Verification of Parameterized Timed Systems

Parosh Aziz Abdulla

Uppsala University, Sweden
parosh@it.uu.se

One of the prominent methods for program verification is that of *model checking* [CES86,QS82]. In the last decade there has been an extensive research effort in order to extend the applicability of model checking to systems with infinite state spaces. There are at least two reasons why a system may be infinite-state:

- A system may operate on data structures with unbounded domains. Examples include real-valued clocks in timed automata [AD94], stacks in push-down automata [BEM97], queues in communicating processes [AJ96], counters in counter machines, etc.
- A system can also be infinite-state because it is *parameterized*. This means that the description of the system is parameterized by the number of components inside the system. In such a case, we would like to verify correctness of the system regardless of the number of processes.

We consider systems which contain both sources of infiniteness; namely parameterized systems of processes each of which behaves as a timed automaton.

Parameterized verification has recently received a lot of attention. One of the earliest works for model checking of parameterized systems was reported by German and Sistla [GS92]. The paper considers systems consisting of an unbounded number of *finite-state processes*, and reduces the problem to a corresponding one for *Vector Addition Systems with States*, a model which is computationally equivalent to Petri nets.

Another important line of research in this area is that of *Regular Model Checking*. Regular model checking was advocated by the paper [KMM$^+$01] as a uniform framework for the analysis of parameterized systems. The idea of regular model checking is to perform symbolic reachability analysis, using regular sets as a symbolic representation of the state space of the parameterized system. Examples of parameterized systems which can be analyzed in this manner are systems consisting of homogeneous finite-state processes connected in linear, ring-formed, or tree-like topologies [AJNS04,ALdR05,BT02,BLW03].

We consider verifying safety properties for *Timed Networks (TNs)* [AJ03]: systems consisting of an arbitrary set of timed automata. Such systems embody both of the two reasons for being infinite-state: they use an infinite data structure (namely clocks which can assume values from the set of real numbers), and they are parameterized in allowing an arbitrary set of processes. TNs cannot be analyzed within earlier frameworks for parameterized verification, since the individual components are no longer finite-state processes.

There are several examples of protocols which can be modelled as TNs. For instance, Fischer's protocol [SBM92] achieves mutual exclusion among an arbitrary set of timed processes. This protocol, together with other protocols, such as the Lynch-Shavit protocol [LS92] and the Phillips audio control protocol [BGK$^+$96], are designed to operate correctly regardless of the number of participants. Therefore, parameterized verification is relevant in order to prove correctness of the entire family in one step.

We will consider a hierarchy of TNs defined by the number of clocks allowed inside each timed automaton. First, we consider *single-clock TNs* in which each process is restricted to operate on one clock. Despite this restriction, a single-clock TN operates on an unbounded number of clocks, and hence its behaviour cannot be captured by that of a timed automaton [AD94]. Therefore, it cannot be analyzed through existing tools for timed automata such as KRONOS and UPPAAL. We present a symbolic algorithm for verifying safety properties for *single clock TNs*. The symbolic representation we use is a variant of that of *zones* widely used in tools for model checking timed automata. The difference is that, in our case, all variables of the zone are existentially quantified. As an example, we use our algorithm to verify parameterized versions of Fischer's protocol and the Lynch-Shavit protocol (i.e., we show their correctness regardless of the number of participating processes).

On the other hand, there are many protocols in the literature which can be modelled as TNs where each of the timed automata has more than one clock. For instance, the Phillips audio protocol has two clocks per process. Also, the system described in [MT01] consists of an arbitrary number of nodes, each of which is connected to a set of LANs. Each node maintains timers to keep track of sending and receiving of messages from other nodes connected to the same set of LANs. In a similar way to Fischer's protocol, it is clearly relevant to ask whether we can verify correctness of the protocol in [BGK$^+$96] regardless of the number of senders, or the protocol in [MT01] regardless of the number of nodes. The question is then whether the decidability result can be extended form single-clock systems to multi-clock systems. We answer this question negatively [ADM04a]. In fact, we show that *two-clock TNs* are Turing-powerful, implying undecidability of all non-trivial verification problems.

Finally, we consider TNs which operate on the discrete time domain. In this case, we show that safety properties can be checked regardless on the number of clocks inside each component.

Considering the undecidability result for the dense-time case, we also study restricted classes of TNs [ADM04b]:

– *Open TNs* in which only strict inequalities are allowed on clock values. We show that undecidability is maintained in this case.
– *Closed TNs* in which only non-strict inequalities are allowed on clock values. In a similar manner to discrete TNs, we show that the problem becomes decidable regardless of the number of clocks inside each process
– *Robust TNs*. Using the *robust semantics*, a computation is accepted only if all neighbouring computations are also accepted. We show that undecidability is maintained in the case of robust TNs.

References

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[ADM04a] P. A. Abdulla, J. Deneux, and P. Mahata. Multi-clock timed networks. In *Proc. LICS' 04* 20^{th} *IEEE Int. Symp. on Logic in Computer Science*, pages 345–354, 2004.

[ADM04b] P. A. Abdulla, J. Deneux, and P. Mahata. Open, closed and robust timed networks. In *Proc. INFINITY '04,* 6^{th} *International Workshop on Verification of Infinite-State Systems*, 2004.

[AJ96] P. A. Abdulla and B. Jonsson. Verifying programs with unreliable channels. *Information and Computation*, 127(2):91–101, 1996.

[AJ03] P. A. Abdulla and B. Jonsson. Model checking of systems with many identical timed processes. *Theoretical Computer Science*, 290(1):241–264, 2003.

[AJNS04] P.A. Abdulla, B. Jonsson, M. Nilsson, and M. Saksena. A survey of regular model checking. In *Proc. CONCUR 2004,* 15^{th} *Int. Conf. on Concurrency Theory*, pages 348–360, 2004.

[ALdR05] P. A. Abdulla, A. Legay, J. d'Orso, and A. Rezine. Simulation-based iteration of tree transducers. In *Proc. TACAS '05,* 11^{th} *Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 3440 of *Lecture Notes in Computer Science*, pages 30–44, 2005.

[BEM97] A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. In *Proc. Intern. Conf. on Concurrency Theory (CONCUR'97)*. LNCS 1243, 1997.

[BGK$^+$96] J. Bengtsson, W. O. D. Griffioen, K.J. Kristoffersen, K.G. Larsen, F. Larsson, P. Pettersson, and W. Yi. Verification of an audio protocol with bus collision using UPPAAL. In *Proc.* 8^{th} *Int. Conf. on Computer Aided Verification*, volume 1102 of *Lecture Notes in Computer Science*, pages 244–256, 1996.

[BLW03] Bernard Boigelot, Axel Legay, and Pierre Wolper. Iterating transducers in the large. In *Proc.* 15^{th} *Int. Conf. on Computer Aided Verification*, volume 2725 of *Lecture Notes in Computer Science*, pages 223–235, 2003.

[BT02] Ahmed Bouajjani and Tayssir Touili. Extrapolating Tree Transformations. In *Proc.* 14^{th} *Int. Conf. on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, 2002.

[CES86] E.M. Clarke, E.A. Emerson, and A.P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specification. *ACM Trans. on Programming Languages and Systems*, 8(2):244–263, April 1986.

[GS92] S. M. German and A. P. Sistla. Reasoning about systems with many processes. *Journal of the ACM*, 39(3):675–735, 1992.

[KMM$^+$01] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. *Theoretical Computer Science*, 256:93–112, 2001.

[LS92] N. A. Lynch and N. Shavit. Timing-based mutual exclusion. In *IEEE Real-Time Systems Symposium*, pages 2–11, 1992.

[MT01] I. A. Mason and C. L. Talcott. Simple network protocol simulation within maude. In Kokichi Futatsugi, editor, *Electronic Notes in Theoretical Computer Science*, volume 36. Elsevier, 2001.

[QS82] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In *5th International Symposium on Programming, Turin*, volume 137 of *Lecture Notes in Computer Science*, pages 337–352. Springer Verlag, 1982.

[SBM92] F. B. Schneider, B. Bloom, and K. Marzullo. Putting time into proof outlines. In de Bakker, Huizing, de Roever, and Rozenberg, editors, *Real-Time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*, 1992.