# Infinite-state Model Checking

Rupak Majumdar
MPI for Software Systems, Kaiserslautern, Germany

Model checking was initially developed for finite-state hardware circuits. However, many systems of interest have *infinite* state spaces. For example, a model of a real-time system contains real-valued clocks, and a model of software may contain a potentially unbounded stack of recursive function calls. The purpose of these lectures is to outline model checking techniques for systems with infinite state spaces.

We will start with a generic description of infinite-state models, and show that the general verification problem is undecidable. Then we will study two general techniques that show decidability of model checking: existence of finite (bi)simulations and well-structuredness of the state space. We will show applications of these techniques to timed automata, Petri nets, and abstract model checking.

Next, we focus on infinite-state models derived from software. For infinite data domains, we argue that decision procedures for more general theories allow us to lift" techniques from finite-state model checking to systems with infinite state spaces.We then focus on infinite state spaces arising out of expressive control structures, such as recursive function calls and asynchronous message passing. We shall show that model checking Boolean programs is decidable by reducing to model checking for pushdown systems. We will show that the verification of multi-threaded recursive programs is undecidable in general. We will demonstrate the connections to software model checking by showing model checking of multi-threaded programs with asynchronous calls is decidable. Finally, we consider some heuristics for infinite-state model checking that seem to work well even though the problem is undecidable. These techniques perform symbolic exploration of the state space together with generalization heuristics that try to guess a suitable inductive invariant. We will discuss the application of these techniques to parameterized verification.

**Annotated References:**
A survey on software model checking, which includes discussion on infinite-state model checking, is [9]. We shall assume familiarity with a basic course on the theory of computation, e.g., at the level of [14]. The book by Clarke, Grumberg, and Peled [5] gives a nice introduction to model checking. Abstract interpretation [6] is a general theoretical framework for studying abstractions of infinite-state models. We will mention predicate abstraction [8] as a specific technique. Timed automata and their decision problems was studied in [2]. Well-structured transition systems were described in [1].

For pushdown systems, we will follow the algorithms of [13, 4]. Ramalingam [12] shows the undecidability of multithreaded pushdown programs.We shall study decidability results for subclasses of multi-threaded recursive programs described in [7, 3].

Interpolation [10] is a technique for generalization from bounded explorations for finding inductive invariants. For parameterized systems, we shall also look at other heuristic methods, such as invisible invariants [11].

## References

[1] P.A.Abdulla, K.Čerāns, B.Jonsson, Y.-K.Tsay. *General Decidability Theorems for Infinite-state Systems.* In: Procs. of the 11th Annual Symposium on Logic in Computer Science, pp. 313-321, IEEE Computer Society Press, 1996.

[2] R.Alur, D.Dill. *A Theory of Timed Automata.* Theoretical Computer Science, Vol. 126, pp. 183-235, 1994.

[3] A.Bouajjani, M.Emmi. *Analysis of Recursively Parallel Programs.* In: POPL, pp. 203-214, 2012.

[4] A.Bouajjani, J.Esparza, O.Maler. *Reachability Analysis of Pushdown Automata: Application to Model Checking.* In: CONCUR 97: Concurrency Theory, LNCS 1243, pp. 135-150, Springer, 1994.

[5] E.Clarke, O.Grumberg, D.Peled. *Model Checking.* MIT Press, 1999.

[6] P.Cousot, R.Cousot. *Abstract Interpretation: a Unified Lattice Model for the Static Analysis of Programs.* In: POPL'77, pp. 238-252, ACM, 1977.

[7] P.Ganty, RMajumdar. *Algorithmic Verification of Asynchronous Programs.* ACM TOPLAS, 2012.

[8] S.Graf, H.Saïdi. *Construction of Abstract State Graphs with PVS.* In: CAV'97, LNCS 1254, pp. 72-83,Springer, 1997.

[9] R.Jhala, R.Majumdar. *Software Model Checking.* ACM Computing Surveys, 2009.

[10] K. L. McMillan. *Lazy Abstraction with Interpolants.* In: CAV'06, LNCS 4144, pp. 123-136, Springer, 2006.

[11] A.Pnueli, S.Ruah, L.D.Zuck. *Automatic Deductive Verification with Invisible Invariants.* In: TACAS'01, LNCS 2031, pp. 8297, Springer, 2001.

[12] G.Ramalingam. *Context-sensitive Synchronization-sensitive Analysis is Undecidable.* ACM Transactions on Programming Languages and Systems, Vol. 22(2), pp. 416-430, 2000.

[13] T.Reps, S.Horwitz, M.Sagiv. *Precise Interprocedural Dataflow Analysis via Graph Reachability.* In: POPL'95, pp. 49-61,ACM, 1995.

[14] M.Sipser. *Introduction to the Theory of Computation.* Course Technology, 2005.