Data Structures – Second Year Report

LIF

May 3, 2004

AMETIST DELIVERABLE 2.3.b

Project acronym: AMETIST
Project full title: Advanced Methods for Timed Systems
Project no.: IST-2001-35304

1 Introduction

Please note that this introduction is very similar to the one given in deliverable 2.3.a, it serves for selfcontained reading to establish the context.

A main emphasis of AMETIST is the automatic analysis of timed systems (such as those of the case studies) with the aims of verification, scheduling and control synthesis. The chain of such automated analysis goes from modeling in a suitable modeling formalism, specification of the analysis goal, via the automatic coding to a more abstract search or constraint satisfaction problem, to the automatic analysis of the transformed problem.

From a theoretical point of view, many such automated analysis problems already do have algorithmic solutions, e.g. the reachability problem of timed automata. However, the problems under consideration are very complex, in some cases NP-complete but typically PSPACE-complete, sometimes harder or undecidable.

In practice, the algorithms used for timed automata related problems have exponential complexity in both time and memory (with a tradeoff between these two measures) thus imposing two bounds on the size of the problems that can be handled: Answers must be computed in an acceptable amount of time for a given amount of memory.

With ever more powerful hardware (faster processors, faster and bigger memory), these limits are pushed back, but it is generally agreed that the potential of improved efficiency on the algorithmic level by far exceeds the potential faster hardware.

Work package 2 is devoted to algorithmics from a theoretical and practical point of view. Task 2.3 in particular considers the question of data structures. In view of the above discussion, it becomes clear that data structures have a crucial role for algorithm efficiency in a double sense:

- Optimized data structures allow faster computation, hence reduce the time requirement of an algorithm.
- Compact data structures may allow to get more out of the memory with respect to the space-time tradeoff of algorithms, hence either allowing to fit bigger problems into the memory or allowing to compute a result faster by a better memory usage.

Data structures cannot be considered in isolation of algorithms and indeed contributions to Task 2.3 often contribute to other tasks as well, notably to the tasks concerned with “structure exploitation” and tasks concerned with particular problem types such as scheduling or control synthesis.

In the introduction to deliverable 2.3.a, we concluded for the first year that *“globally, we consider the ongoing activity and the progress in these lines quite remarkable and witnessing of the yet unexhausted potential of improvements for the algorithms in question.”*. This tendency is unbroken in the second year and we are proud to find that several milestone advances in the area of timed automata took place in the first two years of the project.

2 Analysis methods for timed automata: State of the art

The timed automaton model underlying modeling and analysis of timed systems is based on transition systems (graphs of states as vertices and transitions as edges) and special variables called clocks. These variables allow to model timing constraints in the evolution of a timed automaton:

- An earlier transition a may reset a clock C.
- A later transitions may be restricted by conditions on the value of C.

There is an underlying assumption of global and homogeneous time, hence all clocks advance synchronously with the same speed.

This model was introduced by Alur and Dill in the early 90ies. The authors also introduced an algorithmic basis, the region abstraction, for the automatic analysis of such systems under some hypotheses. Later, notably consortium members AAUand VERIMAGhave developed tools and showed that the model can actually be used for the analysis of interesting systems. This was not initially clear, as the complexity of – for instance – the reachability problem of timed automata is considerable and first experiments were considered disappointing.

State exploration

The currently most widely used approach to timed automata is via symbolic state exploration.

At the beginning of the project AMETIST, the tools for timed automata following this approach have matured considerably and already integrate mature data structures and algorithms. In order to understand the ongoing work in AMETIST, let us consider the basic data structure used for exhaustive exploration of timed automata:

- Concrete states of a timed automaton are composed of discrete states (resulting from the control structure of the original problem) and clock values taken from a continuous or discrete domain (depending on the time model used).

  Typically, there are infinitely many concrete states and – in the case of continuous time – also infinitely many transitions from a given state to successor states. It is obvious that concrete states are not useful for state exploration with graph algorithms.

- Symbolic states replace the clock values by symbolic constraints, for instance polyhedral constraints on clock values. Thus, a single symbolic state represents an infinity of concrete states. The important aspect of symbolic states is that they allow a finite/small representation of the full state space and that they allow to compute symbolic successors.

  The constraints are conjunctions of difference constraints of the form $X-Y \leq c$ or $X-Y < c$, where X and Y are variables and c is a constant, typically a rational or integer number.

  These constraints arise from the fact that, e.g. a clock X was reset by some transition a and later a clock Y was reset at a transition b with a constraint $X \leq c$.

Conjunctions of difference constraints $X - Y \leq c$ thus are of central importance to analysis of timed automata. The reason for this is twofold:

- They provide the expressive power to model a big class of timing related problems.

- There exist efficient algorithms for manipulating conjunctions of such constraints. For example, one way of reading such constraints $X - Y \leq c$ is as an edge in a graph from a vertex X to a vertex Y with a weight $\leq c$.

  In this coding, a set of constraints has a solution iff the corresponding graph does not contain cycles of negative weights. If the latter is not the case, then there also exists a normal form for equivalent sets of constraints, the matrix of the shortest paths between each pair of vertices in the graph.

  Such matrices, known as difference bounds matrices (DBMs), have been introduced by Bellman in 1957 and are widely used in timed automata.

  The all-pairs shortest path algorithm of Floyd-Warshall can be used to compute the normal form.

For a given number of variables that can occur in the constraint sets the dimensions of the DBMs are bounded. Further analysis of the timed automaton model allows to abstract values in these matrices that are beyond certain (positive or negative) thresholds.

Summarizing, the basic structure explored in timed automata analysis algorithms, known also as "simulation graph" is

- A graph of symbolic states consisting of a discrete part and a constraints matrix in some representation.
- Edges/transitions between symbolic states.

This graph is first of all an algorithmic construction, i.e. we assume data structures for the representation of a single symbolic state and an algorithm for computing successor states. In fact, the graph need not even be finite.

A second step is the exploration of a finite fragment of such a graph such to establish a certain property, e.g. reachability of a discrete control state.

The efficiency of such an exploration thus depends on a number of parameters, e.g.

- The actual number of states that need to be explored for establishing the property. In some approaches, the simulation graph is finite, but different abstractions may result in bigger or smaller simulation graphs.
- The size of the data structure for storing a state (in a waiting list). E.g. the DBMs are of quadratic size in the number of variables involved.
- The time (and intermediate space) needed for computing a successor state.
- The memory needed for remembering visited states and the time needed for looking up this information (e.g. by hashing).

Alternative approaches

Alternative approaches for the analysis, which are widely used and also applied in some case studies involve purely discrete models of time. Such modeling allows transitions to occur only at integer moments of time and typically there is a “tick” transition representing the progress of time. This modeling approach allows to reduce the timed model to an untimed one and makes it accessible to analysis methods for discrete systems.

Logic approaches

Different from symbolic state exploration, analysis of reachability and temporal logic properties can be reduced to satisfaction problems of logical formulae, e.g. to the well studied boolean satisfaction problem. In this setting, a (boolean) formula is used to represent the existence of a path the automaton of a certain length and with certain desired properties. At the beginning of AMETIST, no such approach had yet been studied with respect to timed automata.

3 Individual contributions to this task in year 2

In the following, we discuss individual contributions to this task in the second year of AMETIST.

3.1 Bounds abstraction

The representation of symbolic states by difference bounds matrices is not sufficient for obtaining a finite simulation graph: The constraints are actually represented by bounded constraint matrices, but the entries of the matrices may grow without bound.

This is where bounds abstraction comes into play. Bounds abstraction exploits facts about the clock constraints actually used in the timed automaton. Clock differences growing beyond these bounds may safely be considered equivalent with respect to the simulation graph: The structure of the simulation graph as seen from states that only differ in constraints beyond these bounds are actually isomorphic.

Traditionally, this abstraction makes use of the biggest bounds used for some clock in the entire automaton. Last year, it was shown how this analysis can be significantly improved by performing, among other aspects, statewise abstraction.

In [2], in addition, a distinction is introduced between greatest lower bounds and greatest upper bounds, which contribute in different manners to the abstraction. This distinction is particularily strong if certain clocks are only tested against either upper bounds or lower bounds. This improvement concerns the normal form of the stored zones and has pushed the limits of the symbolic state exploration further.

3.2 Symmetry reduction

Many industrial applications and in fact several of the AMETIST case studies have an inherently symmetric structure. Here, symmetries are understood as permutations on the state space which are actually based on permutations on the structure of individual states. For instance, a system may contain several identical components with likewise identical relations to the other components. While it is possible that during the evolution of a system with symmetries, the individual components actually are in different states, the state space itself preserves these symmetries.

Already in year one, we were able to report on symmetry reductions for timed automata. In summary, this work has been consolidated, significantly improved and published. As reported last year, UppAal has been extended with symmetry reductions based on Scalar Sets. This data structure is designed so that all operations on it preserve the full symmetry, for instance it allows to model unordered arrays. Permutations of the scalar set (say the indexes of the unordered array) yield equivalent states. In order to obtain reductions in symbolic state space exploration, the goal is to select one or just a few representatives of an equivalence class of symmetric states. Typically, this is done using a measure and searching for a minimal representative with respect to this measure.

The improvement to last year's work reported in [5], direct result of discussions at a project meeting, concerns the identification of "representative zones" in an equivalence class. Based on the observation that the zone codes the "reset order" of clocks, a significantly smaller, in case of multiset symmetries even canonic set of representatives can be found with remarkable reductions, leading to a new record in the Fischer's protocol benchmark.

3.3 Partial order approach and event zones

Again, in last years deliverable, we reported on a new partial order approach based on "event zones". It was stated: *"One approach to compensate the combinatorial explosion due to the parallel structure of system specifications is known under the name of partial order reductions. Basically, these methods exploit structural knowledge about commuting transitions a and b such that the transition sequences ab and ba are equivalent (leading to the same state) in order to cut branches from the search tree. For untimed systems, certain reduction techniques out of this class have proven to be very effective and to allow analysis of bigger systems.*
For timed systems, past approaches have been much less fruitful, basically because pairs of transitions that do commute in the timed automaton no longer commute on the symbolic simulation graph. On the other hand, previous alternative definitions of the simulation graph preserving commutation suffered from considerable problems with the above mentioned bounds abstraction."

Then, we presented a preliminary version of a new approach, still figuring under the title of "partial order reductions".

Meanwhile, this work has been theoretically consolidated, published [6], and, most importantly, rigorously implemented in the ELSE tool [12]. We have found, and now present it this way, that the approach is not about partial order reduction, but about avoiding the state splitting of classical region and zone automata. And indeed, on some benchmarks (due to the prototype stage of the tool, we were not yet able to consider the case studies) the resulting reductions are highly remarkable.

The second year also has given birth to a new tool, which, while not achieving the performance of UppAal, presents an alternative experimental platform. Due to its alternative algorithmic structure, we have finally given up linking to CADP (what is still reported in [12]) and developed genuine libraries, notably for avoiding memory leaks and memory fragmentation that would otherwise result from enormous search stacks and queues.

3.4 Logic approaches

An approach very different from state exploration is based on the idea of "bounded model checking". The idea is to search for a path of the simulation graph of a specified length and with a desired property by the reduction to a constraint satisfaction problem.

Difference logic is a simple logic based on continuous and boolean variables, where – similar to what has been said before – the continuous variables may be subject to difference constraints $X - Y \leq c$. Differently from the case of state exploration, such constraints may now occur in boolean combinations with the boolean variables, in particular involving disjunctions. The "bounded reachability" approach to timed automata now requires two building blocks: An efficient coding of paths of a given length in difference logic; and a method for efficiently solving constraints expressed in difference logic.

Last year we presented fundamental work and a tool (dedicated SAT solver), that was already quite interesting, but not yet competitive to discrete approaches, since it did not involve conflict analysis. Currently, we present a completely reimplemented tool with conflict analysis and improved algorithms and data structures that can beat the previous implementation by several orders of magnitude [4].

3.5 Priced Timed Automata

Very important to scheduling applications of timed automata is a variant with cost, "Priced Timed Automata". Cost is a linear function of the time passed in a state (for example working on a task) and can vary from state to state. For example, the Axxom case study has been modeled with priced timed automata.

The basic data structure for the analysis of priced timed automata is a "priced zone", i.e. a zone together with a linear cost function (of the clocks). This complicates operations of zones and requires linear programming techniques in testing whether a zone is covered by a better zone that was already explored.

In [9], we show how the simple structure of the linear programs encountered during symbolic minimum-cost reachability analysis of priced timed automata can be exploited in order to substantially improve the performance of the current algorithm. The idea is rooted in duality of linear programs and we show that each encountered linear program can be reduced to the dual problem of an instance of the min-cost flow problem. Thus, we only need to solve instances of the much simpler min-cost flow problem during minimum-cost reachability analysis. Experimental results using Uppaal show a 70-80 percent performance gain. As a main application area, we show how to solve energy-optimal task graph scheduling problems using the framework of priced timed automata.

3.6 Scheduling with timed automata and linear programming

In quickly finding a good solution with depth first search, estimating heuristics are used together with a best first search strategy. The heuristics concern the estimation of the remaining durations and are "optimistic" underapproximations.

As we show in [8], the estimation can be improved by linear programming. The idea is to relax, as is known from MILP approaches, discrete choices into linear functions. The resulting estimations are still optimistic, but less optimistic. In particular, they guide the search much better and lead to fewer explored states in a timed automata search (with specialized zones). However, this better choice comes with an overhead for solving LP problems. It is therefore important to produce simple LP problems tailored to the scheduling application.

3.7 Hybrid systems

With partial funding from AMETIST, different approaches have been developed that are not necessarily restricted to Timed Automata but are applicable to more general classes of hybrid systems.

The idea of combining graph search with embedded programming techniques, as mentioned in the previous section, has been extended to the optimization of hybrid automata with arbitrary nonlinear continuous dynamics [10]. It is shown that the efficiency of determining optimal hybrid behaviors can be increased considerably (compared to pure mixed integer nonlinear programming) by the following principle: (1) the optimization of the continuous and discrete degrees of freedom is separated in a two-stage procedure, where each stage employs a tailored data structure, (2) the discrete degrees of freedom are optimized by tree search including branch-and-bound strategies and cost-based heuristics, and (3) the continuous degrees of freedom are optimized by the solution of embedded nonlinear programming that involves numerical simulation of the hybrid dynamics.

With respect to the algorithmic analysis of hybrid automata, the approach of counterexample-guided abstraction refinement has been further developed [3, 11]. The approach aims at reducing the complexity of verifying safety properties of hybrid automata with nonlinear continuous dynamics by iteratively computing abstractions, generating counterexamples, and validating the latter for the hybrid model. Within the last year, this approach has been improved in two respects: (a) development of a tailored validation strategy that reduces the costly step of computing reachable hybrid sets as much as possible, and (b) introducing a scheme for (in-)validating fragments of counterexamples (rather than complete counterexamples). For several examples, the computational effort for verification could be significantly reduced by this approach.

4 Conclusions and Outlook

The work in this task witnesses of many interesting ideas and a practical progress in the data structures and algorithms for timed automata, as it was expected from the project. The work on these topics is very labor intense, behind the compressed presentations in texts are many thousands lines of code that have been written for the evaluation of the methods. Several improvements that have begun in the first year and have been consolidated in the second year represent record breaking achievements for timed automata tools.

We also want to mention a significant number of ongoing PhD theses with at least partial AMETIST support. In year two two theses with a strong link to the current task have been concluded [7, 1].

A particular challenge of the third year will be the integration of improvements on the level of data structures that are, as we hope, sufficiently othorgonal to be combined: The symmetry approach for timed automata could very fruitfully be combined with event zones, both in turn should see their application domain extended to priced timed automata, etc. While surprises are never to be excluded, the third year of AMETIST should serve to stabilize and integrate the substancial improvements to the timed automata approach achieved in the first two years. Moreover, where this is not yet the case, the improvements should be evaluated on the case studies of the project for a solid assessment of the progress achieved in the project.

References

[1] G. Behrmann. *Data-structure Analysis for Formal Verification.* PhD thesis, Aalborg University, 2003. Phd thesis.

[2] G. Behrmann, P. Bouyer, K. G. Larsen, and R. Pelnek. 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. Available from World Wide Web: `http://www.lsv.ens-cachan.fr/Publis/PAPERS/BBLP-tacas04.ps`.

[3] E. Clarke, A. Fehnker, Z. Han, B.H. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. Journal Foundations of Computer Science*, 14(4):583–604, 2003. Available from World Wide Web: `http://ametist.cs.utwente.nl/INTERNAL/PUBLICATIONS/DORTMUNDPublications/IJFCS03.pdf`.

[4] S. Cotton, E. Asarin, O. Maler, and P. Niebert. Some progress in satisfiabilty checking for difference logic. 2004. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/formats.ps`. Submitted for publication.

[5] M. Hendriks, G. Behrmann, K.G. Larsen, P. Niebert, and F. Vaandrager. Adding symmetry reduction to UPPAAL. 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 43–56. Springer-Verlag, 2003. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/symmetries2.ps`.

[6] 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. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/tacas04.pdf`.

[7] Moez Mahfoudh. *On Satisfaiblity Checking for Difference Logic.* PhD thesis, Université Joseph Fourier, Grenoble, 2003.

[8] 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*, 2004. (submitted).

[9] J. Rasmussen, K. G. Larsen, and K. Subramani. Scheduling using priced timed automata. In *Proc. 10th Int. Conf. of Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2004)*, volume 2988 of *Lecture Notes in Computer Science*, pages 220–235. Springer-Verlag, 2004. Available from World Wide Web: `http://www.springerlink.com/app/home/contribution.asp?wasp=9a0qbvyyrjdjt6rvmewp&referrer=parent&backto=issue,19,43;journal,31,1528;linkingpublicationresults,id:105633,1`.

[10] O. Stursberg. Dynamic optimization of processing systems with mixed degrees of freedom. In *Proc. 7th Int. Symposium on Dynamics and Control of Process Systems*, 2004. Available from World Wide Web: `http://ametist.cs.utwente.nl/INTERNAL/PUBLICATIONS/DORTMUNDPublications/DYCOPS04.pdf`. (to appear).

[11] O. Stursberg, A. Fehnker, Z. Han, and B.H. Krogh. Verification of a cruise control system using counterexample-guided search. *Control Engineering Practice*, 2004. Available from World Wide Web: `http://ametist.cs.utwente.nl/INTERNAL/PUBLICATIONS/DORTMUNDPublications/CEP04a.pdf`. (to appear).

[12] 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. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/else_update.ps`.