Timed automata and constraints
Abstract

Hubert Comon and Yan Jurski
Laboratoire Spécification et Vérification
Ecole Normale Supérieure de Cachan
Hubert.Comon@lsv.ens-cachan.fr

August 18, 1999

The CCL group should be convinced that automata techniques may help in constraint solving. There are a lot of successful examples which have been worked out as part of the CCL activity some of which are summarized in [3]. We can cite for instance applications in theorem proving, which have been carried at MPI (jointly with UPS) [10] or at INRIA (jointly with UPS) [11], set constraints at Lille [8, 13], encompassment constraints at Orsay [7] among others.

Conversely, we claim that constraints may be useful in the analysis of automata properties. We give here an example of such an application of constraints to the analysis of real-time systems.

Timed automata

More precisely, we consider *timed automata* which have been introduced by R. Alur and D. Dill [1] as a possible model of real time systems. Such automata consist of a finite control (states) and are equipped with a finite number of *clocks*. Such real-valued clocks run at the same speed and can be reset by some transitions. Transitions between states are also guarded by inequalities on clock values. Finally, each state is associated with an *invariant* which should be satisfied as long as the control stays in the corresponding state.

A configuration of a timed automaton is given by a pair $(q, \vec{V})$ of a control state and a vector of reals (the clock values). A move from a configuration to another is possible

- either if we stay in the same control state and add some real constant to each component of $\vec{V}$, provided that the invariant at q remains satisfied
- or if the guard of some transition from q to q' is satisfied and the clock values reset by the transitions yield a zero component in the resulting configuration.

We will always assume without loss of generality that there is a clock τ which is never reset.

Example 1 An example of a timed automaton is displayed on figure 1. As usual, invariants are written in the states and guards (also called *enabling conditions*) label the edges. The variables which are reset by a transition are written inside brackets. We assume that there are three clocks x, y, τ.

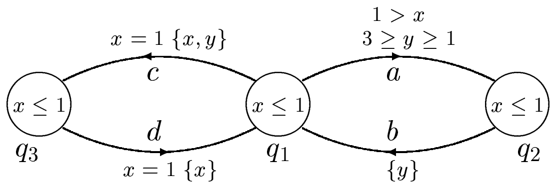

Figure 1: A timed automaton

If q, q' are two states of the automaton, then $\xrightarrow[q,q']{*}$ is the (many steps) reachability relation between configurations $(q, \vec{V})$ and $(q', \vec{V}')$. Our main result is that this binary relation between vectors of reals is expressible in some constraint system, which yields decision procedures for a number of temporal properties of timed automata (some of which are not expressible in known decidable timed temporal logics, see [5]).

The constraint system

The terms are built on rational constants, variables, and addition. The atomic constraints are either inequalities between terms ($t_1 < t_2$ or $t_1 \leq t_2$) or monadic constraints $\mathsf{Int}(x)$. The formulas of the system are arbitrary first-order formulas built on these atomic ones.

The formulas are interpreted in the non negative real (or non negative rational, this does not make any difference) numbers, Int being interpreted as the set of natural numbers.

This constraint system (let us call it $\mathcal{C}$) is decidable as it can be embedded in S1S (see e.g. [2]). Hence, we can use again automata to solve the constraints ! (But this time we will use Büchi automata).

Example 2 If we consider for instance transitions c, d only, we can express $\xrightarrow[q_1,q_1]{*}$ using the formula:

$$\begin{array}{ll} \exists \tau_1, x_1, y_1, t_1, t_2. & \tau_1 = \tau + t_1 \wedge x_1 = x + t_1 \wedge y_1 = y + t_1 \wedge x_1 \leq 1 \\ \wedge & \exists n. \mathsf{Int}(n) \wedge \tau' = \tau_1 + 2n + 1 - t_2 \wedge x' = t_2 \wedge x' \leq 1 \wedge y' = 1 + t_2 \end{array}$$

t_1 is the time spent before the first transition c is fired and is specified on the first line. τ_1, x_1, y_1 are the values of the clocks at that date. Then n is the number of times the loop cd is executed and t_2 is the time spent in q_1 after the last transition d has been fired.

This is typically what we will get from our formula computation.

The main result and its proof

Theorem 1 ([5]) *For every pair of states (q, q'), $\xrightarrow[q,q']{*}$ is expressible in $\mathcal{C}$.*

The full proof can be found in [6]. It can be decomposed in 2 main steps:

1. We prove that any timed automaton can be emulated by a flat timed automaton.

Without giving all definitions, simply keep in mind that *emulation* preserves the binary reachability relation and that *flat automata* are automata which do not contain nested loops. Hence this result states that, as far as clock values are concerned, timed automata can be assumed to be flat without loosing any expressivity.

2. Flat timed automata can be seen as flat counter automata, thanks to Fribourg's trick [9]. Then we use a result in [4] which shows how some fixed points can be computed in the additive theory of real numbers (resp. natural numbers).

Step 1 itself can be decomposed in three intermediate steps:

1.1 We define a finite index right compatible equivalence relation $\sim$ on transition sequences, which generalizes the region construction of [1].

1.2 We prove a commutation lemma: if $w \sim w' \sim w''$ then $w \cdot w' \cdot w'' \approx w' \cdot w \cdot w''$ where $u \approx v$ if and only if u and v define the same relation on clock values.

1.3 We prove a formal language property: if $\sim$ is as above, then there is a set of representatives for $\approx$ which is accepted by a flat (word) automaton.

This relies on classical arguments of combinatorics on words and bit of rewriting.

Conclusion

We defined a translation

Timed automata $\longrightarrow$ Constraints $\longrightarrow$ Büchi automata

which allows to decide properties of timed automata, such as delay analysis.

References

[1] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[2] B. Boigelot, S. Rassart, and P. Wolper. On the expressiveness of real and integer arithmetic automata. In *Proc. 25th Int. Colloqium on Automata Languages and Programming*, volume 1443 of *Lecture Notes in Computer Science*, pages 152–163, 1998.

[3] H. Comon, M. Dauchet, R. Gilleron, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. A preliminary version of this unpublished book is available on `http://l3ux02.univ-lille3.fr/tata` , 1997.

[4] H. Comon and Y. Jurski. Multiple counters automata, safety analysis and presburger arithmetic. In A. Hu and M. Vardi, editors, *Proc. Computer Aided Verification*, volume 1427 of *LNCS*, pages 268–279, Vancouver, 1998. Springer-Verlag.

[5] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. In *Proc. Conf. on Concurrency Theory (CONCUR)*, Lecture Notes in Computer Science. Springer-Verlag, 1999. To appear.

[6] H. Comon and Y. Jurski. Timed automata and the theory of real numbers. Technical report, LSV Research Report, June 1999. avalaible at http://www.lsv.ens-cachan.fr/Publis/publis-lsv-index.html.

[7] Hubert Comon and Florent Jacquemard. Ground reducibility is exptime-complete. In *Proc. IEEE Symp. on Logic in Computer Science*, Varsaw, June 1997. IEEE Comp. Soc. Press.

[8] P. Devienne, J.-M. Talbot, and S. Tison. Solving classes of set constraints with tree automata. In *Proc. Conf. on principles and Practice of Constraint Programming*, volume 1330 of *Lecture Notes in Computer Science*, pages 62–76, 1997.

[9] Laurent Fribourg. A closed form evaluation for extending timed automata. Technical Report 1998-02, Laboratoire Spécification et Vérification, ENS Cachan, March 1998.

[10] Florent Jacquemard, Christoph Meyer, and Christoph Weidenbach. Unification in extensions of shallow equational theories. In Nipkow [12], pages 76–90.

[11] Jean-Pierre Jouannaud and Adel Bouhoula. Automata-driven automated induction. In *Twelfth Annual IEEE Symposium on Logic in Computer Science*, Warsaw,Poland, June 1997. IEEE Comp. Soc. Press.

[12] T. Nipkow, editor. *9th International Conference on Rewriting Techniques and Applications*, volume 1379 of *Lecture Notes in Computer Science*, Tsukuba, Japan, 1998. Springer-Verlag.

[13] J.-M. Talbot, S. Tison, and P. Devienne. Set-based analysis for logic programming and tree automata. In *Proc. Symposium on Static Analysis*, volume 1302 of *Lecture Notes in Computer Science*, pages 127–140, 1997.