Fully Symbolic Bisimulation Minimization of Probabilistic Systems

Develop and implement a fully symbolic bisimulation quotienting algorithm for probabilistic systems

What is probabilistic model checking all about?

In times of ever more complex system designs, it becomes increasingly difficult and often even impossible to manually analyze systems. Yet, for certain systems, e.g. safety-critical ones or ones for which the repair of previously undetected errors may be very cost-intensive, it is crucial to determine whether they behave correctly with respect to the requirements that they are intended to fulfill. Model checking is a fully automated technique to verify properties on a model of the system in a mathematically rigorous way.

Most often real-world models tend to involve stochastic and/or timed behavior. For example, hardware components inside the system may fail with a given probability or it might be of crucial interest that the system satisfies certain timing constraints. Quantitative model checking extends conventional model checking in a way that enables formal analysis of models involving quantitative aspects.

What is to be done?

The runtime of probabilistic model checking is dominated by expensive numerical computations. However, many models contain a lot of redundancy and minimizing the state space has proven to shorten computation times significantly. While preserving important properties of the original model, bisimulation minimization typically requires the full state space of the system to be available prior to minimization. Often this is problematic, because of memory requirements and the time needed to build the model. We devised a method that extracts the bisimulation quotient system from a high-level specification without building the full state space via queries that are dispatched to a satisfiability-solver. This circumvents the aforementioned problem, but suffers from other drawbacks, such as excessive formulae sizes. We believe that investigating whether Binary Decision Diagrams (BDDs) can be used instead of formulae may well be beneficial for the technique. So, the main steps are:

1. Determine how the method can be adapted to BDDs.
2. Implement a prototype in a language of your choice...
3. ... and evaluate its performance.
4. Optionally, there are various extensions that could be investigated, e.g. dealing with non-deterministic systems, additional quantitative measures, etc.

Optimally, you should...

- have programming experience (preferably in C++, but almost any language is fine),
- enjoy the process of research, i.e., experimenting with new ideas,
- have basic knowledge in automata theory,
- be capable of working independently.

All other knowledge/skills can be obtained on-the-fly.

We offer...

- a challenging task that not only requires pure coding skills, but also creativity and ingenuity,
- work at the interface of theory and application,
- a nice coffee machine.

Contact

Christian Dehnert, dehnert@cs.rwth-aachen.de, Tel. 0241/80-21203.