Embedded Software Laboratory

RWTH AACHEN UNIVERSITY

Existential Quantification as Incremental SAT

Jörg Brauer (RWTH Aachen University)

Andy King (Portcullis Computer Security)

Jael Kriener (University of Kent)

20.07.2011 @ CAV

Motivation

- Why SAT?
  - *"CNF and SAT-based quantifier elimination can be exponentially more efficient than [..] BDDs in cases where the resulting fixed points have compact representations in CNF, but not as BDDs."* [McM02]
- SAT-based algorithms are inelegant [Bry08]
  - Require re-engineering of SAT Solver [McM02]
  - Or combination with BDDs [LBC03]

[Bry08] R.E. Bryant, A View From the Engine Room: Computational Support for Symbolic Model Checking
[LBC03] S.K. Lahiri, R.E. Bryant, B. Cook, A Symbolic Approach to Predicate Abstraction, CAV'03
[McM02] K. McMillan, Applying SAT Methods in Unbounded Symbolic Model Checking, CAV'02

Quantifier Elimination in Predicate Abstraction

- State variables: $X = \{x_1, \ldots, x_6\}$
  $Y = \{y_1, \ldots, y_6\}$
- Input state:

$$\xi = (x_1 \wedge \neg x_2 \wedge x_3 \wedge \neg x_4 \wedge x_5 \wedge \neg x_6) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3 \wedge x_4 \wedge \neg x_5 \wedge x_6)$$

- Transition relation:

$$\begin{aligned}\mu = \quad & \neg(x_2 \wedge y_2) \quad \wedge \quad \neg(y_2 \wedge y_1) \quad \wedge \quad ((x_4 \wedge x_6) \Rightarrow y_1) \quad \wedge \\ & (x_3 \Leftrightarrow y_4) \quad \wedge \quad (x_4 \Leftrightarrow y_3) \quad \wedge \quad (x_5 \Leftrightarrow y_6) \quad \wedge \quad (x_6 \Leftrightarrow y_5)\end{aligned}$$

- Compute $\exists X : \mu \wedge \xi$

Quantifier Elimination by Example

- Consider

$$\varphi = (\neg x \vee z) \wedge (y \vee z) \wedge (\neg x \vee \neg w \vee \neg z) \wedge (w \vee \neg z)$$

- Goal: eliminate z from φ such that $\exists z : \varphi$ in CNF
- Introduce fresh variables and replace positive and negative literals

$$\tau(\varphi) = \left\{ \begin{array}{l} (x^- \vee z) \wedge (y^+ \vee z) \wedge (x^- \vee w^- \vee \neg z) \wedge (w^+ \vee \neg z) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{array} \right.$$

Quantifier Elimination by Example

- Passing $\tau(\varphi)$ to SAT solver gives a model

$$\mathcal{M} = \left\{ \begin{array}{llllllllll} w^+ & \mapsto & 1, & w^- & \mapsto & 0, & x^+ & \mapsto & 0, & x^- & \mapsto & 1, \\ y^+ & \mapsto & 0, & y^- & \mapsto & 0, & z & \mapsto & 1 & & & \end{array} \right\}$$

- $\mathcal{M}$ defines implicant $(w \wedge \neg x)$
  - This means $(w \wedge \neg x) \models \exists z : \varphi$
  - Then add blocking clause

Quantifier Elimination by Example

- Passing to SAT solver again gives new model

$$\mathcal{M}' = \left\{ \begin{array}{llllllllll} w^+ & \mapsto & 0, & w^- & \mapsto & 0, & x^+ & \mapsto & 0, & x^- \mapsto 1, \\ y^+ & \mapsto & 1, & y^- & \mapsto & 0, & z & \mapsto & 0 & \end{array} \right\}$$

- Defines new implicant $(\neg x \wedge y)$
- Then $(w \wedge \neg x) \vee (\neg x \wedge y) \models \exists z : \varphi$
- Unsatisfiable in next iteration, i.e.

  $(w \wedge \neg x) \vee (\neg x \wedge y) \equiv \exists z : \varphi$

Intermission: Sorting Networks

- Cardinality constraint $i_1 + i_2 + i_3 = 2$ encoded as $o_1 \wedge o_2 \wedge \neg o_3$ in unary encoding
- Guarantees generation of shortest implicants

D. Knuth: The Art of Computer Programming, Vol. 3
N. Een and N. Sörensson. Translating Pseudo-Boolean Constraints into SAT, JSAT'06

Quantifier Elimination by Example

- So far: ψ equisatisfiable to $\exists z : \varphi$ in DNF
- Then $\neg\psi \equiv \neg\exists z : \varphi$ is in CNF
- Observe:

$$
\begin{array}{lll}
 & \bigvee_{i=1}^{n} \mathsf{imp}_i & \models \quad \neg\psi \\
\Leftrightarrow & \psi & \models \quad \neg \bigvee_{i=1}^{n} \mathsf{imp}_i \\
\Leftrightarrow & \psi & \models \quad \bigwedge_{i=1}^{n} \neg\mathsf{imp}_i \\
\Leftrightarrow & \exists z : \varphi & \models \quad \underbrace{\bigwedge_{i=1}^{n} \neg\mathsf{imp}_i}_{\text{CNF}}
\end{array}
$$

Quantifier Elimination by Example

- Key idea: Simply re-apply technique to

$$\begin{aligned} \neg\psi &= \neg((w \wedge \neg x) \vee (\neg x \wedge y)) \\ &= \neg(w \wedge \neg x) \wedge \neg(\neg x \wedge y) \\ &= (\neg w \vee x) \wedge (x \vee \neg y) \end{aligned}$$

- Gives

$$\tau(\neg\psi) = \left\{ \begin{array}{l} (w^- \vee x^+) \wedge (x^+ \vee y^-) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{array} \right.$$

Quantifier Elimination by Example

$$\tau(\neg\psi) \quad = \quad \left\{ \begin{array}{l} (w^- \vee x^+) \wedge (x^+ \vee y^-) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{array} \right.$$

- Gives $\begin{array}{lcl} (x) & \models & \neg\psi \\ (\neg w \wedge \neg y) & \models & \neg\psi \end{array}$
- Thus $(x) \vee (\neg w \wedge \neg y) \quad \equiv \quad \neg\psi$
- Then

$$\begin{array}{rcl} \exists z : \varphi & \equiv & \psi \\ & \equiv & \neg((x) \vee (\neg w \wedge \neg y)) \\ & \equiv & (\neg x) \wedge (w \vee y) \end{array}$$

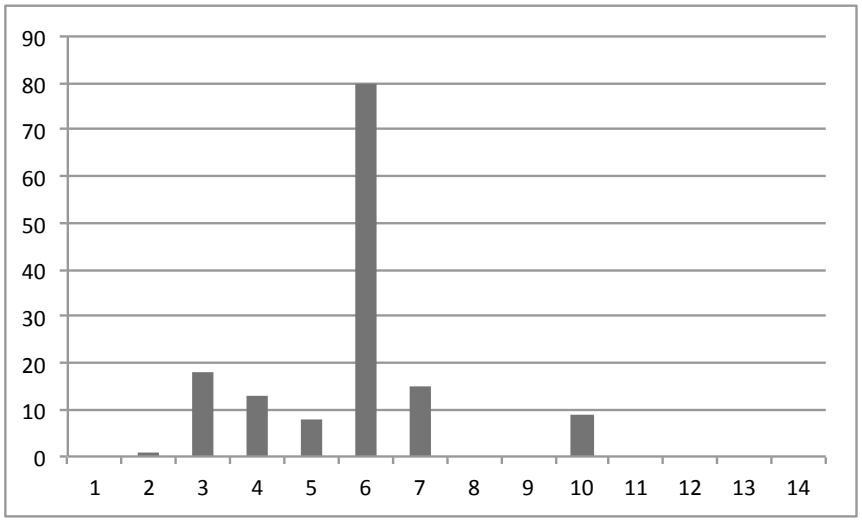
Experimental Setup

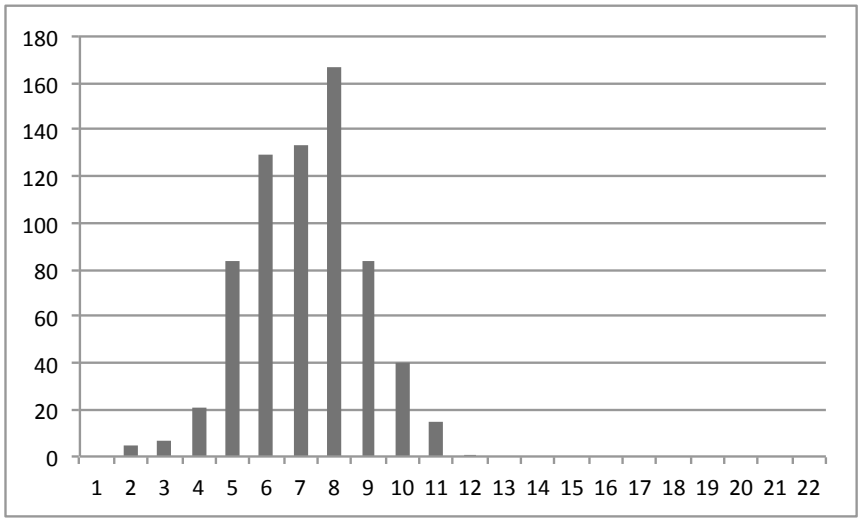
- Implementation using MiniSAT
  - Approx. 100 lines of code
- Different benchmark sets (40 cases)
  - Hardware circuits from ISCAS-89 and 74x series
  - ATmega16 transfer functions for synthesis
  - Value-set abstractions for PLCs
  - Ranging from 66 vars in 119 clauses to 18658 variables in 61696 clauses
- Comparison to hybrid SAT/BDD approach using Cudd library

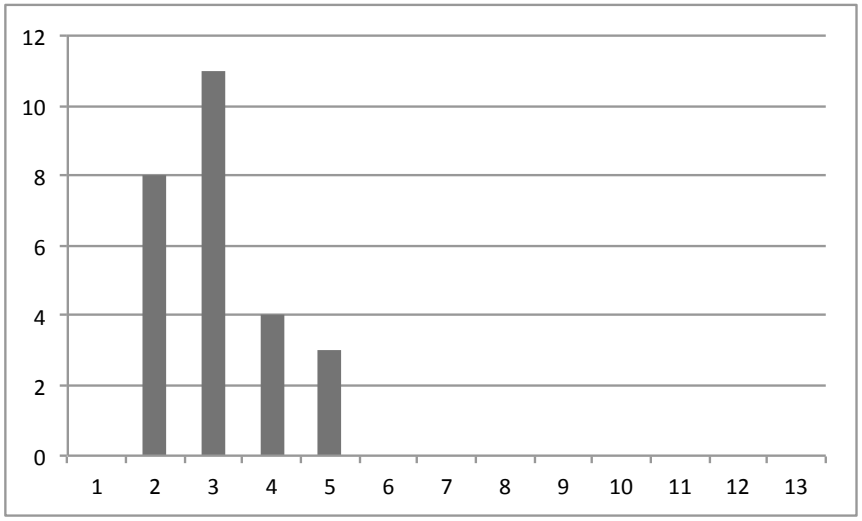
Experimental Results

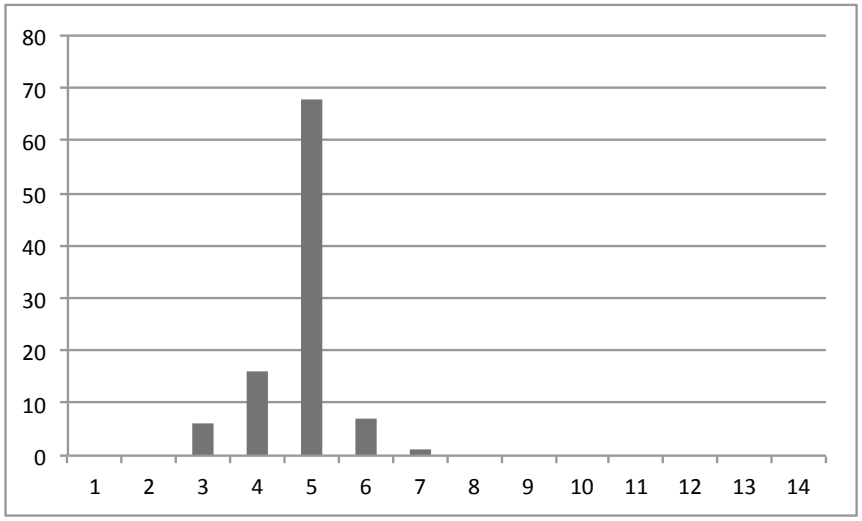
- Runtime
  - Primes win on 32 benchmarks, BDDs on 8
  - Ranging from 0.001s to 7.096s
  - Building BDDs does not come for free
  - But CNF enumeration using Cudd is faster
- Size
  - Primes win on 15 benchmarks, BDDs on 5
  - Identical on 20 benchmarks
- Lessons learnt:
  - BDDs depend a lot on dynamic variable reordering heuristics
  - SAT depends a lot on the chosen encoding (orders of magnitudes)

Distribution of Prime Implicants









So as to not cause offense

- McMillan (CAV‘02)
- Lahiri et al. (CAV‘03 & CAV‘06)
- Monniaux (CAV’10)
- Cavada et al. (FMCAD’07)
- Kettle et al. (TACAS‘06)
- Brauer and King (NFM’11)

Conclusion

- Two-staged algorithm
  - Model enumeration in DNF using prime implicants
    - Sometimes much faster (0.018s vs. 12.811s)
    - Not much slower in worst case
  - Prime implicant generation on negated formula
    - Converges onto quantifier-free formula from above
    - Second phase is thus anytime
  - Some more intrinsics in the paper
- Competitive to hybrid SAT/BDD based approaches
- Systematic rather than heuristic approach
- Easy to implement (~100 LoC on top of MiniSAT)
  - Cleaned up (larger) source code is available from http://www.cs.kent.ac.uk/people/staff/amk/