HARVARD

School of Engineering and Applied Sciences

# 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.,

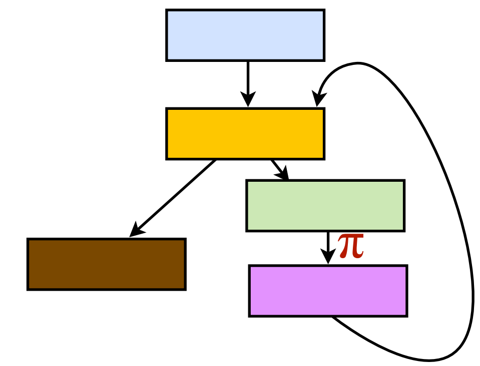
```
int i=0; while (i<n) { i++; j = i;
                       while (j<n) {j++; •}
         }
```

# Solution

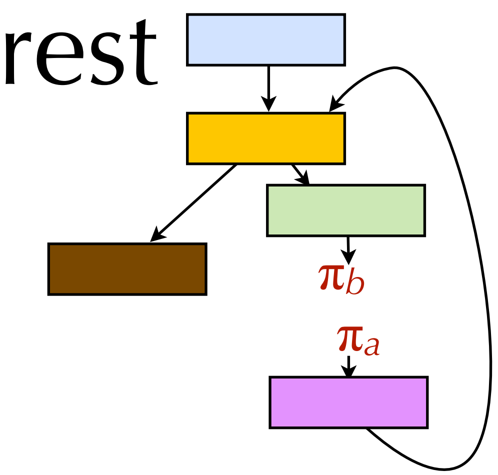
- Bound number of visits to program point $\pi$
- 1. Construct a disjunctive **transition system** that describes relationship of program variables in successive visits to $\pi$
- 2. Generate bounds from transition system using ranking functions.

# In more detail...

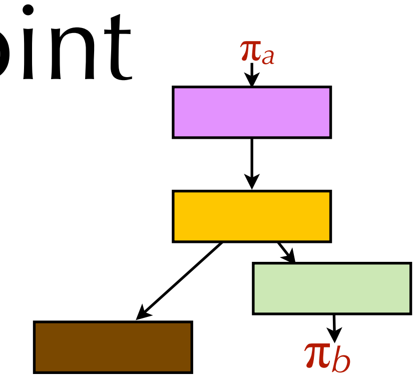
- Construct control flow-graph of procedure




- Split program point of interest




- Consider CFG between split program point




- Now construct transition system with regard to π

# 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

  $T(x,y,z,\ldots,x',y',z',\ldots)$

  such that if x,y,z take on values

  $v_1,v_2,v_3,\ldots$ and $w_1,w_2,w_3,\ldots$

  during consecutive visits to $\pi$ then

  $T(v_1,v_2,v_3,\ldots, w_1,w_2,w_3,\ldots)$ 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

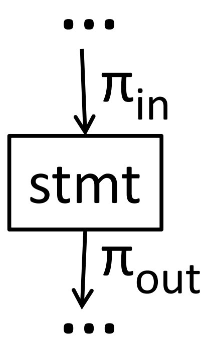
- Abstract interpretation
  - Domain is logical formula, ordering $\sqsubseteq$ is implication $\Rightarrow$
  - Join is disjunction
- Transition systems for atomic statements

$$
\begin{aligned}
\texttt{Translate}(x := e) &= (x' = e) \wedge (\bigwedge_{y \neq x} y' = y) \\
\texttt{Translate}(\texttt{Assume}(\texttt{guard})) &= \texttt{Id} \wedge \texttt{guard}
\end{aligned}
$$

# Composing transition functions

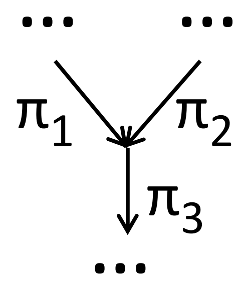
- Initial transition system is Id

(c) Compose




$F[\pi_{out}] = F[\pi_{in}] \circ \text{Translate(stmt)}$

(d) Merge




$F[\pi_3] = F[\pi_1] \vee F[\pi_2]$

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*

$$T \circ T' \stackrel{def}{=} \bigvee_{i,j} s_i \circ s'_j,$$

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

$$s_i(\vec{x}, \vec{x'}) \circ s'_j(\vec{x}, \vec{x'}) \stackrel{def}{=} \exists \vec{x''} \left( s_i[\vec{x''}/\vec{x'}] \wedge s'_j[\vec{x''}/\vec{x}] \right),$$

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

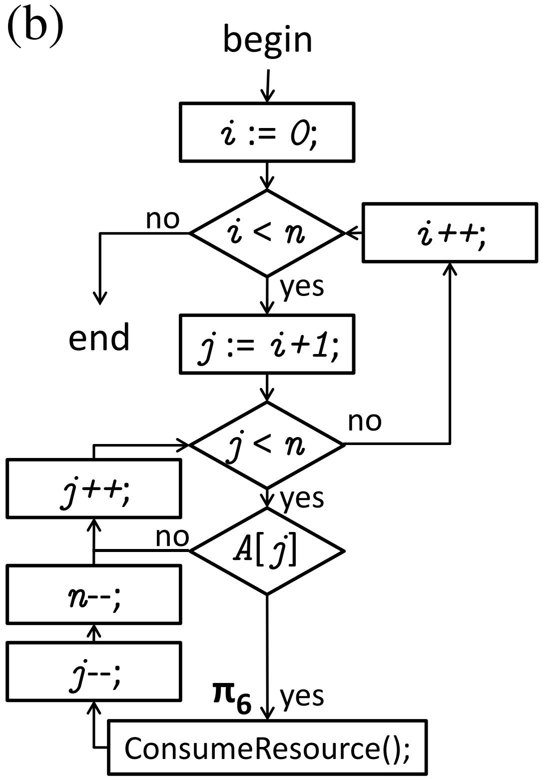
# Nested loops

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

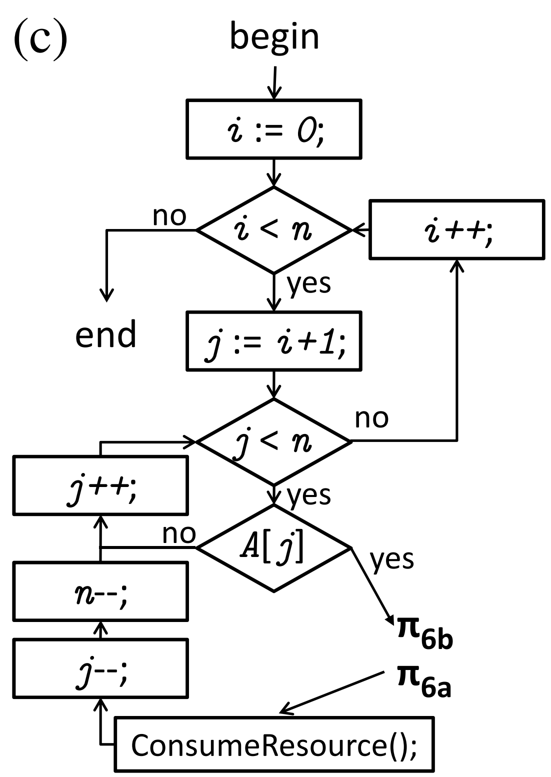
(a)

```
Ex1(uint n, bool[] A)
i := 0;
while (i < n)
    j := i + 1;
    while (j < n)
        if (A[j])
            ConsumeResource();
            j--;
            n--;
        j++;
    i++;
```

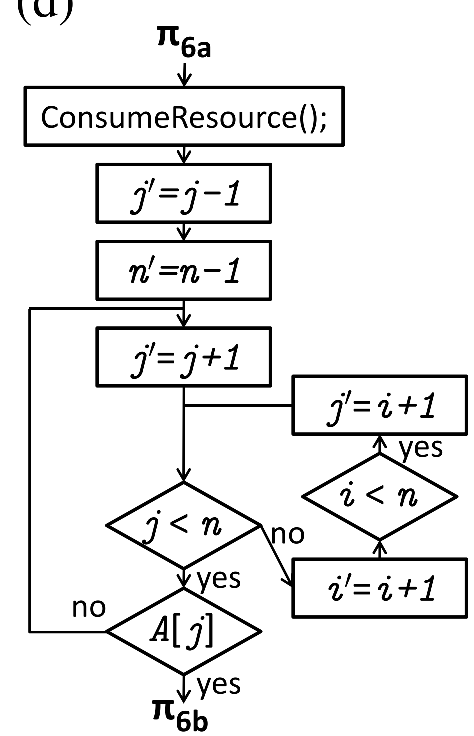
(b)




(c)




(d)



# 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

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*

$$Id \Rightarrow T' \qquad \textit{and} \qquad T' \circ T \Rightarrow T'$$

- 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 $G = g_1 \wedge \ldots \wedge g_n$
  - If $G \Rightarrow e_1 = e_2 \vee e_3 = e_4$ 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
    - $2 \leq x \leq 3 \Rightarrow x=2 \vee x=3$ 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 $\left(\phi \Rightarrow \left(\bigvee_i (x_i = y_i)\right)\right) \implies \left(\bigvee_i (\phi \Rightarrow (x_i = y_i))\right)$
- Suppose $\vee_{j\in 1..m}\, s'_j$ is transitive closure of $\vee_{i\in 1..n}\, s_i$
- Then $Id \Rightarrow \bigvee_{k=1}^{m} s'_k$ 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* $\delta \in \{1, .., m\}$, *a map* $\sigma : \{1, .., m\} \times \{1, .., n\} \mapsto \{1, .., m\}$, *such that for all* $i \in \{1, .., n\}$ *and* $j \in \{1, .., m\}$, *the following holds:*

$$Id \Rightarrow s'_\delta \qquad \textit{and} \qquad (s'_j \circ s_i) \Rightarrow s'_{\sigma(j,i)}$$

# Transitive closure

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

$$\texttt{for } j \in \{1,..,m\} - \{\delta\}\texttt{:}\ s'_j := \texttt{false;}$$
$$s'_\delta := \texttt{Id;}$$
$$\texttt{do } \{$$
$$\quad \texttt{for } i \in \{1,..,n\} \texttt{ and } j \in \{1,..,m\}\texttt{:}$$
$$\qquad s'_{\sigma(j,i)} := \texttt{Join}(s'_{\sigma(j,i)}, s'_j \circ s_i)$$
$$\} \texttt{ while any change in } \bigvee_{j=1}^{m} s'_j$$
$$\texttt{return } \bigvee_{j=1}^{m} s'_j\texttt{;}$$

- Notes:
  - Need a "convexity witness" (δ, σ)
  - 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?

$\texttt{ReachabilityBound}(\pi)$

$T := \texttt{GenerateTransitionSystem}(\pi)$;
$\mathcal{B} := 1 + \texttt{ComputeBound}(T)$;
$\texttt{return TranslateBound}(\mathcal{B}, \pi)$;

$\texttt{GenerateTransitionSystem}(\pi)$

$(\pi_a, \pi_b) := \texttt{Split}(\pi)$;
$\texttt{foreach top-level loop } L$:
  $\pi_L := \texttt{ location before header of } L$;
  $T := \texttt{GenerateTransitionSystem}(\pi_L)$;
  $T_c := \texttt{TransitiveClosure}(T)$;
  $\texttt{Insert Summary}(T_c) \texttt{ before header}$;
  $\texttt{Remove back-edges}$;
$\texttt{Initialize } F[\pi_a] \texttt{ to the transition system Id}$;
$\texttt{Propagate transitions } F \texttt{ using Merge/Compose rules}$;
$\texttt{return } F[\pi_b]$;

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

$\texttt{for } j \in \{1, .., m\} - \{\delta\}\texttt{: } s'_j := \texttt{false}$;
$s'_\delta := \texttt{Id}$;
$\texttt{do } \{$
  $\texttt{for } i \in \{1, .., n\} \texttt{ and } j \in \{1, .., m\}$:
    $s'_{\sigma(j,i)} := \texttt{Join}(s'_{\sigma(j,i)}, s'_j \circ s_i)$
$\} \texttt{ while any change in } \bigvee_{j=1}^{m} s'_j$
$\texttt{return } \bigvee_{j=1}^{m} s'_j$;

# 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* $\mathtt{Rank}(s, r)$.

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

$$\text{If } s \Rightarrow (e > 0 \;\wedge\; e[\vec{x'}/\vec{x}] < e), \text{ then } e \in \texttt{RankC}(s)$$

$$\text{If } s \Rightarrow (e \geq 1 \;\wedge\; e[\vec{x'}/\vec{x}] \leq e/2), \text{ then } \log e \in \texttt{RankC}(s)$$

- Boolean iteration patterns

$$\text{If } s \Rightarrow (e \;\wedge\; \neg(e[\vec{x'}/\vec{x}])), \text{ then } \texttt{Bool2Int}(e) \in \texttt{RankC}(s)$$

- …
- 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: $s_1 \vee s_2$
  - 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