Bit-Blasting for Termination Analysis

Johannes Waldmann, HTWK Leipzig, Germany

September 8, 2015

Constraints for Linear Interpretations

- (typical) exercise: Find montone linear functions $a : x \mapsto a_1 x + a_0, b : x \mapsto b_1 x + b_0 : \mathbb{N} \to \mathbb{N}$ such that $\forall x \in \mathbb{N} : a(b(x)) > b(a(x))$
- Application: these a, b prove termination of string rewriting system $\{ab \to ba\}$
- Constraints: monotonicity: $a_1 > 0, b_1 > 0$,
  map into domain: $a_0 \geq 0, b_0 \geq 0$,
  $a(b(x)) = a_1 b_1 x + a_1 b_0 + a_0$,
  $b(a(x)) = b_1 a_1 x + b_1 a_0 + b_0$,
  compare coefficients: $a_1 b_0 + a_0 > b_1 a_0 + b_0$
- general task: solve system of inequalities between polynomials over $\mathbb{N}$ (SMT-LIB: QF_NIA)

QF_NIA Example

```
(set-logic QF_NIA)
(set-option :produce-models true)
(declare-fun P () Int) (declare-fun Q () Int)
(declare-fun R () Int) (declare-fun S () Int)
(assert (and (< 0 P) (<= 0 Q) (< 0 R) (<= 0 S)))
(assert (> (+ (* P S) Q) (+ (* R Q) S)))
(check-sat)(get-value (P Q R S))
```

```
$ z3 ab-ba.smt2
sat
((P 14) (Q 9) (R 11) (S 7))
```

just for demonstration, we don’t usually do it like this

Matrix Interpretations

- (typical) exercise: find matrices over $\mathbb{N}$
  $$A, B, C \in \begin{pmatrix} \geq 1 & \dots & \\ \vdots & \geq 0 & \vdots \\ & \dots & \geq 1 \end{pmatrix} \text{ with}$$
  $$A^2 - BC, B^2 - AC, C^2 - AB \in \begin{pmatrix} & \dots & > 0 \\ \geq 0 & & \vdots \end{pmatrix}$$
- application: this proves termination of
  $\{aa \to bc, bb \to ac, cc \to ab\}$
  (was open for some years, solved in 2005)
- general task (again): solve system of inequalities between polynomials over $\mathbb{N}$.

Matrix Interpretation Example

```
main_standard solve = do
  res :: Maybe [Matrix 5 Integer] <- solve $ do
    ms @ [a,b,c] :: [Matrix 5 (Natural 3)] <-
         replicateM 3 $ unknown_positive_s
    rule_s [a,a] [b,c] ; rule_s [b,b] [a,c] ; rul
    return $ C.decode ms
  print res

rule_s lhs rhs = do
  let word (x:xs) = foldM times x xs
  l <- word lhs ; r <- word rhs
  assert_matrix_greater_s l r
```

DSL (embedded in Haskell) for SAT encoding

Backend: Minisat-API or Glucose via DIMACS format

Exotic Matrix Interpretations: Fuzzy

- exercise: find matrices over the fuzzy semiring $\mathbb{F} = (\mathbb{N} \cup \{\infty\}, \min, \max)$ $A, B \in \begin{pmatrix} < \infty & \dots \\ \vdots & * \end{pmatrix}$ with $A^2B^2 >_\infty B^3A^3$ where $x >_\infty y$ iff ($x > y$ or $x = \infty = y$)
- this proves termination of $aabb \to bbbaaa$ (a famous test case for automated termination, "Zantema's Problem", `tpdb-4.0/SRS/Zantema/z001.srs`)
- general task: boolean combination (note "or") of difference constraints (SMT-LIB: QF_IDL)

Exotic Matrix Interpretations: Tropical

- exercise: find matrices
  over the tropical semiring $\mathbb{T} = (\mathbb{N} \cup \{\infty\}, \min, +)$
  $A, B \in \begin{pmatrix} < \infty & \dots \\ \vdots & * \end{pmatrix}$ with $A^2B^2 >_\infty B^3A^3$
  where $x >_\infty y$ iff ($x > y$ or $x = \infty = y$)
- (again) proves termination of $aabb \to bbbaaa$
- boolean combination of linear inequalities
  (SMT-LIB: QF_LIA)

Flavours of Constraint Programming

- (mixed) integer linear programs
- finite domain constraints
- boolean satisfiability (SAT)
  DPLL (propagation, backtracking)
  with CDCL (clause learning, backjumping),
  preprocessing (var. and clause elimination)
- SAT modulo Theory (SMT)
  T: linear inequalities (LRA), difference constraints (IDL), bitvector operations (BV)
  "lazy approach": DPLL(T)
- "eager approach" for BV: *bit-blasting*

Methods to solve Polynomial Constraints

- matrices over $\mathbb{N}$: QF_NIA is mostly hopeless
  Tarski, QEPCAD
- fix bit width, use QF_BV (bit vectors)
- but their arithmetics is silently overflowing
- for small widths, use hand-crafted bit-blasting
- matrices over $\mathbb{T}, \mathbb{F}$: the boolean part dominates
  (the “or” in “min” is used very often)
- again, QF_BV or bit-blasting

. . . solve Pol. Constraints (cont.)

- use QF_LRA (!) [BLN$^+$09, YKS14]
  for determining coefficients of $x \mapsto a_i x + b_i$,
  bit-blast a_i, obtain linear inequalitites for b_i
  better than QF_NIA, relation to QF_BV not clear
- each termination prover somehow bit-blasts,
  but deeply buried as subroutine in proof search
  no uniform testbed, no reliable comparison
- no-one has seriously used “classical” constraint
  programming (Gecode, . . .)
  or its modern variants (Zinc)
- if you can beat our bit-blasting, you’re very much
  welcome (win the Termination Competition)

Termination Competitions

- since 2003, yearly
- input: rewrite system R,
  out: YES (R terminates), NO, MAYBE/timeout

extensions:

- variants of rewriting (strategies, modulo AC,...)
- programming languages (Haskell,Prolog,Java,C)
- complexity (derivation lengths)
- certification (of proofs of (non) termination)

termcomp 2015:

- 10 participants, 10^4 problems, 10^7 sec (CPU)
- 10 h (wall), `http://www.starexec.org/`

(SAT) Constraints for Termination

- precedence for path orders:
  Kurihara, Kondo [KK99] (using BDD for solving)
  Stuckey et al. [CLS08]
- coefficients for interpretations:
  polynomials [FGM$^+$07],
  matrices [HW06, EWZ08]
- looping derivations
  in string rewriting [ZSHM10], in term rewriting
- semantic labelling w.r.t. finite model
  use Haskell-to-SAT compiler CO4,
  ongoing PhD thesis of Alexander Bau

Bit-Blasting for General Arithmetics

- standard approach: circuit $\xrightarrow{\text{Tseitin}}$ CNF
- addition: ripple-carry (linear depth) or something tricky (with log depth)?
  overhead of carry-lookahead is too much (for small widths, and for larger, multiplication is the blocker anyway)
- multiplication: "school" method (repeated add-and-shift) or . . . ? (fake) Wallace multiplier (with dumb addition at the very end)
- in any case: integrated (early) overflow detection

General Bitblasting Examples

```
half_adder x y = do
  r <- B.xor2 x y ; c <- B.and [x,y]
  return (r,c)

add xs ys = do
  let go (Just c) [] [] = do
        B.assert [ B.not c ] ; return []
      go Nothing (x:xs) (y:ys) = do
           (r,c) <- half_adder x y
           (r:) <$> go (Just c) xs ys
      go (Just c) (x:xs) (y:ys) = do
           (r,c) <- full_adder x y c
           (r:) <$> go (Just c) xs ys
  go Nothing xs ys
```

Bit-Blasting for Narrow Arithmetics

- approach: no extra variables (no Tseitin)
  — use a minimal equivalent CNF
- example: (non-overflowing) 3 bit addition
  has a CNF with 24 clauses (on 9 variables)
  Ripple-Carry adder has 2 extra vars (carries)
  and $2 * 14 + 7$ clauses (2 full, 1 half adder)
- intermediate approach: very few extra variables,
  use minimal satisfiable-equivalent CNF
  (ongoing MSc. thesis by Martin Finke)
- divide and conquer for larger widths

Narrow Bitblasting Example

```
mul3 [x1,x2,x3] [x4,x5,x6] = do
 res@[x7,x8,x9] <- replicateM 3 boolean
 let a = assert
 a [x4,not x7] ; a [x4,x5,not x8]
 a [not x3,not x6] ; a [not x3,not x5]
 a [not x3,not x4,x9] ; a [not x3,x4,not x9]
 a [x3,x5,x6,not x9] ; a [not x2,not x6]
 a [not x2,not x5,x9] ; a [not x2,not x5,not x7]
 a [not x2,not x4,x8] ; a [x2,not x5,x6,not x9]
 a [x2,x5,not x8] ; a [not x1,not x6,x9]
 a [not x1,not x5,x8] ; a [not x1,not x4,x7]
 a [x1,not x7] ; a [x1,not x6,not x9]
 a [x1,x4,not x8] ; a [x1,x2,not x8]
 return res
```

Optimal CNFs - with respect to what?

- circuit optimisation aims to reduce size (area), depth (delay), fan-out (current),. . .
- for bitblasting, actual aim is DPLL run-time (for the complete formula)
- correlation to size/shape of (sub)formulas is loose and/or unknown
- can we measure *propagatability*? (unit propagation is what speeds DPLL)

Tightness of CNF encodings

- CNF F is *(UP) tight for conflicts* if for each partial assignment σ than cannot be extended to a model of F, $F\sigma$ contains a conflict clause (creates a conflict clause by UP)
- CNF F is *(UP) tight for propagation* if for each partial assignment σ and each unique extension to $v \notin \mathrm{dom}(\sigma)$, $F\sigma$ contains a unit clause for v (creates such a clause by UP)
- it is not clear how to encode this (efficiently) for the CNF optimisation problem
- but can add clauses afterwards for tightness.

Example: non-tight for Conflicts

- semantics: non-overflowing 3 bit multiplication (9 variables)
- implementation: school method
- partial assignments (LSB is left)
  that cannot be extended to a model
  but that do not give conflict by unit prop:
  $\ldots * .10 = 1.1$ meaning $. * \{2, 3\} = \{5, 7\}$
  $\ldots * 11. = .01$ meaning $. * \{3, 7\} = \{4, 5\}$
  $..0 * ..0 = 1.1$ meaning $\{\leq 3\} * \{\leq 3\} = \{5, 7\}$

Examples: non-tight for Propagation

half adder:

- implementation: $(r = x \oplus y, c = x \wedge y)$
  with $4 + 3$ clauses
- semantics implies $r \Rightarrow \neg c$
  but this cannot be proven by unit prop

3 bit non-overflowing multiplication

- implementation: school method,
  tight half and full adders
- $\ldots * .1. = 1..$ implies $.0. * .1. = 11.$
- but this cannot be unit-propagated
- (unit prop. finds $1.. * .1. = 1..$ though)

CNFs for the Fuzzy Semiring (max,min)

- our constraints are just $x > y$, have small model property.
- should use order encoding for numbers, $[n] = (\underbrace{1, \ldots, 1}_{n}, \underbrace{0, \ldots, 0}_{w-n})$.
- then min/max are cheap (element-wise and/or)
- k-ary min
  not by nested 2-ary min ($3(k-1)w$ clauses)
  but one element-wise (k-ary) or ($(k+1)w$ clauses)

Symmetry Breaking

(AFAIK, no-one considered this for termination)

- for standard matrix interpretations, can permute indices $\{1, d\}$ and $\{2, \ldots, d - 1\}$.
- for exotic, $\{2, \ldots, d\}$.

Adding Redundant Constraints

- fuzzy numbers are order-encoded (monotone sequence of bits)
- fuzzy operations (min,max) (bit-wise and, or) produce monotone values
- can add monotonicity constraint for the results (redundant but possibly helpful)

data: z001 fuzzy dim 9 bits 6 with glucose on kernkraft (for re-paired version)
baseline: 118 min, with monotonicity: 8 min,

Fine Points of Tropical Bit Blasting

- recall $\mathbb{T} = \mathbb{N} \cup \{+\infty\}$, min, plus.
- encoding of $\mathbb{T}$: one “finite” bit, a (binary) number (“contents”)
- semiring addition (min)

  ```
  Tropical <$> (finite s || finite t)
      <*> min (contents s) (contents t)
  ```

  needs an extra condition to work
- semiring multiplication (plus)

  ```
  Tropical <$> (finite s && finite t)
      <*> plus (contents s) (contents t)
  ```

  will *not* work as expected

Optimisations specific to Matrix Products

- instead of `a*a*b*b > b*b*b*a*a*a` (8 Mult.)
- compute
  `a2 = a*a; b2 = b*b; a2*b2 > b2*b*a*a2`
  (6 Mult.)
- in general [BLNW13]:
  eliminate common substrings,
  (implementation: repeatedly remove pairs)
  this uses associativity of matrix multiplication,

Solving Matrix Constraints by Completion

- increase entries in matrices, along a path
- path may contain “fresh” nodes
  (increase matrix dimension)
- this is a form of local search
- for standard matrix constraints, use as heuristics
- for fuzzy matrix constraints, there is a semi-algorithm [EHW06] (if a solution exists, it will be found) that can do this very quickly (creating huge matrices)

Challenges

general:

- solve matrix constraints over $\mathbb{N}, \mathbb{T}, \mathbb{A}, \mathbb{F}$
- improve bit-blasting for QF_BV solvers

concrete open questions

- for $a^2 \to bc, b^2 \to ac, c^2 \to ab$ over $\mathbb{N}$: solution with dimension < 5?
- ... is there an upper triangular solution? (0 below diag., ≤ 1 on diag.) (any dim.)
- for $a^2 b^2 \to b^3 a^3$ over $\mathbb{F}$: dimension < 9?
- for $a^2 b^2 \to b^3 a^3$ over $\mathbb{T}$: dimension $<$ previous?
- for $a^2 b^2 \to b^3 a^3$ over $\mathbb{N}$: upper triangular?

- Cristina Borralleras, Salvador Lucas, Rafael Navarro-Marset, Enric Rodríguez-Carbonell, and Albert Rubio.
  Solving non-linear polynomial arithmetic via SAT modulo linear arithmetic.
  In Renate A. Schmidt, editor, *Automated Deduction - CADE-22, 22nd International Conference on Automated Deduction, Montreal, Canada, August 2-7, 2009. Proceedings*, volume 5663 of *Lecture Notes in Computer Science*, pages 294–305. Springer, 2009.
- Alexander Bau, Markus Lohrey, Eric Nöth, and Johannes Waldmann.

Compression of rewriting systems for termination analysis.
In Femke van Raamsdonk, editor, *24th International Conference on Rewriting Techniques and Applications, RTA 2013, June 24-26, 2013, Eindhoven, The Netherlands*, volume 21 of *LIPIcs*, pages 97–112. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2013.

Michael Codish, Vitaly Lagoon, and Peter J. Stuckey.
Solving partial order constraints for LPO termination.
JSAT, 5(1-4):193–215, 2008.

- Jörg Endrullis, Dieter Hofbauer, and Johannes Waldmann.
  Decomposing terminating rewrite relations.
  Workshop on Termination, Seattle, 2006.

- Jörg Endrullis, Johannes Waldmann, and Hans Zantema.
  Matrix interpretations for proving termination of term rewriting.
  J. Autom. Reasoning, 40(2-3):195–220, 2008.

- Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl.

SAT solving for termination analysis with polynomial interpretations.
In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing - SAT 2007, 10th International Conference, Lisbon, Portugal, May 28-31, 2007, Proceedings*, volume 4501 of *Lecture Notes in Computer Science*, pages 340–354. Springer, 2007.

Dieter Hofbauer and Johannes Waldmann.
Termination of string rewriting with matrix interpretations.
In Frank Pfenning, editor, *Term Rewriting and Applications, 17th International Conference, RTA*

2006, Seattle, WA, USA, August 12-14, 2006, Proceedings, volume 4098 of *Lecture Notes in Computer Science*, pages 328–342. Springer, 2006.

Masahito Kurihara and Hisashi Kondo.
Heuristics and experiments on BDD representation of boolean functions for expert systems in software verification domains.
In Norman Y. Foo, editor, *Advanced Topics in Artificial Intelligence, 12th Australian Joint Conference on Artificial Intelligence, AI '99, Sydney, Australia, December 6-10, 1999, Proceedings*, volume 1747 of *Lecture Notes in*

Computer Science, pages 353–364. Springer, 1999.

Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe.
Nagoya termination tool.
CoRR, abs/1404.6626, 2014.

Harald Zankl, Christian Sternagel, Dieter Hofbauer, and Aart Middeldorp.
Finding and certifying loops.
In Jan van Leeuwen, Anca Muscholl, David Peleg, Jaroslav Pokorný, and Bernhard Rumpe, editors, *SOFSEM 2010: Theory and Practice of Computer Science, 36th Conference on Current*

Trends in Theory and Practice of Computer Science, Spindleruv Mlýn, Czech Republic, January 23-29, 2010. Proceedings, volume 5901 of *Lecture Notes in Computer Science*, pages 755–766. Springer, 2010.