed automata. In *International Workshop on Hybrid and Real-Time Systems (HART'97)*, volume 1201 of *Lecture Notes in Computer Science*, pages 331–345, Grenoble, France, March 1997. Springer.

[Hen08] T.A. Henzinger. Two challenges in embedded systems design: Predictability and robustness. *Philosophical Transactions of the Royal Society*, 366:3727–3736, 2008.

[HHM99] Thomas A. Henzinger, Benjamin Horowitz, and Rupak Majumdar. Rectangular hybrid games. In *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings*, volume 1664 of *Lecture Notes in Computer Science*, pages 320–335. Springer, 1999.

[HK99] Thomas A. Henzinger and Peter W. Kopke. Discrete-time control for rectangular hybrid automata. *Theor. Comput. Sci.*, 221(1-2):369–392, 1999.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *J. Comput. Syst. Sci.*, 57(1):94–124, 1998.

[HMP92] T. A. Henzinger, Z. Manna, and A. Pnueli. What good are digital clocks? In *Proc. 19th Int. Coll. Automata, Languages, and Programming (ICALP'92)*, number 623 in LNCS, pages 545–558. Springer, 1992.

[HMP05] Thomas A. Henzinger, Rupak Majumdar, and Vinayak S. Prabhu. Quantifying similarities between timed systems. In Paul Pettersson and Wang Yi, editors, *Proceedings of the 3rd International Conferences on Formal Modelling and Analysis of Timed Systems, (FORMATS'05)*, volume 3829 of *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, September 2005.

[HR00] Thomas A. Henzinger and Jean-François Raskin. Robust undecidability of timed and hybrid systems. In *3th International Workshop on Hybrid Systems: Computation and Control (HSCC'00)*, volume 1790 of *Lecture Notes* *in Computer Science*, pages 145–159, Pittsburgh, PA, USA, March 2000. Springer.

[HS06] Thomas A. Henzinger and Joseph Sifakis. The embedded systems design challenge. In *FM 2006: Formal Methods, 14th International Symposium on Formal Methods, Hamilton, Canada, August 21-27, 2006, Proceedings*, volume 4085 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2006.

[HWT92] G. Hoffmann and Howard Wong-Toi. The input-output control of real-time discrete event systems. In *IEEE Real-Time Systems Symposium*, pages 256–265, 1992.

[JR11] Rémi Jaubert and Pierre-Alain Reynier. Quantitative robustness analysis of flat timed automata. In *Proc. 14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'11)*, volume 6604 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2011.

[Kri99] Padmanabhan Krishnan. Distributed timed automata. *Electr. Notes Theor. Comput. Sci.*, 28:5–21, 1999.

[MLR08] Morgan Magnin, Didier Lime, and Olivier (H.) Roux. Symbolic state space of stopwatch petri nets with discrete-time semantics (theory paper). In *Proceedings of the 29th international conference on Applications and Theory of Petri Nets*, PETRI NETS '08, pages 307–326, Berlin, Heidelberg, 2008. Springer-Verlag.

[MPS95] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems (an extended abstract). In *STACS*, pages 229–242, 1995.

[OLS11] James Jerson Ortiz, Axel Legay, and Pierre-Yves Schobbens. Distributed event clock automata - extended abstract. In *CIAA*, pages 250–263, 2011.

[OW03] Joel Ouaknine and James Worrell. Revisiting digitization, robustness, and decidability for timed automata. In *In Proceedings of LICS 03*, pages 198–207. IEEE Computer Society Press, 2003.

[Pur98] Anuj Puri. Dynamical properties of timed automata. In Anders P. Ravn and Hans Rischel, editors, *5th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'98)*, volume 1486 of *Lecture Notes in Computer Science*, pages 210–227, Lyngby, Danemark, September 1998. Springer.

[Pur00] Anuj Puri. Dynamical properties of timed systems. *Discrete Event Dynamic Systems*, 10(1-2):87–113, January 2000.

[PZ91] Louchka Popova-Zeugmann. On time petri nets. *Elektronische Informationsverarbeitung und Kybernetik*, 27(4):227–244, 1991.

[San11] Ocan Sankur. Untimed language preservation in timed systems. In *Proc. of MFCS'11*, Lecture Notes in Computer Science. Springer, 2011.

[SBM11] Ocan Sankur, Patricia Bouyer, and Nicolas Markey. Shrinking timed automata. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2011, December 12-14, 2011, Mumbai, India*, volume 13 of *LIPIcs*, pages 90–102. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[SF07] Mani Swaminathan and Martin Fränzle. A symbolic decision procedure for robust safety of timed systems. In *TIME 2007*, page 192. IEEE Computer Society, 2007.

[SFK08] Mani Swaminathan, Martin Fränzle, and Joost-Pieter Katoen. The surprising robustness of (closed) timed automata against clock-drift. In Giorgio Ausiello, Juhani Karhumäki, Giancarlo Mauri, and C.-H. Luke Ong, editors, *5th IFIP International Conference On Theoretical Computer Science (TCS 2008)*, pages 537–553, Milano, Italy, September 2008. Springer.

[WDR04] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Almost asap semantics: From timed models to timed implementations. In *7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 296–310, Philadelphia, PA, USA, March 2004. Springer.

[WDR05] Martin De Wulf, Laurent Doyen, and Jean-François Raskin. Systematic implementation of real-time models. In John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, editors, *International Symposium of Formal Methods (FM'05)*, volume 3582 of *Lecture Notes in Computer Science*, pages 139–156, Newcastle, UK, July 2005. Springer.