The Reachability-Bound Problem

Gulwani and Zuleger

PLDI 10

CS252r Spring 2011
The Reachability-Bound problem

• Find a symbolic worst case bound on the number of times a program point is reached
  • Intra-procedural: consider a program point within a procedure
  • Symbolic: give the bounds in terms of the procedure inputs
  • Bound the total number of times program point reached, not just number of times in inner loop
    • e.g., int i=0; while (i<n) { i++; j = i; while (j<n) {j++; •} }

© 2011 Stephen Chong, Harvard University
Solution

• Bound number of visits to program point $\pi$

1. Construct a disjunctive **transition system** that describes relationship of program variables in successive visits to $\pi$

2. Generate bounds from transition system using ranking functions.
In more detail...

- Construct control flow-graph of procedure
- Split program point of interest
- Consider CFG between split program point
- Now construct transition system with regard to $\pi$
Transitions

• Let live variables at $\pi_a$ be denoted $x,y,z,...$ and their counterparts at $\pi_b$ be denoted $x',y',z',...$

• A transition for $\pi$ is a relation $T(x,y,z,...,x',y',z',...)$ such that if $x,y,z$ take on values $v_1,v_2,v_3,...$ and $w_1,w_2,w_3,...$ during consecutive visits to $\pi$ then $T(v_1,v_2,v_3,..., w_1,w_2,w_3,...)$ holds.

• Assume a transition is expressed as a conjunction of formulas over $x,y,z,...,x',y',z',...$

• A transition system for $\pi$ is disjunction of transitions
Finding transition systems

• Abstract interpretation
  • Domain is logical formula, ordering $\sqsubseteq$ is implication $\Rightarrow$
  • Join is disjunction

• Transition systems for atomic statements

\[
\text{Translate}(x := e) = (x' = e) \land (\bigwedge_{y \neq x} y' = y)
\]

\[
\text{Translate(Assume(guard))} = \text{Id} \land \text{guard}
\]
Composing transition functions

- Initial transition system is $\text{Id}$

\[
\begin{array}{c}
\text{(c) Compose} \\
\begin{array}{c}
\cdots \\
\uparrow \pi_{in} \\
\text{stmt} \\
\downarrow \pi_{out} \\
\cdots
\end{array} \\
F[\pi_{out}] = F[\pi_{in}] \circ \text{Translate(stmt)}
\end{array}
\quad
\begin{array}{c}
\text{(d) Merge} \\
\begin{array}{c}
\cdots \\
\downarrow \pi_1 \\
\downarrow \pi_2 \\
\downarrow \pi_3 \\
\cdots
\end{array}
\end{array}
\]

**Definition 6** (Composition of Transition Systems). Given two transition systems $T(\vec{x}, \vec{x}') = \bigvee_i s_i$ and $T'(\vec{x}, \vec{x}') = \bigvee_j s'_j$, we define their binary composition to be

\[T \circ T' \overset{\text{def}}{=} \bigvee_{i,j} s_i \circ s'_j,
\]

where $s_i \circ s'_j$ denotes the transition

\[s_i(\vec{x}, \vec{x}') \circ s'_j(\vec{x}, \vec{x}') \overset{\text{def}}{=} \exists \vec{x}'' \left( s_i[\vec{x}''/\vec{x}'] \land s'_j[\vec{x}''/\vec{x}] \right),
\]

where $s_i[\vec{x}''/\vec{x}']$ denotes the substitution of $\vec{x}'$ by $\vec{x}''$ in $s_i$. 
Nested loops

• But what about nested loops?
• E.g.,

Ex1(uint n, bool[] A)
1 i := 0;
2 while (i < n)
3 j := i + 1;
4 while (j < n)
5 if (A[j])
6 ConsumResrce();
7 j--;  
8 n--;  
9 j++;  
10 i++;
Transitive closure

• Idea:
  • Compute transition system for one iteration of nested loop;
  • Take transitive closure of transition system
  • Use transitive closure as summary of nested loop

\textbf{Definition 8} (Transitive Closure). We say that $T'(\vec{x}, \vec{x}')$ is a transitive closure of a transition system $T(\vec{x}, \vec{x}')$ if

\[ \text{Id} \Rightarrow T' \quad \text{and} \quad T' \circ T \Rightarrow T' \]

• How to find transitive closure?
  • Analogous to finding a loop invariant
  • Can use a widening operator to guarantee termination
  • But can take advantage of additional structure in domain...
Convexity

• A theory is **convex** if
  • For all $G = g_1 \land \ldots \land g_n$
  • If $G \Rightarrow e_1 = e_2 \lor e_3 = e_4$ then either $G \Rightarrow e_1 = e_2$ or $G \Rightarrow e_3 = e_4$

• E.g. convex theory
  • Rational linear arithmetic

• E.g. non-convex theory
  • Integer linear arithmetic
    • $2 \leq x \leq 3 \Rightarrow x = 2 \lor x = 3$ but not the case that $2 \leq x \leq 3 \Rightarrow x = 2$ or that $2 \leq x \leq 3 \Rightarrow x = 3$
Convexity-like assumption

- Convexity: \( \left( \phi \Rightarrow \left( \bigvee_i (x_i = y_i) \right) \right) \Rightarrow \left( \bigvee_i (\phi \Rightarrow (x_i = y_i)) \right) \)

- Suppose \( \forall j \in 1..m \ s_j' \) is transitive closure of \( \forall i \in 1..n \ s_i \)

- Then \( \text{Id} \Rightarrow \bigvee_{k=1}^m s_k' \) and \( s_j' \circ s_i \Rightarrow \bigvee_{k=1}^m s_k' \)

- Distributing implication over disjunction, as for convexity gives:

**Definition 10** (Convexity-like Assumption).

Let \( T' = \bigvee_{j=1}^m s_j'(\vec{x}, \vec{x}') \) be a transitive closure for a transition system \( T = \bigvee_{i=1}^n s_i(\vec{x}, \vec{x}') \), where each \( s_i \) and \( s_j' \) is a conjunctive relation. We say that the transitive closure \( \bigvee_{j} s_j' \) satisfies the convexity-like assumption if there exists an integer \( \delta \in \{1, \ldots, m\} \), a map \( \sigma : \{1, \ldots, m\} \times \{1, \ldots, n\} \mapsto \{1, \ldots, m\} \), such that for all \( i \in \{1, \ldots, n\} \) and \( j \in \{1, \ldots, m\} \), the following holds:

\[
\text{Id} \Rightarrow s'_\delta \quad \text{and} \quad (s_j' \circ s_i) \Rightarrow s'_{\sigma(j,i)}
\]
Transitive closure

TransitiveClosure(\bigvee_{i=1}^{n} s_i)

1. for \( j \in \{1,\ldots,m\} - \{\delta\} \): \( s'_j := \text{false}; \)
2. \( s'_\delta := \text{Id}; \)
3. do 
4.     for \( i \in \{1,\ldots,n\} \) and \( j \in \{1,\ldots,m\} \):
5.         \( s'_{\sigma(j,i)} := \text{Join}(s'_{\sigma(j,i)}, s'_j \circ s_i) \)
6.     while any change in \( \bigvee_{j=1}^{m} s'_j \)
7. return \( \bigvee_{j=1}^{m} s'_j \);

• Notes:
  
• Need a “convexity witness” (\( \delta, \sigma \))

• May need a widening operator instead of the Join to ensure termination

• If algorithm terminates (using Join) then is precise!
  • i.e., at least as precise as any other transitive closure
Where are we at?

ReachabilityBound(\(\pi\))
1 \(T := \text{GenerateTransitionSystem}(\pi)\);
2 \(B := 1 + \text{ComputeBound}(T)\);
3 return \(\text{TranslateBound}(B, \pi)\);

GenerateTransitionSystem(\(\pi\))
1 \((\pi_a, \pi_b) := \text{Split}(\pi)\);
2 foreach top-level loop \(L\):
3 \(\pi_L := \text{location before header of } L\);
4 \(T := \text{GenerateTransitionSystem}(\pi_L)\);
5 \(T_c := \text{TransitiveClosure}(T)\);
6 Insert Summary(\(T_c\)) before header;
7 Remove back-edges;
8 Initialize \(F[\pi_a]\) to the transition system \(\text{Id}\);
9 Propagate transitions \(F\) using Merge/Compose rules;
10 return \(F[\pi_b]\);

TransitiveClosure(\(\bigvee_{i=1}^{n} s_i\))
1 for \(j \in \{1, \ldots, m\} - \{\delta\}\): \(s'_j := \text{false}\);
2 \(s'_\delta := \text{Id}\);
3 do {
4 for \(i \in \{1, \ldots, n\}\) and \(j \in \{1, \ldots, m\}\):
5 \(s'_{\sigma(j,i)} := \text{Join}(s'_{\sigma(j,i)}, s'_j \circ s_i)\)
6 } while any change in \(\bigvee_{j=1}^{m} s'_j\)
7 return \(\bigvee_{j=1}^{m} s'_j\);
Ranking function

• **Ranking functions** are used to prove termination
  • Integer function bounded below by zero, and decreases in each iteration

**Definition 13** (Ranking Function for a Transition). We say that an integer-valued function \( r(\vec{x}) \) is a ranking function for a transition \( s(\vec{x}, \vec{x}') \) if it is bounded below by 0 and if it decreases by at least 1 in each execution of the transition, i.e.,

  - \( s \Rightarrow (r > 0) \)
  - \( s \Rightarrow (r[\vec{x}' / \vec{x}] \leq r - 1) \)

We denote this by \( \text{Rank}(s, r) \).

We say that a ranking function \( r_1(\vec{x}) \) is more precise than a ranking function \( r_2(\vec{x}) \) if \( r_1 \leq r_2 \) (because in that case, \( r_1 \) provides a more precise bound for the transition than \( r_2 \)).
Finding ranking functions

• Use pattern-based matching
  • Fast, effective, quite precise
  • Makes calls to SMT solver to figure out if pattern matches
  • RankC(s) outputs a set of expressions that are ranking functions

• Arithmetic iteration patterns

- If \( s \Rightarrow (e > 0 \land e[x'/x] < e) \), then \( e \in \text{RankC}(s) \)
- If \( s \Rightarrow (e \geq 1 \land e[x'/x] \leq e/2) \), then \( \log e \in \text{RankC}(s) \)

• Boolean iteration patterns

- If \( s \Rightarrow (e \land \neg(e[x'/x])) \), then \( \text{Bool2Int}(e) \in \text{RankC}(s) \)

• ...

• May fail to find a ranking function
Bounding computation

• If transition system consists of single transition, and r is its ranking function, then max(0,r) is a symbolic bound
• If transition system has more than one transition, it gets harder
• Suppose transition system has 2 transitions: $s_1 \lor s_2$
  • In certain cases, can take max of ranking functions
  • In certain cases, can take sum of ranking functions
  • In certain cases, can take multiplication of ranking functions
• Generalize for system with more than 2 transitions
• May fail to find a symbolic bound