Automatic Abstraction for Intervals using Boolean Formulae

Jörg Brauer (RWTH Aachen University)
Andy King (Portcullis Computer Security Limited)

15.09.2010 @ SAS

Motivating Example (1/2)

```
1: INC R0;
2: MOV R1, R0;
3: LSL R1;
4: SBC R1, R1;
5: EOR R0, R1;
6: SUB R0, R1;
```

- Goal: Affine transfer functions that relate interval boundaries
- Wraps are ubiquitous on 8-bit architecture
- Guard wrapping inputs using octagons [Min06]

Motivating Example (2/2)

```
1: INC R0;
2: MOV R1, R0;
3: LSL R1;
4: SBC R1, R1;
5: EOR R0, R1;
6: SUB R0, R1;
```

$$(127 \leq r0 \leq 127) \Rightarrow (r0_l^\star = -128 \wedge r0_u^\star = -128)$$

$$(-128 \leq r0 \leq -2) \Rightarrow (r0_l^\star = -r0_u - 1 \wedge r0_u^\star = -r0_l - 1)$$

$$(-1 \leq r0 \leq 126) \Rightarrow (r0_l^\star = r0_l + 1 \wedge r0_u^\star = r0_u + 1)$$

- Key idea: Boolean encodings of semantics
- Compute abstractions of affine relations and guards separately using SAT

Guards for Wrapping

- Consider instruction `ADD r0 r1`
- Boolean encoding (outputs are primed):

$$\begin{aligned}\varphi(\mathbf{c}) \quad = \quad & (\wedge_{i=0}^{7} \mathbf{r0}'[i] \oplus \mathbf{r0}[i] \oplus \mathbf{r1}[i] \oplus \mathbf{c}[i]) \wedge \neg \mathbf{c}[0] \wedge \\ & (\wedge_{i=0}^{6} \mathbf{c}[i+1] \leftrightarrow (\mathbf{r0}[i] \wedge \mathbf{r1}[i]) \vee (\mathbf{r0}[i] \wedge \mathbf{c1}[i]) \vee (\mathbf{r1}[i] \wedge \mathbf{c}[i])\end{aligned}$$

- For example, enforce overflow:

$$\varphi(\mathbf{c})' \quad = \quad \varphi(\mathbf{c}) \wedge (\neg \mathbf{r0}[7] \wedge \neg \mathbf{r1}[7] \wedge \mathbf{r0}'[7])$$

- Then $\varphi(\mathbf{c})'$ characterizes overflow-case only

Characterization of Optimal Bounds

- Guards are of the form $\pm v_1 \pm v_2 \leq d$
- d is characterized as (similar to [Mon09]):
  - It is an upper bound for any $\pm v_1 \pm v_2$
  - For any other upper bound d', we have $d \leq d'$
- The „for any“ manifests itself in terms of universal quantification
  - Which is trivial for CNF formulae
  - Simply strike out all literals

Guards in Boolean Logic

- Safety:

$$\nu \quad = \quad \forall r0 : \forall r1 : (\varphi \Rightarrow \pm r0 \pm r1 \leq d)$$

- Optimality:

$$\mu \quad = \quad \forall r0 : \forall r1 : \forall d' : ((\varphi \Rightarrow \pm r0 \pm r1 \leq d') \Rightarrow d \leq d')$$

- Then solve $\nu \wedge \mu$ using SAT after q-elimination
- Observe that μ interacts with ν to impose an optimal bound

Boolean Characterization for Intervals

- Very similar formulation for relation between input- and output-intervals (but more technically involved)
- Also uses two-staged formulation to
  - First characterize safe output intervals
  - And then impose optimality
- However, still need to compute affine relations

Key Idea: Affine Closure

- Obtain a solution of formula using SAT
- Represent solution as matrix
- Add disequality to obtain new solutions
- Join with previous matrix
- Add disequality to obtain new solutions
- ...
- Eventually stabilizes since domain is finite

Example: Affine Closure

$$\varphi = \begin{cases} (\neg w[0]) \wedge \left(\wedge_{i=0}^{6} w[i+1] \leftrightarrow (v[i] \oplus \wedge_{j=0}^{i-1} v[j])\right) & \wedge \\ (\neg x[0]) & \wedge \\ \left(\wedge_{i=0}^{6} x[i+1] \leftrightarrow (w[i] \wedge x[i]) \vee (w[i] \wedge y[i]) \vee (x[i] \wedge y[i])\right) & \wedge \\ \left(\wedge_{i=0}^{7} z[i] \leftrightarrow w[i] \oplus x[i] \oplus y[i]\right) & \wedge \\ ((v[7] \leftrightarrow v[6]) \wedge (v[6] \leftrightarrow v[5])) \wedge ((y[7] \leftrightarrow y[6]) \wedge (y[6] \leftrightarrow y[5])) & \end{cases}$$

- Compute affine relations between variables z, v and y
- Could also be our Boolean characterization of intervals

Example: Affine Closure

- 1st solution: $(v = 0, y = 0, z = 2)$

$$\left[\begin{array}{ccc|c} 0 & 0 & 0 & 1 \end{array}\right] \sqcup \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right] = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

- 2nd solution: $(v = -1, y = 0, z = 0)$

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 \end{array}\right] \sqcup \left[\begin{array}{ccc|c} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 0 & -1 & -2 \\ 0 & 1 & 0 & 0 \end{array}\right]$$

- 3rd solution: $(v = 0, y = 1, z = 3)$

$$\left[\begin{array}{ccc|c} 2 & 0 & -1 & -2 \\ 0 & 1 & 0 & 0 \end{array}\right] \sqcup \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 3 \end{array}\right] = \left[\begin{array}{ccc|c} 2 & 1 & -1 & -2 \end{array}\right]$$

- Result: $2 \cdot v + y - z = -2$

Applying Transfer Functions

- Amounts to linear programming
- Given an octagonal guard g and input intervals i
- Treat affine transfer function f as target function and maximize/minimize f subject to $g \wedge i$
- Solve using Simplex or branch-and-bound (runtime vs. precision)

Example: Linear Programming

- Input:

  $i = (-3 \leq r0 \leq 4)$

$$(127 \leq r0 \leq 127) \Rightarrow (r0_l^\star = -128 \wedge r0_u^\star = -128)$$

$$(-128 \leq r0 \leq -2) \Rightarrow (r0_l^\star = -r0_u - 1 \wedge r0_u^\star = -r0_l - 1)$$

$$(-1 \leq r0 \leq 126) \Rightarrow (r0_l^\star = r0_l + 1 \wedge r0_u^\star = r0_u + 1)$$

- Solving the two remaining linear programs then yields: $r0_l^\star = 0$

  $r0_u^\star = 5$

Related Work

- [Min06] A. Mine: The Octagon Abstract Domain (HOSC 2006)
- [Mon09] D. Monniaux: Automatic Modular Abstractions for Linear Constraints (POPL 2009)
- [KS10] A. King, H. Søndergaard: Automatic Abstraction for Congruences (VMCAI 2010)
- [RSY04] T. Reps, M. Sagiv, G. Yorsh: Symbolic Implementation of the Best Transformer (VMCAI 2004)

Summary

- Deriving transfer functions for bit-vector programs using SAT
- Combination of octagons and affine equalities
- Applying a transfer function amounts to linear programming

Future Work

- Obtain executable transfer functions to dismiss the need for linear programming
- Transfer functions for loops
- Affine relations could be substituted with more expressive domain, say, polynomials of bounded degree

Thank you very much!