MODEL CHECKING INFINITE STATE SYSTEMS USING INFERENCE OF REGULAR LANGUAGES

Pavel Rozehnal
Master Degree Programme (2), FIT BUT
E-mail: xrozeh03@stud.fit.vutbr.cz

Supervised by: Tomas Vojnar
E-mail: vojnar@fit.vutbr.cz

ABSTRACT

Regular model checking is a method for verifying infinite-state systems based on coding their configurations as words over a finite alphabet, sets of configurations as finite automata, and transitions as finite transducers. This paper introduces a new general approach to regular model checking based on inference of regular languages. The method builds upon the observation that for infinite-state systems whose behavior can be modelled using length-preserving transducers, there is a finite computation for obtaining all reachable configurations.

The method is based on the Angluin's L^* algorithm for inference of regular languages that is used in an original way for finding out an invariant of the examined system, which can in turn be used for checking various safety properties of the system.

1 INTRODUCTION

Most of computing systems contain bugs, and hence an important research direction in computer science is looking for automated tools capable of finding these bugs. One promising approach in this area is *model checking (MC)*. MC is a process of checking whether a given model satisfies a given property. The concept is general and applies to all kinds of logics and their models. The various approaches to MC can be categorized to ones that handle finite-state systems (like systems having a finite number of reachable configurations) and those that handle infinite-state systems. MC has been originally developed for finite-state systems, its generalization to infinite-state systems is relatively new. The generalization is motivated by the growing importance of infinite-state systems. Such systems include pushdown systems, (lossy) FIFO-channel systems, systems with counters, or parameterized and dynamic networks of processes. Finite-state MC is usually based on generating and exploring all reachable states (or, for efficiency, at least all heuristically chosen relevant states). Infinite-state MC cannot enumerate all reachable states one-by-one. Instead, their infinite sets are often represented symbolically in a finite way using various formalisms like logics or automata. All states that are members of such finitely represented sets of states are handled at the same time. The key problem is then how to make the computation of reachable states represented in a symbolic way terminate as often as possible. A simple iterative approach (an iterative firing of the system transitions) will usually not converge. One of the symbolic approaches to infinite-state MC is the so-called regular MC.

2 REGULAR MODEL CHECKING

Regular Model Checking (RMC) is based on representing configurations of systems as words over an alphabet Σ, their sets as regular sets (finite automata) over Σ and behavior of systems as *transducers*[1] τ over the alphabet Σ. Apart from a transducer τ determine transitions of the examined system, the user is supposed to supply regular sets *Init* and *Bad*. Then, the basic safety verification problem in RMC is

$$\tau^*(\mathit{Init}) \cap \mathit{Bad} \neq \emptyset$$

More, $\tau^*(\mathit{Init})$ represents all configurations of the system reachable from initial configurations *Init*. Alternatively, one can compute τ^* and subsequently check property

$$\tau^* \cap \tau_{\mathit{Bad}} \neq \emptyset$$

where τ_{Bad} describes prohibited transitions.
However, the problem of this second approach is that computing τ^* which is often more difficult then computing $\tau^*(\mathit{Init})$ and there even exist cases where $\tau^*(\mathit{Init})$ is regular and τ^* is not.
The main problem in RMC is how to make the computation of $\tau^*(\mathit{Init})$ finish as the simple iterative approach computing

$$\tau^*(\mathit{Init}) = \mathit{Init} \cup \tau(\mathit{Init}) \cup \tau(\tau(\mathit{Init})) \cup \dots$$

will usually not stop.
Below, we sketch a new approach to this problem as an alternative to the existing approaches based on automata state partitioning, widening, abstracting, a inference of languages.

3 NEW APPROACH TO REGULAR MODEL CHECKING

The approach is based on a novel use of methods of inference of regular languages and has been proposed in cooperation with members of research group T. Vojnar, P. Habermehl (LIAFA, Paris) and M. Leucker (TU, München).
Our algorithm uses the Angluin's L^*[1] algorithm to find (learn) an approximation of the set $\tau^*(\mathit{Init})$ by generating *an observation table* from which the invariant *Inv* is inferred. This algorithm requires an oracle (a Minimally Adequate Teacher) which provides answers for membership and equivalence queries. The main idea behind Angluin's L^* algorithm is to systematically explore membership of certain strings in the unknown target language and create a DFA with the minimum number of states as a conjecture for the target set. If the conjecture is incorrect, the string returned by the teacher is used to correct the conjuncture, possibly after more membership queries. The algorithm uses a two-dimensional observation table which maintains a prefix closed set S of possible states of the target DFA as rows and a suffix closed set E denoting experiments needed to distinguish between these states. If we are able to find out the observation table which satisfies the conditions of the so-called consistency and closedness [1] we can extract the conjectured DFA.
In order to be able to apply the Angluin algorithm to compute an invariant of the system as a safe overaproximation of its reachable states in our setting, we need to be able to answer membership and equivalence queries. We answer membership queries by checking whether a given state can be reached and/or whether it reaches bad states. This check is decidable as the set of states forwards/backwards reachable for a single configuration is finite in length preserving systems.

[1]Transducers have transitions inscribed by pairs of letters `a/b` with the meaning of `a` to be rewrite to `b`.

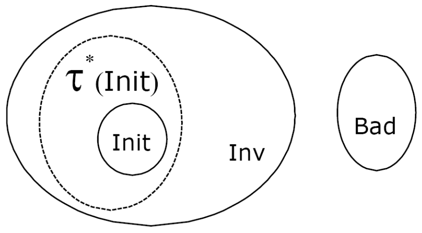

Figure 1: The main relations among regular sets

Then the equivalence queries are approximated by checking whether the inferred set is an invariant (i.e. a set closed with the application τ) and whether it is a safe overapproximation of the reachable states (includes *Init*, but does not intersect *Bad*). Sometimes, we are not able to answer memberhip queries in an unambiguous way (a given state may, but needs not, be included in the invariant). To resolve such cases, we use the Bierman algorithm which is able to resolve no possibility of asking any queries and it has to supply a minimal possible DFA that accepts/rejects the given string.

4 IMPLEMENTATION

We are currently implementing the approach in Prolog (we have chosen the YAP Prolog system) using the FSA library[2] for manipulation with finite-state automata. As a SAT solver, we are going to use the fast and simple implementation MiniSat based on *the chaff* algorithm. The interconnection between YAP Prolog and MiniSat will be done by exploiting the ability of YAP to encapsulate C functions in predicates.

5 CONCLUSION

We have sketched a new approach to RMC based an inference of regular languages. The next step in our work is to finish a prototype implementation of this algorithm and evaluate its performance.

ACKNOWLEDGMENT

This work was supported by the Czech Grant Agency within projects 102/07/0322, the Czech-French Barrande project 2-06-27, and the Czech Ministry of Education under the Research Plan No. MSM 0021630528.

REFERENCES

[1] Angluin, Dana. Learning Regular Sets from Queries and Counterexamples. Information and Computation, pages 87-106, 1987.

[2] Ahmed Bouajjani, Bengt Jonsson, Marcus Nilsson, and Tayssir Touili: Regular model checking. In Computer Aided Verification, pages 403-418, 2000.

[3] P. Habermehl and T. Vojnar: Regular model checking using inference of regular languages, http://citeseer.ist.psu.edu/habermehl04regular.html, 2004.

[2]http://www.let.rug.nl/˜vannoord/Fsa/