THETA: a Framework for Abstraction Refinement-Based Model Checking

Tamás Tóth\textsuperscript{1}, Ákos Hajdu\textsuperscript{1,2}, András Vörös\textsuperscript{1,2}, Zoltán Micskei\textsuperscript{1}, István Majzik\textsuperscript{1}

\textsuperscript{1}Budapest University of Technology and Economics
Department of Measurement and Information Systems

\textsuperscript{2}MTA-BME Lendület Cyber-Physical Systems Research Group

FMCAD 2017, Vienna, Austria, 05.10.2017.
Motivation: a framework for
- Abstraction refinement-based algorithms
- Easy development, evaluation and combination
- Supporting various formalisms
- Applicable where systems have different aspects (e.g. CPS)

Our solution: Theta
- Open source: github.com/FTSRG/theta
Theta – Characteristics

Generic
Various kinds of formal models

Configurable
Different algorithms and strategies

Modular
Reusable and combinable modules
Generic – Formalisms

- **Symbolic transition systems**
  - Low level formalism
  - Based on SMT formulas

- **Control flow automata**
  - Programs as graphs
  - Edges annotated with statements

- **Timed automata**
  - Clock variables
  - Operations over clocks

- **Support for new formalisms**
  - Reusable components, e.g. expressions

\[
I := x = 0 \land y = 0 \\
T := x' = y + 1 \land y' = 2 \cdot y
\]
Generic – Language frontends

- Symbolic transition systems [FORTE’16]
  - AIGER format
  - Intermediate language for PLCs

- Control flow automata [VPT’17]
  - Subset of C
  - Size reduction techniques

- Timed automata [FORMATS’17]
  - UPPAAL XTA
THETA: a Framework for Abstraction Refinement-Based Model Checking

Modular – Architecture

Formalisms and language front-ends
- AIGER
- PLC
- Transition systems
- C programs
- Control flow automata
- UPPAAL XTA
- Timed automata

Verification back-end
- Abstract domain
- Interpreter
  - Init func.
  - Transfer func.
  - Action func.
- Abstraction refinement loop
  - Abstractor
  - ART
  - Refiner

SMT solver interface
Modular – Extensibility

- New algorithms

Formalisms and language front-ends

- AIGER
- PLC
- Transition systems
- C programs
- Control flow automata
- UPPAAL XTA
- Timed automata

Verification back-end

- Abstract domain
- Interpreter
  - Init func.
  - Transfer func.
  - Action func.

Abstraction refinement loop

- Abstractor
- ART
- Refiner

SMT solver interface
Modular – Extensibility

- New formalisms

**Formalisms and language front-ends**

- AIGER
- PLC
- Transition systems
- C programs
- Control flow automata
- UPPAAL XTA
- Timed automata

**Verification back-end**

- Abstract domain
  - Interpreter
    - Init func.
    - Transfer func.
    - Action func.

- Abstraction refinement loop
  - Abstractor
  - ART
  - Refiner

- SMT solver interface
Configurable – Parameters

Abstract domain
- Predicate
- Explicit value
- Zone
- Location
- Composition

Refinement strategy
- Binary interp. forw.
- Binary interp. backw.
- Sequence interp.
- Unsat core

Search strategy
- BFS
- DFS
- Dist. to error
- Random

Initial precision
- Empty
- Property-based

Precision granularity
- Global
- Local

Predicate split
- Atoms
- Conjuncts
- Whole

78 configs for control flow automata
52 configs for transition systems
15 configs for timed automata
Configurable – Use Cases

- Developing and evaluating **new algorithms**
  - Extending predicate abstraction with explicit values [FORTE’16]
  - Lazy reachability checking of timed automata [FORMATS’17]

- **Diverse results** support configurability

Comparison of execution time in case of different analysis configurations on various models
Conclusions

- **Theta**: Model checking framework
  - Generic, modular, configurable
  - Various formalisms and frontends
  - Abstraction refinement algorithms

- **Current and future work**
  - Extend the C frontend (LLVM)
  - Experiment with novel algorithms
  - Increase input models in experiments
  - Automatic configuration selection

→ github.com/FTSRG/theta
References


