Boolector: An Efficient SMT Solver for Bit-Vectors and Arrays

Robert Brummayer and Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University Linz, Austria

TACAS 2009

York, UK

March 24th, 2009

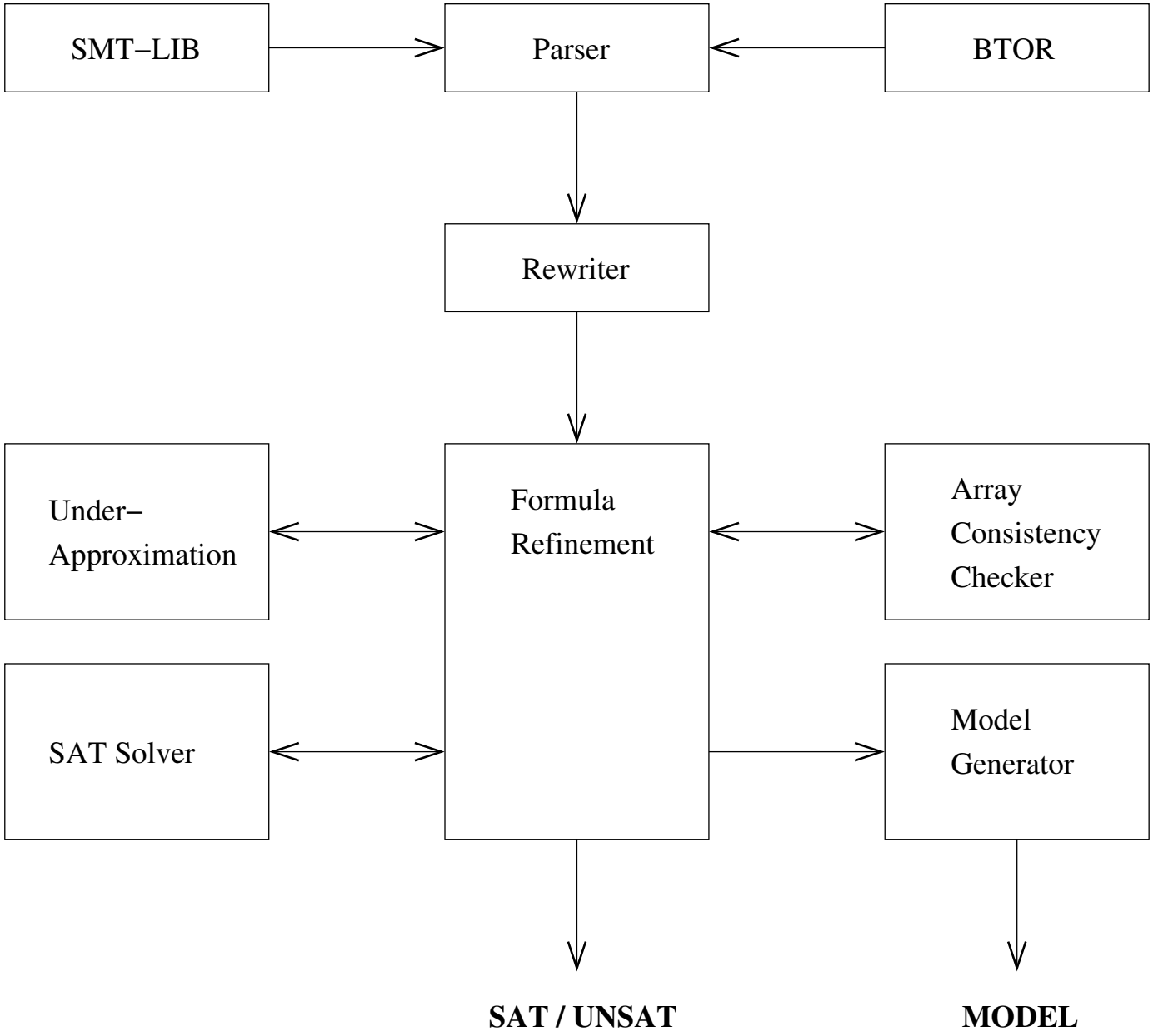
Satisfiability Modulo Theories (SMT)

R. Brummayer, FMV, JKU Linz

- Generalization of the Boolean Satisfiability Problem (SAT)
- Satisfiability with respect to background theories
- Software and Hardware verification
- SMT Solvers
  - Boolector, Z3, CVC3, STP, Barcelogic, MathSAT, Spear, OpenSMT, ...
- Theories
  - Decidable fragments of first-order logic (typically quantifier-free)
    - Theory of bit-vectors and extensional theory of arrays

Boolector: Schematic Overview

R. Brummayer, FMV, JKU Linz



Rewriting Engine

R. Brummayer, FMV, JKU Linz

- Goal is to simplify Formula as soon and as much as possible
- Basic level 1 rewrite rules, e.g. $a \wedge \neg a \Leftrightarrow \bot$
  - Locally applied during formula construction
- Level 2 rewrite rules perform global substitutions
  - Topological sorting allows substitutions in one pass
  - Static analysis techniques
- Level 3 rewrite rules perform arithmetic normalization
- Rules of quadratic worst-case complexity bounded in recursion depth

- Replace reads in ϕ by fresh abstraction variables to obtain $\bar{\phi}$
- Let SAT solver "guess" solution
  - If SAT solver cannot find a solution, ϕ is unsat
- Explicitly check if model is consistent with theory
- If check succeeds, ϕ is sat
- If check fails
  - Add lemma to refine formula
  - Let SAT solver "guess" a new solution

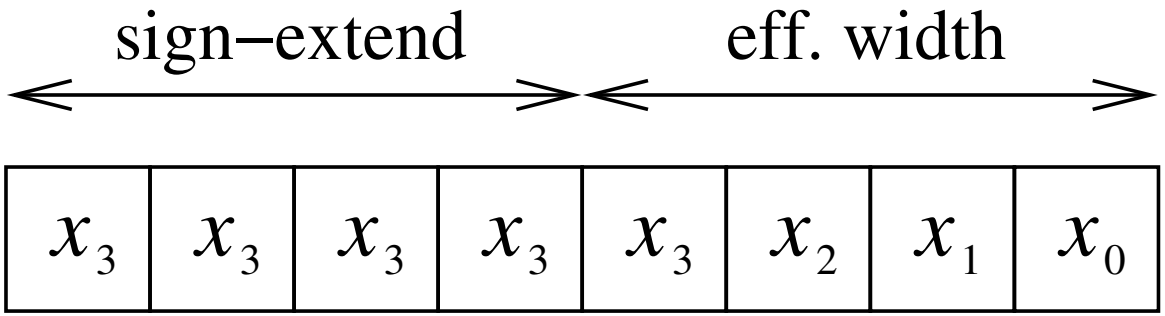
- Axiom of extensionality
  - $a = b \quad \Leftrightarrow \quad \forall i\,(read(a,i) = read(b,i))$
- Replace array equality $a = b$ by boolean abstraction variable $\alpha(a = b)$
- For each array equality $a = b$ add $a \neq b \;\Rightarrow\; \exists\lambda(read(a,\lambda) \neq read(b,\lambda))$
  - Guarantees that there exists a witness for inequality
- If underlying decision procedure assigns $\alpha(a = b)$ to $\top$
  - Explicitly check if model is consistent with extensional theory

- Propagation-based approach
- Annotate each array node with its set of reads ρ
- Propagate reads according to array axioms until fix-point is reached
- Handling extensionality
  - For each $\alpha(a = b)$ where $\sigma(\alpha(a = b)) = \top$
    - add every read $\in \rho(a)$ to $\rho(b)$ and vice versa.
- Finally, check read congruence on all arrays

Supported Under-Approximation Techniques (1/4)

R. Brummayer, FMV, JKU Linz

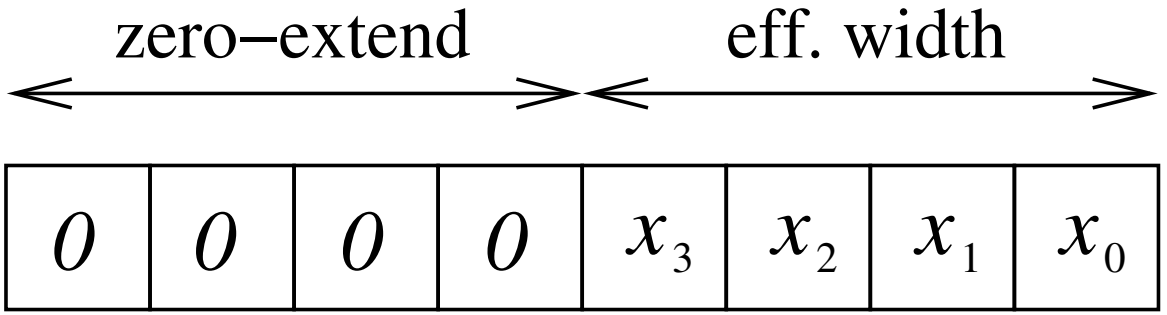
- Sign-Extension of BV variables
  - Let the n least significant bits be variable
  - We call n the effective bit-width
  - Sign-extend the n^{th} bit
  - Appropriate in software verification with two's complement semantics



Supported Under-Approximation Techniques (2/4)

R. Brummayer, FMV, JKU Linz

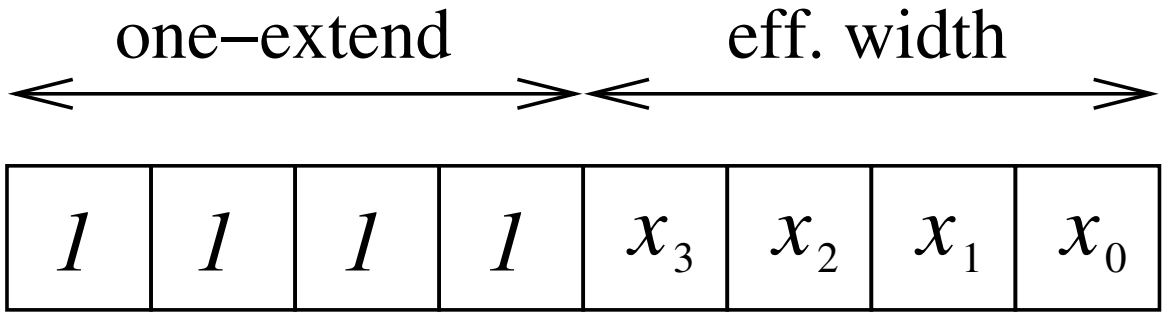
- Zero-Extension (Unsigned-Extension) of BV variables
  - Let the n least significant bits be variable
  - Set remaining bits to zero
  - Appropriate in verification with unsigned semantics
    - May speed up decision procedure more than sign-extension



Supported Under-Approximation Techniques (3/4)

R. Brummayer, FMV, JKU Linz

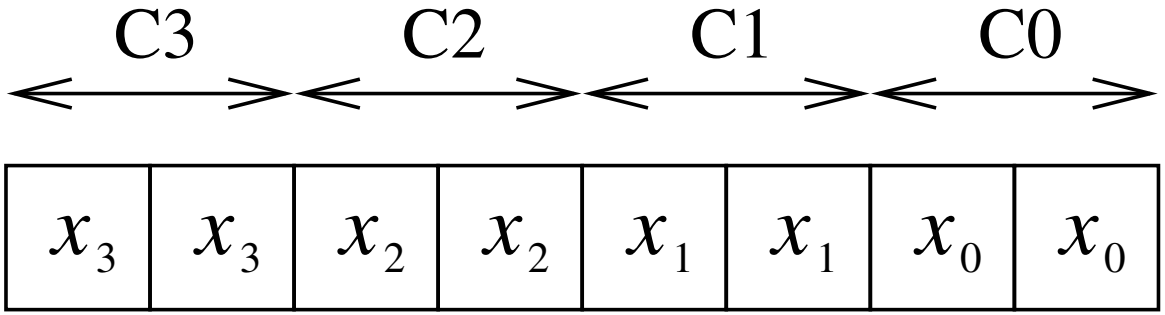
- One-Extension of BV variables
  - Let the n least significant bits be variable
  - Set remaining bits to one
  - Beneficial in context of negative numbers
    - May speed up decision procedure more than sign-extension



Supported Under-Approximation Techniques (4/4)

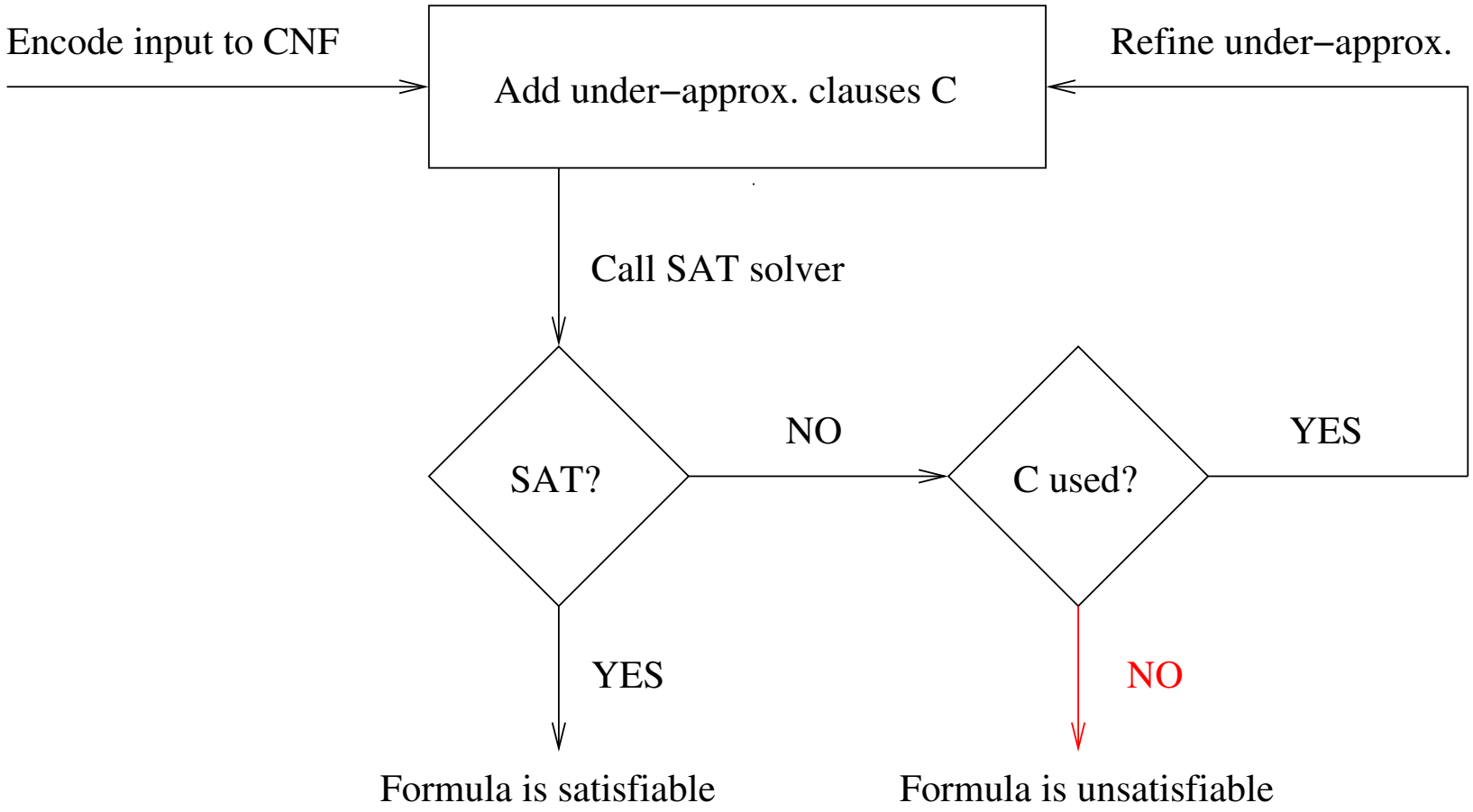
R. Brummayer, FMV, JKU Linz

- Equivalence class splitting of BV variables
  - Split bit-vector into n equivalence classes
  - Refinement variants
    - Either refine overall class splitting
    - Or find out which classes to refine



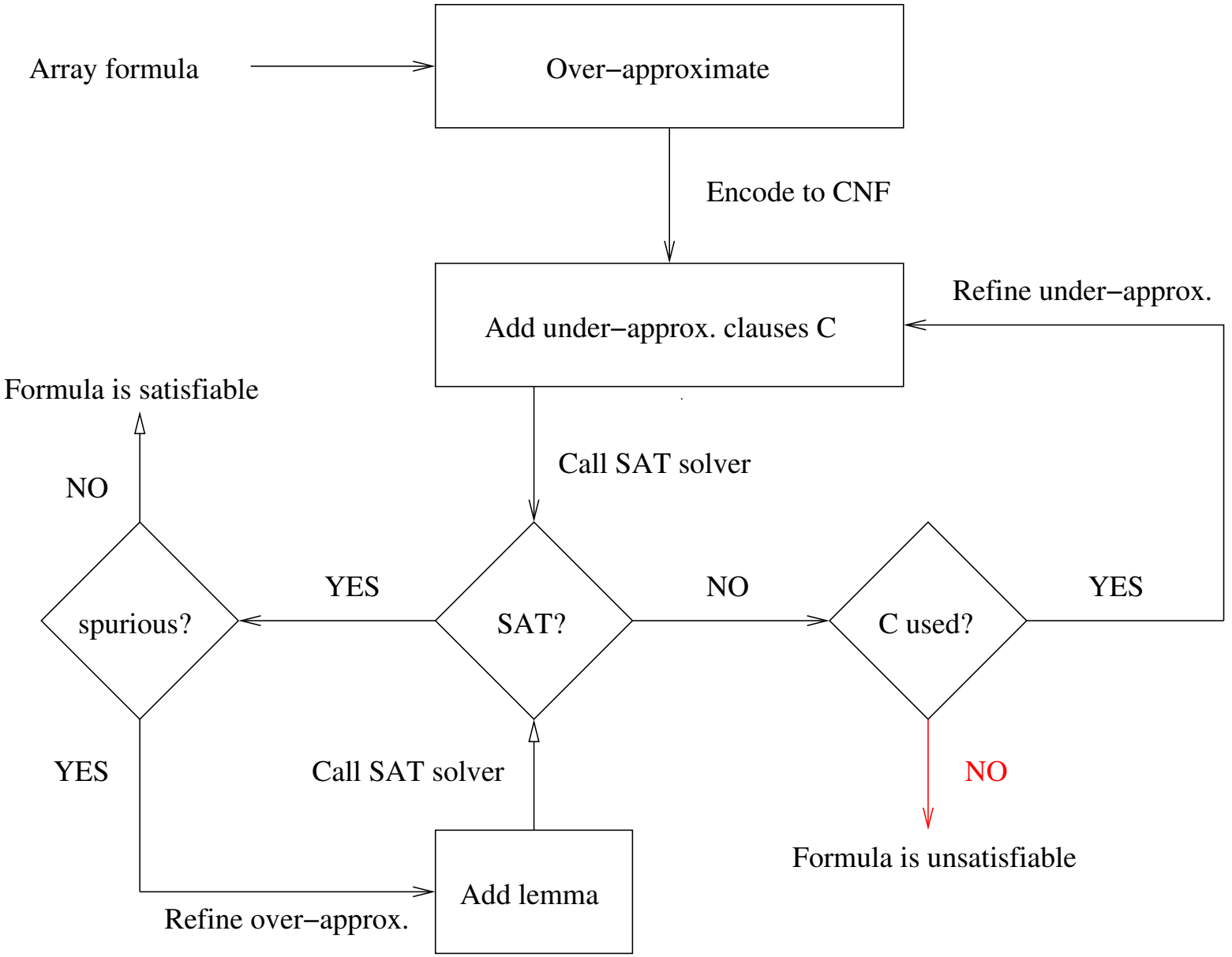
Early Unsat Termination [BrummayerBiere09]

R. Brummayer, FMV, JKU Linz



Combining Over- and Under-Approximation [BrummayerBiere09]

R. Brummayer, FMV, JKU Linz



Additional Features

R. Brummayer, FMV, JKU Linz

- Word-level model checking [BrummayerBiereLonsing08]
  - BTOR Format provides ”next“ operator for registers and memories
  - Boolector can be used as bounded model checker
    - Searching witnesses, k-induction with and without simple paths
- Rich C API
- Model Generation
  - Bit-vector and array variables
  - Consistent assignments for terms and formulas in ϕ
- Pretty Printer: Conversion between SMT-LIB 1.2 and BTOR format

Conclusion

R. Brummayer, FMV, JKU Linz

- Rewriting Engine
- Over-approximation techniques for arrays
- Under-approximation techniques for bit-vector variables and reads
- Combining over-approximation and under-approximation
- Additional features
- Boolector 1.0 available for research
  - http://fmv.jku.at/boolector