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;

\[ (127 \leq r_0 \leq 127) \Rightarrow (r_0^* = -128 \land r_0^* = -128) \]
\[ (-128 \leq r_0 \leq -2) \Rightarrow (r_0^* = -r_0 - 1 \land r_0^* = -r_0 - 1) \]
\[ (-1 \leq r_0 \leq 126) \Rightarrow (r_0^* = r_0 + 1 \land r_0^* = r_0 + 1) \]

- 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):

  \[
  \varphi(c) = (\land_{i=0}^{7} r0'[i] \oplus r0[i] \oplus r1[i] \oplus c[i]) \land \neg c[0] \land \\
  (\land_{i=0}^{6} c[i+1] \leftrightarrow (r0[i] \land r1[i]) \lor (r0[i] \land c1[i]) \lor (r1[i] \land c[i])
  \]

- For example, enforce overflow:

  \[
  \varphi(c)' = \varphi(c) \land (\neg r0[7] \land \neg r1[7] \land r0'[7])
  \]

- Then \( \varphi(c)' \) 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:**
  \[ \nu = \forall r_0 : \forall r_1 : (\varphi \Rightarrow r_0 \pm r_1 \leq d) \]

- **Optimality:**
  \[ \mu = \forall r_0 : \forall r_1 : \forall d' : ((\varphi \Rightarrow r_0 \pm r_1 \leq d') \Rightarrow d \leq d') \]

- Then solve \( \nu \land \mu \) using SAT after q-elimination
- Observe that \( \mu \) interacts with \( \nu \) 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

\[
\varphi = \begin{cases} 
(\neg w[0]) \land (\land_{i=0}^{6} w[i + 1] \leftrightarrow (v[i] \oplus \land_{j=0}^{i-1} v[j])) \\
(\neg x[0]) \\
(\land_{i=0}^{6} x[i + 1] \leftrightarrow (w[i] \land x[i]) \lor (w[i] \land y[i]) \lor (x[i] \land y[i])) \\
(\land_{i=0}^{7} z[i] \leftrightarrow w[i] \oplus x[i] \oplus y[i]) \\
\end{cases}
\]

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

• **1st solution:** \((v = 0, y = 0, z = 2)\)
  \[
  \begin{bmatrix}
  0 & 0 & 0 & 1 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 2 \\
  \end{bmatrix}
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 2 \\
  \end{bmatrix}
  =
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 2 \\
  \end{bmatrix}
  \]

• **2nd solution:** \((v = -1, y = 0, z = 0)\)
  \[
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 2 \\
  \end{bmatrix}
  \begin{bmatrix}
  1 & 0 & 0 & -1 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  \end{bmatrix}
  =
  \begin{bmatrix}
  2 & 0 & -1 & -2 \\
  0 & 1 & 0 & 0 \\
  \end{bmatrix}
  \]

• **3rd solution:** \((v = 0, y = 1, z = 3)\)
  \[
  \begin{bmatrix}
  2 & 0 & -1 & -2 \\
  0 & 1 & 0 & 0 \\
  \end{bmatrix}
  \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 1 \\
  0 & 0 & 1 & 3 \\
  \end{bmatrix}
  =
  \begin{bmatrix}
  2 & 1 & -1 & -2 \\
  \end{bmatrix}
  \]

• **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 $g \land i$
- Solve using Simplex or branch-and-bound (runtime vs. precision)
Example: Linear Programming

- Input:
  
  $i = (-3 \leq r0 \leq 4)$
  
  \[
  \Rightarrow (127 \leq r0 \leq 127) \Rightarrow (r0^*_i = -128 \land r0^*_u = -128)
  \]
  
  \[
  \Rightarrow (-128 \leq r0 \leq -2) \Rightarrow (r0^*_i = -r0_u - 1 \land r0^*_u = -r0_i - 1)
  \]
  
  \[
  \Rightarrow (-1 \leq r0 \leq 126) \Rightarrow (r0^*_i = r0_i + 1 \land r0^*_u = r0_u + 1)
  \]

- Solving the two remaining linear programs then yields:
  
  $r0^*_i = 0$
  
  $r0^*_u = 5$
Related Work

• [Min06] A. Mine: The Octagon Abstract Domain (HOSC 2006)
• [Mon09] D. Monniaux: Automatic Modular Abstractions for Linear Constraints (POPL 2009)
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!