MARIA: Modular Reachability Analyzer for Algebraic System Nets

Application

MARIA is a powerful tool designed to aid engineers in modelling and solving concurrency related problems in parallel and distributed computing systems.

MARIA was written at the Laboratory for Theoretical Computer Science of Helsinki University of Technology. Many ideas are based on the preceding tool PROD, but the modular design is completely new.

Benefits

- Construct a formal model to avoid or to solve ambiguities in verbal specifications
- Detect design flaws in the planning stage, before writing any code
- Powerful formalism eases model construction both for humans and automatic translators
- Freely available under the GNU Copyleft
- Written in portable C++; runs on personal workstations as well as big mainframes

Features

- Based on algebraic system nets, a truly *formal* class of high-level Petri nets
- Sophisticated data types and powerful operations with solid error checking bring additional safety
- Textual input format; an EMACS mode is supplied
- Both textual and graphical user interfaces
- Interactive simulation and visualisation of state spaces and error traces
- Exhaustive reachability analysis and LTL model checking with fairness constraints
- Probabilistic search for checking safety properties
- An option for translating models to C code for speeding up exhaustive analysis

Analysing Industrial Systems

The expressive power of MARIA's formalism is close to high-level programming and specification languages (such as Java and SDL). The powerful queue and stack operations of the language make it possible to model complex communications without introducing superfluous intermediate states that complicate analysis.

Users do not need to be familiar with the internal formalism of the analyser. An automatic translator, a *language-specific front-end* can hide the underlying formalism from the user:

- translate user programs to the internal formalism
- allow desired properties to be specified in terms of the application
- display erroneous behaviour as an execution sequence of the application

A generic formalism has some advantages over specialised formalisms. Implementing new analysis methods immediately benefits all languages for which a translation exists.

A front-end for SDL has been implemented, and some plans for a Java front-end exist. The data type system of MARIA makes it easy to translate expressions and message passing.

Overview

The tool can be run both interactively (textually and graphically) and in batch mode. There are several modes of operation:

- exhaustive reachability analysis
- interactive simulation: generate successors for the states selected by the user
- on-the-fly verification of safety and liveness properties specified in temporal logic
- unfolding nets, optionally reduced with a "coverable marking" algorithm

```
typedef unsigned (1,5,10,50,100,500) money_t;

place customer money_t: 10,2#5;
place cashier money_t: 3#5,10#1;

trans smaller
in {
  place customer: big;
  place cashier: small, (big/small-1)#small;
}
gate big > small
out {
  place cashier: place customer;
  place customer: place cashier;
};

trans bigger
in {
  place customer: small, (big/small-1)#small;
  place cashier: big;
}
gate big > small
out {
  place customer: place cashier;
  place cashier: place customer;
};
```

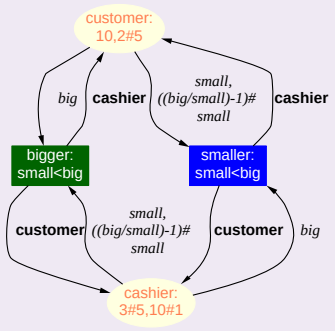

Figure 1: An algorithm for changing money

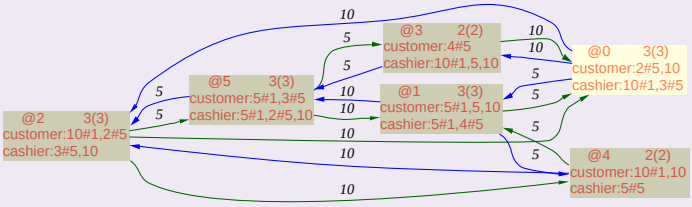

Figure 2: The states of the money-changing algorithm

The tool manages the reachability graph (reachable states and actions between states) in disk files. Keeping all data structures on disk has some advantages:

- the analysis can be interrupted and continued later
- the generated reachability graph can be explored on a different computer
- memory capacity is not a limit: a high-level model with 15,866,988 states and 61,156,129 events was analysed in 5 MB of RAM (and 1.55 GB of disk)

A probabilistic verification option does not use disk files but constructs a set of reachable states in memory. It is most useful when verifying safety properties non-interactively.

Graph Exploration

The reachability graph can be explored and browsed in many ways. Expressions and temporal logic formulae can be evaluated in different states. It is possible to find the shortest path between a specified state and a set of states. MARIA can compute the strongly connected components of the graph.

Evaluating temporal logic expressions in a partially generated reachability graph causes new states to be added to the graph, until a counterexample violating the property being verified is found or the property is found to hold.

Simulation can be a useful tool when developing new Petri Net models. The graph explorer performs simulation in a very transparent way: when one lists the successors of an unprocessed state, the successors will be generated, stored in the reachability graph and shown to the user.

Figure **??** illustrates a simple system both in the MARIA input language (as highlighted by EMACS) and its graphical presentation. Figure **??** illustrates the state space of this system. The graphs in both figures were produced by MARIA and manually edited before laying them out with GRAPHVIZ.

Powerful Algebraic Operations

In addition to basic programming language constructs, there are built-in operations for

- managing buffers (queues and stacks),
- basic multi-set arithmetics (union, intersection, difference, mappings), and
- aggregation: multi-set summation, existential and universal quantification.

Aggregation over a dynamic range of indexes is a particularly powerful construct and extremely useful when modelling distributed algorithms. For instance, it is possible to use compact notation for modelling a server that sends a message to all other servers. In order to change the number of servers in the system, only one data type definition needs to be modified.

Basic algebraic operations check for exceptional conditions, such as overflows or division by zero. All operations detect constraint violations. The allowed values can be restricted arbitrarily even for structured types.

Availability

MARIA has been developed in the UNIX environment. Part of it works in any system for which a standard-compliant C++ compiler is available. The tool can be obtained from `http://www.tcs.hut.fi/maria/`.

Even if you are not familiar with our approach, please do not hesitate to contact us. We are constantly looking for interesting systems and models that can help us to improve our tool.