Verified Model Checking of Timed Automata

Simon Wimmer <wimmers@in.tum.de> Peter Lammich <lammich@in.tum.de>
Institut für Informatik, Technische Universität München

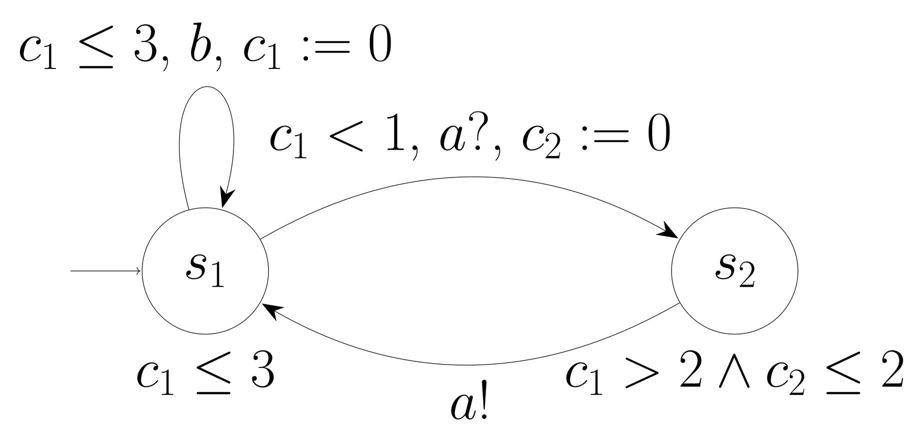
Objectives

- Model checking for the common class of diagonal-free Timed Automata
- Feature parity with UPPAAL (for model checking)
- Complete Verification with Isabelle/HOL
- Reasonable Performance

Timed Automata

Clocks

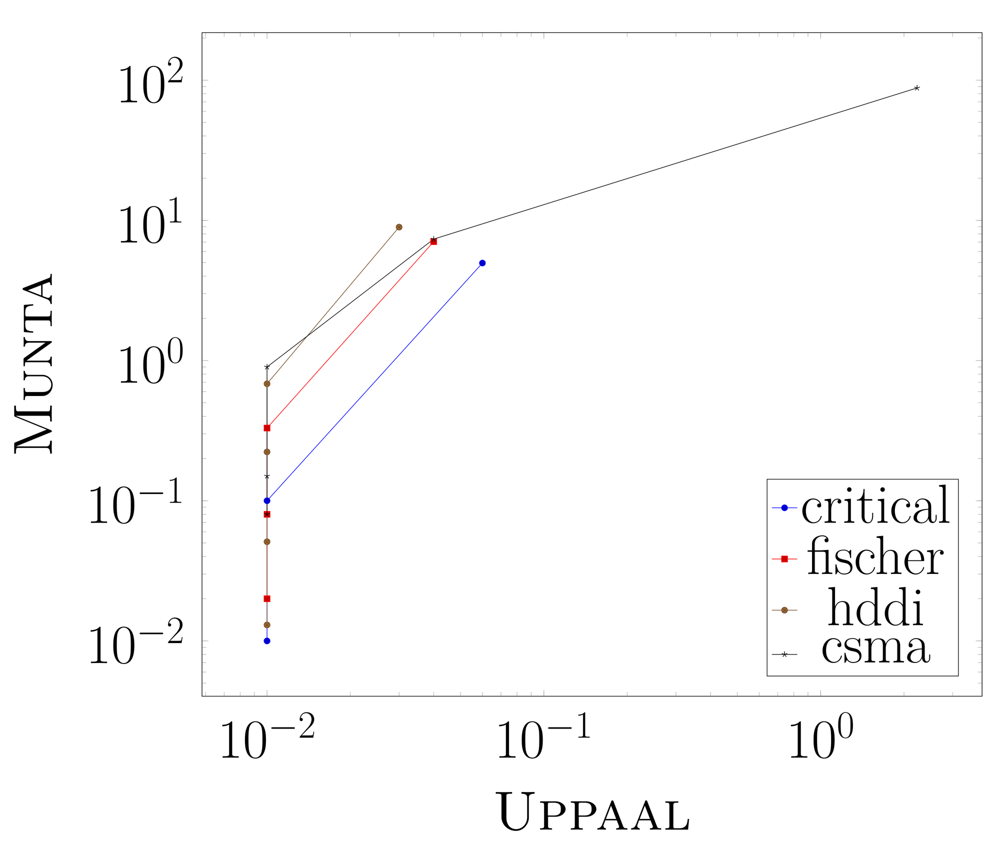
- Resets and guards on edges, invariants on nodes
- Real-valued semantics ⇒ *infinite* state space



Model Checking

- Concrete states $(l, u) \rightsquigarrow$ abstract states (l, Z) (for node l, clock valuation u, and Z a set of clock valuations)
- *Infinitely* many *zones* Z ⇒ Approximations!
- Soundness of approximations is peculiar[1]

Experiments



From Theory to Model Checking

- Starting point: abstract formalization of reachability checking for Timed Automata[4]
- Real Model Checking is more:

Modeling

- Modeling language: UPPAAL *bytecode!*
- Networks of Automata with discrete integer state variables (global)

Algorithms

- Worklist Algorithm for reachability: *subsumption*
- Operations on Difference Bound Matrices (DBMs): represent zones
- Floyd-Warshall algorithm

Program Analysis

Not all UPPAAL bytecode represents a valid automaton ⇒ we apply simple means of *program analysis* to accept a subset of valid inputs

Abstraction and Simulation

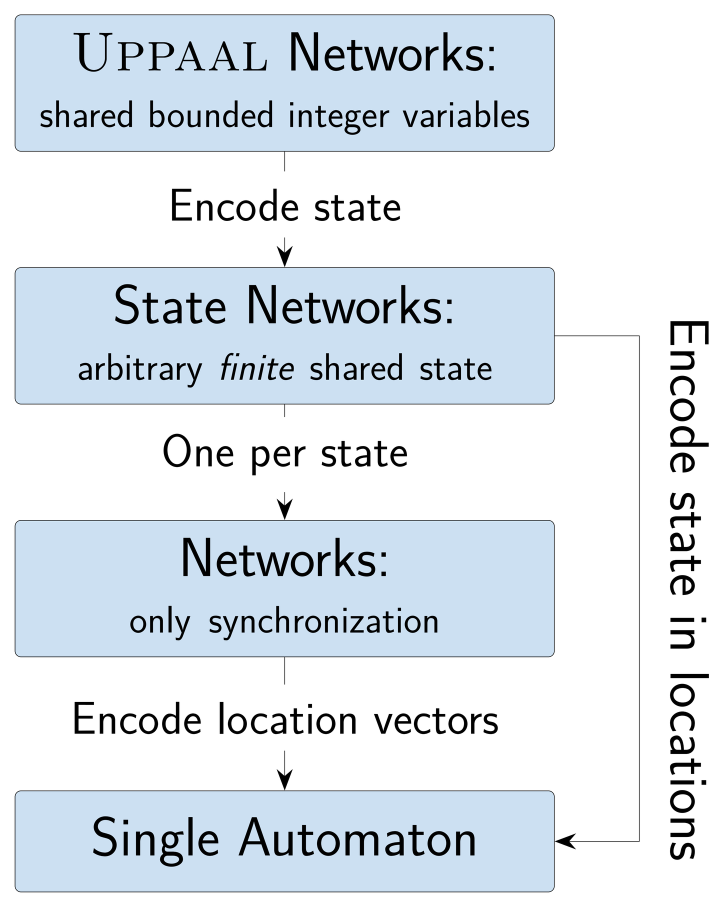
- Generalized framework for transition systems and their abstractions
- Simulation and subsumption graphs[2]
- Relations between infinite runs in the concrete system and cycles in the abstract system

Refinement, Refinement, Refinement

Much of this work is really an exercise in refinement:

- Abstract Operations on DBMs ⇒ functional impl. on maps ⇒ imperative impl. on arrays
- Semantics on reals ⇒ concrete models with integer constraints
- Complex networks with bytecode semantics ⇒ Single product automaton (*On the fly!*)

Product Construction



Isabelle Infrastructure

Different parts of recent Isabelle/HOL infrastructure are crucial for this work:

- Codatatypes and Coinduction: liveness
- Eisbach: product construction
- Transfer: reals ↔ integers
- The Imperative Refinement Framework[3]: imperative implementations

Older but important tools:

- Code Generation
- Locales: to build logical frameworks
- Sledgehammer: free proofs

Work in Progress

Temporal Logics

- LTL: Büchi emptiness
- (T)CTL à la UPPAAL: $\mathbf{A}\Diamond\varphi$, $\mathbf{A}\Box\,(\varphi \Rightarrow \mathbf{A}\Diamond\,\psi)$
- *Obstacle*: UPPAAL semantics & zenoness

Algorithms

- Simple Algorithm for $\mathbf{A}\Diamond$
- Combined with reachability, this gives $\mathbf{A}\Box\,(\varphi \Rightarrow \mathbf{A}\Diamond\,\psi)$

Modeling Language

- Better program analysis on the input: accept larger subclasses of valid bytecode

On-the-fly construction

Simple Trick: A single automaton is represented as an invariant assignment and a transition function. After performing the product construction, we give equivalent functional implementations, thus obtaining an on-the-fly construction.

Future

- Certification: reachability and Büchi emptiness
- Modeling: Broadcast channels, urgent and committed locations, ...
- Closing the Loop: Verified model transformation & parsing

[1] P. Bouyer. Untameable Timed Automata! In *STACS 2003*, volume 2607 of *LNCS*. Springer, 2003. doi:10.1007/3-540-36494-3_54.
[2] F. Herbreteau, B. Srivathsan, T.-T. Tran, and I. Walukiewicz. Why liveness for timed automata is hard, and what we can do about it. In *FSTTCS 2016*, volume 65 of *LIPIcs*. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.48.
[3] P. Lammich. Refinement to Imperative/HOL. In C. Urban and X. Zhang, editors, *ITP 2015*, volume 9236 of *LNCS*. Springer, 2015. doi:10.1007/978-3-319-22102-1_17.
[4] S. Wimmer. Formalized Timed Automata. In *ITP 2016*, volume 9807 of *LNCS*. Springer, 2016. doi:10.1007/978-3-319-43144-4_26.

github.com/wimmers/munta