

On Projection using SAT/SMT



Jörg Brauer (RWTH Aachen University)
Andy King (Portcullis Computer Security)
Jael Kriener (University of Kent)

05.03.2012 @ MPI, Saarbrücken

Myself

- Diploma in CS from University of Kiel (2008)
 - Static analysis @ NICTA (2006-2008)
- Since 10/2008: PhD student @ RWTH
 - Leading [mc]square project
- Research interests
 - Automatic abstraction
 - Analysis of binaries
 - Decision procedures (to some extent)

Two Topics

1. Existential quantifier elimination as incremental SAT solving
2. Approximate projection for (non-) linear constraints

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: $X = \{x_1, \dots, x_6\}$
 $Y = \{y_1, \dots, y_6\}$

- 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:

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

- 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

$$\tau(\varphi) = \begin{cases} (x^- \vee z) \wedge (y^+ \vee z) \wedge (x^- \vee w^- \vee \neg z) \wedge (w^+ \vee \neg z) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{cases}$$

Quantifier Elimination by Example

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

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

- \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

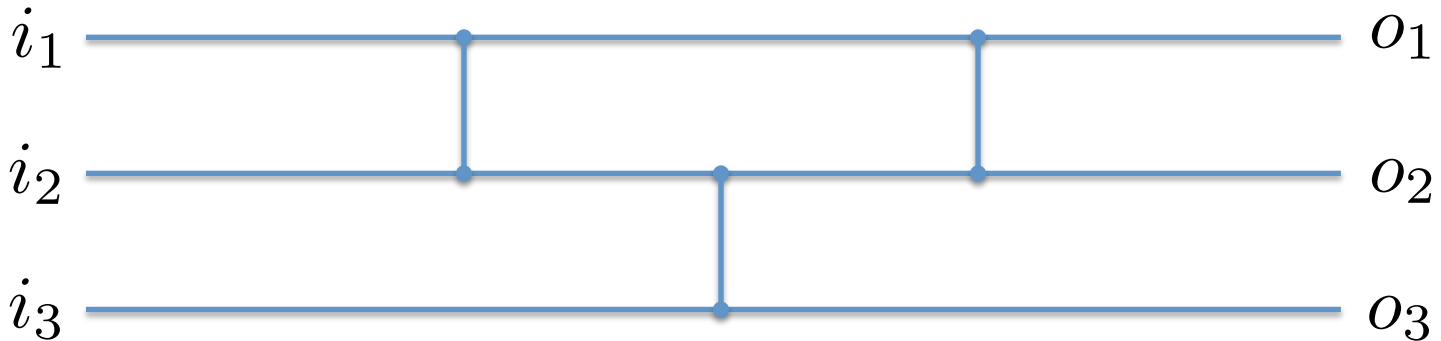
- Passing to SAT solver again gives new model

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

- 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:

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

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

$$\tau(\neg\psi) = \begin{cases} (w^- \vee x^+) \wedge (x^+ \vee y^-) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{cases}$$

Quantifier Elimination by Example

$$\tau(\neg\psi) = \begin{cases} (w^- \vee x^+) \wedge (x^+ \vee y^-) \wedge \\ (\neg w^+ \vee \neg w^-) \wedge (\neg x^+ \vee \neg x^-) \wedge (\neg y^+ \vee \neg y^-) \end{cases}$$

- Gives $(x) \models \neg\psi$
 $(\neg w \wedge \neg y) \models \neg\psi$

- Thus $(x) \vee (\neg w \wedge \neg y) \equiv \neg\psi$

- Then

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

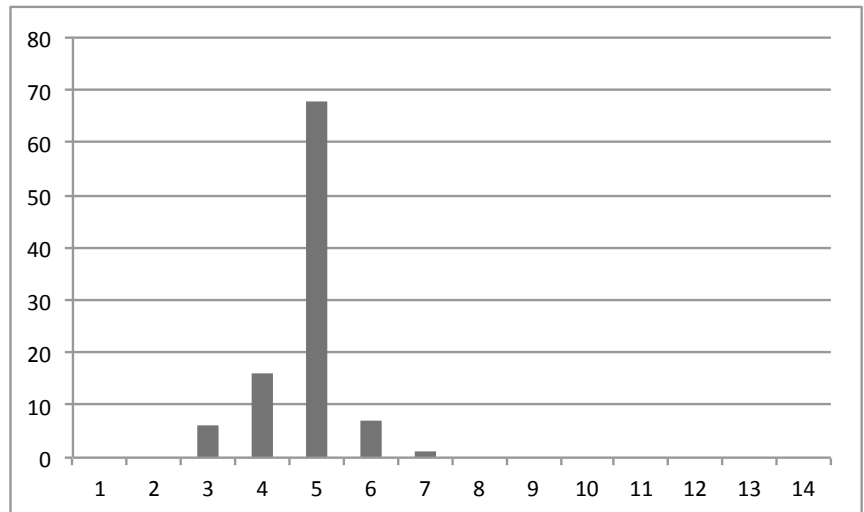
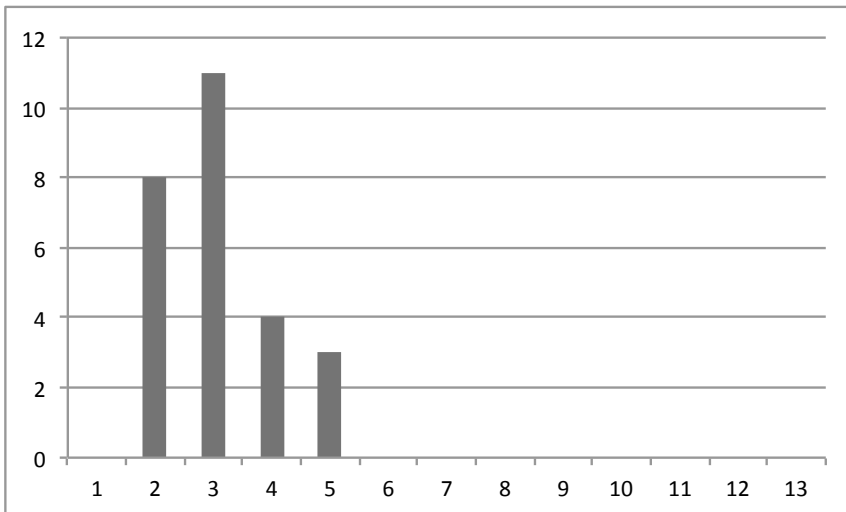
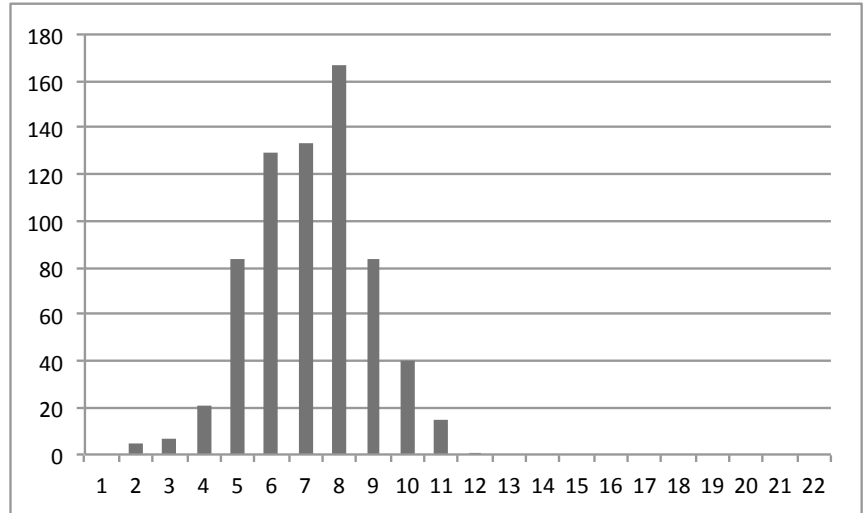
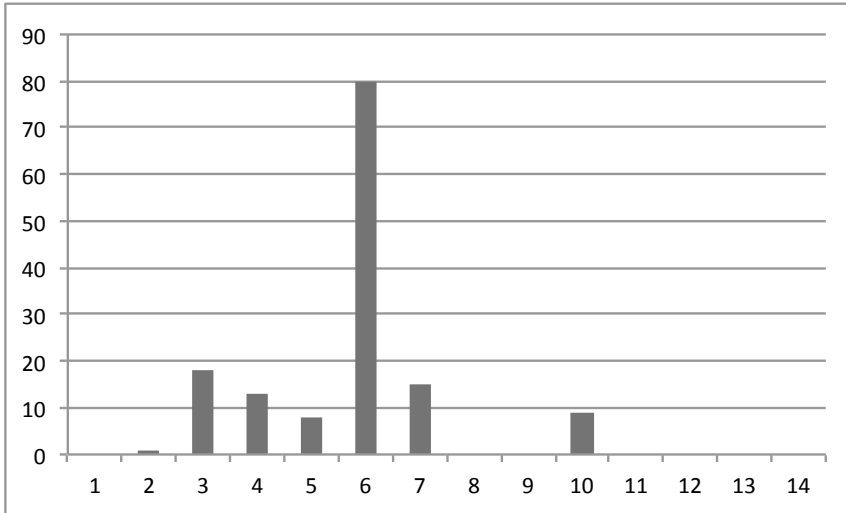
Experimental Setup

- 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)
- Brauer and King (CAV'11)
- Goldberg and Manolios (arXiv'12)

Discussion

- 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/>
 - There are faster techniques though

Two Topics

1. Existential quantifier elimination as incremental SAT solving
2. Approximate projection for (non-) linear constraints



Some ideas

The Ultimate Goal (1/3)

- We have a conjunction of integral non-linear inequalities:

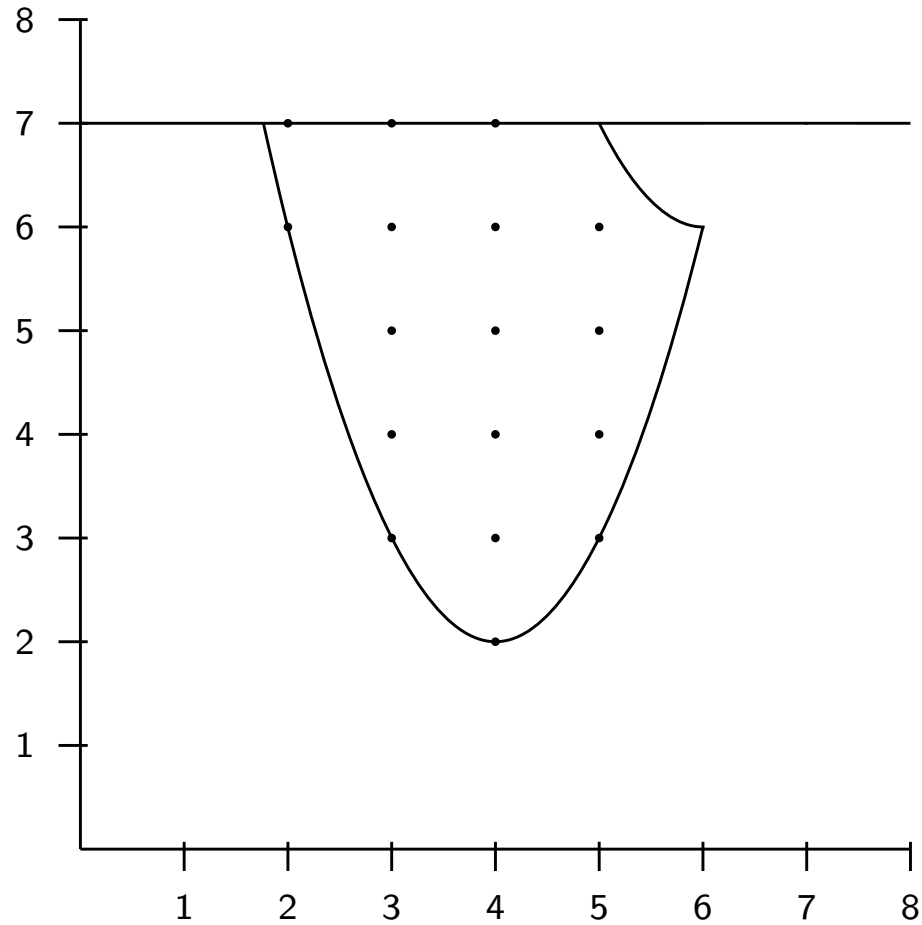
$$y \geq (x - 4)^2 + 2$$

$$\wedge \quad 0 \leq y \leq 7$$

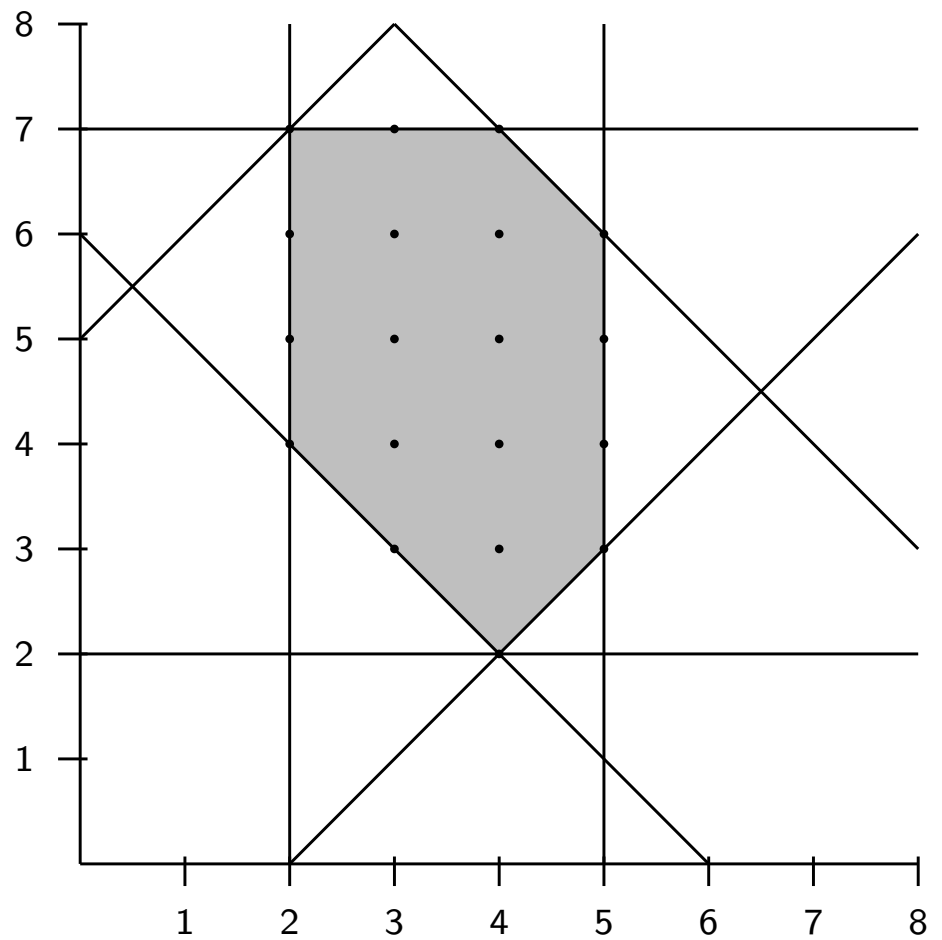
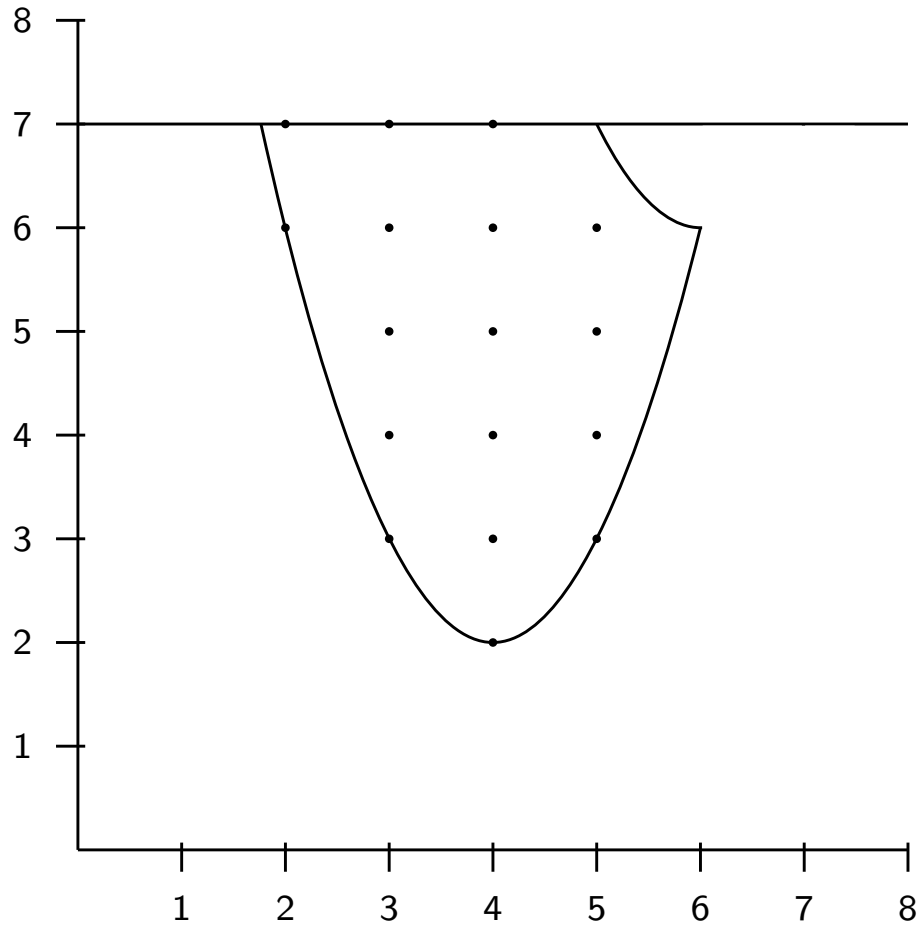
$$\wedge \quad y < (x - 5)^2 + 6$$

- Such systems are rather heavyweight
- Over-approximate representation using some linear template domain possible?

The Ultimate Goal (2/3)



The Ultimate Goal (3/3)



Simpler Example

- Consider
$$\varphi = \left\{ \begin{array}{l} y = x^2 + 2 \\ \wedge 0 \leq x \leq 10 \\ \wedge 0 \leq y \leq 50 \end{array} \right\}$$
 Variable bounds are important here!
- Approximate using octagons, i.e., $\pm x \pm y \leq c$
- Take $x + y \leq c$, which entails $0 \leq c \leq 60$
- Want to compute least sound c

Refining $x + y \leq c$

$$\varphi = \begin{cases} y = x^2 + 2 \\ \wedge 0 \leq x \leq 10 \\ \wedge 0 \leq y \leq 50 \end{cases} \quad 0 \leq c \leq 60$$

- We know $(0 \leq c \leq 30) \vee (31 \leq c \leq 60)$
- Is $\varphi \wedge (31 \leq c \leq 60)$ satisfiable?
 - Just query an SMT solver
 - Model $(x = 6) \wedge (y = 38)$ entails $x + y = 44$
 - Thus $(44 \leq c \leq 60)$

Refining $x + y \leq c$

$$\varphi = \begin{cases} y = x^2 + 2 \\ \wedge 0 \leq x \leq 10 \\ \wedge 0 \leq y \leq 50 \end{cases} \quad (44 \leq c \leq 60)$$

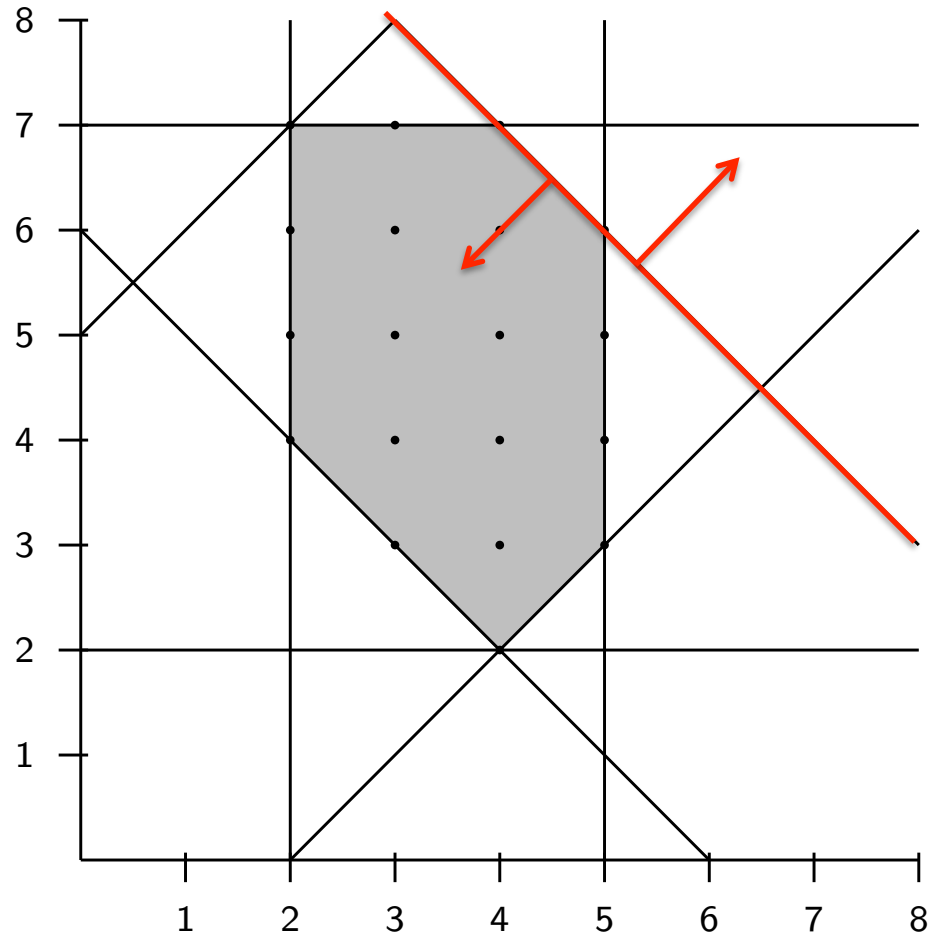
- Simply split $(44 \leq c \leq 60)$ into $(44 \leq c \leq 52) \vee (53 \leq c \leq 60)$
- Is $\varphi \wedge (53 \leq c \leq 60)$ satisfiable?
- No! Hence $(44 \leq c \leq 52)$

Refining $x + y \leq c$

$$\varphi = \begin{cases} y = x^2 + 2 \\ \wedge 0 \leq x \leq 10 \\ \wedge 0 \leq y \leq 50 \end{cases} \quad (44 \leq c \leq 52)$$

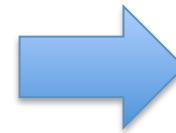
- Just keep goin'!
- Eventually gives $c = 44$, hence $x + y \leq 44$
- Could have stopped prematurely though
 - Kind of an anytime projection algorithm
 - $x + y \leq 47$ is sound, too

The Result



Solving Non-Linear Constraints in Z3

1. Simplify formula
2. Bit-blast and run SAT solver
 - a. If SAT return SAT
 - b. Otherwise check soundness of transformation
3. Run general-purpose SMT engine
 - a. Simplex
 - b. Bound propagation
 - c. Gomory cuts
 - d. Interval splitting
 - e. Gröbner bases



Sound + complete for
(very) bounded non-linear
integer arithmetic (QF_NIA)

Related Work

- Lot of work on QE for linear and non-linear systems, from Fourier-Motzkin till 2011
- Approximate methods based on refinements?
 - None known to me (I'm no expert though)
 - Hints?
- Similar strategies found in abstract interpretation
 - Codish et al. (TPLP'08)
 - Brauer, King and Kowalewski (FMICS'10)
 - Brauer and King (ESOP'11)

Concluding Discussion

- Projection from (non-) linear system to (non-) linear system
- Approximations based on incremental refinements in some template domain
 - Necessitates bounded problems
 - Opinions? Any useful?
- Use off-the-shelf SMT solver and straightforward encodings
 - No need to be a genius

Excellent

