Analysis & Tools: State Space Representations

LIF

June 1, 2003

AMETIST DELIVERABLE 2.3.a

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/`

1 Introduction

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.

Workpackage 2 is devoted to algorithmics from a theoretical and practical point of view. Task 2.3 in particular considers the questions of algorithms and 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 part of Task 2.1 concerned with "structure exploitation" and Task 2.2, which is concerned with scheduling and control synthesis.

The actual contributions specific to Task 2.3 in the first year of AMETIST are concerned with timed automata and two distinct constraints solving approaches, discrete and continuous symbolic state exploration on the one hand and reduction to logic constraint solving on the other hand.

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.

In a last section, we will outline some challenges for the second and third year.

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 AAU and VERIMAG have 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. [3], for instance summarizes the heuristics integrated into UPPAAL of AAU, including recent improvements developed in the first year of AMETIST.

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 the following, we discuss individual contributions to this task in the first 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.

In [1], static analysis is used for a finer, state dependent abstraction. For each discrete control state, the abstraction used may be a different one. As a result, substantially reduced symbolic state spaces are possible.

On the other hand, [5] shows by the help of a counter example, that these abstraction are not possible in the presence of difference constraints on clocks in transition conditions, a fact that has been overlooked for years. The interest in comparing clocks in transitions is that they provide a way of having much more concise models, in particular for scheduling applications.

Not only are the abstractions used for normal timed automata wrong when considering clock difference constraints in transitions; it is shown that no abstraction of the same type is possible. Yet subclasses of systems where this abstraction is possible despite clock difference conditions in transitions are possible.

3.2 Loop acceleration

Related to the bounds abstraction is an approach to solve the problem of "different time scales" of components in timed automata. Different time scales do often occur in real-time systems, e.g., a polling real-time system samples the environment many times per second, whereas the environment may only change a few times per second. When these systems are modeled as (networks of) timed automata, the validation using symbolic model checking techniques can significantly be slowed down by unnecessary fragmentation of the symbolic state space. In [9] a technique is introduced to compensate a part of the resulting explosion with the help of a "loop acceleration" technique. The idea is to consider an arbitrary number of executions of certain loops and to compute a constraint for the union of all these loops. Under some restrictive but practically relevant conditions, loops allowing such an acceleration can be detected and symbolically explored in a single step, thus covering all intermediate states at once. It is shown that this syntactical adjustment does not alter reachability properties and that it indeed is effective. The exact acceleration technique is illustrated with run-time date obtained with the model checkers UPPAAL and KRONOS.

3.3 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.

For discrete systems, symmetries have been used with resulting remarkable reductions. One particularly successful approach is based on an inherently symmetric modeling data structure, *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.

In [7, 8], the scalar set approach is carried over to timed automata and integrated into UPPAAL. On the theoretical side, this implies lifting the symmetry from concrete to symbolic states. Permutations on the Scalar set then imply permutations on the constraint matrices. Several heuristics

for obtaining minimal representatives on the level of constraint matrices are explored.

The first experimental findings are very encouraging: The resulting version of UPPAAL allows to analyze some systems that are far beyond the reach of classical timed automata. An error in a communications protocol until now believed to be correct was found with the symmetry reduction.

3.4 Exploration and hashing

Storing visited states in order to avoid double exploration is the essential method of using memory for acceleration of the search. In the space-time trade-off, the more states are remembered, the more memory is needed, but the fewer states are explored twice.

However, the actual gain of discovering that a certain state has been visited before may go far beyond the time needed for the exploration of a single state and the computation of its immediate successors. In fact, the entire graph reachable from this state is not revisited.

Two important approaches for improving algorithm efficiency in this respect are to reduce the amount of memory required for storing individual states, and to store just a well chosen part of all states.

In [10], techniques for compressing the representation of symbolic states are shown, that allow to remember bigger numbers of visited states. The most important aspect of this approach is to transform arbitrary conjunctions of differences into a canonical minimal form, which is often a sparse matrix and which allows a fast test of constraint implication.

In [2], heuristics to store "important" states only based on a static coverability analysis is explored. The fundamental idea is to store a sufficient subset of states such that on each explored path a sufficient number of visited states and inevitable states are remembered. The method guarantees termination and turns out work very well on a selection of case studies.

3.5 Partial order approach and event zones

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.

In [6, 15], a radically different approach is chosen, that on the one hand does lead to an infinite simulation graph that preserves all commutations, but on the other hand allows to use essentially the same abstraction as classical timed automata to cut a finite/small fragment out of it.

A first result, without any partial order reduction in the sense of the word, is that the part of the simulation graph actually explored can be substantially smaller than the simulation graph of classical timed automata. This is due to the fact that the phenomen of zone splitting as a result of interleavings in the classical simulation graph is fully avoided.

On the other hand, in [4, 11], a new approach to partial order reductions based on the structure of executions is explored. While this approach is currently only developed for discrete systems (thus applicable, e.g. to discrete timed automata), this work is intended to be used also for timed systems on the basis of [6]. Indeed, both approaches have the same structure of simulation graph and a similar cutting technique in common.

3.6 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.

Indeed, automatic translations for a fragment of IF to difference logic have been developed [14]. On the other hand, a specialized SAT solver for difference logic, which generalizes a version of the Davis-Putnam method from boolean constraints to mixed constraints in difference logic, has been developed [12, 13]. The emphasis of the approach is to integrate the solution of difference constraints using DBMs with the Davis-Putnam method.

In comparison with state space exploration methods, this method is not exhaustive (it only allows to reason about paths of predefined length), but has the potential to allow the analysis of systems much bigger than is possible with the exploration 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. It will contributed to an overall performance gain of the timed automata tools maintained in the project consortium.

Of course, we hope to continue to progress in a similar manner in the following year, with the additional challenges posed by the case studies: These exposed some surprises concerning complexity, that suggest dedicated improvements of data structures and algorithms, with the ultimate goal of going further with these case studies than is currently possible.

References

[1] G. Behrmann, P. Bouyer, E. Fleury, and K.G. Larsen. Static guard analysis in timed automata verification. In *In Proc. 9th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003) Warsaw, Poland*, volume 2619 of *Lecture Notes in Computer Science*, pages 254–277. Springer Verlag, 2003. Available from World Wide Web: `http://www.lsv.ens-cachan.fr/Publis/PAPERS/BBFL-tacas-2003.ps`.

[2] G. Behrmann, K.G. Larsen, and R. Pelanek. To store or not to store. Lecture Notes in Computer Science. Springer Verlag, 2003.

[3] Gerd Behrmann, Johan Bengtsson, Alexandre David, Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL implementation secrets. In *Proc. of 7*th *International Symposium on Formal Techniques in Rea-Time and Fault Tolerant Systems*, 2002. Available from World Wide Web: `http://www.docs.uu.se/docs/rtmv/papers/bbdlpw-ftrtft02.ps.gz`.

[4] S. Bornot, R. Morin, P. Niebert, and S. Zennou. Black box unfolding with local first search. In *TACAS'2002*, volume 2280 of *LNCS*, page 386 ff., 2002. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/tacas02.pdf`.

[5] Patricia Bouyer. Untameble timed automata. volume 2607 of *Lecture Notes in Computer Sicnece*, pages 620–631. Springer, 2003. Available from World Wide Web: `http://www.lsv.ens-cachan.fr/Publis/PAPERS/Bou-stacs2003.ps`. In Proc. 20th Ann. Symp. Theoretical Aspects of Computer Science (STACS'2003), Berlin, Germany, Feb. 2003.

[6] S. Zennou D. Lugiez, P. Niebert. Clocked mazurkiewicz traces for partial order reductions of timed automata, 2003. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/clockedmazu.pdf`.

[7] M. Hendriks. Enhancing Uppaal by exploiting symmetry. Report NIII-R0208, Nijmegen Institute for Computing and Information Sciences, University of Nijmegen, October 2002. Available from World Wide Web: `http://www.cs.kun.nl/ita/publications/papers/martijnh/NIII-R0208.pdf`.

[8] M. Hendriks, G. Behrmann, K.G. Larsen, and F.W. Vaandrager. Adding symmetry reduction to Uppaal, May 2003. Available from World Wide Web: `http://www.cs.kun.nl/ita/publications/papers/fvaan/symmetry.html`. Submitted.

[9] M. Hendriks and K.G. Larsen. Exact acceleration of real-time model checking. *Electronic Notes in Theoretical Computer Science*, 65(6), April 2002. Available from World Wide Web: `http://www.cs.kun.nl/ita/publications/papers/martijnh/TPTS02.pdf`.

[10] K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Compact data structure and state-space reduction for model-checking real-time systems. *In Real-Time Systems - The International Journal of Time-Critical Computing System*, 25(1), 2003. Available from World Wide Web: `http://www.docs.uu.se/docs/rtmv/papers/llpw-rts02.ps.gz`.

[11] D. Lugiez, P. Niebert, and S. Zennou. Dynamic bounds and transition merging for local first search. In *Model Checking Software*, volume 2318 of *LNCS*, pages 221–229, 2002. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/sw02.pdf`.

[12] M. Mahfoudh, P. Niebert, E. Asarin, and O. Maler. A satisfiability checker for difference logic. In *SAT*, 2002. Available from World Wide Web: `http://www-verimag.imag.fr/~maler/Papers/solver.ps`.

[13] Moez Mahfoudh. *On Satisfaiblity Checking for Difference Logic*. PhD thesis, UJF Grenoble. Available from World Wide Web: `http://www-verimag.imag.fr/~maler/Papers/thesis-moez.ps`. submitted Mars 2003.

[14] P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, N. Jain, and O. Maler. Verification of timed automata via satisfiability checking. In W. Damm and E-R Olderog, editors, *FTRTFT*, volume 2469 of *LNCS*, pages 225–244. Springer, 2002. Available from World Wide Web: `http://www-verimag.imag.fr/PEOPLE/Oded.Maler/Papers/timedbmc.ps`.

[15] Sarah Zennou, Manuel Yguel, and Peter Niebert. Else: A new symbolic state generator for timed automata, 2003. Available from World Wide Web: `http://www.cmi.univ-mrs.fr/~niebert/docs/else.pdf`.