Framework Report (v3b)*

VERIMAG

May 20, 2006

AMETIST DELIVERABLE 0.2.3

Project acronym: AMETIST
Project full title: Advanced Methods for Timed Systems
Project no.: IST-2001-35304
Project Co-ordinator: Frits Vaandrager
Project Start Date: 1 April 02
Duration: 36 months
Project home page: http://ametist.cs.utwente.nl/

*This is a revised version of deliverable 0.2.3.

1 Introduction

The purpose of this report is to summarize the developments that took place within the AMETIST project and put them in a larger scientific and technological context. We start with a general overview of *model-based* design and analysis of systems from which the approach advocated by AMETIST has emerged. The principles underlying this approach are those underlying the verification of reactive computer systems. We then move to the more specific goal of the project, namely the exportation and adaptation of this approach to a wide class of systems where *quantitative timing information* plays a major role. It happens sometime that such systems are treated by techniques that do not follow these principles for one or more of the following reasons:

1. The communities in question follow their old ways and gave not assimilated the computer science way of looking at such problems;
2. Being "semantically correct" is not the most urgent issue in these problem compared to being able to express many *real-life details* and treat hard computational problems. It is sometimes believed that taking the trouble to formalize cleanly one does not progress toward a solution and often adds another level of artificial complexity.

Finally we mention some of the major achievements of the project and assess their contribution toward fulfilling that vision, or perhaps adapting it to real life in a more realistic manner.

2 Model-Based System Development

A large part of engineering and applied mathematics is concerned with building *mathematical models* of systems and using these models to validate the correct functioning of the system and to choose between design alternatives in order to optimize system performance. The nature of the system in question determines the types of mathematical models that are useful for its analysis. The internal operation of a combustion engine will be modeled by partial differential equations, the car dynamics — by ordinary differential equations and an automatic driver by, say, a finite-state automaton.

The fact that a class of models is used in an application domain is not always correlated with its adequacy for solving the domain's problem. In many cases it is a combination of the latter with historical and cultural coincidences. Practitioners, in general, do not have the time to build new clean and rigorous models.[1] The either use what was invented by theoreticians in the past or develop ad-hoc (and, sometimes, ingenious) models that allow them to solve the concrete problems they face within the time frame they have been allocated. It usually requires the intervention of theoreticians in order to clean and generalize these models.

Academics, on the other hand, tend to live in an imaginary world and have the privilege of being allowed to ignore the feed-back from reality about the short- and mid-term relevance of their models. Hence they can publish inertial papers that solve problems whose only significance is internal to the academic community to which they belong[2]. This was not the main goal of AMETIST but rather the establishment of timed automata as the underlying model for a large class of problems and application domains in the same sense that differential equations underly a large part of physics and traditional engineering, or that transition systems are used in software and hardware engineering.

[1] In retrospect, one may say that the real world does not always admit clean and rigorous models.

[2] This is not meant to undermine fundamental purely-theoretical research. Certain (but not all) mathematical objects are worth being studied for their own internal sake.

The class of models advocated in AMETIST has its origins in the domain often called *formal verification* whose goal is to prove that certain systems behave correctly for *all the external contexts* in which they can find themselves. Essentially these are models of discrete dynamical systems whose most notable features are:

1. The different *components* of the system in question are clearly identified and the model is given in terms of a *composition* of simpler models.
2. The *interaction* and mutual *influence* between the components is easily visible from the interconnection scheme of the components.
3. The *external environment* of the system is also modeled as one or more components of the same kind.
4. Each component is modeled by an *automaton*, the archetypical model of *discrete dynamical system*. The state transitions are labeled (and enabled) by *interaction conditions* between the components and by *input* and *output events*.
5. Putting all components together yields a *global automaton* in which the origin of each transition can be traced back to the participating components.
6. The global system model is the basis of all design activities such as *simulation, validation, evaluation and optimization*. These activities are supported by *tools* that provide for automatic and semi-automatic analysis.

This approach has been extended in the last decade to treat time-dependent behaviors using the *timed automaton* model, proposed by Alur and Dill. The project aimed at establishing this framework as a unifying model for a large class of timing related problems and to *scheduling* in particular.

To contrast this approach we discuss the ways these problems are treated today in industry and academia. This is an intentional straw man, whose description is quite caricaturized but, nevertheless, it may represent the existing spirit in some areas. In many application domains the models are of a more ad-hoc nature without making a distinction between the essential and the accidental details of the problem. The latter may be the syntax of a given programming language, scheduling policies of a given operating systems, etc.[3] This approach may lead to practical solutions but not always to solutions which are scalable to more general problems. The prime example in the success of Science, Newtonian mechanics, abstracted from the specific details of masses, small stones and planets and gave a general rule governing the behavior or both, although every practitioner will tell you that the "application" of throwing stones and the "application" of predicting where the stars will appear next evening, are quite different. A more down-to-earth example is the relative success of formal verification which can be partly attributed to the use of the transition system model (automaton) which abstracts away from such details. Of course, one needs at the end to treat such details in order to solve concrete problems, but much of the insights obtained on the more abstract model could not have been reached on models cluttered with details.

Another feature of commonly-used approaches is the modeling of the system in a form which is already geared toward a *particular solution technique*, although this is not always the most natural and suitable model for the problem at hand. To take an example, someone who is accustomed to

[3] One may argue that these details, which the theoretician will call "accidental" are what makes real problems (and life in general) so complex, horrible and wonderful, but the role of science is to generate useful abstractions whenever possible.

linear programming will tend to model problems as linear optimization and if the phenomena persists and refuses to be modeled that way, there will be still an attempt to keep the models close to linear programming, e.g. by adding integer variables. This approach, summarized by the proverb "*When you have a hammer, everything looks like a nail.*", can be justified for practitioners who have to solve problems routinely and quickly[4] and it may be useful in the short-term, but cannot be recommended at times of "paradigm shift" where *new phenomena* are to be modeled, or when two phenomena which used to be modeled separately need to be modeled together.

We strongly believe that *phenomena come first* and that it is more useful to understand them and devise new formal models whose semantics corresponds faithfully to these phenomena rather than to rush and translate them into one's favorite computational problem. An attack of the computational problems associated with the system analysis should follow only *after* the nature of the problem is understood. The be fair to industry, we should not forget that researchers and engineers work on different time scales and contractual constraints, and the re-use of what have been already done in the past might be the only way to meet deadlines.

It should be noted that when we say "semantics" or "timed automata" we do not refer to 90% of the work published in these domains. By semantics we do not mean fancy formalisms full of Greek letters and complex definitions. We pick from it the basic idea that a system description, *any system description*, denotes a *set of behaviors*, and that these behaviors are the objects of study, those according to which we *evaluate* system *correctness* and *performance*. Likewise, by timed automata, we do not necessarily refer to a particular definition, analysis method or a tool, but rather to the more essential mathematical model of a discrete dynamical system with clock variables.

3 Project Highlights

At the end of the AMETIST project we can try to summarize the achievement along two major axes, *expressiveness* and *performance*. Expressiveness, is the big promise of TA technology, which is claimed to facilitate the rigorous formulation of complex planning and scheduling situation which cannot be expressed naturally within the standard academic models of operations research, and which are solved in practice using somewhat ad hoc methods. Performance is the other side of the coin, an ongoing fight against the complexity inherent in TA-based analysis and synthesis techniques, a complexity which grows as models become larger and richer.

3.1 Expressiveness

The work on enhancing the expressiveness of the TA framework can be roughly partitioned into two sub-classes, *model-directed* and *application-directed*. By model-directed enhancement we refer to extensions of the TA model by new features and to the characterization of new scheduling problems and their translation to timed automata. While these extensions are *inspired* by practical problems, the main outcomes of this type of work are new rigorous models and algorithms for analysis and synthesis for these models. By application-directed work we refer to the modeling efforts intended to cope with the particularities of specific case-studies and commercial tools in order to provide *complete solutions* to real-life problems in a form acceptable by their intended end users. These two directions are complementary and can be viewed as digging a tunnel starting from both sides of the mountain, hoping that they meet in the middle.

[4] After all, scientist do not change their favorite tools too often.

3.1.1 Model-directed Extensions

Priced Timed Automata

In ordinary timed automata essentially only time can be optimized as it can be represented by an additional clock that measures its value. There are many situations in which the relevant cost functions are richer, involving cost variables that grow at *different rates* at different states or are associated with certain transitions. Some motivating examples are memory and power consumption in computers, setup and machine occupation costs in manufacturing as well as non uniform penalties for missing deadlines in systems with *soft constraints*. Priced timed automata [17, 8, 19, 20, 18, 14, 15] constitute a natural extension of TA with such cost variables. Although the dynamics of these variables renders the automata *hybrid* rather than timed, these variables do not really participate in the rest of the system dynamics (they do not appear in transition guards) and many problem are still decidable for this model. Some new significant decidability results and algorithms for priced timed automata have been obtained in the second and third years of the project and have been integrated into to UPPAAL CORA tool.

Stochastic Extensions

Probabilities can be added to timed automata in two ways. The first is by adding probabilities to transitions, an extensions which transform reachable sets from unions of zones into probabilities on zones. The second and more challenging extension is to replace delay bounds by probabilities on delays, similar to the way this done in continuous time Markov chains. Numerous topics related to both extensions were investigated within the project leading to results that clarify some of the relation between timing and probability, in particular for continuous-time Markov chains[7, 6, 5] and branching time [4]. A stochastic variant of the Axxom case study has been treated in [16].

New Scheduling Problems

At the beginning of the project, one class of problems of scheduling under uncertainty was explored, where the uncertainty is associated with task *durations*. For this problem interesting results that propose an optimality criterion more refined than simple worst-case performance. A scheduler synthesis algorithm was implemented for this class of problems and the schedules it generated on simple examples turned out to be much more *adaptive* to the *real duration* of tasks than other types of solutions [1, 2].

In the second and third year the complementary problem of scheduling under *discrete uncertainty* has been tackled. It covers the situation where the choice of tasks that need to be executed may depend on the *results* of other tasks, results that become known only after the termination of these tasks. Such situations are very common in scheduling of real time programs, where the results correspond to testing conditions inside *if* statements, but it can also be found in manufacturing, for example when certain production steps may terminate successfully or fail. A modeling framework for this problem using *conditional dependency graphs*, transformed into timed automata with discrete adversaries have been developed. Several exact and heuristic algorithms for synthesizing optimal and sub-optimal scheduling policies for this problem have been implemented. The most recent progress with heuristic depth-first search allowed us to synthesize adaptive schedulers for problems with 400 tasks and up to 20 conditions [21].

3.1.2 Application-directed Extensions

Real life problems, like those faced by Axxom customers, do not fall exactly within a stylized class of problems like the job shop. Such problems have additional constraints concerning the relative distance

between tasks, occupation of certain resources, such as mixing tasks, during the execution of several steps, different penalties for missing deadlines, etc. Many of these details are hidden inside various Excell tables used by the Orion PI tool. Much effort have been put during AMETIST lifetime in trying to understand these features, give them a rigorous semantics and devise a language to express them in a way that translates smoothly into timed automata. This work is described in detail in Deliverable 3.4.4 and we mention only the work done on using timed automata [22, 11, 12], using stochastic models [16] and optimization-based methods [31, 30]. Although this work is less fun than inventing new mathematical models or algorithms, it is of great importance for the future acceptance of TA based methods. Another extension related to the Axxom case study is concerned with using probabilities to model machine failures, hopefully in a more refined way than the current "macro level" treatment of failures in Axxom tools.

3.2 Performance

The efforts in scaling TA technology can be divided into three classes. The first seeks help from *existing* techniques of *optimization* and *constraint satisfaction*. The second direction is concerned with improvements in the building blocks of *existing TA verification tools* while the last one attempts to specialize the algorthimics of TA for those sub classes of automata associated with optimal scheduling problems.

3.2.1 Optimization and Constraint Satisfaction

During the second and third years the applicability of mixed integer linear programming (MILP) as a tool for TA analysis has been explored. An interesting observation is that relaxed models (with "integers" interpreted as reals) may sometimes give more interesting lower bounds on the costs that extend a partial solution and hence can be used for this purpose with reachability-based algorithms. All these result were integrated into the prototype tool TAOPT [29, 28, 30].

For bounded horizon problems, questions related to timed automata reachability are transformed into *satisfiability* problems for *difference logic*. Scheduling problem do translate naturally to this logic without going through automata. During the project we have developed a series of solvers for this logic, culminating in the current version of the performant solver DL-SAT [23] and `jat` [24]. The tool ELSE [32] can serve as a front end for such solvers by generating efficient difference logic formulae (exploiting partial-order ideas) that correspond to bounded model-checking formulations for timed automata.

3.2.2 Improvements in TA Technology

The basic cycle of TA verification tools consists of taking a zone (a conjunction of difference constraints on clocks) and applying to it some operations in order to produce its successors on which the same process is iterated. The number of these zones and the size of their representations is a major bottleneck for TA verification. Zones are typically represented as difference bounds matrices (DBMs) of size quadratic in the number of clocks, and it has already been known that their dimensionality can be reduced in each state to the number of clocks active in that state. More recent work in AMETIST shows that performing a finer analysis of the structure of the TA, may yield for some states DBM representations which can be as good as linear in the number of clocks [9, 10]. The integration of these results in a new version of Uppaal will contribute to a significant performance improvement.

Among the other important contributions to improving performance of TA tools we mentioned ideas inspired by partial order methods [3, 26] symmetry reduction [25] and more clever memory management during exploration [13].

3.2.3 Specialized Technology

The zone based technology has its roots in verification where the temporal uncertainty is viewed as coming from the *external environment* and the system should be correct with respect to *all* environment choices. Around the beginning of the project it was observed that when uncertainty is associated with the scheduler decisions, for example in problems of scheduling under certainty, sometimes there is a unique successor among the uncountably many which gives the optimum ("non-lazy" schedules) [2]. Consequently the problem can be solved without using zones at all but rather using *vectors* of clock variables. This way certain problems can be formulated as shortest paths in *discrete weighted graphs* and solved much more efficiently. They can also benefit from existing search algorithms for on game graphs in order to find sub-optimal schedules for scheduling with discrete uncertainties.

Even in the case of dense uncertainty coming from the environment/adversary side, there might be some clever ways to avoid zones. When the adversary has a choice in some interval $I = [a, b]$ we may still "relax" the problem by assuming only a finite subset I' of the interval. Solving the problem of synthesizing an optimal scheduling strategy with respect to I' may have the following consequences:

1. The actual value of the chosen strategy with respect to I maybe *worse* than the value computed based on I'. This is more problematic for *qualitative* criteria where the system may be correct with respect to I' but not with respect to I. For quantitative criteria this is less critical because we already accept sub optimal solutions when the problem is large
2. During execution the adversary can make a choice in $I - I'$ and the system will find itself in a state for which the optimal action has not been computed. However in many problems some default rules can be used to determine the action at such a state based on the optimal computed action in its neighborhood.

4 Toward a Unifying Framework

Working with discrete and continuous systems, being exposed to control, verification, scheduling and other domains, one cannot but observe that many problems treated under different names within different disciplines, have more resemblance if we look at them through an appropriate abstraction that filters their domain-specific details. Among these problems and techniques we mention the algorithmic approach to discrete systems verification by forward or backward fixpoint computation, the derived reachability algorithms for continuous and hybrid systems, bounded model checking (using satisfiability solvers to verify correctness for a bounded horizon), computational techniques for optimal control such as dynamic programming and model-predictive control, simulation, search methods in AI and Markov decision processes. Much of our effort during project was concerned with building a general unifying game-theoretic scheme, for which various system design and validation problems are concrete instances, most notably the problem of scheduling under uncertainty.

The paper [27] lays down a *unified* and *domain-independent* model for control in the presence of adversaries. The model uses the metaphor of a two-player game between the controller to be synthesized and the environment it is supposed to control. Control synthesis is viewed as finding an optimal (or satisfactory) strategy for the controller, where optimality is parameterized by two factors:

1. The cost function associated with *individual* trajectories
2. The way the costs of all possible adversary-induced trajectories under a given strategy are combined to compute the overall value of the strategy (worst case, average case).

Verification, and open-loop control ("ballistic" control, planning) are obtained as special cases where either one of the players is suppressed, i.e. is assumed to be deterministic with no choice. On this model we identify three generic approaches for solving controller synthesis problems:

- For *bounded time-horizon* the problem can be posed as a constrained optimization problem as is done in model-predictive control and bounded model-checking (in the latter, it is often the case that *feasibility* of the constraints is emphasized rather than cost optimality).
- The other class of methods, roughly characterized as *dynamic programming* (DP), is based on iterative computation of a value function (cost-to-go), which determines the optimal cost and action at every state of the system. For discrete systems such techniques are used in backward verification and synthesis for automata and for Markov decision processes. In continuous systems the value function is often computed as a solutions of some partial differential equations due to Hamilton, Jacobi, Bellman and Isaacs (HJBI).
- The third approach is to perform a forward search in the space of strategies and trajectories, an approach used by game-playing programs. In verification, when the search is not exhaustive over all adversarial behaviors, this activity can be viewed as testing. For control this approach is not mainstream but it seems to be popular in some related domains such as reinforcement learning and robot motion planning. The advantage of this approach compared to DP is that the value function needs to be computed only for the reachable part of the state space, a fact that may moderate the curse of dimensionality.

The separation between the domain-independent abstract scheme and the specific computational aspects of each domain, may facilitate the development of a "universal" controller synthesis tool based on this model. The specificity of each domain will be manifested by the type of computational engine used, for example a SAT solver for discrete systems and an LP solver when the dynamics is linear. The study of solvers for hybrid constrained optimization problems, the basic computational tool for all verification and synthesis problems, is becoming an important research issue by itself, as described in previous sections.

5 Concluding Remarks

With all the accumulated effort invested in Ametist, there are reasons to believe that at the end of the project timed automata technology finds itself in a much better shape and quite closer to industrial acceptance than it was. We summarize the progress with respect to the main objectives of the project, namely, *scalability*, *convenience* and *accessibility*.

- *Scalability*: This is a major obstacle for acceptance of the technology which has been subject to numerous intensive attacks during the project lifetime. Specialized solutions tailored for scheduling problems and optimizations of existing tools have led to significant improvements that make TA-based methods not inferior to other commonly-used methods. As mentioned

in Deliverable 1.1, many real-life solution to scheduling problems do not try to solve the intractable optimization problems but rather use heuristics that constrain the solution space to the point that a program can find a solution.

The above observation should not be used as an excuse to neglect scalbility altogether. The project has brought us closer to the long-awaited performance breakthrough which, like the "BDD revolution" in untimed verification, is a pre-requisite for the proliferation of timed automata to all the application domains that can benefit from their rich and useful modeling capabilities.

- *Convenience*: The pleasant user interface of UppAal clearly makes the working of timed automata much more intuitive for the layman. It has been demonstrated in the project that specialized descriptions of scheduling problems can be transformed automatically into timed automaton description, something that many potential users cannot do. The project did not develop a general language/formalism for scheduling for reasons related to the issues described in the next paragraph.

- *Accessibility*: By this term we understand the increased awareness in the client disciplines and application domains of the potential of timed automata and what they can do for the scheduling professional. Unlike domains such as formal verification where it is sufficient to convince someone in Intel or Mircosoft in order to generate a huge impact, the scheduling arena is much more fragmented, among application domains, in-house vs.commercial tools, and geographic locations. The project made the partners in contact with Axxom, a small SME which had the opportunity to observe closely what timed automata can (and cannot) do for its problems. Like any encounter with reality, the collaboration with Axxom was humblifying and will certainly help partners in their future efforts to make TA technology more widely accessible.

  Let us remark that one of the apparent major obstacles for gaining acceptance for timed automata technology lies in timed automata being a quite non-standard model from the engineer's point of view. Purely continuous systems (differential equations) as well as discrete time models (difference equations, synchronous automata) are very well understood, while discrete systems working on "asynchronous" dense time are much less intuitive to grasp upon a first encounter. Perhaps some more pedagogical considerations should be employed while writing papers on the domain, rather than addressing (and impressing) a small community of experts.

References

[1] Y. Abdeddaïm, E. Asarin, and O. Maler. On optimal scheduling under uncertainty. In H. Gargamel and J. Hatcliff, editors, *Proc. TACAS*, volume 2619 of *LNCS*. Springer, 2003.

[2] Y. Abdeddaïm, E. Asarin, and O. Maler. Scheduling with timed automata. *Theoretical Computer Science*, 2005. to appear.

[3] Y. Abdeddaïm and P. Niebert. On the use of partial order methods in scheduling. In *Ninth International Conference on Project Management and Scheduling (PMS 04)*, 2004. to be published as online abstract.

[4] C. Baier, P.R. D'Argenio, and M. Größer. Partial order reducction for probabilistic branching time. In *3rd Workshop of Quantitative Aspects of Programming Languages, emphQAPL'05*, 2005.

[5] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.

[6] C. Baier, B. Haverkort, H. Hermanns, and J.-P. Katoen. Efficient computation of time-bounded reachability probabilities in uniform continuous-time markov decision processes. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, Barcelona, Spain, 2004. Lecture Notes in Computer Science, Springer-Verlag.

[7] Christel Baier, Joost-Pieter Katoen, Holger Hermanns, and Boudewijn Haverkort. Simulation for continuous-time Markov chains. In L. Brim, P. Jancar, M. Kretinsky, and A. Kucera, editors, *Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 338–354. Springer-Verlag, 2002.

[8] G. Behrmann. Guiding and cost optimizing uppaal. Web-page, 2002.

[9] G. Behrmann, P. Bouyer, E. Fleury, and K. G. Larsen. Static guard analysis in timed automata verification. In *Proc. 9th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003)*, volume 2619 of *Lecture Notes in Computer Science*, pages 254–277. Springer-Verlag, 2003.

[10] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelánek. Lower and upper bounds in zone based abstractions of timed automata. In *Proc. 10th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 312–326. Springer-Verlag, 2004.

[11] G. Behrmann, E. Brinksma, M. Hendriks, and A. Mader. Scheduling lacquer production by reachability analysis – a case study. In *Proceedings of the 16-th IFAC World Congress*, 2005. Extended abstract, to appear.

[12] G. Behrmann, E. Brinksma, M. Hendriks, and A. Mader. Scheduling lacquer production by reachability analysis – a case study. In *Workshop on Parallel and Distributed Real-Time Systems*. IEEE Computer Society Press, 2005. To appear.

[13] G. Behrmann, K. G. Larsen, and R. Pelánek. To store or not to store. In *Proc. of 15th Int. Conf. Computer Aided Verification (CAV'2003)*, volume 2725 of *Lecture Notes in Computer Science*, pages 433–445. Springer-Verlag, 2003.

[14] G. Behrmann, K.G. Larsen, and J.I. Rasmussen. Optimal scheduling using priced timed automata. *Performance Evaluation Review, ACM Sigmetric*, 2005. To appear.

[15] G. Behrmann, K.G. Larsen, and J.I. Rasmussen. Priced timed automata: Algorithms and applications. In *Proceedings of FMCO'04*, Lecture Notes in Computer Science. Springer Verlag, 2005. To appear.

[16] H.C. Bohnenkamp, H. Hermanns, R. Klaren, A. Mader, and Y.S. Usenko. Synthesis and stochastic assessment of schedules for lacquer production. In *Proc. QEST'04*, LNCS, September 2004. To appear.

[17] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. BRICS Report Series RS-04-4, Basic Research In Computer Science, 2004.

[18] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Staying alive as cheaply as possible. In Rajeev Alur and George J. Pappas, editors, *Proceedings of the 7th International Conference on Hybrid Systems: Computation and Control (HSCC'04)*, volume 2993 of *Lecture Notes in Computer Science*, pages 203–218, Philadelphia, Pennsylvania, USA, March 2004. Springer-Verlag.

[19] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Optimal strategies in priced timed game automata. In Kamal Lodaya and Meena Mahajan, editors, *Proceedings of the 24th Conference on Fundations of Software Technology and Theoretical Computer Science (FSTTCS'04)*, volume 3328 of *Lecture Notes in Computer Science*, pages 148–160, Chennai, India, December 2004. Springer.

[20] Patricia Bouyer, Franck Cassez, Emmanuel Fleury, and Kim G. Larsen. Synthesis of optimal strategies using HyTech. In Luca De Alfaro, editor, *Proceedings of the Workshop on Games in Design and Verification (GDV'04)*, volume 119 of *Electronic Notes in Theoretical Computer Science*, pages 11–31, Boston, Massachusetts, USA, February 2005. Elsevier Science Publishers.

[21] M. Bozga, A. Kerbaa, and O. Maler. Optimal scheduling of acyclic branching programs on parallel machines. In *Real-Time Systems Symposium (RTSS)*, pages 208–217. IEEE Press, 2004.

[22] M. Bozga and O. Maler. Timed automata approach for the axxom case study. Technical report, Verimag, 2003.

[23] S. Cotton, E. Asarin, O. Maler, and P. Niebert. Some progress in satisfiabilty checking for difference logic. In *FORMATS/FTRTFT'04*, number 3253 in LNCS, pages 263–276. Springer, 2004.

[24] Scott Cotton. DPLL and difference constraints. Master's thesis, Max-Planck Doctoral School Saarbrucken, June 2005. Verimag.

[25] M. Hendriks, G. Behrmann, K.G. Larsen, P. Niebert, and F.W. Vaandrager. Adding symmetry reduction to Uppaal. In *Proceedings First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003)*, September 6-7 2003, Marseille, France, volume 2791 of *LNCS*. Springer Verlag, 2004. Full version available as Technical Report NIII-R0407, NIII, University of Nijmegen, February 2004.

[26] Denis Lugiez, Peter Niebert, and Sarah Zennou. A partial order semantics approach to the clock explosion problem of timed automata. In Kurt Jensen and Andreas Podelski, editors, *Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference, TACAS 2004*, volume 2988 of *LNCS*, pages 296–311. Springer-Verlag, 2004.

[27] O. Maler. On optimal and sub-optimal control in the presence of adversaries. In *Workshop on Discrete Event Systems (WODES)*, pages 1–12. IFAC, 2004. Invited talk.

[28] S. Panek, O. Stursberg, and S. Engell. Job-shop scheduling by combining reachability analysis with linear programming. In *Proc. 7th Int. Workshop on Discrete Event Systems*, pages 199–204, 2004.

[29] S. Panek, O. Stursberg, and S. Engell. Optimization of timed automata models using mixed-integer programming. In *Formal Modeling And Analysis of Timed Systems*, volume 2791 of *LNCS*, pages 73–87. Springer, 2004.

[30] S. Panek, O. Stursberg, and S. Engell. Efficient synthesis of production schedules by optimization of timed automata. *Control Engineering Practice*, 2005. submitted.

[31] Sebastian Panek and Sebastian Engell. Value chain optimisation / improvements in the solution by mathematical programming. internal report ametist, University of Dortmund, May 2004. Case study 4, deliverable 3.4.3.

[32] Sarah Zennou, Manuel Yguel, and Peter Niebert. *ELSE*: A new symbolic state generator for timed automata. In Kim G. Larsen and Peter Niebert, editors, *Proceedings of the 1st International Workshop on Formal Modelling and Analysis of Timed Systems, FORMATS 2003*, volume 2791 of *LNCS*, pages 263–270. Springer-Verlag, 2003.