Verification of Data Paths Using Unbounded Integers: Automata Strike Back

Tobias Schuele and Klaus Schneider

Reactive Systems Group, University of Kaiserslautern
Gottlieb-Daimler-Straße 48, 67663 Kaiserslautern, Germany

Haifa Verification Conference
October 23-26, 2006

Outline

Hardware Verification Using...

... Propositional Logic

- if the system to be verified is given as a netlist of gates
- allows BDD/SAT–based symbolic/bounded model checking
- takes into account overflows etc. and bugs in synthesis tools
- large data paths hardly tractable without prior abstraction
- arithmetic operations must be broken down to prop. formulas

... More Powerful Base Logics

- higher level of abstraction than gate level (unbounded bitwidth)
- support for arithmetical and bitvector operations (AND, SRA)
- Presburger arithmetic: decidable predicate logic over integers
- UCLID: Presburger arithmetic, uninterpreted functions, arrays

Decision Procedures for Presburger Arithmetic (PA)

Syntactic Procedures (ILP)

- quantifier elimination [Presburger 1929], [Cooper 1972]
- SUP–INF method [Bledsoe 1975], [Shostak 1977]
- Fourier–Motzkin variable elimination, Omega test [Pugh 1992]

Automata–Based Procedures

- construct an automaton $\mathcal{A}_\Phi$ accepting the models of a formula Φ
- [Büchi 1960], [Boudet, Comon 1996], [Wolper, Boigelot 2000]
- BDD–based: [Schuele, Schneider 2002], [Ganesh et al. 2002]

SAT–Based Procedures

- propositional encoding of Fourier–Motzkin elim. [Strichman 2002]
- red. to finite domain using solution bounds [Seshia, Bryant 2005]
- restricted to quantifier–free Presburger arithmetic (as SUP–INF)

Quantifier–Free PA + Bitvector Operations (QFPAbit)

Syntax

- terms: addition, const. multiplication, bitvector operations
- formulas: propositional variables, relations, Boolean operations
- example: $(a \vee b) \wedge (x \leq 2y) \vee (y = x \overrightarrow{\wedge} z)$

Semantics

- semantics based on two's complement encoding
- example: $-6 = \langle 1010 \rangle_{\mathbb{Z}}$ and $5 = \langle 0101 \rangle_{\mathbb{Z}}$
- $-6 \overrightarrow{\vee} 5 = \langle 1111 \rangle_{\mathbb{Z}} = -1$ and $-6 \overrightarrow{\wedge} 5 = \langle 0000 \rangle_{\mathbb{Z}} = 0$

Expressive Power

- 'x is a power of two' not expressible in pure (QF)PA
- but expressible in QFPAbit: $1 + (x - 1 \overrightarrow{\vee} x) = 2x \wedge x > 0$
- ▶ QFPAbit is strictly more expressive than QFPA

Alternating Finite Automata (AFAs)

Foundations

- sequential circuit, symbolic description of a deterministic finite automaton with an explicit partitioning of the transition relation
- state variables $\vec{q} = (q_0, \ldots, q_m)$, input variables $\vec{v} = (v_0, \ldots, v_n)$
- *initial function* *transition functions* *final states*

$$I(\vec{q}, \vec{v}) \qquad \begin{array}{l} q_0' \leftrightarrow \delta_0(\vec{q}, \vec{v}) \\ \vdots \\ q_m' \leftrightarrow \delta_m(\vec{q}, \vec{v}) \end{array} \qquad \begin{array}{l} b_0 \\ \vdots \\ b_m \end{array}$$

- equational structure: efficiently unwind transition functions
- no new state variables as usually required in BMC

Translation of QFPAbit Formulas to AFAs

Words and Assignments

$$\begin{array}{rcr} x & = & 5 \\ y & = & -2 \\ z & = & 3 \end{array} \quad \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$$

Bitvector Expressions

- separate bitvector parts from arithmetic parts
- $(x \overrightarrow{\wedge} y) + z = s$ satisfiable iff $(x \overrightarrow{\wedge} y) = t \wedge (t + z = s)$ satisfiable
- construction of AFAs by bitwise evaluation of the given equation

Arithmetic Equations

- recursion equation: $T = 0$ iff $T \equiv_2 0$ and $\lfloor T/2 \rfloor = 0$
- transitions functions compute $\lfloor T/2 \rfloor$ and check $T \equiv_2 0$
- initial function checks whether $\lfloor -T/2 \rfloor = 0$ holds

Checking Emptiness of AFAs

Unwinding AFAs

- successively substitute transition functions for state variables
- initial function satisfiable $\Rightarrow$ language not empty
- ▶ semi–decision procedure, cannot prove emptiness

Computing Solution Bounds

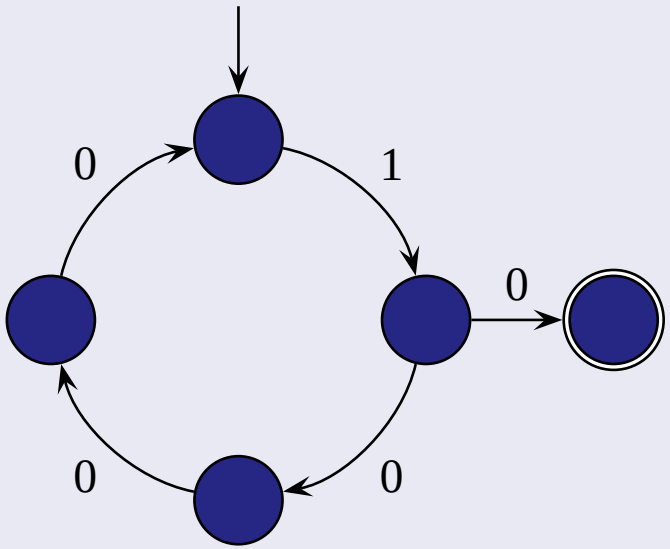
- every satisfiable quantifier–free Presburger formula Φ has a solution whose size is polynomially bounded in the size of Φ

  [Borosh, Treybig 1976], [Papadimitriou 1978], [Seshia, Bryant 2005]
- compute bound on the number of unwinding steps (completeness threshold in bounded model checking)
- ▶ no polynomial bounds for QFPAbit formulas

Solution Bounds for QFPAbit Formulas

Describing Rings in QFPAbit

R_k = 'every k-th bit is one'



Combining Rings in QFPAbit

- $p(i)$: i-th prime number
- $S_n = -2 \overrightarrow{\bigwedge_{i=1}^{n}} R_{p(i)}$
- $\varnothing$ of S_n: $\prod_{i=1}^{n} p(i) \sim e^n$
- size of smallest solution exponential in formula size

Checking Emptiness Using Induction

Checking Safety Properties Using Induction [Sheeran et al. 2000]

- given a finite transition system and a property $\mathcal{P}$, check whether $\mathcal{P}$ invariantly holds on all paths originating in an initial state
- base case: path of length k such that $\mathcal{P}$ holds on every state
- ind. step: path can be extended by one state such that $\mathcal{P}$ holds

Algorithm for Checking Emptiness of an AFA

if not sat(*initial function*) **then return true;**
loop
  if not valid(*base case*) **then return false;**
  if valid(*induction step*) **then return true;**
  // unwind transition functions, i.e., increase induction depth
end;

Experimental Results (I)

Averest Benchmarks

Benchmark	Averest		NuSMV		Cad. SMV	
	AFA	DFA	8 bit	32 bit	8 bit	32 bit
BinarySearch	0,3	-	16,7	182,9	2,3	99,8
	0,2	22,4	18,7	-	12,6	-
BubbleSort	117,3	-	3,1	347,0	0,1	0,1
	102,6	-	9,6	-	2,4	93,3
FastMax	0,1	0,8	1,3	398,9	0,0	0,1
	0,3	1,8	3,6	-	-	-
ParallelPrefixSum	1,3	-	4,1	-	0,1	0,1
	7,3	-	-	-	453,7	-
Partition	1,1	-	147,0	-	39,7	-
	0,4	-	256,1	-	112,1	-
SortingNetwork	3,0	-	534,2	-	0,0	0,1
	2,1	-	-	-	-	-

Experimental Results (II)

SMT–LIB Benchmarks

Benchmark	Averest	CVC Lite	MathSat	Yices
ckt_PROP0_tf_15	< 1	24.9	42.2	< 1
ckt_PROP0_tf_20	< 1	-	66.0	< 1
FISCHER5-4-fair	4.2	5.4	< 1	< 1
FISCHER5-6-fair	-	15.1	< 1	< 1
FISCHER5-8-fair	-	67.6	1.9	< 1
MULTIPLIER_4	< 1	172.9	1.2	< 1
MULTIPLIER_6	5.6	-	42.1	1.0
MULTIPLIER_7	33.7	-	255.4	10.3
ADDER_6	1.2	-	26.9	< 1
ADDER_8	23.7	-	-	94.3
ADDER_10	115.0	-	-	-
wisa3	6.4	226.0	-	< 1
wisa4	4.1	28.0	-	1.1

Summary and Conclusion

Summary

- polynomial time translation of PA formulas to finite automata
- reduction of satisfiability problem to propositional logic
- ability to benefit from advances in SAT solving
- does not require computation of CNF or DNF
- low overhead for propositional subformulas ($a \wedge \neg a \wedge x = 2y$)
- strict superset of Presburger arithmetic, bitvector operations
- automata encode all models $\Rightarrow$ easy to find the smallest solution

Future Work

- combination with other procedures (Nelson-Oppen/Shostak)
- combination with syntactic methods to deal with quantifiers