

SAT-based Abstraction Refinement for (Timed) Constraint Automata

Stephanie Kemper
 Centrum voor Wiskunde en Informatica (CWI)
 S.Kemper@cwi.nl

Failures within embedded systems of automotive industry, railway technology, and avionics usually have disastrous consequences. Dominant features of such safety-critical systems are that safety not only depends on the **values** of variables communicated between components, but also on (inter-component) actions and reactions that arrive **in time**.

For instance, a train controller needs to apply the brakes (reaction) as a response to driving faster (speed value) than the current track situation permits (action). Yet,

this response has to be executed in time before reaching an open gate, or it will be useless.

Reachability of certain error states thus has to be excluded prior to implementation, using symbolic model checking techniques [4]. The performance of applied model checking techniques thereby is restricted by the state explosion problem, caused by parallel composition of components and their communication devices (component connectors) to build the system.

The System Model

We use (timed) constraint automata [1, 2] (TCA) as the underlying formal model, to describe event-based behaviour and possible data flow (see figure 1). States stand for possible configurations (e.g., values of variables), transitions represent possible data flow and its effects on these configurations.

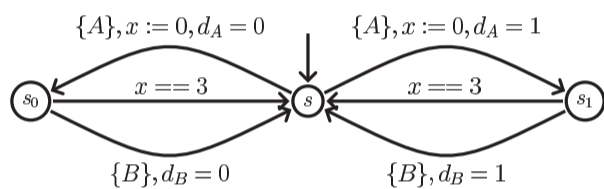


Figure 1: FIFO1 buffer with capacity 1 and timeout, able to store data items 0 or 1. Data item is lost after 3 time units, if not consumed before.

Our Approach

We have defined an encoding of TCA in propositional logic with linear arithmetic. Checking reachability of error state s then amounts to satisfiability (SAT) checking of a formula (rather than checking the automaton itself), using abstraction refinement techniques [3] to cope with the state explosion problem.

To reduce complexity, parts of the formula – representing automaton parts considered irrelevant to reachability of s – are removed (abstraction). If – due to over-approximation – a false counterexample is detected, we use Craig interpolants [5] to identify wrongly abstracted parameters in the refinement step.

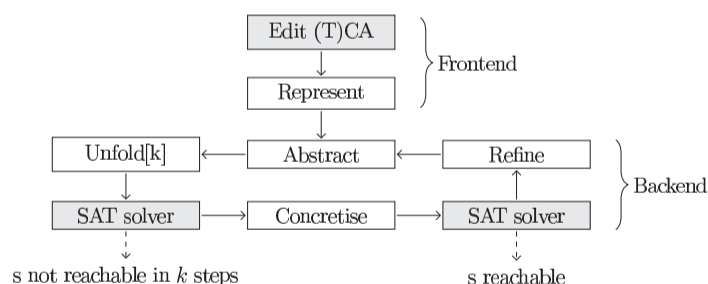


Figure 2: SAT-based Abstraction Refinement Framework

Research Challenges

There were three major challenges we had to deal with:

1. Define a representation of TCA in propositional logic.

To reduce formula complexity, the representation has to ensure that the representation of a system of automata is obtained by linear combination of individual representations, rather than by forming the exponential cross product.

Furthermore, it has to fulfil the following properties

- formula unsatisfiable $\Leftrightarrow s$ not reachable in k steps, and
- formula satisfiable $\Leftrightarrow s$ reachable.

2. Define an abstraction technique working on propositional formulas.

The abstraction technique has to ensure that abstraction performed on the automaton (e.g., removal of states or transitions) can be reflected on formula level

(and vice versa!), such that the abstract formula is a weak over-approximation of the original one.

3. Prove correctness of all these definitions.

Main Results 2004

We integrated all our results in a complete abstraction refinement framework (see figure 2). In particular, we achieved the following:

1. The formula representation of parallel composition is **linear** in the number of automata. In this, we overcome the state explosion problem and can check much larger systems, compared to exponential composition.
2. Our logic based abstraction and refinement steps work **uniformly** for different elements of TCA, represented by formulas. Furthermore, they are purely syntactical, and can thus be performed **fully automatic**.
3. Thanks to the purely syntactical character of abstraction and refinement, our framework can be used for **other automata models** (e.g. hybrid automata) as well, by only changing the frontend.
4. We proved all steps in the framework to be **correct**, such that all results achieved for the formula representation immediately hold for the underlying automaton (and vice versa).

Future Work

We recently began to work on a new automaton model, inspired by TCA, which combines characteristics of components and component connectors in a much more advanced way than TCA, and enables us to model systems accurately with only one system model. Furthermore, we want to adapt our framework (frontend) to these new “network automata” as well as to other models, to be able to model check systems specified with a number of different models.

Another direction of research is the following: our automatic abstraction and refinement steps contain heuristics – in fact, every fully automatic approach has rely on heuristics. We believe that these heuristics can be drastically improved, by identifying and examining the underlying algebraic and logical principles.

References

- [1] Farhad Arbab, Christel Baier, Frank S. de Boer, and Jan J. M. M. Rutten. Models and temporal logics for timed component connectors. In *SEFM*, pages 198–207. IEEE Computer Society, 2004.
- [2] Christel Baier, Marjan Sirjani, Farhad Arbab, and Jan J. M. M. Rutten. Modeling component connectors in Reo by constraint automata. *Science of Computer Programming*, 61(2):75–113, 2006.
- [3] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.
- [4] Kenneth L. McMillan. *Symbolic Model Checking*. PhD thesis, Norwell, MA, USA, 1993.
- [5] K.L. McMillan. An interpolating theorem prover. *Theor. Comput. Sci.*, 345(1):101–121, 2005.

Part of this research has been funded by the Dutch BSIK/BRICKS project AFM3.1