Control Explicit—Data Symbolic Model Checking

Petr Bauch

supervised by: Jiří Barnat

November 22, 2012

ParaDiSe
Parallel & Distributed
Systems Laboratory

Introduction

to Execution-Based Verification

Formal Verification

given a program, decide, using rigorous mathematical methods, if the program is correct (against behaviour specification)

- undecidable in theory (of computability)

$$\varphi : \mathbb{N} \to \mathbb{N}, \psi : \mathbb{N} \to \mathbb{N}; \varphi \stackrel{?}{=} \psi$$

- difficult in practice

power of specification

formal $\longrightarrow$ informal

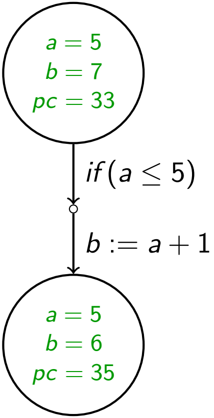
small $\longleftarrow$ large

size of program

Transition System

verifying that a program satisfies a formula by checking that its transition system is a model of the formula

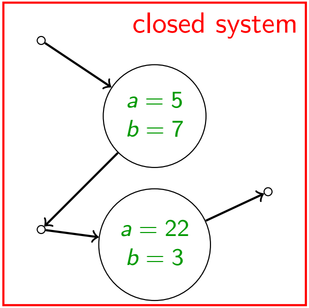
- execution generates a transition system
- states are: evaluation of variables, program counter(s), heap, stack content
- transitions are: execution of commands
- branching is caused by: conditions, cycles, thread interleaving

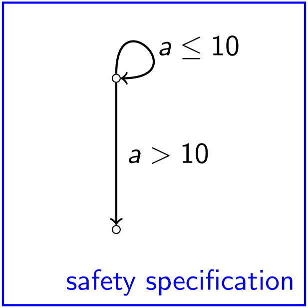


Execution-Based Verification

execute the system while concurrently checking the specification

- Testing (arbitrary program)

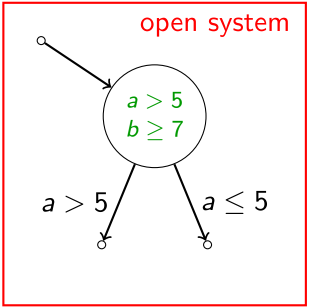


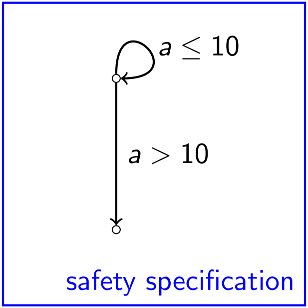


Execution-Based Verification

execute the system while concurrently checking the specification

- Symbolic Execution (no multiplication or cycles)



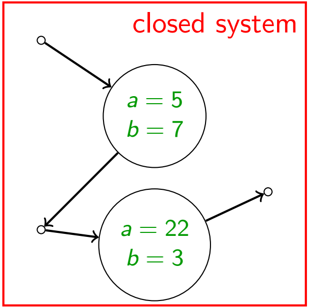


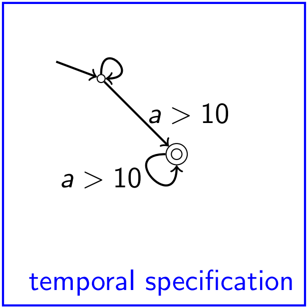
J. King. Symbolic Execution and Program Testing. *Commun. ACM*, 19(7):385–394, 1976.

Execution-Based Verification

execute the system while concurrently checking the specification

- Model Checking (small, parallel programs; communication protocols)





M. Vardi and P. Wolper. An Automata-Theoretic Approach to Automatic Program Verification. In *Proc. of LICS*, pages 332–344, 1986.

Long-Term Objectives

overcome the limitations of individual methods

- testing: more user friendly
  - automatic (Godefroid, et. al., 2008)
  - parallelism (Simsa, et. al., 2010)
- symbolic execution: more robust
  - temporal properties (Braione, et. al., 2008)
  - parallelism (Păsăreanu, et. al., 2003)
  - cycles (Trtík, et. al., 2012)
- model checking: smaller state space
  - symbolic representation (McMillan, et. al., 1992)
  - bounded approach (Biere, et. al., 1999)
  - parallel approach (Barnat, et. al., 2001)

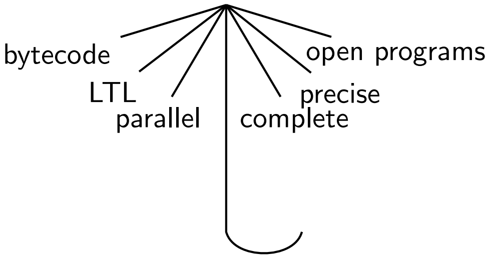
Explicit vs Symbolic Model Checking

- explicit:
  states forming the transition system are stored explicitly
  better for asynchronous, control-flow nondeterministic systems
  allow parallel processing
- symbolic:
  transition system is encoded into a logical formula
  better for synchronous, data-flow nondeterministic systems
  parallelisation is difficult

C. Eisner and D. Peled. Comparing Symbolic and Explicit Model Checking of a Software System. In *SPIN* volume 2318 of *LNCS*, pages 230–239. Springer, 2002.

State of the Art

towards complete and precise verification of parallel software
against temporal specification



umbrellaed verification

Symbolic Execution

- states (path conditions) represented as linear constraints
  use state-of-the-art libraries [1], e.g. Omega library [2]
  – fast, unbounded variable evaluation, Presburger arithmetic

1. A. Coen-Porisini and G. Denaro and C. Ghezzi and M. Pezzé. Using Symbolic Execution for Verifying Safety-Critical Systems. In *Proc. of ESEC/FSE*, pages 142–151, 2001.

2. W. Pugh. A Practical Algorithm for Exact Array Dependence Analysis. *Communications of the ACM*, 35(8):102–114, 1992.

Symbolic Execution

- handle nondeterminism, parallelism by generating execution interleavings
- no state equivalence → no accepting cycle detection → no temporal specification

S. Khurshid, C. Păsăreanu, and W. Visser. Generalized Symbolic Execution for Model Checking and Testing. In *TACAS*, volume 2619 of *LNCS*, pages 553–568. Springer, 2003.

Symbolic Execution

- handle loops: unwind [1], invariants [2], path counters [3]

1. J. King. Symbolic Execution and Program Testing. *Commun. ACM*, 19(7):385–394, 1976.

2. S. Khurshid, C. Păsăreanu, and W. Visser. Generalized Symbolic Execution for Model Checking and Testing. In *TACAS*, volume 2619 of *LNCS*, pages 553–568. Springer, 2003.

3. J. Strejček and M. Trtík. Abstracting Path Conditions. In *Proc. of ISSTA*, pages 155–165, 2012.

Symbolic Execution

- handle complex operations: state-of-the-art decision procedures [1], SMT solvers (bit-vectors, arrays), trigonometric functions [2]

1. S. Anand and C. Păsăreanu and W. Visser. JPF-SE: A Symbolic Execution Extension to Java PathFinder. In *TACAS*, volume 4424 of *LNCS*, pages 134–138. Springer, 2007.

2. M. Souza and M. Borges and M. d'Amorim and C. Păsăreanu. CORAL: Solving Complex Constraints for Symbolic PathFinder. In *NFM* volume 6617 of *LNCS*, pages 359–374. Springer, 2011.

Symbolic Execution

- handle bytecode as input language

C. Păsăreanu and N. Rungta. Symbolic PathFinder: Symbolic Execution of Java Bytecode. In *Proc. of ASE*, pages 170–180, 2010.

Explicit Model Checking

- model checking real code: C [1], Microcode [2], Simulink [3], LLVM [4]

1. M. Musuvathi and D. Park and A. Chou and D. Engler and D. Dill. CMC: A Pragmatic Approach to Model Checking Real Code. *OSR*, 36:75–88, 2002.

2. B. Schlich. *Model checking of Software for Microcontrollers*. PhD thesis, Aachen University, 2008.

3. J. Barnat, J. Beran, L. Brim, T. Kratochvíla, and P. Ročkai. Tool Chain to Support Automated Formal Verification of Avionics Simulink Designs. In *FMICS*, volume 7437 of *LNCS*, pages 78–92. Springer, 2012.

4. J. Barnat, L. Brim, and P. Ročkai. Towards LTL Model Checking of Unmodified Thread-Based C&C++ Programs. In *NFM*, volume 7226 of *LNCS*, pages 252–256. Springer, 2012.

Symbolic Model Checking

- standard representation (Binary Decision Digram) is exponential for integer multiplication [1] → model checking with Binary Moment Diagram [2] and Boolean Expression Diagram [3]

1. R. Bryant. On the Complexity of VLSI Implementations and Graph Representations of Boolean Functions with Application to Integer Multiplication. *IEEE Trans. Comput.*, 40(2):205–213, 1991.

2. R. Bryant and Y.-A. Chen. Verification of Arithmetic Circuits with Binary Moment Diagrams. In *Proc. of DAC*, pages 535–541, 1995.

3. P. Williams, A. Biere, E. Clarke, and A. Gupta. Combining Decision Diagrams and SAT Procedures for Efficient Symbolic Model Checking. In *CAV*, volume 1855 of *LNCS*, pages 124–138. Springer, 2000.

Symbolic Model Checking

- smaller state space using bounded approach, allows to use state-of-the-art satisfiability procedures, such as SAT [1] or SMT (satisfiability modulo theory) [2]

1. A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu. Symbolic Model Checking Using SAT Procedures instead of BDDs. In *Proc. of DAC*, pages 317–320, 1999.

2. A. Armando, J. Mantovani, and L. Platania. Bounded Model Checking of Software Using SMT Solvers Instead of SAT Solvers. In *SPIN*, volume 3925 of *LNCS*, pages 146–162. Springer, 2006.

Explicit/Symbolic Combination

handle combination of control and data -flow

- multiple explicit states in one symbolic [1,2]
- explicit property, symbolic system description [3,4]

1. A. Cimatti, M. Roveri, and P. Bertoli. Searching Powerset Automata by Combining Explicit-State and Symbolic Model Checking. In *TACAS*, volume 2031 of *LNCS*, pages 313–327. Springer, 2001.

2. A. Duret-Lutz, K. Klai, D. Poitrenaud, and Y. Thierry-Mieg. Self-Loop Aggregation Product — A New Hybrid Approach to On-the-Fly LTL Model Checking. In *ATVA*, volume 6996 of *LNCS*, pages 336–350. Springer, 2011.

3. A. Biere, E. Clarke, and Y. Zhu. Multiple State and Single State Tableaux for Combining Local and Global Model Checking. In *Correct System Design*, volume 1710 of *LNCS*, pages 163-179. Springer, 1999.

4. R. Sebastiani, S. Tonetta, and M. Vardi. Symbolic Systems, Explicit Properties: On Hybrid Approaches for LTL Symbolic Model Checking. In *CAV*, volume 3576 of *LNCS*, pages 100–246. Springer, 2005.

Research Goals

- combine explicit and symbolic approaches to model checking
- explicit control, symbolic data
- investigate symbolic representation for software model checking
- propose new representation for better performance

the goal is umbrellaed verification

Achieved Results

- extended DiVinE with support for symbolic data representation (multi-states)
- proposed requirements on the symbolic representation for use in explicit model checking
- case study experiments: DVE, Simulink
- experiments show potential towards umbrellaed verification