Verifying Concurrent Systems with Automata over Infinite Alphabets

Radu IOSIF (VERIMAG, Grenoble)
mailto:Radu.Iosif@univ-grenoble-alpes.fr

1 Background

The verification of algorithms and computer systems has been considered as a research problem starting with the seminal works of Alan Turing, Robert W. Floyd and C.A.R. Hoare. Recently, this domain has experienced significant advances. For instance, the works of Amir Pnueli, Allen Emerson, Edmund Clarke and Joseph Sifakis (founder of the VERIMAG laboratory) on temporal logics and model checking have been rewarded with two Turing prizes (in 1996 and 2007), the Nobel Prize equivalent for computer scientists.

The theory of finite-state automata is recognized as being central to verification of hardware, communication protocols and software systems, being at the core of widespread verification techniques, such as *model checking* [5], where systems are specified as finite-state automata and properties defined using temporal logic [4]. Many results in automata theory rely on the *finite alphabet hypothesis*, which guarantees the existence of determinization, complementation and language inclusion algorithms. Using finite alphabets for the specification of properties and models is however very restrictive, when dealing with real-life computer systems: programs handle data from very large domains, that can be assumed to be infinite (64-bit integers, floating point numbers, strings of characters, etc.) and their correctness must be specified in terms of the data values.

Recently, work carried out by researchers at VERIMAG has led to the definition of powerful extensions of finite automata to infinite alphabets. We started from classical finite-alphabet *alternating automata* [2] in which there are two kinds of transitions: *universal* (the execution continues with all successor states) and *existential* (the execution choses one successor state to continue). We extended this model to accept input from any infinite data domain, such as integers, strings, sets, real numbers or algebraic data types (lists, queues, arrays) that has a well-defined first-order theory, supported by a mainstream Satisfiability Modulo Theories (SMT) solver tool, such as CVC4[1] or Z3[2]. The result is the class of First Order Alternating Automata (FOADA) [3]. In this model, all automata-theoretic boolean operations (union, intersection, complement) are possible in linear time. On the other hand, the emptiness problem is undecidable, in general, but can be dealt with by means of a semi-algorithm that is efficient in most practical cases. An implementation of the automata library is avaliable online: `https://github.com/cathiec/FOADA`

2 Challenges and Goal

The goal of the internship is to use FOADA in order to develop a verification method for concurrent systems consisting of an unbounded number of replicated processes. This problem, known as *parametric verification*, asks whether a system composed of n replicated processes is safe, for all $n \geq 2$. By safety we mean that every execution of the system stays clear of a set of global error configurations, such as deadlocks or mutual exclusion violations. Even if we assume each process to be finite-state and every interaction to be a synchronization of actions without exchange of data, the problem remains challenging because we ask for a general proof of safety, that works *for any number of processes*.

The efficiency of a verification method crucially relies on its ability to synthesize an *invariant*, i.e., an infinite set of configurations that contains the initial configurations, is closed under the transition relation, and excludes the error configurations. In general, automatically synthesizing invariants is challenging and computationally expensive. A solution consists in inferring invariants directly from the structure (and not on the execution semantics) of the interaction network in which the processes are placed, being thus called *structural invariants* [1].

The goal of the internship is using FOADA to define structural invariant of parameterized concurrent systems. Since the input to a FOADA is a finite sequence of data elements, such as integers, it can be used to

[1] http://cvc4.cs.stanford.edu/wiki/About_CVC4
[2] https://rise4fun.com/z3/tutorial

define a set of configurations of a given parameterized system, in which each data element represents the local data of a process. Equally, the error states can be captured by FOADA and the verification amounts to deciding emptiness of a product automaton. We aim at defining a systematic encoding for different classes of Petri Net invariants (traps, siphons, linear invariants) as FOADA and performing an experimental assessment of their quality, on a set of benchmarks (cache coherence protocols, mutual exclusion protocols, control systems, etc.).

The internship consists of both theoretical and implementation work.

3 How to Apply

Send your CV and a transcript of your university grades to:

mailto:Radu.Iosif@univ-grenoble-alpes.fr

References

[1] M. Bozga, R. Iosif, and J. Sifakis. Checking deadlock-freedom of parametric component-based systems. In *25th Intl. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS).*, 2019.

[2] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *J. ACM*, 28(1):114–133, 1981.

[3] R. Iosif and X. Xu. Alternating automata modulo first order theories. In *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part II*, pages 43–63, 2019.

[4] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, SFCS '77, pages 46–57. IEEE, 1977.

[5] M. Vardi and P. Wolper. Reasoning about infinite computations. *Information and Computation*, 115(1):1 – 37, 1994.