Analysis & Tools: Tools and Tool Interaction

AAU

Revised version, October 30, 2006

AMETIST DELIVERABLE 2.5.b

1 Introduction & Summary

During the three years of AMETIST the performance and usability of the existing tools for analysing timed automata models have improved enormously. In addition to traditional verification of timed models, emphasis has been put into retargeting the technology towards optimal scheduling and performance analysis and have been pursued in a number of new tools developed within the consortium (e.g. UPPAAL Cora, ELSE, TAOpt, MOTOR). The following sections extends the information given in D0.1.5 on "Tools Strategy" (Section 7) with respect to the tools developed within AMETIST their (final) status and possible interaction.

2 Tools

2.1 UPPAAL

UPPAAL is a tool for modeling, simulation and verification of real-time systems. The tool has been developed jointly by BRICS at Aalborg University and the Department of Computer Systems at Uppsala University since 1995.

The current official released version of UPPAAL is 3.4.10 (May 11, 2005) which compared with the former 3.2 line offers significant improvements in performance, the modeling language as well as the graphical user interface. UPPAAL 3.4 supports a number of operating systems including Microsoft Windows 95 or newer, Microsoft Windows NT 4 or newer, Linux on INTEL, SUN Solaris on UltraSparc as well as MAC OS.

On December 21, 2004 the first version of the 3.5 line of UPPAAL was released and with UPPAAL 3.5.6 (released May 3, 2005) being the latest version in this line. UPPAAL 3.5.x allows the user to define complex functions using a fully integrated imperative programming language (syntactically this language is very similar to C). This new feature has already now proven to make it substantially simpler to make natural models of real-time systems containing complicated discrete data-structures (e.g. routing tables, ready queues, etc.).

Distributed UPPAAL
In [5] the principle behind the distributed version of UPPAAL is described as well as experimental evaluation. The distributed version currently only supports reachability but scales up linearly with the number of processes. The ambition which is currently pursued in the context of the ARTIST2 Network of Excellence in Embedded Systems is to connect local PC-cluster into a European Verification Grid. Experiments on the Nordic Grid, NorduGrid, have already been made.

Dissemination
During the third year of AMETIST the UPPAAL and UPPAAL Cora tools have been presented at several PhD schools and at tutorials. A new UPPAAL tutorial note fully compatible with version 3.4 has been written as a contribution to a summer school on Formal Methods for Real-Time Systems (Bertinoro September 2004) [10]:

- Formal Methods for the Design of Computer, Communication and Software Systems: Real Time, 13-18 September, 2004, Bertionoro University Residential Center, Italy.
- Third international symposium on Formal Methods for Components and Objects (FMCO 2004), Leiden, The Netherlands, November 2-5, 2004.
- PRISE: Principles of Software Engineering, Buenos Aires, Argentina, November 22-27,2004.
- MOVEP'04: Modeling and Verifying Parallel Processes, Brussels, Belgium, 13-17 December 2004.
- GVD'05: German Verification Day, Oldenburg, Germany, March, 2005.

Finally, on October 23, 2004, Gerd Behrmann received the prestigious Spar Nord Foundation Prize (35.000 Euro) for his Ph.D. thesis [4] containing numerous contributions to the development of UPPAAL (see `http://www.ciss.dk/` for more information).

2.2 UPPAAL Cora

UPPAAL Cora (former C-UPPAAL) is a branch of UPPAAL with dedicated support for optimal reachability of PTA, priced timed automata[1]. That is, timed automata models can be augmented with a single real-valued cost variable with different growth rates in locations and impulse cost on edges. The type of reachability problems solvable by UPPAAL Cora is finding the minimum cost of reaching some goal location.

The search can be guided with respect to two internal meta variables called *heur* and *remaining* that are associated with each state and which, respectively, provide an ordering among states to be explored and an underestimate on the remaining cost of reaching the goal from a given state. The search strategies supported by UPPAAL Cora are depth-first, random depth-first, best depth-first, breadth-first, smallest/largest *heur* first, and best-first. Pruning is performed with respect to *remaining*.

Furthermore, being compatible with UPPAAL 3.5.x, UPPAAL Cora allows the user to define functions in a C-like language and use these when modelling automata.

Case Study
During the third year of AMETIST, UPPAAL Cora has been extensively and successfully applied to the Axxom lacquer production case study by the Twente and Nijmegen partners in collaboration with Aalborg [6, 7]. We refer to deliverable 3.4.3 on the Axxom case-study for more detailed information and information about performance.

New Scheduling Problems
New optimal scheduling problems in the context of timed automata and priced timed automata have been considered with prototype implementations.

Firstly, the substantially more difficult problem of synthesizing cost-optimal winning *strategies* in the presence of an adversary has been solved in [16]. The method described forms the basis of an algorithmic method for computing cost-optimal winning strategies. In [17] it is demonstrated how this method may be implemented in HyTech.

Infinite scheduling calls for strategies which enable a particular process to execute forever. In this setting optimality may be formulated as a strategy which minimizes the cost-time ratio in the long run. In [17] it is shown that the minimum limit cost-time ratio problem is in fact computable and that the optimal strategy (when one exists) is also computable.

More recently, we have considered conditional optimal reachability for multi-priced timed automata. More precisely, we consider the problem of determining the minimal cost of reaching a given target state with respect to some *primary* cost variable while respecting upper bound constraints on the remaining (secondary) cost variables. We show in [40] that this problem is computable using a zone-based algorithm.

In [11] we prove computability of the problem of parameter synthesis for the cost-bounded liveness problem for PTAs, that is, the problem of synthesizing the *maximal* cost before a given target state is guaranteed to be reached. The problem has important applications to worst-case execution time analysis as demonstrated in the paper on a number of task-graph scheduling problems where tasks have uncertain execution times.

Finally, in [22] the first on-the-fly algorithm for synthesising (time-optimal) winning strategies for timed game automata has been given and implemented using the DBM-library of UPPAAL [39]. The prototype tool has been applied to a synthesized version of the well-known Production Cell case-study.

Dissemination

[1] UPPAAL Cora may be downloaded from `http://www.cs.auc.dk/~behrmann/cora/`.

In several studies PTAs have proven to be a very natural formalism for modeling a number of optimal scheduling and planning problems ranging from cost-optimal task-graph scheduling, aircraft landing to vehicle routing problems. In [8, 9] we present the methodology of UPPAAL Cora to two different research communities, namely (a) Performance Analysis and (b) Planning and Scheduling. In particular, the AMETIST research on applying timing technology to scheduling and planning has been noticed by the Planning and Scheduling community: in the beginning of the project Oded Maler gave an invited tutorial on timed automata at ICAPS and at this year's ICAPS Kim G. Larsen has been invited to give a key-note lecture on UPPAAL Cora in particular and the findings of AMETIST in general.

2.3 UPPAAL Tron

A number of contributions of applying the timed automata technology to the planning of real-time testing has been made. In [38] methods for on-line real-time conformance testing with respect to a given timed automaton model is provided and implemented in the tool UPPAAL Tron (branch of UPPAAL). Also, optimal off-line testing of real-time systems with guaranteed coverage of the specification, is obtained using optimal reachability of timed automata [33, 34].

On May 16, 2004, the version 1.1 of the testing tool UPPAAL Tron was released[2] and presented at FATES'04 [38].

UPPAAL Tron is a testing tool for black-box conformance testing of embedded and real-time systems. Given a formal timed automata model of the system under test (SUT) and its assumed operating environment, UPPAAL Tron automatically generates and executes timed test sequences. UPPAAL Tron is an on-line testing tool which means that it continuously executes test events on the SUT as they are being generated, and events from the SUT are checked against the model. The observed behavior is required to be timed trace included in the specification.

The system under test is attached to UPPAAL Tron via a test-adapter (an SUT specific software layer) and considered as a black-box since its states cannot be directly observed; only communication events via input/output channels. The user supplies UPPAAL Tron with the closed timed automata network of SUT model in parallel composition together with assumptions on the environment.

The explicit environment model is an important feature, as it can be used to generate realistic test events. It may also be fully-permissible, meaning that the environment (testing tool in this case) can offer any input at any moment and accept any output at any moment. Finally it can be used to guide the test (which is randomized) to produce particularly interesting behaviors. In [37] an application of UPPAAL Tron to an industrial case study (a thermostat) is provided with promising error detecting capabilities and execution performance.

In addition, off-line methods for real-time testing using the optimal reachability features of UPPAAL Cora have been developed and implemented [33, 34].

2.4 S-UPPAAL

In [31], we describe a prototype extension of UPPAAL with symmetry reduction. The symmetric data type scalarset, which is also used in the Murphi model checker, was added to UPPAAL's system description language to support the easy static detection of symmetries. Our prototype tool uses state swaps, described and proven sound in [30], to reduce the space and memory consumption of UPPAAL. For all examples that we experimented with (both academic toy examples and industrial cases), we obtained a drastic reduction of both computation time and memory usage, exponential in the size of the scalar sets used. We aim at integrating symmetry reduction as implemented in the prototype tool S-UPPAAL in the standard release of UPPAAL.

[2] See `http://www.cs.auc.dk/~marius/tuppaal/`

2.5 TAopt: A prototype tool combining TA and MILP

Starting from the second year, the work of the Dortmund group has focused on combining two types of approaches to scheduling: The intention is to take advantage of the simplicity of modeling with timed automata (including modularity and synchronization), but also of the relaxation techniques and heuristics that are known from mixed-integer programming (MIP). As a first step in this direction, the group in Dortmund developed a translation procedure that automatically generates MIP representations of optimization problems formulated initially for timed automata [47]. The second step was to develop a specific MIP representation for certain subclasses of scheduling problems. Such a representation leads to smaller MIP models and therefore increases the efficiency of the combined approach [46]. As a possible use of this translation, an iterative solution procedure was developed in the second year of the project, combining cost-optimal search for TA with the solution of subproblems formulated by MIP. The key idea is to use relaxations in the MIP step to guide the graph search for TA and prune the reachable state space in a branch-and-bound fashion. The objective for the third year was to provide a tool that realizes the interaction of MIP and graph search for TA.

Implementation of the approach combining MIP with graph search
The main part of the work in year three was thus devoted to the implementation of a corresponding tool, which is called TAopt. As can be seen from Fig. 1, the embedded LP solution is iteratively performed in the main loop of the optimization algorithm of TAopt. The tool CPLEX, which is known to be one of the most efficient tools for solving linear programs, is used for the LP step. Before running CPLEX the first time, the problem is specified algebraically according to a formulation investigated in [29, 49, 45]. The algebraic model is used to derive relaxed MIP models for the nodes generated by the search algorithm: For any node explored within the main loop, the history stored in the partial trace ending in the current node is used to fix relaxed decision variables in the embedded MIP model. Thus, the step *LP generator* iteratively reduces the degrees of freedom for the algebraic model. The solution obtained from the LP solver corresponds to an underestimate of the cost-to-go from the root node to the optimal leaf node of the reachability tree. This, in turn, is useful for pruning non-optimal branches of the tree and in guiding the search according to a best-lower-bound criterion.

In addition to the computation of lower bounds, different state-space reduction techniques as non-laziness and the sleep-set method have been implemented in TAopt to reduce the computational effort.

TAopt performs cost-minimizing reachability analysis for priced TA with costs on transitions and cost rates on locations. Unlike recent versions of UPPAAL, it does not employ priced zones to represent symbolic states, but single clock vectors are used instead to represent the dominating point of time in each symbolic state. Consequently, the symbolic traces potentially considered by TAopt are restricted to traces which satisfy the property of being immediate as described in [1]. This restriction allows for efficient implementation, but does not guarantee to reach the optimal solution for general cost functions. However, it is sufficient for the optimization criteria minimal tardiness, makespan, or storage cost minimization with hard deadlines.

Recent Results
TAopt has been successfully applied in optimizing various benchmark instances of job-shop problems from the OR community [48, 46]. These results can be summarized as follows: (1) The non-laziness setting and the sleep set method led to considerable reductions of the state space and the solution effort. (2) For most of the considered benchmark instances, the combined approach shows significantly better results than the pure automata-based search without embedded LP. (3) Pruning the state space with branch-and-bound techniques helps in reducing the memory consumption of reachability analysis techniques; (4) For the investigated job-shop scheduling problems, the results obtained by TAopt are competitive with those found in literature for the tools Kronos and CPLEX, if the latter is used for solving the problem completely by MIP.

Perspectives
Current work aims at developing TAopt to a level of maturity which makes it possible to: (1)

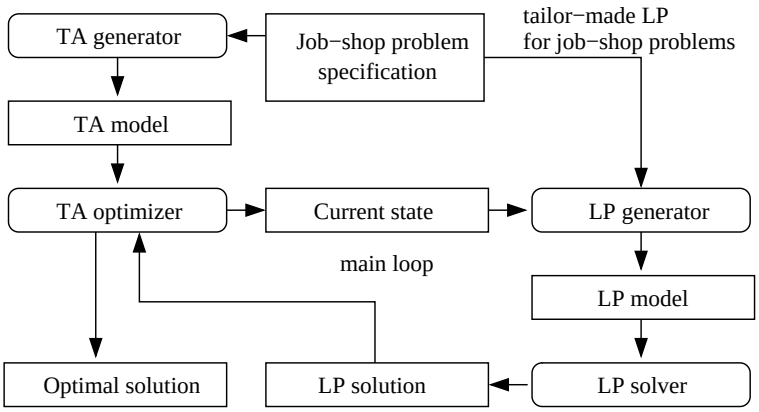

Figure 1: Structure of the implementation: the solution of linear programs (LP) with the tool CPLEX is embedded into TAopt.

easily model more general classes of scheduling problems (e.g. resource task networks with timing constraints, changeover procedures and various types of costs), and (2) to efficiently synthesize schedules for different optimization criteria. In addition, it is investigated to which degree TAopt can be used to convert TA models (created in tools like UPPAAL) into pure algebraic models for MIP. Following this path, TAopt could play the role of linking the tools developed within AMETIST to the industrially relevant domain of representing scheduling problems by mixed-integer formulations.

2.6 The ELSE Tool

Developed during the first two years of AMETIST, the "ELSE"-tool [50] is a new symbolic state explorer for timed automata. Originally aimed as a prototype for experiments with partial order semantics of timed automata, it is in the process of becoming a serious exploration engine.

In [41] the suggested approach applied in ELSE for the symbolic exploration of timed automata is investigated. The approach solves a particular aspect of the combinatorial explosion occurring in the widely used clock zone automata, the splitting of symbolic states depending on the order of transition occurrences, even if these transitions concern unrelated components in a parallel system. The goal is to preserve independence (commutation of transitions) from the original timed automaton to the symbolic level, thus fully avoiding state splitting, yet avoiding problems of previous similar approaches with maximal bounds abstraction. This goal is achieved by (1) lifting the theory of Mazurkiewicz traces to timed words and symbolic state exploration, (2) examining symbolic path exploration from a formal language point of view, and (3) by splitting the concerns of (abstraction free) successor computation and zone comparison by a new abstraction related to maximal bounds. The theory results in data structures and algorithms that experimentally is found to provide good reductions.

2.7 The DLSAT Solver for Difference Logic

Difference logic, consisting of propositional logic augmented with difference constraints of the form $x - y < c$ is the natural formalism for expressing timing-related problems. In particular this is the type of constraints obtained while expressing optimal scheduling as a constrained optimization problem, as well as the formalism for expressing finite-horizons runs of timed automata and hence for bounded model-checking of timed automata. Bounded model-checking (BMC) for discrete systems profited from the impressive progress in the performance of propositional SAT solvers. Systems that made reachability computation explode, can now be treated in a satisfactory manner

using such solvers. In a series of works (collaboration between VERIMAG and LIF) we have tried to apply these ideas to timed automata via difference logic.

Our first experience with the art (or more appropriately, the black magic) of SAT solvers was through the MX-Solver developed within the thesis of Moez Mahfoudh [43, 42], during which we have studied different approaches for enhancing SAT to treat enriched logics. Our approach was based on a dynamic interaction between the Boolean and numerical parts. Each time a difference constraint was implied by the current assignment (i.e. became a unit clauses) it was put in a large difference-bounds matrix (DBM) which was checked for consistency using algorithms for negative cycle detection. Note that the DBMs used for SAT are several orders of magnitude larger that those used in TA verification. The performance of this solver on problems coming from BMC for timed automata was still much inferior than that of standard TA verification tools. It was much superior, however, to several other solvers developed around the same time (the topic was in the air) on problems dominated by difference constraints such as job-shop scheduling.

The second round of work on the topic started with the Master Thesis of Scott Cotton [24]. The first version of his DLSAT solver [23] employed conflict analysis and learning as well as a variety of techniques that improved the performance results significantly. The second (and still unreported) version of DLSAT has already obtained some impressive results. To begin with, for purely-propositional formulae it beats a world-class solver such as zChaff on some instances. Secondly, for some notorious job-shop scheduling problems such as FT10 it found the optimum (!) while most solvers would explode much before that. Finally the results on bounded model-checking for TA are much more encouraging than the previous results (for example it is able to check a formula obtained from a 25-unfolding of a 64-gate circuit (2^{64} states and 65 clocks) but more work still needs to be done concerning more efficient translations of BMC to SAT. It might be the case that like other asynchronous systems, TA are not amenable for efficient BMC. The ongoing work on the ELSE tool, will eventually produce an automatic translation from timed automata written in the IF format to optimized difference logic formulae for BMC. Like for untimed systems, the use of the solver is not restricted to BMC, and it can serve as an alternative computation engine inside (unbounded) reachability algorithms.

2.8 IF-SCHED

Early experiments with job-shop scheduling using IF convinced us that the classical zone-based algorithms constitute an overkill for many scheduling problems. Consequently VERIMAG has developed an independent point-based tool for finding shortest path in timed automata and used it to solve the job-shop problem, the task-graph problem and the preemtive job-shop problem. Note that for preemptive scheduling we use stopwatch automata that typically complicate zone-based algorithms, but cause no problem for our point-based method. In all the abovementioned problems our tools performance matched that of state-of-the-art tools described in the academic literature.

Later we have extended the tool in two directions. The first was the treatment of temporal uncertainty in task duration. The approach that we took, dynamic programming for timed automata, required the use of zones and used the DBM library of IF. Since IF is limited to forward analysis it could not be used by itself for this problem. The second direction is the introduction of discrete uncertainty (conditional precedence graph) and the resolution of the associated problem using search algorithms on game graphs.

All these tools are rather rudimentary in terms of user interface as most of the effort is in improving the algorithmics. The scheduling problem is specified in a classical way (jobshop, task graph, conditional precedence graph) and transformed automatically into timed automata. Some of the results of this work has been transferred to IF: it is now possible to add cost and use search algorithms for finding optimal runs or runs that satisfy certain cost constraints. Non-lazy schedules which are the basis of our point-based methods were also implemented in IF, although not in a direct way. The first results on the Axxom case-study have been obtained using this extension of IF.

Our strategy for the future is to build a specialized tool for scheduling applications based on all the abovementioned algorithmic results, with a richer schedule description language that can cover more complex real-life constraints such as the one manifested in the new Axxom case study. We hope to have some results within the project lifetime but most of the effort is planned for future projects.

2.9 IF-DC

An alternative method for coping with state explosion is divide-and-conquer, that is to decompose the system into components and create a conservative approximation of each components having a smaller number of states and clocks than the concrete systems. We developed an automatic abstraction method for acyclic closed systems and now we are working on its extension to open systems. This work currently focuses on timed automata derived from timed digital circuits (because of their relative simplicity and regular structure) but it could be extended naturally to more general types of timed automata. The tool has a front-end that translate digital circuits expressed in common industrial format (SDF) into IF timed automata, and uses the exploration and minimizations methods of the latter to generate the abstractions.

If successful, the techniques behind IF-DC can probably be integrated in tools such as IF and UPPAAL, but this will only happen after Ametist has ended.

2.10 MoDeST and MoToR

The specification language MoDeST (**Mo**delling and **De**scription language for **S**tochastic and **T**imed systems) has been developed at the University of Twente. The language supports the modular description and analysis of reactive systems while covering both functional and non-functional system aspects such as hard and soft real-time, and quality-of-service aspects. The formal semantics of MoDeST is given in terms of Stochastic Timed Automata [25, 13], a formalism that encompasses a number of well-known formalisms: labelled transitions system, probabilistic automata, timed automata, probabilistic timed automata, generalised semi-Markov processes, and Interactive Markov Chains, and more.

Important rationales behind the development of MoDeST have been:

- *Orthogonality.* Timing and probabilistic aspects can easily be added to (or omitted from) a specification if these aspects are of (no) relevance.
- *Usability.* Syntax and language constructs have been designed to be close to some other commonly used languages. The syntax resembles that of the programming language `C` and the modelling language Promela.
- *Practical considerations.* The design of the language and the development of accompanying prototypical tool support have taken place hand-in-hand.
- *Expressiveness.* Several concepts – all well studied and widely accepted in the fields of e.g., computer-aided verification and concurrency theory – have been considered:
  (1) *Action nondeterminism*
  (2) *Probabilistic branching*
  (3) *Clocks*
  (4) *Delay nondeterminism*
  (5) *Random variables*

A data-part increases the convenience of the language, and allows communication via shared variables. Parallelism is expressed by processes, synchronisation is achieved by synchronisation a la LOTOS, or by means of a test-and-set mechanism inherent to the semantics.

The universality of MoDeST makes the languages interesting in two aspects: i) it is ideal as an intermediate semantical basis for higher-level formalisms like, for example, UML state-charts; ii) its rich semantical basis makes it possible to employ, with some restrictions, different analysis approaches to identical or similar MoDeST models. The latter point led to the development of the MoToR: the **Mo**DeST **To**ol Envi**R**onment.

MoToR
In order to facilitate the analysis of MoDeST models, the prototype tool MoToR has been developed. MoToR aims at supporting a variety of analysis algorithms tailored to the different kinds of analysable submodels expressible in Stochastic Timed Automata. The idea behind MoToR is to connect MoDeST to existing tools, rather than re-implementing existing analysis algorithms anew. MoToR has been connected to the performance evaluation environment Möbius [27], which is used to conduct stochastic discrete-event simulations of MoDeST models. MoToR is available for download from `http://fmt.cs.utwente.nl/tools/motor/`.

Case-studies
Apart from the Axxom case-study (see below), the MoDeST/MoToR/Möbius tandem has been used for other case-studies. The [36] paper gives a survey. In [14], the modelling and analysis of a device-absence detection protocol in plug-and-play networks is described. The analysis of the protocol with MoToR/Möbius reveals a fairness problem inherent to the original protocol design, and guided by the simulation results, a new protocol is developed which does not show the problems discovered before.

During the third year of AMETIST, MoDeST and MoToR/Möbius have been successfully used for the Axxom case study [15]. The effect of faulty behaviour on the hard real-time scheduling problem from the domain of lacquer production is investigated. The scheduling problem is first solved using the timed model-checker UPPAAL. The resulting schedules are then embedded in a MoDeST failure model of the lacquer production line, and analysed with the discrete-event simulator of Möbius. The results obtained allow to assess the quality of the schedules with respect to timeliness, utilisation of resources, and sensitivity to different assumptions about the reliability of the production line.

Future of MoDeST/MoToR
In cooperation with Saarland University, Germany, MoToR is currently extended to allow translation of MoDeST specifications into UPPAAL models. This step allows also immediately to use MoDeST/MoToR as in input formalism for the timed testing tool TorX [12], which is developed and maintained at the University of Twente. In cooperation with the University Aachen, Germany, MoToR will be extended to connect to the Markov-chain model-checking tool ETMCC [32] and its successor MRMC. MoToR will form the basis of extensions of Markov-Reward model-checking to Markov decision processes.

3 Tool Interaction, Formats and Libraries

The tools that have been developed by the AMETIST project are research vehicles, build to demonstrate and assess the effectiveness of our ideas on modelling and analysis. Translating models from one tool to another is generally difficult: different tools support a different way of modelling systems, and if one applies a generic translation to port a model of some system X from tool A to tool B, analysis results are usually not as good as those obtained for a model of system X made directly for tool B. Also, since the tools are being improved/extended all the time, a generic translator from tool A to tool B is typically outdated within months after being developed. In the hardware industry, where model checking has been and is used very successfully, we see that major companies such as Intel develop their own tools. They do not use the tools developed by academics but rather the *ideas* and algorithms behind those tools. For these reasons, AMETIST has deliberately decided not to make major investment in automatic translations between tools.

Nevertheless, of course, interaction between tools is necessary/essential in a scientific setting and in order to stimulate takeup by industry. In Deliverable 3.5.3 [3] it is observed that in 7 out

of the 18 case studies carried out within AMETIST more than one tool has been used. Timed automata tools are combined with theorem provers, computer algebra tools, finite state model checkers, tools that support analysis of stochastic models, testing tools, etc. We see this as a very natural and clear trend. For these case studies, model translations have been written either by hand or using ad hoc translation programs. Producing these translations typically was a matter of hours. In order to enable third parties to use different analysis tools in a single verification case, it is important to provide clearly specified interfaces and the ability to import/export models to and from a tool. From the AMETIST tools in particular UPPAAL, IF, MoToR and ELSE come equipped with such interfaces.

Finally, in order to stimulate further takeup by both industry and academia, we believe it is important to make certain key components from our verification tools seperately available for use by others.

3.1 Tool Interaction

AMETIST has witnessed numerous examples of interaction between tools, both tools developed internally by members of the consortium as well externally developed tools. The examples include the following:

- [15] describes the application of MoDeST, MoToR, UPPAAL and Möbius to the Axxom case-study. The schedules synthesized by UPPAAL are embedded in a MoDeST failure model, and analyzed wrt. expected performance with the discrete event simulator of Möbius.
- [26] reports on the automatic verification of timed probabilistic properties of the IEEE 1394 root contention protocol combining two existing tools: the real-time model-checker Kronos and the probabilistic model-checker Prism.
- [21] presents a layered end-to-end approach for the design and implementation of embedded software on a distributed platform. The approach comprises a high-level modeling and simulation layer (Simulink), a middle-level programming and validation layer (SCADE/Lustre) and a low-level execution layer (TTA). We provide algorithms and tools to pass from one layer to the next. [20] presents a method of translating discrete-time Simulink models to Lustre programs. The method has been implemented and used to translate part of an industrial automotive controller provided by Audi.
- [44] presents a technique and a tool for model-checking operational UML models based on a mapping of object oriented UML models into a framework of communicating extended timed automata - in the IF format - and the use of the existing model-checking and simulation tools for this format. UML to Extended Timed Automata
- [2] provides an implementation in Java of a graphical interactive simulator of the SuperSingle mode of the Cybernetix case-study. The simulator allows to make simple configurations to experiment with variations of the SuperSingle mode and error handling situations. Moreover, simulation sequences can be stored to and loaded and loaded from files,allowing the coupling with the analysis in other tools such as UPPAAL.
- [28] Specifies and validates the SET protocol (Secure Electronic Transaction) for e-commerce using UPPAAL (for handling of timing constraints) and RAPTURE [35] for dealing with QoS aspects.
- UPPAAL Cora has been extended with the ability to generate concrete (optimal) traces. Based on this feature a prototype tool for translating schedules synthesized by UPPAAL Cora into the Orion tool has been implemented allowing alternative visualization and validation of the schedules provided. This translator has been applied in the Axxom case study [6, 7].

3.2 Formats

Within the AMETIST project the XML format of UPPAAL has been developed. Also the IF common modeling language has been developed by research groups of AMETIST member institutions. Both formats are freely available and already now downloaded by several users from outside the consortium.

The IF (= Intermediate Format) common modeling language is a very expressive modeling language[3]. The IF language [19, 18] allows the description of systems consisting of processes running in parallel and communicating through message passing via communication buffers, through signal exchange or through shared variables. Each process may use several clocks to measure time and transitions may be guarded by timing constraints and decorated with explicit deadlines. The language provide several type constructors such as enumeration, range, array, record, abstract as well as predefined basic types in order to simplify complex data description and manipulation. The language includes dynamic creation and destruction of process and signal route (channel) instances. This makes system configuration to be dynamic, that is, the number of components running (and in turn, the number of clocks ...) may change during execution. The common language integrates hierarchical states (to structure automata) and composed transitions basic control statements such as if-then-else and while-do are provided to structure automata transitions

3.3 Libraries

A number of libraries for third party developers have been released with UPPAAL 4.0:

1. As a service to the research community – and after numerous requests – we have decided to make the core data structures of UPPAAL publicly available in form of a DBM library downloadable from UPPAAL's homepage.

   DBMs are efficient data structures to represent clock constraints in timed automata . They are used in UPPAAL as the core data structure to symbolically represent timing information. The library features all the common operations such as up (delay, or future), down (past), general updates, different extrapolation functions, etc. on DBMs and federations. The library also supports an efficient method for performing subtractions [39].

   The DBM library has an extensive test suite with an extra alternative implementation of the algorithms. This implementation has also been tested on countless case studies.

   A DBM library is released under the GPL and contains language bindings for C, C++ and Ruby.

2. The UPPAAL Timed Automata Parser Library is distributed separately from UPPAAL under the LGPL licence (`libutap`[4]) and is ideal for implementing model transformation tools or analysis tools. The library supports the following file formats:

   - The TA file format is the oldest of the three formats. It is a clear text, human readable description of a network of timed automata. In this description, an automaton is called a process. The format does not handle templates.
   - The XTA format is very similar to the TA format, except that processes are really parameterised templates that can be instantiated to form processes.
   - The XML format is an XML conforming version of the XTA format. Elements like templates, locations, transitions and labels are described using tags. The level of abstraction in the format is chosen so that the format is independent of the actual syntax of declarations, invariants, guards, etc. Thus all labels are simply treated as strings without structure.

[3] `http://www-verimag.imag.fr/PEOPLE/async/IF/index.html`
[4] See `http://www.cs.auc.dk/~behrmann/utap/`

3. Java parsers and client stubs for the verification backend are distributed with UPPAAL and may be used by, for instance, domain specific tools (the documentation can be found at the UPPAAL website).

4. UPPAAL can be used as a compiler for UPPAAL models, translating the model to a bytecode representation.

References

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

[2] M. Agopian, *A simulation tool for the SuperSingle mode*, 2003, Not a paper, a tool.

[3] AMETIST, *Miscellaneous case studies: Final report*, 2005, Deliverable 3.5.3 from the IST project AMETIST.

[4] G. Behrmann, *Data-structure analysis for formal verification*, Ph.D. thesis, Aalborg University, 2003.

[5] ———, *Distributed reachability analysis in timed automata*, Software Tools for Technology Transfer **7** (2005), no. 1, 19–30.

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

[7] ———, *Scheduling lacquer production by reachability analysis – a case study*, Workshop on Parallel and Distributed Real-Time Systems, IEEE Computer Society Press, 2005, To appear.

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

[9] ———, *Priced timed automata: Algorithms and applications*, Proceedings of FMCO'04, Lecture Notes in Computer Science, Springer Verlag, 2005, To appear.

[10] Gerd Behrmann, Alexandre David, and Kim G. Larsen, *A tutorial on uppaal*, Formal Methods for the Design of Real-Time Systems, Lecture Notes in Computer Science, no. 3185, Springer Verlag, 2004, pp. 200–236.

[11] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen, *Beyond liveness: Efficient parameter synthesis for time bounded liveness*, To appear, 2004.

[12] H. Bohnenkamp and A. Belinfante, *Timed testing with TorX*, Formal Methods 2005 (John Fitzgerald, Ian Hayes, and Andrzej Tarlecki, eds.), LNCS, vol. 3582, Springer-Verlag, 2005.

[13] H. C. Bohnenkamp, P.R. d'Argenio, H. Hermanns, and J.-P. Katoen, *Modest: A compositional modeling formalism for real-time and stochastic systems*, Tech. Report TR-CTIT-04-46, Centre for Telematics and Information Technology, University of Twente, November 2004.

[14] H. C. Bohnenkamp, J. Gorter, J. Guidi, and J.-P. Katoen, *Are you still there? A lightweight algorithm to monitor node presence in self-configuring networks*, Proceedings of the 2005 International Conference on Dependable Systems and Networks (DSN 2005), IEEE Computer Society, June 2005.

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

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

[17] ———, *Synthesis of optimal strategies using HyTech*, Proceedings of the Workshop on Games in Design and Verification (GDV'04) (Boston, Massachusetts, USA) (Luca De Alfaro, ed.), vol. 119, Electronic Notes in Theoretical Computer Science, no. 1, Elsevier Science Publishers, February 2005, pp. 11–31.

[18] M. Bozga, S. Graf, and L. Mounier, *If-2.0: A validation environment for component-based real-time systems*, Proceedings of CAV'02 (Copenhagen, Denmark) (K.G. Larsen Ed Brinksma, ed.), LNCS, vol. 2404, Springer, July 2002, pp. 343–348.

[19] M. Bozga, S. Graf, L. Mounier, and I. Ober, *IF tutorial*, SPIN'04 Workshop on Model-Checking of Software, Barcelona, Spain, April 2004.

[20] P. Caspi, A. Curic, A. Maignan, C. Sofronis, and S. Tripakis, *Translating discrete-time Simulink to Lustre*, Embedded Software (EMSOFT'03), LNCS, vol. 2855, Springer, 2003.

[21] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert, *From Simulink to SCADE/Lustre to TTA: a layered approach for distributed embedded applications*, Languages, Compilers, and Tools for Embedded Systems (LCTES'03), ACM, 2003.

[22] F. Cassez, A. David, E. Fleury, K.G. Larsen, and D. Lime, *Efficient on-the-fly algorithms for the analysis of timed games*, To appear in Proceedings of CONCUR 2005 (Luca de Alfaro Martin Abadi, ed.), Lecture Notes in Computer Science, Springer Verlag, 2005.

[23] S. Cotton, E. Asarin, O. Maler, and P. Niebert, *Some progress in satisfiabilty checking for difference logic*, FORMATS/FTRTFT'04, LNCS, no. 3253, Springer, 2004, pp. 263–276.

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

[25] Pedro R. D'Argenio, Holger Hermanns, Joost Pieter Katoen, and Ric Klaren, MoDeST – *a modelling and description language for stochastic timed systems*, Process Algebra and Probabilistic Methods (PAPM-ProbmiV 2001) (Luca de Alfaro and Stephen Gilmore, eds.), Lecture Notes in Computer Science, vol. 2165, Springer Verlag, September 2001, pp. 87–104.

[26] C. Daws, M.Z. Kwiatkowska, and G. Norman, *Automatic verification of the IEEE 1394 root contention protocol with KRONOS and PRISM*, STTT **5** (2004), no. 2-3, 221–236.

[27] D. Deavours, G. Clark, T. Courtney, D. Daly, S. Derasavi, J. Doyle, W.H. Sanders, and P. Webster, *The Möbius framework and its implementation*, IEEE Tr. on Softw. Eng. **28** (2002), no. 10, 956–970.

[28] G. Diaz, K. Larsen, J. Pardo, F. Cuartero, and V. Valero, *An approach to handle real time and probabilistic behaviors in e-commerce: validating the set protocol*, SAC '05: Proceedings of the 2005 ACM symposium on Applied computing (New York, NY, USA), ACM Press, 2005, pp. 815–820.

[29] Sebastian Engell and Sebastian Panek, *Mathematical model formulation for the axxom case study*, Tech. report, University of Dortmund, May 2003.

[30] M. Hendriks, *Enhancing Uppaal by exploiting symmetry*, Report NIII-R0208, Nijmegen Institute for Computing and Information Sciences, University of Nijmegen, October 2002.

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

[32] H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle, *A markov chain model checker*, Tools and Algorithms for the Construction and Analysis of Systems, 6th International Conference, TACAS 2000 (S. Graf and M. Schwartzbach, eds.), LNCS, vol. 1785, Springer-Verlag, 2000, pp. 347–362.

[33] Anders Hessel, Kim G. Larsen, Brian Nielsen, Paul Pettersson, and Arne Skou, *Time-Optimal Test Cases for Real-Time Systems*, 3rd International Workshop on FORMAL APPROACHES TO TESTING OF SOFTWARE (FATES 2003) (Montral, Qubec, Canada), October 2003, In affiliation with the 18th IEEE International Conference on AUTOMATED SOFTWARE ENGINEERING (ASE 2003).

[34] ———, *Time-Optimal Test Cases for Real-Time Systems–extended abstract*, 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS), September 2003, Invited Talk by Paul Pettersson.

[35] B. Jeannet, P.R. D'Argenio, and K.G. Larsen, RAPTURE*: A tool for verifying Markov Decision Processes*, Tools Day'02, Brno, Czech Republic (I. Cerna, ed.), Technical Report, Faculty of Informatics, Masaryk University Brno, 2002.

[36] J.-P. Katoen, Henrik Bohnenkamp, H. Hermanns, and J. Klaren, *Embedded software analysis with motor*, LNCS, pp. 268–294, Springer, Bertinoro, Italy, 2004.

[37] K.G. Larsen, M. Mikucionis, B. Nielsen, and A. Skou, *Testing real-time embedded software using UPPAAL-Tron: An inudstrial case study*, to appear in Proceedings of EMSOFT 2005, Lecture Notes in Computer Science, Springer Verlag, 2005.

[38] Kim Larsen, Marius Mikucionis, and Brian Nielsen, *Online Testing of Real-time Systems using Uppaal*, International workshop on Formal Approaches to Testing of Software (Co-located with IEEE Conference on Automates Software Engineering 2004, Linz, Austria.) (Jens Grabowski and Brian Nielsen, eds.), September 2004.

[39] Kim G. Larsen, Alexandre David, John Haakansson, and Paul Pettersson, *Minimal dbm substraction*, Proceedings of 16th Nordic Workshop on Programming Theory (Paul Pettersson and Wang Yi, eds.), Uppsala technical report, no. 2004-041, Uppsala University, October 2004, pp. 17–21.

[40] Kim G. Larsen and Jacob I. Rasmussen, *Optimal conditional reachability for multi-priced timed automata*, In Proceedings of FoSSACS 2005, Lecture Notes in Computer Science, no. 3441, 2005, pp. 234–249.

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

[42] Moez Mahfoudh, *On satisfaiblity checking for difference logic*, Ph.D. thesis, UJF Grenoble, May 2003.

[43] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, N. Jain, and O. Maler, *Verification of timed automata via satisfiability checking*, FTRTFT (W. Damm and E-R Olderog, eds.), LNCS, vol. 2469, Springer, 2002, pp. 225–244.

[44] Iulian Ober, Susanne Graf, and Ileana Ober, *Model checking of UML models via a mapping to communicating extended timed automata*, SPIN'04 Workshop on Model Checking of Software, 2004, vol. LNCS 2989, 2004.

[45] S. Panek, S. Engell, and C. Lessner, *Scheduling of a pipeless multi-product batch plant using mixed-integer programming combined with heuristics*, Proc. European Symposium on Computer Aided Process Engineering, ESCAPE 15, 2005, accepted.

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

[47] ———, *Optimization of timed automata models using mixed-integer programming*, Formal Modeling And Analysis of Timed Systems, LNCS, vol. 2791, Springer, 2004, pp. 73–87.

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

[49] 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.

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