Project no.: **ICT-FP7-STREP-214755**

Project full title: **Quantitative System Properties in Model-Driven Design**

Project Acronym: **QUASIMODO**

Deliverable no.: D3.2

Title of Deliverable: Tool for implementability checking

Contractual Date of Delivery to the CEC:	**Month 18**
Actual Date of Delivery to the CEC:	**Month 18 (february 1, 2009)**
Organisation name of lead contractor for this deliverable:	**Université Libre de Bruxelles (CFV)**
Author(s):	**Jean-François Raskin**
Participants(s):	**P04 RWTH, CNRS, CFV, SU, AAU**
Work package contributing to the deliverable:	**WP 3**
Nature:	**R**
Version:	**1.0**
Total number of pages:	**10**
Start date of project:	1 Jan. 2008 **Duration:** 36 month

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2X013)

Dissemination Level

PU	Public	X
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to a group specif ed by the consortium (including the Commission Services)	
CO	Conf dential, only for members of the consortium (including the Commission Services)	

Abstract:

In this deliverable, we summarize the progress made in the Quasimodo project on theory, algorithms and tools for robust analysis of timed automata. We f rst recall the context and main challenges related to notions of robustness for timed automata. We review the recent works done on the theory necessary to build eff cient tools. We describe the results that have been obtained towards eff cient algorithms based on zones, and f nally, we report on the status of their implementation in the tool UPPAAL.

Keyword list: Robustness of timed automata, implementability of timed automata.

Contents

Bibliography

[1] Franck Cassez, Thomas A. Henzinger, and Jean-François Raskin. A Comparison of Control Problems for Timed and Hybrid Systems. In *HSCC'02*, Lecture Notes in Computer Science 2289, pp. 134–148, Springer Verlag, 2002.

[2] Christel Baier, Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Marcus GröSer. Almost-sure model checking of inf nite paths in one-clock timed automata. In *Proceedings of the 23rd Annual IEEE Symposium on Logic in Computer Science (LICS'08)*, pages 217-226, Pittsburgh, PA, USA, June 2008. IEEE Computer Society Press.

[3] Nathalie Bertrand, Patricia Bouyer, Thomas Brihaye, and Nicolas Markey. Quantitative model-checking of one-clock timed automata under probabilistic semantics. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*, pages 55-64, Saint Malo, France, September 2008. IEEE Computer Society Press.

[4] Patricia Bouyer, Nicolas Markey, and Pierre-Alain Reynier. Robust analysis of timed automata via channel machines. In *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.

[5] 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.

[6] Alexandre David, Piotr Kordy, Rom Langerak, Kim Larsen, Jan Willem Polderman: Practical Robustness Analysis of Timed Automata. November 2008. Submitted.

[7] Mani Swaminathan, Martin Fränzle, and Joost-Pieter Katoen. The Surprising Robustness of (Closed) Timed Automata against Clock-Drift. In *Proceedings of the 5th IFIP International Conference on Theoretical Computer Science (IFIP TCS)*, pages 537-553, Milano, Italy, September 2008. Springer.

Abbreviations

AAU: Aalborg University, DK

CFV: Centre Fèdèrè en Vèrif cation, B

CNRS: National Center for Scientif c Research, FR

ESI: Embedded Systems Institute, NL

ESI/RU: Radboud University Nijmegen, NL

RWTH: RWTH Aachen University, D

SU: Saarland University, D

1 Introduction

Timed and hybrid systems are dynamical systems with both discrete and continuous components. A paradigmatic example of a hybrid system is a digital embedded control program for an analog plant environment, like a furnace or an airplane: the controller state moves discretely between control modes, and in each control mode, the plant state evolves continuously according to physical laws. A natural model for hybrid systems is the *hybrid automaton*, which represents discrete components using finite-state machines and continuous components using real-numbered variables which evolution is governed by differential equations or differential inclusions. Several verification and control problems have been studied for hybrid automata or interesting subclasses. Tools like HYTECH have proven useful to analyze high-level descriptions of embedded controllers in continuous environments.

When a high level description of a controller has been proven *correct* it would be valuable to ensure that an implementation of that design can be obtained in a systematic way in order to ensure the *conservation of correctness*. This is often called program refinement: given a high-level description P_1 of a program, refine that description into another description P_2 such that the "important" properties of P_1 are maintained. Usually, P_2 is obtained from P_1 by reducing nondeterminism. To reason about the correctness of P_2 w.r.t. P_1, we often use a notion of simulation (due to Milner) which is powerful enough to ensure conservation of LTL properties for example.

In this task, we have studied how this elegant schema can be adapted in the context of real-time embedded controllers. To reach this goal, there are several difficulties to overcome. First, the notion of time used by hybrid automata is based on a dense set of values (usually the real numbers). This is unarguably an interesting notion of time at the modeling level but when implemented, a digital controller manipulates timers that are digital clocks. Digital clocks have finite precision and take their values in a discrete domain. As a consequence, any control strategy that requires clocks with infinite precision can not be implemented. Second, hybrid automata can be called "*instantaneous devices*" in that they are capable of instantaneously react to time-outs or incoming events by taking discrete transitions without any delay. Again, while this is a convenient way to see reactivity and synchronization at the modeling level, any control strategy that relies for its correctness on that instantaneity can not be implemented by any physical device no matter how fast it is. Those problems are known and have already attracted some attention from our research community. For example, it is well-known that timed automata may describe controllers that control their environment by playing a so called zeno strategy, that is, by taking an infinite number of actions in a finite amount of time. This is widely considered as unacceptable even by authors making the synchrony hypothesis. But even if we prove our controller model non-zeno, that does not mean that it can be implemented. In fact, it is shown in [1] that there are (very simple) timed automata that respect a syntactic criterion that ensures nonzenoness but requiring faster and faster reactions, say at times $0, \frac{1}{2}, 1, 1\frac{1}{4}, 2, 2\frac{1}{8}, 3, 3\frac{1}{16}, \ldots$. So, timed automata may model control strategies that can not be implemented because the control strategy does not maintain a minimal bound between two control actions. A direct consequence is that we can not hope to define for the entire class of timed automata a notion of refinement such that if a model of a real-time controller has been proven correct then it can be systematically implemented in a

way that preserves its correctness.

The inf nite precision and instantaneity characteristics of the traditional semantics given to timed automata is very closely related to the *synchrony hypothesis* that is commonly adopted in the community of synchronous languages. Roughly speaking, the synchrony hypothesis can be stated as follows: "*the program reacts to inputs of the environment by emitting outputs instantaneously*". The rationale behind the synchrony hypothesis is that the speed at which a digital controller reacts is usually so high w.r.t. the speed of the environment that the reaction time of the controller can be neglected and considered as nil. This hypothesis *greatly simplifies* the work of the designer of an embedded controller: he/she does not have to take into account the performances of the platform on which the system will be implemented. We agree with this view at the modeling level. But as any hypothesis, the synchrony hypothesis *should be validated* not only by informal arguments but formally if we want to transfer correctness properties from models to implementations. In previous works, we have shown how this can be done *formally* and *elegantly* using a semantics called the Almost ASAP semantics (AASAP-semantics).

The AASAP-semantics is a parametric semantics that leaves as a parameter the *reaction time* of the controller. This semantics relaxes the synchrony hypothesis in that it does not impose the controller to react instantaneously but imposes on the controller to react *within* Δ *time units* when a synchronization or a control action has to take place (is urgent). The designer acts as if the synchrony hypothesis was true, i.e. he/she models the environment and the controller strategy without referring to the reaction delay. This reaction delay is taken into account during the *verification phase*: we compute the largest Δ for which the controller is still receptive w.r.t. to the environment in which it will be embedded and for which the controller is still correct w.r.t. to the properties that it has to enforce (to avoid the environment to enter bad states for example).

During 2008 and 2009, the teams of our consortium have worked on foundations necessary to build algorithms and tools for robust verif cation of timed automata [1]. The results of this foundational work have been implemented into the tool UPPAAL. Those results are detailed in the next sections.

2 Foundations of robustness and implementability

2.1 Region based safety checking

2.1.1 Participants

- Martin De Wulf and Jean-François Raskin, CFV, Université Libre de Bruxelles, Belgium
- Laurent Doyen, CNRS/LSV, Cachan, France
- Nicolas Markey, CNRS/LSV, Cachan, France

[1] We recall here the main results on the theory underlying robust analysis of timed automata. Part of those results are described in more details into delevirable D3.1 of the Quasimodo project.

2.1.2 Contribution

In this paper, we tackle the basic problem of reachability under the relaxed semantics. We solve the safety verif cation problem for this robust semantics: given a timed automaton and a set of bad states, our algorithm decides if there exist positive values for the parameters Δ and ϵ such that the timed automaton never enters the bad states under the relaxed semantics. We also prove that our algorithm requires polynomial space, i.e., it has the same theoretical complexity as safety verif cation algorithms in the classical semantics. This has been published in [5].

2.2 Robustness analysis under f nite life-time or resynchronization

2.2.1 Participants

- Mani Swaminathan and Martin Fränzle, Uni. Oldenburg, Germany
- Joost-Pieter Katoen, RWTH, Aachen, Germany

2.2.2 Contribution

The unsafe states that become reachable with drifting clocks (but which are unreachable with perfect clocks) are obtained by iterating unboundedly many times through the (progress) cycles of the TA, assuming an inf nite system's life-time. Moreover, unbounded relative drift between clocks is considred which does not take into account the regular resynchronization of clocks that is performed in many implementations of real-time systems.

We address these two issues, with two main contributions:

1. Under closed guards, invariants, and targets, the standard zone-based FRA of TA performed by tools such as UPPAAL is shown to be exact for robust safety for TA with an *arbitrary, but finite* life-time. That is, for any i, there is $\varepsilon_i > 0$ such that $\mathit{Reach}_i^{\varepsilon_i}\, cal\, G = \emptyset$ where $\mathit{Reach}_i^{\varepsilon_i}$ is the reachable state space after i iterations under maximum perturbation ε_i of the clocks. Robust safety thus does not imply the existence of a homogeneous $\varepsilon > 0$ that is independent of the number of iterations, but avoids the target state G by some strictly positive value of the perturbation for any *arbitrary, but finite* number of iterations.

2. We consider clock-drifts with he possibility if regular clock resynchronization. This results in a *bounded* relative clock-drift. Under the assumption of closed guards, invariants, and targets, we show that the standard zone-based FRA of TA (like in UPPAAL) is exact for robust safety of TA with regular clock resynchronization. In this case, a certif cation of robust safety imposes no restriction on the life-time of the system—it implies avoidance of the (closed) target by all $0 < \varepsilon < 1$ (where the ε now parameterizes the maximum relative bounded clock-drift subject to periodic resynchronization) independent of the number of iterations.

This work is published as [7].

2.3 Probabilistic approach to robustness

2.3.1 Participants

- Christel Baier and Marcus GröSer, Uni. Dresden, Germany
- Nathalie Bertrand, IRISA, Rennes, France
- Patricia Bouyer and Nicolas Markey, CNRS/LSV, Cachan, France
- Thomas Brihaye, CFV, Univ. Mons-Hainaut, Belgium

2.3.2 Contribution

The mathematical aspect of real-time model-checking has another disavantage: it detects every single failure, even if it is highly unlikely to occur. While this exhaustivity is often seen as a strength of this method, it might be desirable to sometimes ignore those unlikely paths. To cope with this problem, we have def ned a probabilistic semantics for timed automata. Roughly, in a given conf guration, a transition that is f rable only at a f nite number of single dates will have zero probability if some other transition, from the same conf guration, is allowed on (at least) a non-empty interval of dates.

We proposed an algorithm for almost-surely model-checking ω-regular properties ; we also proposed a method to compute (or approximate) the probability of an ω-regular property under that semantics. These two results have been published in [2, 3].

3 Zone-based algorithm for robust safety of timed automata

3.1 Participants

- Alexandre David, Aalborg University, AAU
- Piotr Kordy, Twente University, ESI
- Rom Langerak, Twente University, ESI
- Kim G. Larsen, Aalborg University, AAU
- Jan Willem Polderman, Aalborg University, ESI

3.2 Contribution

We propose a practical algorithm for the analysis of robustness of timed automata, that is, the correctness of the model in the presence of small drifts on the clocks. The algorithm is an extension of the region based algorithm of Puri and uses the idea of stable zone of Daws and Kordy. The algorithm is a depth f rst search based on on-the-f y reachability using zones.

Puri proposes extended reachability concept. Extended reachability is set of states that are reachable if we allow any clock drift - even inf nitely small clock drift. He shows that this problem is decidable by introducing algorithm to calculate extended reachability. That algorithm is based on detecting strongly connected components in the region graph of a timed automaton. Since it is region based it is not effective in practice for complexity reasons. The idea behind algorithm is simple. Whenever we touch a region that is on some strongly connected component we add it to the set of reachable set of states. The algorithm is correct under the assumptions that all clocks are bounded and all cycles are progress cycles. Progress cycle is a cycle in the region graph where all clocks are reset.

The zone based algorithm is based on the notion of a stable zone. Stable zone is a set of states that associated with a cyclic sequence of edges. Given sequence of edges, stable zone associated with that sequence is the largest set of states such that starting from any point in the stable zone we can always reach some other point in stable zone using both forward and backward reachability along associated sequence of edges. The important property of stable zone is that if we can reach some point in stable zone, we can reach whole stable zone in extended semantics. It is possible to calculate stable zone using pre and post operators, which are currently used in model checking tools (e. g. UPPAAL) for timed automata.

If we want to use stable zone instead of strongly connected components, we can have potentially inf nite number of stable zones as there are inf nite number of cyclic paths of edges. Considering only simple cyclic paths is not enough as two cyclic paths may have empty stable zones but the combination has a non-empty stable zone.

The problem can be divided into two parts:

- case where we allow only progress cycles
- case where we allow any type of cycle in timed automaton

For the case where we allow only progress cycles, it is possible to prove the following theorem:

Theorem 1 *Let there be progress (all clocks are reset) cyclic sequence of edges* σ_1, $\sigma_2 \ldots \sigma_n$ *starting in common location. Let* σ_{all} *be any combination of* $\sigma_1 \ldots \sigma_n$*. If* σ_i *(for some* $0 < i \leq n$*) contained in* σ_{all} *has non empty stable zone* W_{σ_i}*, then stable zone* $W_{\sigma_{all}}$ *is reachable from* W_{σ_i}*.*

This theorem states that if we have a cyclic sequence of edges σ_i with non-empty stable zone, then from that stable zone we can reach any stable zone that has complex sequence of edges containing σ_i. This allows to stop exploring a cycle as soon as it has non empty stable zone. On the other hand if sequence of edges does not contain stable zone then it is not possible to take that sequences of edges inif tely many times. Based on Theorem 1 we created symbolic algorithm that calculates extended reachability. In the algorithm we detect cyclic paths on-the-f y, storing on a stack the sequence of states that we have visited already. Similar mechanism is used in algorithm for liveness checking. If we detect that we visit the same state we check if associated stable zone exists and if it does we add it to the reachable set of states. Depending it there exist a stable zone we continue exploring from current state or not. If we choose to continue we are guaranteed to f nish because stable zone does not exists so we will be not able to

cycle inf nitely often. The algorithm has been implemented in the experimental tool based on the development version of UPPAAL. The tool can be downloaded from the following web address: `http://www.cs.aau.dk/~adavid/uppaal-dev.zip`.

For the case where we allow any type of cycle in timed automaton the algorithm as it is does not work properly.