Inferring Expected Runtimes of Probabilistic Programs

Jürgen Giesl

LuFG Informatik 2, RWTH Aachen University, Germany

joint work with Marcel Hark and Fabian Meyer
Termination analysis of imperative programs: ranking functions
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for **complexity analysis**
Termination analysis of imperative programs: ranking functions

Goal: use ranking functions for complexity analysis

Problem: complexity from combination of ranking functions
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for **complexity analysis**
- **Problem**: complexity from combination of ranking functions

```plaintext
while i > 0 do
  i = i - 1
done

while x > 0 do
  x = x - 1
done
```
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for complexity analysis
- **Problem**: complexity from combination of ranking functions

**Termination**: lexicographic combination of

\[
\begin{align*}
    f_1(x, i) &= i \\
    f_2(x, i) &= x
\end{align*}
\]

```
while i > 0 do
    i = i - 1
done
```

```
while x > 0 do
    x = x - 1
done
```
Termination analysis of imperative programs: ranking functions

Goal: use ranking functions for complexity analysis

Problem: complexity from combination of ranking functions

Termination: lexicographic combination of

\[ f_1(x, i) = i \]
\[ f_2(x, i) = x \]

while \( i > 0 \) do
\[ i = i - 1 \]
done

while \( x > 0 \) do
\[ x = x - 1 \]
done

Complexity: linear
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for **complexity analysis**
- **Problem**: complexity from combination of ranking functions

**Termination**: lexicographic combination of
\[
\begin{align*}
    f_1(x, i) &= i \\
    f_2(x, i) &= x
\end{align*}
\]

**Complexity**: linear

```
while i > 0 do
    x = x + i
    i = i - 1
done
```

```
while x > 0 do
    x = x - 1
done
```
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for **complexity analysis**
- **Problem**: complexity from **combination** of ranking functions

**Termination**:

\[ f_1(x, i) = i \]
\[ f_2(x, i) = x \]

**Complexity**: quadratic

while \( i > 0 \) do
  \( x = x + i \)
  \( i = i - 1 \)
done

while \( x > 0 \) do
  \( x = x - 1 \)
done
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for **complexity analysis**
- **Problem**: complexity from combination of ranking functions

**Termination**: lexicographic combination of

\[ f_1(x, i) = i \]
\[ f_2(x, i) = x \]

**Complexity**: quadratic

\[ i_0 + \text{"size"}(x) \]

```
while i > 0 do
    x = x + i
    i = i - 1
done
```

```
while x > 0 do
    x = x - 1
done
```
Complexity Analysis for Integer Programs

- **Termination analysis** of imperative programs: ranking functions
- **Goal**: use ranking functions for complexity analysis
- **Problem**: complexity from combination of ranking functions

**Termination**: lexicographic combination of
\[ f_1(x, i) = i \]
\[ f_2(x, i) = x \]

**Complexity**: quadratic
\[ i_0 + \text{"size"}(x) \]

**Solution**: modular approach which alternates between finding runtime and size bounds
Complexity Analysis for Probabilistic Programs

Probabilistic ranking functions

Expected value of ranking function must decrease by at least 1

\[
f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + \frac{1}{2} \cdot f_1(x, i) + 1
\]

Probabilistic ranking functions for each loop

\[
f_1(x, i) = 2 \cdot i
\]

\[
f_2(x, i) = x
\]

Expected runtime: quadratic

\[
2 \cdot i_0 + \text{"expected size"}(x)
\]

while \( i > 0 \) do
  \[
x = x + i
  i = i - 1
\]
done

while \( x > 0 \) do
  \[
x = x - 1
\]
done

Solution: modular approach which alternates between finding runtime and size bounds
Complexity Analysis for Probabilistic Programs

Probabilistic ranking functions

Expected value of ranking function must decrease by at least 1

\[ f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + 1 \]

Probabilistic ranking functions for each loop

\[ f_1(x, i) = 2 \cdot i \]

\[ f_2(x, i) = x \]

Expected runtime: quadratic

\[ 2 \cdot i + \text{"expected size"}(x) \]

```
while i > 0 do
    \{ x = x + i \} [\frac{1}{2}] \{ x = x \}
    i = i - 1
done

while x > 0 do
    x = x - 1
done
```
Complexity Analysis for Probabilistic Programs

- Probabilistic ranking functions

\[
\begin{align*}
\text{while } i > 0 & \text{ do} \\
\{ & x = x + i \\
& i = i - 1 \} \left[ \frac{1}{2} \right] \{ & x = x \\
& i = i \} \\
\text{done} \\
\text{while } x > 0 & \text{ do} \\
& x = x - 1 \\
\text{done}
\end{align*}
\]
Complexity Analysis for Probabilistic Programs

- **Probabilistic** ranking functions
- **Expected** value of ranking function must decrease by at least 1

```
while i > 0 do
    \[
    \begin{cases}
    x = x + i \\
    i = i - 1
    \end{cases}
    \]
    \[
    \left[\frac{1}{2}\right]
    \begin{cases}
    x = x \\
    i = i
    \end{cases}
    \]
done

while x > 0 do
    x = x - 1
done
```
Complexity Analysis for Probabilistic Programs

- **Probabilistic** ranking functions
- **Expected** value of ranking function must decrease by at least 1

**Probabilistic ranking functions** for each loop

\[
\begin{align*}
  f_1(x, i) &= 2 \cdot i \\
  f_2(x, i) &= x
\end{align*}
\]

```plaintext
while i > 0 do
  \{ x = x + i \} [\frac{1}{2}] \{ x = x \} \i
  i = i - 1
\}
done

while x > 0 do
  x = x - 1
done
```
Complexity Analysis for Probabilistic Programs

- **Probabilistic ranking functions**

- **Expected** value of ranking function must decrease by at least 1

\[
f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + \frac{1}{2} \cdot f_1(x, i) + 1
\]

**Probabilistic ranking functions** for each loop

\[
f_1(x, i) = 2 \cdot i \\
f_2(x, i) = x
\]

**while** \( i > 0 \) **do**

\[
\begin{align*}
x &= x + i \\
i &= i - 1
\end{align*}
\]

\( \left\lfloor \frac{1}{2} \right\rfloor \)

**done**

**while** \( x > 0 \) **do**

\[
\begin{align*}
x &= x - 1
\end{align*}
\]

**done**
Complexity Analysis for Probabilistic Programs

- **Probabilistic ranking functions**

- **Expected** value of ranking function must decrease by at least 1

\[ f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + \frac{1}{2} \cdot f_1(x, i) + 1 \]

**Probabilistic ranking functions** for each loop

\[ f_1(x, i) = 2 \cdot i \]
\[ f_2(x, i) = x \]

**Expected runtime:** quadratic

\[
\text{while } i > 0 \text{ do }
\{
\begin{align*}
   x &= x + i \\
   i &= i - 1
\end{align*}
\}
\quad \left[ \frac{1}{2} \right]
\{
\begin{align*}
   x &= x \\
   i &= i
\end{align*}
\}\]
\text{done}

\[
\text{while } x > 0 \text{ do }
\{ x = x - 1 \}
\text{done} \]
Complexity Analysis for Probabilistic Programs

- **Probabilistic ranking functions**

- **Expected** value of ranking function must decrease by at least 1

\[ f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + \frac{1}{2} \cdot f_1(x, i) + 1 \]

**Probabilistic ranking functions** for each loop

\[ f_1(x, i) = 2 \cdot i \]
\[ f_2(x, i) = x \]

**Expected runtime**: quadratic

\[ 2 \cdot i_0 + \text{“expected size”}(x) \]

```
while i > 0 do
    \{ x = x + i \} \quad \left\{ \begin{array}{l}
        x = x \\
        i = i - 1
    \end{array} \right\}
\quad \left[ \begin{array}{l}
        [\frac{1}{2}]
    \end{array} \right]
\quad \left\{ \begin{array}{l}
        x = x \\
        i = i
    \end{array} \right\}
\quad \text{done}
```

```
while x > 0 do
    x = x - 1
\quad \text{done}
```
Complexity Analysis for Probabilistic Programs

- **Probabilistic ranking functions**

- **Expected** value of ranking function must decrease by at least 1
  \[
  f_1(x, i) \geq \frac{1}{2} \cdot f_1(x + i, i - 1) + \frac{1}{2} \cdot f_1(x, i) + 1
  \]

**Probabilistic ranking functions** for each loop
- \( f_1(x, i) = 2 \cdot i \)
- \( f_2(x, i) = x \)

**Expected runtime**: quadratic
- \( 2 \cdot i_0 + \text{“expected size”}(x) \)

**Solution**: modular approach which alternates between finding \textit{runtime} and \textit{size} bounds
Runtime and Size Bounds

while \( i > 0 \) do
\[
\begin{align*}
\{ & x = x + i \\
& i = i - 1 \}
\end{align*}
\]
\[
\begin{align*}
\{ & x = x \\
& i = i \}
\end{align*}
\]
done

while \( x > 0 \) do
\[
\begin{align*}
x &= x - 1
\end{align*}
\]
done
Runtime and Size Bounds

while $i > 0$ do
  \{
  x = x + i
  \}
  \frac{1}{2}
  \{
  x = x
  \}
  \{ i = i - 1 \}
  \frac{1}{2}
  \{
  i = i
  \}

done

while $x > 0$ do
  $x = x - 1$

done

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

Runtime bound $R(t)$:
- $R(t_0) = 1$
- $R(t_1) = i_0$
- $R(t_2) = x_0 + i_2$

Size bound $S(t, v)$:
- $S(t_1, x) = x_0 + i_2$

Overall runtime is bounded by $R(t_0) + \ldots + R(t_3) = 1 + i_0 + 1 + x_0 + i_2$. 
Runtime and Size Bounds

while $i > 0$ do
    $x = x + i$
    $i = i - 1$
done

while $x > 0$ do
    $x = x - 1$
done
**Goal:** find complexity bounds w.r.t. the sizes (absolute values) of the input variables.

Overall runtime is bounded by

\[ R(t_0) + \ldots + R(t_3) = 1 + i_0 + 1 + x_0 + i_2. \]
Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $R(t)$: bound on number of times that transition $t$ occurs in executions

```
while i > 0 do
  x = x + i
  i = i - 1
while x > 0 do
  x = x - 1
```

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $R(t)$: bound on number of times that transition $t$ occurs in executions

```
while i > 0 do
  x = x + i
  i = i - 1
while x > 0 do
  x = x - 1
```

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $R(t)$: bound on number of times that transition $t$ occurs in executions

```
while i > 0 do
  x = x + i
  i = i - 1
while x > 0 do
  x = x - 1
```

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $R(t)$: bound on number of times that transition $t$ occurs in executions

```
while i > 0 do
  x = x + i
  i = i - 1
while x > 0 do
  x = x - 1
```
**Goal**: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound** $R(t)$:
  bound on number of times that transition $t$ occurs in executions

  $R(t_0) = 1$

---

**Goal**

- **Runtime and Size Bounds**

```plaintext
while i > 0 do
  x = x + i
  i = i - 1

end
while x > 0 do
  x = x - 1
end
```

**Goal**:

- **Runtime bound** $R(t)$:
  - $R(t_0) = 1$
  - $R(t_1) = i_0$
  - $R(t_2) = x_0 + i_2$

- **Size bound** $S(t, v)$:
  - $S(t_1, x) = x_0 + i_2$

**Overall runtime is bounded by**

$$R(t_0) + \ldots + R(t_3) = 1 + i_0 + 1 + x_0 + i_2.$$
Runtime and Size Bounds

**Goal:** find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound $\mathcal{R}(t)$:**
  - bound on number of times that transition $t$ occurs in executions
  - $\mathcal{R}(t_0) = 1$
  - $\mathcal{R}(t_1) = i_0$

---

```
while i > 0 do
  x = x + i
  i = i - 1
while x > 0 do
  x = x - 1
```

---

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables.
Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $\mathcal{R}(t)$:
  bound on number of times that transition $t$ occurs in executions
  
  $\mathcal{R}(t_0) = 1$  
  $\mathcal{R}(t_1) = i_0$  
  $\mathcal{R}(t_2) = 1$  
  $\mathcal{R}(t_3) = x_0 + i_2$

Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- Runtime bound $\mathcal{R}(t)$:
  bound on number of times that transition $t$ occurs in executions
  
  $\mathcal{R}(t_0) = 1$  
  $\mathcal{R}(t_1) = i_0$  
  $\mathcal{R}(t_2) = 1$  
  $\mathcal{R}(t_3) = x_0 + i_2$

Overall runtime is bounded by $\mathcal{R}(t_0) + \ldots + \mathcal{R}(t_3) = 1 + i_0 + 1 + x_0 + i_2$. 

The diagram shows a flowchart with transitions $t_0, t_1, t_2, t_3$ and conditions for each transition: 
- $t_0$: if $i > 0$, then $x = x + i$, $i = i - 1$.
- $t_1$: if $i \leq 0$, then $t_1$.
- $t_2$: if $x > 0$, then $x = x - 1$.
Runtime and Size Bounds

**Goal:** find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound** $R(t)$:
  
  bound on number of times that transition $t$ occurs in executions

  $R(t_0) = 1$
  
  $R(t_1) = i_0$
  
  $R(t_2) = 1$
  
  $R(t_3) = x_0 + i_0^2$

```plaintext
while i > 0 do
    x = x + i
    i = i - 1
while x > 0 do
    x = x - 1
```

- **Size bound** $S(t, v)$:
  
  bound on size of $v$ after using transition $t$ in program executions

  $S(t_1, x) = x_0 + i_0^2$

**Overall runtime** is bounded by

$R(t_0) + \ldots + R(t_3) = 1 + i_0 + 1 + x_0 + i_0^2$. 

![Diagram]
Runtime and Size Bounds

**Goal:** find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound** $\mathcal{R}(t)$:
  - bound on number of times that transition $t$ occurs in executions
  \[
  \mathcal{R}(t_0) = 1, \quad \mathcal{R}(t_1) = i_0, \quad \mathcal{R}(t_2) = 1, \quad \mathcal{R}(t_3) = x_0 + i_0^2
  \]

- **Size bound** $\mathcal{S}(t, v)$:
  - bound on size of $v$ after using transition $t$ in program executions

![Diagram](image)
Goal: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound \( R(t) \):**
  bound on number of times that transition \( t \) occurs in executions
  
  \[
  R(t_0) = 1 \quad R(t_2) = 1 \\
  R(t_1) = i_0 \quad R(t_3) = x_0 + i_0^2
  \]

- **Size bound \( S(t, v) \):**
  bound on size of \( v \) after using transition \( t \) in program executions
  
  e.g., \( S(t_1, x) = x_0 + i_0^2 \)
Runtime and Size Bounds

**Goal**: find complexity bounds w.r.t. the sizes (absolute values) of the input variables

- **Runtime bound** $\mathcal{R}(t)$:
  bound on number of times that transition $t$ occurs in executions

  $\mathcal{R}(t_0) = 1$  
  $\mathcal{R}(t_1) = i_0$  
  $\mathcal{R}(t_2) = 1$  
  $\mathcal{R}(t_3) = x_0 + i_0^2$

- **Size bound** $\mathcal{S}(t, v)$:
  bound on size of $v$ after using transition $t$ in program executions
  
  e.g., $\mathcal{S}(t_1, x) = x_0 + i_0^2$

Overall runtime is bounded by $\mathcal{R}(t_0) + \ldots + \mathcal{R}(t_3) = 1 + i_0 + 1 + x_0 + i_0^2$. 
Runtime Bounds from Ranking Functions

Initial bounds

$R(t_0) = 1, \ R(t_2) = 1$ as $t_0$ and $t_2$ are not in loops

Graph:
- $\ell_0 \rightarrow \ell_1 \rightarrow \ell_2 \rightarrow \ell_0$
- $t_0$ to $\ell_1$
- $t_1$ if $i > 0$
- $x = x + i$
- $i = i - 1$
- $t_2$ if $i \leq 0$
- $t_3$ if $x > 0$
- $x = x - 1$
Initial bounds

\( R(t_0) = 1, R(t_2) = 1 \) as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( r \) for program \( P \)

\[ r(\ell) = i \] for all locations \( \ell \)

Thus:

\( t_1 \)

\( \text{if}(i > 0) \)
\( x = x + i \)
\( i = i - 1 \)

\( t_2 \)

\( \text{if}(i \leq 0) \)

\( t_3 \)

\( \text{if}(x > 0) \)
\( x = x - 1 \)
Initial bounds
\[ R(t_0) = 1, \ R(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( \tau \) for program \( \mathcal{P} \)
- \( \tau \) maps locations to \( \mathbb{R}[v_1, \ldots, v_n] \)

```
t_1 := if (i > 0)
x := x + i
i := i - 1

\text{if}(i \leq 0)
t_2

\text{if}(x > 0)
x := x - 1

\text{if}\ \text{else}
```

Thus:
\[ t_1 \in \mathcal{P} \succ \]
Initial bounds
\[ R(t_0) = 1, \ R(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( r \) for program \( P \)
- \( r \) maps \( \text{locations} \) to \( \mathbb{R}[v_1, \ldots, v_n] \)
- Non-Increase: no transition in \( P \) increases value of \( r \)

Diagram:
- \( \ell_0 \) to \( t_0 \)
  - if \( (i > 0) \)
    - \( x = x + i \)
    - \( i = i - 1 \)
- \( \ell_1 \) to \( t_1 \)
- \( \ell_2 \) to \( t_2 \)
  - if \( (i \leq 0) \)
- \( \ell_3 \) to \( t_3 \)
  - if \( (x > 0) \)
    - \( x = x - 1 \)
Initial bounds
\[ R(t_0) = 1, \quad R(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( r \) for program \( P \)
- \( r \) maps locations to \( \mathbb{R}[v_1, \ldots, v_n] \)
- **Non-Increase:** no transition in \( P \) increases value of \( r \)
- **Decrease:** value of \( r \) decreases by at least 1 for \( P_{\succ} \subseteq P \)
Runtime Bounds from Ranking Functions

**Initial bounds**
\[ R(t_0) = 1, \ R(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

**Ranking function \( r \) for program \( P \)**
- \( r \) maps *locations* to \( \mathbb{R}[v_1, \ldots, v_n] \)
- **Non-Increase:** no transition in \( P \) increases value of \( r \)
- **Decrease:** value of \( r \) decreases by at least 1 for \( P_\succ \subseteq P \)
- **Boundedness:** \( r \geq 0 \) after \( P_\succ \subseteq P \)
Runtime Bounds from Ranking Functions

**Initial bounds**

\[ \mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

**Ranking function \( \tau \) for program \( \mathcal{P} \)**

- \( \tau \) maps locations to \( \mathbb{R}[v_1, \ldots, v_n] \)
- **Non-Increase**: no transition in \( \mathcal{P} \) increases value of \( \tau \)
- **Decrease**: value of \( \tau \) decreases by at least 1 for \( \mathcal{P}_\succ \subseteq \mathcal{P} \)
- **Boundedness**: \( \tau \geq 0 \) after \( \mathcal{P}_\succ \subseteq \mathcal{P} \)

- \( \tau(\ell) = i \) for all locations \( \ell \)

---

**Diagram**

- \( \mathcal{L}_0 \)
- \( \mathcal{L}_1 \)
- \( \mathcal{L}_2 \)
- \( t_0 \)
- \( t_1 \) if \( (i > 0) \)
- \( x = x + i \)
- \( i = i - 1 \)
- \( t_2 \) if \( (i \leq 0) \)
- \( t_3 \) if \( (x > 0) \)
- \( x = x - 1 \)
Runtime Bounds from Ranking Functions

Initial bounds
\( \mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1 \) as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( r \) for program \( \mathcal{P} \)
- \( r \) maps locations to \( \mathbb{R}[v_1, \ldots, v_n] \)
- **Non-Increase**: no transition in \( \mathcal{P} \) increases value of \( r \)
- **Decrease**: value of \( r \) decreases by at least 1 for \( \mathcal{P}_\succ \subseteq \mathcal{P} \)
- **Boundedness**: \( r \geq 0 \) after \( \mathcal{P}_\succ \subseteq \mathcal{P} \)

- \( r(\ell) = i \) for all locations \( \ell \)
- Thus: \( t_1 \in \mathcal{P}_\succ \)
Runtime Bounds from Ranking Functions

Initial bounds

\[ R(t_0) = 1, \ R(t_2) = 1 \] as \( t_0 \) and \( t_2 \) are not in loops

Ranking function \( r \) for program \( P \)

- for all \( t \in P_{\succ} \), set \( R(t) = r(l_0) \)
- **Non-Increase**: no transition in \( P \) increases value of \( r \)
- **Decrease**: value of \( r \) decreases by at least 1 for \( P_{\succ} \subseteq P \)
- **Boundedness**: \( r \geq 0 \) after \( P_{\succ} \subseteq P \)

- \( r(l) = i \) for all locations \( l \)
- Thus: \( t_1 \in P_{\succ} \)
Runtime Bounds from Ranking Functions

Initial bounds
\[ \mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1, \mathcal{R}(t_1) = i_0 \]

Ranking function \( r \) for program \( P \)
- for all \( t \in P \), set \( \mathcal{R}(t) = r(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases value of \( r \)
- **Decrease**: value of \( r \) decreases by at least 1 for \( P \preceq \subseteq P \)
- **Boundedness**: \( r \geq 0 \) after \( P \preceq \subseteq P \)
  - \( r(\ell) = i \) for all locations \( \ell \)
  - Thus: \( t_1 \in P \preceq \)
Current bounds
\( R(t_0) = 1, R(t_2) = 1, R(t_1) = i_0 \)
Modular Runtime Bounds from Ranking Functions

Current bounds
\( R(t_0) = 1, R(t_2) = 1, R(t_1) = i_0 \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)

- Diagram:
  - \( \ell_0 \) (start state)
  - \( t_0 \) to \( \ell_1 \)
    - \( t_1 \) if \( i > 0 \)
      - \( x = x + i \)
      - \( i = i - 1 \)
  - \( \ell_1 \) to \( \ell_2 \)
    - \( t_2 \) if \( i \leq 0 \)
  - \( \ell_2 \) to \( \ell_1 \)
    - \( t_3 \) if \( x > 0 \)
      - \( x = x - 1 \)
Modular Runtime Bounds from Ranking Functions

Current bounds

\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0 \]

Computing runtime bound for \( t \in \mathcal{P}' \)

\[ R(t) = r(\ell) \]

\( \ell \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{t_3\} \)
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0 \]

Computing runtime bound for \( t \in P' \)
\[ R(t) = r(\ell) \]
- \( \ell \): entry location of \( P' \)

- Modular use of ranking function for subset \( P' = \{t_3\} \)
- \( r(\ell_2) = x \)

Diagram:
- If \( i > 0 \):
  - \( x = x + i \)
  - \( i = i - 1 \)
- If \( i \leq 0 \):
  - \( t_3 \)
  - \( x = x - 1 \)
Modular Runtime Bounds from Ranking Functions

Current bounds
\( R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0 \)

Computing runtime bound for \( t \in \mathcal{P}' \)
\[
R(t) = r(\ell)
\]

- \( \ell \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{t_3\} \)

- \( r(\ell_2) = x \)
  
  Thus: \( t_3 \in \mathcal{P}' \)

Diagram:
- \( \ell_0 \) to \( t_0 \)
- \( t_1 \) if \( i > 0 \) then \( x = x + i \), \( i = i - 1 \)
- \( t_2 \) if \( i \leq 0 \) then \( t_3 \)
- \( t_3 \) if \( x > 0 \) then \( x = x - 1 \)
Modular Runtime Bounds from Ranking Functions

Current bounds

\[ R(t_0) = 1, R(t_2) = 1, R(t_1) = i_0, R(t_3) = r(l_2) \]

Computing runtime bound for \( t \in \mathcal{P}' \)

\[ R(t) = r(l) \]

- \( l \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{t_3\} \)

- \( r(l_2) = x \)

Thus: \( t_3 \in \mathcal{P}' \)

![Graph showing the modular runtime bounds and ranking function calculations](image-url)
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = r(\ell_2) \]

Computing runtime bound for \( t \in \mathcal{P}' \)

\[ R(t) = r(\ell) \]

- \( \ell \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)

- \( r(\ell_2) = x \) Thus: \( t_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

Diagram:
- \( \ell_0 \) to \( t_0 \)
- \( t_1 \) if \( i > 0 \)
  - \( x = x + i \)
  - \( i = i - 1 \)
- \( \ell_1 \) to \( t_2 \)
- \( t_2 \) if \( i \leq 0 \)
  - \( t_3 \)
  - \( x > 0 \)
  - \( x = x - 1 \)
- \( \ell_2 \) to \( t_3 \)
Modular Runtime Bounds from Ranking Functions

**Current bounds**
\[ R(t_0) = 1, \quad R(t_2) = 1, \quad R(t_1) = i_0, \quad R(t_3) = r(\ell_2) \]

**Computing runtime bound for** \( t \in P' \)
\[ R(t) = r(\ell) \]
- \( \ell \): entry location of \( P' \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( r(\ell_2) = x \) \quad Thus: \( t_3 \in P'_\leq \)
- Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.
- For global result:
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ \mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1, \mathcal{R}(t_1) = i_0, \mathcal{R}(t_3) = r(\ell_2) \]

Computing runtime bound for \( t \in \mathcal{P}' \)

\[
\mathcal{R}(t) = r(\ell)
\]

- \( \ell \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{t_3\} \)

- \( r(\ell_2) = x \)

- Thus: \( t_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = r(l_2) \]

Computing runtime bound for \( t \in P' \)

\[ R(t) = r(\ell) \]

- \( \ell \): entry location of \( P' \)

- Modular use of ranking function for subset \( P' = \{t_3\} \)

- \( t(l_2) = x \)
  
  Thus: \( t_3 \in P'_x \)

- Executions of \( P' \) starting in \( l_2 \) use \( t_3 \) at most \( r(l_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))

\[ \Rightarrow \] multiply \( t_2 \)'s runtime bound \( R(t_2) \) with local bound \( r(l_2) \)
Modular Runtime Bounds from Ranking Functions

Current bounds

\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = R(t_2) \cdot r(\ell_2) \]

Computing runtime bound for \( t \in \mathcal{P}' \)

% \[ R(t) = r(\ell) \]

- \( \ell \): entry location of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)

- \( r(\ell_2) = x \)

Thus: \( t_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))

⇒ multiply \( t_2 \)'s runtime bound \( R(t_2) \) with local bound \( r(\ell_2) \)
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = R(t_2) \cdot r(\ell_2) \]

Computing runtime bound for \( t \in P' \)
\[ R(t) = R(t') \cdot r(\ell) \]
- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)

- \( r(\ell_2) = x \)
  - Thus: \( t_3 \in P'_\perp \)

- Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))

\[ \Rightarrow \text{multiply } t_2 \text{'s runtime bound } R(t_2) \text{ with local bound } r(\ell_2) \]
Modular Runtime Bounds from Ranking Functions

Current bounds

\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = R(t_2) \cdot r(\ell_2) \]

Computing runtime bound for \( t \in P' \)

\[ R(t) = R(t') \cdot r(\ell) \]

- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( r(\ell_2) = x \)  
  Thus: \( t_3 \in P'_\ell \)

- Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider value of \( P'' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{multiply } t_2 \text{'s runtime bound } R(t_2) \text{ with local bound } r(\ell_2) \]

\[ t_1 \]
\[ \text{if}(i > 0) \]
\[ x = x + i \]
\[ i = i - 1 \]

\[ t_2 \]
\[ \text{if}(i \leq 0) \]

\[ t_3 \]
\[ \text{if}(x > 0) \]
\[ x = x - 1 \]
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = R(t_2) \cdot r(\ell_2) \]

Computing runtime bound for \( t \in P' \)

\[ R(t) = R(t') \cdot r(\ell) \]

- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( r(\ell_2) = x \)
  
  Thus: \( t_3 \in P'_x \)

- Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider value of \( P'' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } r(\ell_2) \text{ by } r(\ell_2)[x/S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, R(t_2) = 1, R(t_1) = i_0, R(t_3) = R(t_2) \cdot r(l_2) [x / S(t_2, x)] \]

Computing runtime bound for \( t \in P' \)
\[ R(t) = R(t') \cdot r(l) \]
- \( l \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( r(l_2) = x \)
  Thus: \( t_3 \in P'_\prec \)

Executions of \( P' \) starting in \( l_2 \) use \( t_3 \) at most \( r(l_2) = x \) times.

For global result:
- consider how often \( P' \) is reached (by \( t_2 \))
- consider value of \( P' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } r(l_2) \text{ by } r(l_2) [x / S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = R(t_2) \cdot r(\ell_2)[x / S(t_2, x)] \]

Computing runtime bound for \( t \in P' \)
\[ R(t) = R(t') \cdot r(\ell)[v / S(t', v)] \]

- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

Modular use of ranking function for subset \( P' = \{ t_3 \} \)

- \( t(\ell_2) = x \)
  
  Thus: \( t_3 \in P' \)

Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

For global result:
- consider how often \( P' \) is reached (by \( t_2 \))
- consider value of \( P'' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(\ell_2) \text{ by } t(\ell_2)[x / S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

Current bounds
\[
\mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1, \mathcal{R}(t_1) = i_0, \mathcal{R}(t_3) = 1 \cdot r(l_2)[x / S(t_2, x)]
\]

Computing runtime bound for \( t \in \mathcal{P}' \)
\[
\mathcal{R}(t) = \mathcal{R}(t') \cdot r(l)[v / S(t', v)]
\]

- \( l \): entry location of \( \mathcal{P}' \)
- \( t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)
- \( t(l_2) = x \)
  
  Thus: \( t_3 \in \mathcal{P'} \)
- Executions of \( \mathcal{P}' \) starting in \( l_2 \) use \( t_3 \) at most \( r(l_2) = x \) times.
- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
  - consider value of \( \mathcal{P}' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(l_2) \text{ by } t(l_2)[x / S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

Current bounds

\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = 1 \times x \ [x / S(t_2, x)] \]

Computing runtime bound for \( t \in \mathcal{P}' \)

\[
R(t) = R(t') \cdot r(\ell)[v / S(t', v)]
\]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)
- \( r(\ell_2) = x \)  
  Thus: \( t_3 \in \mathcal{P}' \)
- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.
- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
  - consider value of \( \mathcal{P}' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } r(\ell_2) \text{ by } r(\ell_2)[x / S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

**Current bounds**

\[ R(t_0) = 1, \quad R(t_2) = 1, \quad R(t_1) = 1, \quad R(t_3) = 1 \cdot S(t_2, x) \]

**Computing runtime bound for \( t \in P' \)**

\[ R(t) = R(t') \cdot r(\ell)[v / S(t', v)] \]

- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( r(\ell_2) = x \)  
  Thus: \( t_3 \in P' \)
- Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider value of \( P'' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } r(\ell_2) \text{ by } r(\ell_2)[x / S(t_2, x)] \]
Modular Runtime Bounds from Ranking Functions

Current bounds
\[ R(t_0) = 1, \quad R(t_2) = 1, \quad R(t_1) = i_0, \quad R(t_3) = 1 \cdot (x_0 + i_0^2) \]

Computing runtime bound for \( t \in P' \)
\[ R(t) = R(t') \cdot r(\ell)[v/S(t', v)] \]
- \( \ell \): entry location of \( P' \)
- \( t' \): pre-transition of \( P' \)

Modular use of ranking function for subset \( P' = \{ t_3 \} \)
- \( t(\ell_2) = x \)
  Thus: \( t_3 \in P'_\leq \)

Executions of \( P' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

For global result:
- consider how often \( P' \) is reached (by \( t_2 \))
- consider value of \( P' \)'s initial variable \( x \) in full run

⇒ replace \( t(\ell_2) \) by \( t(\ell_2)[x/S(t_2, x)] \)
Modular Runtime Bounds from Ranking Functions

**Runtime bounds**

\[ \mathcal{R}(t_0) = 1, \mathcal{R}(t_2) = 1, \mathcal{R}(t_1) = i_0, \mathcal{R}(t_3) = x_0 + i_0^2 \]

**Computing runtime bound for** \( t \in \mathcal{P}' \)

\[ \mathcal{R}(t) = \mathcal{R}(t') \cdot r(\ell)[v / S(t', v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( t' \): pre-transition of \( \mathcal{P}' \)

use size bounds to compute runtime bounds

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)
  
  \[ r(\ell_2) = x \]
  
  Thus: \( t_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
  - consider value of \( \mathcal{P}' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } r(\ell_2) \text{ by } r(\ell_2)[x / S(t_2, x)] \]
**Modular Runtime Bounds from Ranking Functions**

### Runtime bounds

\[ R(t_0) = 1, \ R(t_2) = 1, \ R(t_1) = i_0, \ R(t_3) = x_0 + i_0^2 \]

### Computing runtime bound for \( t \in \mathcal{P}' \)

\[ R(t) = R(t') \cdot r(\ell)[v / S(t', v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( t' \): pre-transition of \( \mathcal{P}' \)

**Use size bounds to compute runtime bounds**

- Modular use of ranking function for subset \( \mathcal{P}' = \{ t_3 \} \)
- \( r(\ell_2) = x \)  
  Thus: \( t_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( t_3 \) at most \( r(\ell_2) = x \) times.

- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
  - consider value of \( \mathcal{P}' \)’s initial variable \( x \) in full run

### Overall runtime

Overall runtime is bounded by

\[ R(t_0) + \ldots + R(t_3) = 1 + i_0 + 1 + x_0 + i_0^2. \]
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds $R_E(g_0) = 1$, $R_E(g_2) = 1$ as $g_0$ and $g_2$ are not in loops

Probabilistic ranking function $r$ for program $P$

Non-Increase: no transition in $P$ increases expected value of $r$

Decrease: expected value of $r$ decreases by 1 for $P ≻ ⊆ P$

Boundedness: $r \geq 0$ after $P ≻ ⊆ P$

$\ell_0$

$t_0$

$t_1$

if $(i > 0)$

$x = x + i$

$i = i - 1$

$t_2$

if $(i \leq 0)$

$t_3$

if $(x > 0)$

$x = x - 1$
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds

\[
R_E(g_0) = 1, \quad R_E(g_2) = 1 \quad \text{as } g_0 \text{ and } g_2 \text{ are not in loops}
\]

Probabilistic ranking function \( r \) for program \( P \)

- Non-Increase: no transition in \( P \) increases expected value of \( r \)
- Decrease: expected value of \( r \) decreases by 1 for \( P \succ \subseteq P \)
- Boundedness: \( r \geq 0 \) after \( P \succ \subseteq P \)

\[
\ell_0 \\
t_0 \in g_0
\]

\[
\ell_1 \\
\frac{1}{2} : t_1 \in g_1 \quad \text{if} (i > 0) \\
x = x + i \\
i = i - 1
\]

\[
\ell_2 \\
\frac{1}{2} : t_2 \in g_2 \quad \text{if} (i \leq 0) \\
t_2 \in g_2
\]

\[
\ell_3 \\
t_3 \in g_3 \quad \text{if} (x > 0) \\
x = x - 1
\]

\[
\ell_4 \\
\frac{1}{2} : t_4 \in g_1 \quad \text{if} (i > 0) \\
2i \geq 1 \\
2i \geq \frac{2}{2} \cdot r(\ell_1) [x/x + i, i/i - 1] \\
2i + 1
\]
**Initial bounds**

\( \mathcal{R}_E(g_0) = 1, \mathcal{R}_E(g_2) = 1 \) as \( g_0 \) and \( g_2 \) are not in loops.

**Diagram**

- \( \ell_0 \) with \( t_0 \) belonging to \( g_0 \)
- \( \ell_1 \) with conditions:
  - \( \frac{1}{2} : t_1 \in g_1 \)
  - \( \text{if}(i > 0) \)
  - \( x = x + i \)
  - \( i = i - 1 \)
- \( \ell_2 \) with condition:
  - \( \text{if}(i \leq 0) \)
  - \( t_2 \in g_2 \)
- \( \ell_3 \) with conditions:
  - \( t_3 \in g_3 \)
  - \( \text{if}(x > 0) \)
  - \( x = x - 1 \)
- \( \ell_4 \) with condition:
  - \( \frac{1}{2} : t_4 \in g_1 \)
  - \( \text{if}(i > 0) \)
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds
$$R_E(g_0) = 1, R_E(g_2) = 1$$ as $g_0$ and $g_2$ are not in loops

Probabilistic ranking function $r$ for program $P$
- $r$ maps locations to $\mathbb{R}[v_1, \ldots, v_n]$
- **Non-Increase**: no transition in $P$ increases value of $r$
- **Decrease**: value of $r$ decreases by 1 for $P \succ P$
- **Boundedness**: $r \geq 0$ after $P \succ P$

Diagram:
- $\ell_0$: $t_0 \in g_0$
- $\ell_1$: $t_1 \in g_1$
  - $\frac{1}{2} : t_1 \in g_1$
  - $\text{if} (i > 0)$
  - $x = x + i$
  - $i = i - 1$
- $\ell_2$: $t_2 \in g_2$
  - $\frac{1}{2} : t_4 \in g_1$
  - $\text{if} (i > 0)$
  - $t_2 \in g_2$
  - $\text{if} (i \leq 0)$
- $\ell_3$: $t_3 \in g_3$
  - $\text{if} (x > 0)$
  - $x = x - 1$
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds
\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

Probabilistic ranking function \( \tau \) for program \( \mathcal{P} \)
- \( \tau \) maps locations to \( \mathbb{R}[v_1, \ldots, v_n] \)
- **Non-Increase**: no transition in \( \mathcal{P} \) increases value of \( \tau \)
- **Decrease**: value of \( \tau \) decreases by 1 for \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)
- **Boundedness**: \( \tau \geq 0 \) after \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)

---

\[
\begin{align*}
R_E(g_0) &= 1, \\
R_E(g_2) &= 1
\end{align*}
\]
as \( g_0 \) and \( g_2 \) are not in loops

\[
E(g_0) = 1, \ E(g_2) = 1
\]

**Non-Increase**
- No transition in \( \mathcal{P} \) increases value of \( \tau \)

**Decrease**
- Value of \( \tau \) decreases by 1 for \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)

**Boundedness**
- \( \tau \geq 0 \) after \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)

Diagram:
- \( \ell_0 \rightarrow \ell_1 \):
  - \( t_0 \in g_0 \)
  - \( \frac{1}{2} : t_1 \in g_1 \)
  - \( \text{if}(i > 0) \)
  - \( x = x + i \)
  - \( i = i - 1 \)
- \( \ell_1 \rightarrow \ell_2 \):
  - \( \text{if}(i > 0) \)
  - \( t_2 \in g_2 \)
- \( \ell_2 \rightarrow \ell_0 \):
  - \( \text{if}(i \leq 0) \)
  - \( t_3 \in g_3 \)
  - \( \text{if}(x > 0) \)
  - \( x = x - 1 \)

- Value of \( \tau \) decreases by 1 for \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)
- Boundedness: \( \tau \geq 0 \) after \( \mathcal{P}_{\succ} \subseteq \mathcal{P} \)
Expected Runtime Bounds from Probabilistic Ranking Functions

**Initial bounds**
\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \text{ as } g_0 \text{ and } g_2 \text{ are not in loops} \]

**Probabilistic ranking function \( r \) for program \( P \)**
- for all \( g \in P \), set \( R_E(g) = r(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases value of \( r \)
- **Decrease**: value of \( r \) decreases by 1 for \( P \subseteq P \)
- **Boundedness**: \( r \geq 0 \) after \( P \subseteq P \)
**Expected Runtime Bounds from Probabilistic Ranking Functions**

**Initial bounds**
\[ \mathcal{R}_E(g_0) = 1, \mathcal{R}_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

**Probabilistic ranking function \( \tau \) for program \( \mathcal{P} \)**
- for all \( g \in \mathcal{P}_\succ \), set \( \mathcal{R}_E(g) = \tau(\ell_0) \)
- **Non-Increase**: no transition in \( \mathcal{P} \) increases expected value of \( \tau \)
- **Decrease**: value of \( \tau \) decreases by 1 for \( \mathcal{P}_\succ \subseteq \mathcal{P} \)
- **Boundedness**: \( \tau \geq 0 \) after \( \mathcal{P}_\succ \subseteq \mathcal{P} \)

```
\[ \frac{1}{2} : t_1 \in g_1 \]  
\[ \text{if}(i > 0) \]
\[ x = x + i \]
\[ i = i - 1 \]

\[ \frac{1}{2} : t_4 \in g_1 \]  
\[ \text{if}(i > 0) \]

\[ t_2 \in g_2 \]
\[ \text{if}(i \leq 0) \]
```

```
\[ t_3 \in g_3 \]
\[ \text{if}(x > 0) \]
\[ x = x - 1 \]
```
Expected Runtime Bounds from *Probabilistic* Ranking Functions

**Initial bounds**

\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

**Probabilistic ranking function** \( \tau \) for program \( \mathcal{P} \)

- for all \( g \in \mathcal{P}_\succ \), set \( R_E(g) = \tau(\ell_0) \)

  - **Non-Increase**: no transition in \( \mathcal{P} \) increases *expected* value of \( \tau \)
  
  - **Decrease**: *expected* value of \( \tau \) decreases by 1 for \( \mathcal{P}_\succ \subseteq \mathcal{P} \)
  
  - **Boundedness**: \( \tau \geq 0 \) after \( \mathcal{P}_\succ \subseteq \mathcal{P} \)
**Expected Runtime Bounds from Probabilistic Ranking Functions**

**Initial bounds**
\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

**Probabilistic ranking function** \( r \) for program \( P \)

- For all \( g \in P_{\gg} \), set \( R_E(g) = r(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases expected value of \( r \)
- **Decrease**: expected value of \( r \) decreases by 1 for \( P_{\gg} \subseteq P \)
- **Boundedness**: \( r \geq 0 \) after \( P_{\gg} \subseteq P \)

- \( r(\ell) = 2 \cdot i \) for all locations \( \ell \)

**Diagram**

- \( \ell_0 \)
  - \( t_0 \in g_0 \)
  - \( \frac{1}{2} : t_1 \in g_1 \) if \( i > 0 \)
    - \( x = x + i \)
    - \( i = i - 1 \)
  - \( \frac{1}{2} : t_4 \in g_1 \) if \( i > 0 \)

- \( \ell_1 \)
  - \( t_2 \in g_2 \) if \( i \leq 0 \)
    - \( t_3 \in g_3 \) if \( x > 0 \)
    - \( x = x - 1 \)
Expected Runtime Bounds from Probabilistic Ranking Functions

**Initial bounds**

\[ R_E(g_0) = 1, \; R_E(g_2) = 1 \text{ as } g_0 \; \text{and} \; g_2 \; \text{are not in loops} \]

**Probabilistic ranking function** \( r \) for program \( P \)

- for all \( g \in P_{\succ} \), set \( R_E(g) = r(\ell_0) \)
- **Non-Increase:** no transition in \( P \) increases expected value of \( r \)
- **Decrease:** expected value of \( r \) decreases by 1 for \( P_{\succ} \subseteq P \)
- **Boundedness:** \( r \geq 0 \) after \( P_{\succ} \subseteq P \)

- \( r(\ell) = 2 \cdot i \) for all locations \( \ell \)
- Thus: \( g_1 \in P_{\succ} \)

\[ \frac{1}{2} : t_1 \in g_1 \]
\[ \text{if}(i > 0) \quad x = x + i \quad i = i - 1 \]

\[ \frac{1}{2} : t_4 \in g_1 \]
\[ \text{if}(i > 0) \]

\[ t_2 \in g_2 \]
\[ \text{if}(i \leq 0) \]

\[ t_3 \in g_3 \]
\[ \text{if}(x > 0) \quad x = x - 1 \]
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds
\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

Probabilistic ranking function \( r \) for program \( P \)
- For all \( g \in P_\succ \), set \( R_E(g) = r(\ell_0) \)
- Non-Increase: no transition in \( P \) increases expected value of \( r \)
- Decrease: expected value of \( r \) decreases by 1 for \( P_\succ \subseteq P \)
- Boundedness: \( r \geq 0 \) after \( P_\succ \subseteq P \)

\[ r(\ell) = 2 \cdot i \] for all locations \( \ell \)
- Thus: \( g_1 \in P_\succ \)

\[ r(\ell_1) \geq \frac{1}{2} \cdot r(\ell_1)[x / x + i, \ i / i - 1] + \frac{1}{2} \cdot r(\ell_1)[x / x, \ i / i] + 1 \]
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds
\[ R_E(g_0) = 1, \quad R_E(g_2) = 1 \text{ as } g_0 \text{ and } g_2 \text{ are not in loops} \]

Probabilistic ranking function \( \tau \) for program \( P \)
- for all \( g \in P_\succcurlyeq \), set \( R_E(g) = \tau(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases *expected* value of \( \tau \)
- **Decrease**: *expected* value of \( \tau \) decreases by 1 for \( P_\succcurlyeq \subseteq P \)
- **Boundedness**: \( \tau \geq 0 \) after \( P_\succcurlyeq \subseteq P \)

- \( \tau(\ell) = 2 \cdot i \) for all locations \( \ell \)
- Thus: \( g_1 \in P_\succcurlyeq \)

\[
2 \cdot i \geq \frac{1}{2} \cdot \tau(\ell_1)[x/x+i, i/i-1] + \frac{1}{2} \cdot \tau(\ell_1)[x/x, i/i] + 1
\]
Expected Runtime Bounds from Probabilistic Ranking Functions

Initial bounds
\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \text{ as } g_0 \text{ and } g_2 \text{ are not in loops} \]

Probabilistic ranking function \( \tau \) for program \( P \)
- for all \( g \in P_\succ \), set \( R_E(g) = \tau(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases expected value of \( \tau \)
- **Decrease**: expected value of \( \tau \) decreases by 1 for \( P_\succ \subseteq P \)
- **Boundedness**: \( \tau \geq 0 \) after \( P_\succ \subseteq P \)

- \( \tau(\ell) = 2 \cdot i \) for all locations \( \ell \)
- Thus: \( g_1 \in P_\succ \)

\[
2 \cdot i \geq \frac{1}{2} \cdot 2 \cdot (i - 1) + \frac{1}{2} \cdot \tau(\ell_1)[x / x, i / i] + 1
\]

Diagram:
- \( \ell_0 \rightarrow t_0 \in g_0 \)
- \( \ell_1 \):
  - \( \frac{1}{2} : t_1 \in g_1 \) if \( i > 0 \)
  - \( x = x + i \) if \( i > 0 \)
  - \( i = i - 1 \)
- \( \ell_2 \):
  - \( \frac{1}{2} : t_4 \in g_1 \) if \( i > 0 \)
- \( \ell_3 \):
  - \( t_3 \in g_3 \) if \( x > 0 \)
  - \( x = x - 1 \)
**Expected Runtime Bounds from Probabilistic Ranking Functions**

**Initial bounds**

\[ R_E(g_0) = 1, \ R_E(g_2) = 1 \] as \( g_0 \) and \( g_2 \) are not in loops

**Probabilistic ranking function \( \tau \) for program \( P \)**

- for all \( g \in P_\succ \), set \( R_E(g) = \tau(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases expected value of \( \tau \)
- **Decrease**: expected value of \( \tau \) decreases by 1 for \( P_\succ \subseteq P \)
- **Boundedness**: \( \tau \geq 0 \) after \( P_\succ \subseteq P \)

- \( \tau(\ell) = 2 \cdot i \) for all locations \( \ell \)

- **Thus**: \( g_1 \in P_\succ \)

\[
2 \cdot i \geq \frac{1}{2} \cdot 2 \cdot (i - 1) + \frac{1}{2} \cdot 2 \cdot i + 1
\]
**Expected Runtime Bounds from Probabilistic Ranking Functions**

**Initial bounds**
\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0 \]

**Probabilistic ranking function \( r \) for program \( P \)**
- for all \( g \in P_A \), set \( R_E(g) = r(\ell_0) \)
- **Non-Increase**: no transition in \( P \) increases *expected* value of \( r \)
- **Decrease**: *expected* value of \( r \) decreases by 1 for \( P_A \subseteq P \)
- **Boundedness**: \( r \geq 0 \) after \( P_A \subseteq P \)

- \( r(\ell) = 2 \cdot i \) for all locations \( \ell \)

**Thus**: \( g_1 \in P_A \)

\[
2 \cdot i \geq \frac{1}{2} \cdot 2 \cdot (i - 1) + \frac{1}{2} \cdot 2 \cdot i + 1
\]
Modular Expected Runtime Bounds from Probabilistic Ranking Functions

Current bounds

\[ R_E(g_0) = 1, \ R_E(g_2) = 1, \ R_E(g_1) = 2 \cdot i_0 \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**
\[ \mathcal{R}_E(g_0) = 1, \quad \mathcal{R}_E(g_2) = 1, \quad \mathcal{R}_E(g_1) = 2 \cdot i_0 \]

**Computing runtime bound for** \( g \in \mathcal{P}' \)
\[ \mathcal{R}(g) = \mathcal{R}(g') \cdot \mathcal{R}(t) [v / S (g', v)] \]
- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g' \): pre-transition of \( \mathcal{P}' \)

\[ \begin{align*}
\ell_0 & \quad t_0 \in g_0 \\
\ell_1 & \quad \frac{1}{2} : t_1 \in g_1 \\
& \quad \text{if}(i > 0) \\
& \quad x = x + i \\
& \quad i = i - 1 \\
\ell_2 & \quad t_2 \in g_2 \\
& \quad \text{if}(i \leq 0) \\
\ell_3 & \quad t_3 \in g_3 \\
& \quad \text{if}(x > 0) \\
& \quad x = x - 1 \\
\ell_4 & \quad t_4 \in g_1 \\
& \quad \text{if}(i > 0) \\
\end{align*} \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds

\[ \mathcal{R}_E(g_0) = 1, \mathcal{R}_E(g_2) = 1, \mathcal{R}_E(g_1) = 2 \cdot i_0 \]

Computing *expected* runtime bound for \( g \in \mathcal{P}' \)

\[ \mathcal{R}(g) = \mathcal{R}(g') \cdot t(\ell)[v/S(g', \nu)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g' \): pre-transition of \( \mathcal{P}' \)

\[
\begin{align*}
\ell_0 & \quad t_0 \in g_0 \\
\ell_1 & \quad \frac{1}{2} : t_1 \in g_1 \\
& \quad \text{if}(i > 0) \\
& \quad x = x + i \\
& \quad i = i - 1 \\
\ell_2 & \quad t_2 \in g_2 \\
& \quad \text{if}(i \leq 0) \\
\ell_3 & \quad t_3 \in g_3 \\
& \quad \text{if}(x > 0) \\
& \quad x = x - 1 \\
\ell_4 & \quad \frac{1}{2} : t_4 \in g_1 \\
& \quad \text{if}(i > 0) \\
\end{align*}
\]
**Modular Expected Runtime Bounds from Probabilistic Ranking Functions**

**Current bounds**

\[ \mathcal{R}_E(g_0) = 1, \mathcal{R}_E(g_2) = 1, \mathcal{R}_E(g_1) = 2 \cdot i_0 \]

**Computing expected runtime bound for \( g \in \mathcal{P}' \)**

\[ \mathcal{R}_E(g) = \mathbb{E}( \mathcal{R}(g') \cdot t(\ell) [\nu / S(g', \nu)] ) \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g' \): pre-transition of \( \mathcal{P}' \)

\[ \begin{align*}
\ell_0: t_0 & \in g_0 \\
\ell_1: t_1 & \in g_1 \\
\ell_2: t_2 & \in g_2 \\
\ell_3: t_3 & \in g_3 \\
\ell_4: t_4 & \in g_1
\end{align*} \]

- if \( i > 0 \):
  - \( x = x + i \)
  - \( i = i - 1 \)

- if \( i \leq 0 \):
  - \( t_2 \in g_2 \)

- if \( x > 0 \):
  - \( x = x - 1 \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds

\[ \mathcal{R}_E(g_0) = 1, \mathcal{R}_E(g_2) = 1, \mathcal{R}_E(g_1) = 2 \cdot i_0 \]

Computing *expected* runtime bound for \( g \in \mathcal{P}' \)

\[ \mathcal{R}_E(g) = \mathbb{E}(\mathcal{R}(g')) \cdot t(\ell)[v/S(g',v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g' \): pre-transition of \( \mathcal{P}' \)

Expected value *not* multiplicative!

\[
\begin{align*}
\ell_0 & \quad t_0 \in g_0 \\
\ell_1 & \quad \frac{1}{2} : t_1 \in g_1 \\
& \quad \text{if}(i > 0) \\
& \quad x = x + i \\
& \quad i = i - 1 \\
\ell_2 & \quad \frac{1}{2} : t_2 \in g_2 \\
& \quad \text{if}(i > 0) \\
\ell_3 & \quad t_3 \in g_3 \\
& \quad \text{if}(x > 0) \\
& \quad x = x - 1 \\
\ell_4 & \quad t_4 \in g_4 \\
& \quad \text{if}(i > 0) \\
\end{align*}
\]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0 \]

**Computing expected runtime bound for \( g \in \mathcal{P}' \)**

\[ R_E(g) = R(t') \cdot E( r(\ell) [v / S (g', v)] ) \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

Expected value *not* multiplicative!

**Expected value not multiplicative!**

**Diagram:**

- \( \ell_0 \): entry location of \( \mathcal{P}' \)
- \( t_0 \in g_0 \)
- \( t_1 \in g_1 \) if \( i > 0 \)
  - \( x = x + i \)
  - \( i = i - 1 \)
- \( t_2 \in g_2 \) if \( i \leq 0 \)
- \( t_3 \in g_3 \) if \( x > 0 \)
  - \( x = x - 1 \)
- \( t_4 \in g_1 \) if \( i > 0 \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0 \]

Computing *expected* runtime bound for \( g \in P' \)

\[ R_E(g) = R(t') \cdot E(\tau(\ell) [v / S(g', v)]) \]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

*Expected value not multiplicative!*  
\( \Rightarrow \) restrict to *linear* ranking functions \( \tau \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds

\[ R_E(g_0) = 1, \; R_E(g_2) = 1, \; R_E(g_1) = 2 \cdot i_0 \]

Computing *expected* runtime bound for \( g \in P' \)

\[ R_E(g) = R(t') \cdot \tau(\ell) [v / S_E(g', v)] \]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

Expected value *not* multiplicative!

⇒ restrict to *linear* ranking functions \( \tau \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**
\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0 \]

**Computing expected runtime bound for \( g \in P' \)**
\[
R_E(g) = R(t') \cdot t(\ell) [v / S_E(g', v)]
\]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0 \]

Computing *expected* runtime bound for \( g \in P' \)

\[ R_E(g) = R(t') \cdot t(\ell)[v/S_E(g', v)] \]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(\ell_2) = x \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

### Current bounds

\[ R_E(g_0) = 1, \; R_E(g_2) = 1, \; R_E(g_1) = 2 \cdot i_0, \; R_E(g_3) = t(\ell_2) \]

### Computing expected runtime bound for \( g \in \mathcal{P}' \)

\[
R_E(g) = R(t') \cdot t(\ell) [v / S_E(g', v)]
\]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{g_3\} \)
- \( t(\ell_2) = x \)  
  Thus: \( g_3 \in \mathcal{P}' \)

\[
\frac{1}{2} : t_1 \in g_1 \\
\text{if}(i > 0) \\
x = x + i \\
i = i - 1
\]

\[
\frac{1}{2} : t_2 \in g_2 \\
\text{if}(i > 0) \\
t_3 \in g_3 \\
\text{if}(x > 0) \\
x = x - 1
\]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, R_E(g_2) = 1, R_E(g_1) = 2 \cdot i_0, R_E(g_3) = t(l_2) \]

**Computing expected runtime bound for \( g \in \mathcal{P}' \)**

\[ R_E(g) = R(t') \cdot t(l)[v / S_E(g', v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{g_3\} \)
- \( t(l_2) = x \)
  
  Thus: \( g_3 \in \mathcal{P}' \)

- Executions of \( \mathcal{P}' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

### Current bounds

\[ R_E(g_0) = 1, \; R_E(g_2) = 1, \; R_E(g_1) = 2 \cdot i_0, \; R_E(g_3) = t(\ell_2) \]

### Computing expected runtime bound for \( g \in \mathcal{P}' \)

\[ R_E(g) = R(t') \cdot t(\ell)[v / S_E(g', v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{g_3\} \)
- \( t(\ell_2) = x \quad \text{Thus:} \quad g_3 \in \mathcal{P}_\succ \)
- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.
- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))

#### Diagram

- \( t_0 \in g_0 \)
- \( t_1 \in g_1 \) if \( i > 0 \)
- \( t_2 \in g_2 \) if \( i \leq 0 \)
- \( t_3 \in g_3 \) if \( x > 0 \)
- \( x = x - 1 \)
- \( t_4 \in g_1 \) if \( i > 0 \)
- \( x = x + i \)
- \( i = i - 1 \)
**Modular Expected Runtime Bounds from Probabilistic Ranking Functions**

Current bounds

\[ R_E(g_0) = 1, \, R_E(g_2) = 1, \, R_E(g_1) = 2 \cdot i_0, \, R_E(g_3) = t(l_2) \]

Computing expected runtime bound for \( g \in P' \)

\[ R_E(g) = R(t') \cdot t(l)[v / S_E(g', v)] \]

- \( l \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)

- \( t(l_2) = x \)  
  Thus: \( g_3 \in P' \)

- Executions of \( P' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))

⇒ multiply \( t_2 \)'s non-probabilistic runtime bound \( R(t_2) \) with local bound \( t(l_2) \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

Current bounds
\[
R_E(g_0) = 1, \ R_E(g_2) = 1, \ R_E(g_1) = 2 \cdot i_0, \ R_E(g_3) = R(t_2) \cdot t(\ell_2)
\]

Computing expected runtime bound for \( g \in P' \)
\[
R_E(g) = R(t') \cdot t(\ell) [v / S_E(g', v)]
\]
- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(\ell_2) = x \)
  - Thus: \( g_3 \in P'_> \)
- Executions of \( P' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  
  \[ \Rightarrow \text{multiply } t_2 \text{'s non-probabilistic runtime bound } R(t_2) \text{ with local bound } t(\ell_2) \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0, \quad R_E(g_3) = 1 \cdot t(l_2) \]

**Computing expected runtime bound for \( g \in P' \)**

\[ R_E(g) = R(t') \cdot t(l)[v / S_E(g', v)] \]

- \( l \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(l_2) = x \)
  Thus: \( g_3 \in P' \)
- Executions of \( P' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))

\[ \Rightarrow \text{multiply} \ t_2 \text{'s non-probabilistic runtime bound} \ R(t_2) \text{with local bound} \ t(l_2) \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0, \quad R_E(g_3) = t(l_2) \]

**Computing expected runtime bound for** \( g \in P' \)

\[ R_E(g) = R(t') \cdot t(l)[v / S_E(g', v)] \]

- \( l \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)
- Modular use of ranking function for subset \( P' = \{g_3\} \)
- \( t(l_2) = x \) Thus: \( g_3 \in P' \Rightarrow \)
- Executions of \( P' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))

\( \Rightarrow \) multiply \( t_2 \)'s *non-probabilistic* runtime bound \( R(t_2) \) with local bound \( t(l_2) \)
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, \; R_E(g_2) = 1, \; R_E(g_1) = 2 \cdot i_0, \; R_E(g_3) = t(l_2) \]

**Computing expected runtime bound for \( g \in P' \)**

\[ R_E(g) = R(t') \cdot t(l)[v/S_E(g', v)] \]

- \( l \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(l_2) = x \) Thus: \( g_3 \in P'_\succ \)
- Executions of \( P' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider expected value of \( P' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{multiply } t_2 \text{'s non-probabilistic runtime bound } R(t_2) \text{ with local bound } t(l_2) \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Current bounds**

\[ R_E(g_0) = 1, \ R_E(g_2) = 1, \ R_E(g_1) = 2 \cdot i_0, \ R_E(g_3) = t(l_2) \]

**Computing expected runtime bound for \( g \in \mathcal{P}' \)**

\[ R_E(g) = R(t') \cdot t(l)[v / S_E(g', v)] \]

- \( l \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

**Modular use of ranking function for subset \( \mathcal{P}' = \{ g_3 \} \)**

\[ t(l_2) = x \]

**Executions of \( \mathcal{P}' \) starting in \( l_2 \) use \( g_3 \) at most \( t(l_2) = x \) times.**

**For global result:**

- consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
- consider *expected* value of \( \mathcal{P}' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(l_2) \text{ by } t(l_2)[x / S_E(g_2, x)] \]
Modular Expected Runtime Bounds from Probabilistic Ranking Functions

Current bounds
\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0, \quad R_E(g_3) = t(\ell_2)[x/S_E(g_2, x)] \]

Computing expected runtime bound for \( g \in P' \)
\[ R_E(g) = R(t') \cdot t(\ell)[v/S_E(g', v)] \]
- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(\ell_2) = x \)
  
Thus: \( g_3 \in P'_{\succ} \)
- Executions of \( P' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider expected value of \( P' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(\ell_2) \text{ by } t(\ell_2)[x/S_E(g_2, x)] \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

### Current bounds

\[
R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0, \quad R_E(g_3) = x \left[ \frac{x}{S_E(g_2, x)} \right]
\]

### Computing expected runtime bound for \( g \in P' \)

\[
R_E(g) = R(t') \cdot t(\ell) \left[ v / S_E(g', v) \right]
\]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

#### Modular use of ranking function for subset \( P' = \{ g_3 \} \)

- \( t(\ell_2) = x \)  
  Thus: \( g_3 \in P'_> \)

- Executions of \( P' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider *expected* value of \( P'\)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(\ell_2) \text{ by } t(\ell_2) \left[ \frac{x}{S_E(g_2, x)} \right] \]
Modular Expected Runtime Bounds from Probabilistic Ranking Functions

Current bounds

\[ R_E(g_0) = 1, \ R_E(g_2) = 1, \ R_E(g_1) = 2 \cdot i_0, \ R_E(g_3) = S_E(g_2, x) \]

Computing expected runtime bound for \( g \in P' \)

\[ R_E(g) = R(t') \cdot t(\ell)[v / S_E(g', v)] \]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(\ell_2) = x \)
  Thus: \( g_3 \in P'_\succ \)

- Executions of \( P' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.

- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider expected value of \( P' \)'s initial variable \( x \) in full run

\( \Rightarrow \) replace \( t(\ell_2) \) by \( t(\ell_2)[x / S_E(g_2, x)] \)
Modular Expected Runtime Bounds from Probabilistic Ranking Functions

**Expected runtime bounds**

\[ R_E(g_0) = 1, \quad R_E(g_2) = 1, \quad R_E(g_1) = 2 \cdot i_0, \quad R_E(g_3) = x_0 + i_0^2 \]

**Computing expected runtime bound for \( g \in \mathcal{P}' \)**

\[ R_E(g) = R(t') \cdot t(\ell)[v / S_E(g', v)] \]

- \( \ell \): entry location of \( \mathcal{P}' \)
- \( g', t' \): pre-transition of \( \mathcal{P}' \)

- Modular use of ranking function for subset \( \mathcal{P}' = \{g_3\} \)
- \( t(\ell_2) = x \) Thus: \( g_3 \in \mathcal{P}'_> \)
- Executions of \( \mathcal{P}' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.
- For global result:
  - consider how often \( \mathcal{P}' \) is reached (by \( t_2 \))
  - consider expected value of \( \mathcal{P}' \)'s initial variable \( x \) in full run

\[ \Rightarrow \text{replace } t(\ell_2) \text{ by } t(\ell_2)[x / S_E(g_2, x)] \]
Modular *Expected* Runtime Bounds from *Probabilistic* Ranking Functions

**Expected runtime bounds**

\[ R_E(g_0) = 1, \ R_E(g_2) = 1, \ R_E(g_1) = 2 \cdot i_0, \ R_E(g_3) = x_0 + i_0^2 \]

**Computing expected runtime bound for** \( g \in P' \)

\[ R_E(g) = R(t') \cdot t(\ell) [v / S_E(g', v)] \]

- \( \ell \): entry location of \( P' \)
- \( g', t' \): pre-transition of \( P' \)

- Modular use of ranking function for subset \( P' = \{ g_3 \} \)
- \( t(\ell_2) = x \)
  - Thus: \( g_3 \in P' \) \( \succ \)
- Executions of \( P' \) starting in \( \ell_2 \) use \( g_3 \) at most \( t(\ell_2) = x \) times.
- For global result:
  - consider how often \( P' \) is reached (by \( t_2 \))
  - consider *expected* value of \( P'' \)'s initial variable \( x \) in full run

Overall *expected* runtime is bounded by

\[ R_E(g_0) + \ldots + R_E(g_3) = 1 + 2 \cdot i_0 + 1 + x_0 + i_0^2. \]
Size Bounds

Size bounds
\[ S(t_0, i) = i_0, \ S(t_0, x) = x_0 \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0 \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0 \]
Size Bounds

Size bounds

\[ S(t_0, \nu) = \nu_0, \ S(t_1, i) = i_0 \]

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = \ \mathcal{LC}(t, \nu) \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)

Diagram:

- \( \ell_0 \) to \( \ell_1 \): \( t_0 \)
- \( \ell_1 \): \( t_1 \)
  - \( \text{if}(i > 0) \): \( x = x + i \)
  - \( i = i - 1 \)
- \( \ell_2 \): \( t_2 \)
  - \( \text{if}(i \leq 0) \)
- \( \ell_3 \): \( t_3 \)
  - \( \text{if}(x > 0) \): \( x = x - 1 \)
Size Bounds

Size bounds

\[ S(t_0, \nu) = \nu_0, \ S(t_1, i) = i_0 \]

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = \mathcal{LC}(t, \nu) \]

\( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)

\[ \mathcal{LC}(t_1, x) = i \]

- \( t_1 \)
  - \text{if} \( i > 0 \)
  - \( x = x + i \)
  - \( i = i - 1 \)

- \( t_2 \)
  - \text{if} \( i \leq 0 \)
  - \( x = x - 1 \)
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = \mathcal{LC}(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = \mathcal{LC}(t, v) \]

\( \mathcal{LC}(t, v) \): local change by one application of \( t \)

\( \mathcal{LC}(t_1, x) = i \)
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = \mathcal{LC}(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = \mathcal{LC}(t, v) \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

- For global result:

\[ \text{if}(i > 0) \quad x = x + i \]
\[ \text{if}(i \leq 0) \quad i = i - 1 \]
**Size Bounds**

**Size bounds**

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = \mathcal{LC}(t_1, x) \]

**Computing size bound for variable \( v \) after transition \( t \)**

\[ S(t, v) = \mathcal{LC}(t, v) \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))

![Diagram](image-url)
# Size Bounds

## Size bounds

\[
S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = \mathcal{LC}(t_1, x)
\]

## Computing size bound for variable \( v \) after transition \( t \)

\[
S(t, v) = S(t_0, v) + \mathcal{LC}(t, v)
\]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))

\[ \Rightarrow \text{add size bound } S(t_0, x) \text{ to } \mathcal{LC}(t_1, x) \]
Size Bound

Size bounds
\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = S(t_0, x) + LC(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = LC(t, v) \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( LC(t_1, x) = i \)
- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))

\[ \Rightarrow \text{add size bound } S(t_0, x) \text{ to } LC(t_1, x) \]
Size Bounds

Size bounds

$S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = S(t_0, x) + \ LC(t_1, x)$

Computing size bound for variable $v$ after transition $t$

$S(t, v) = S(t', v) + \ LC(t, v)$

- $\LC(t, v)$: local change by one application of $t$
- $t'$: pre-transition of $t$

$\LC(t_1, x) = i$

For global result:
- consider value of $x$ before reaching $t_1$ (after $t_0$)

$\Rightarrow$ add size bound $S(t_0, x)$ to $\LC(t_1, x)$
Size Bounds

Size bounds
\[ S(t_0, \nu) = \nu_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + \ \mathcal{LC}(t_1, x) \]

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = S(t', \nu) + \ \mathcal{LC}(t, \nu) \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))

\[ \Rightarrow \text{add size bound } S(t_0, x) \text{ to } \mathcal{LC}(t_1, x) \]
Size Bounds

Size bounds
\[ S(t_0, v) = v_0, \; S(t_1, i) = i_0, \; S(t_1, x) = x_0 + \mathcal{LC}(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = S(t', v) + \mathcal{LC}(t, v) \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed

⇒ add size bound \( S(t_0, x) \) to \( \mathcal{LC}(t_1, x) \)
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + \ \mathcal{LC}(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + \ \mathcal{LC}(t, v) \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \mathcal{LC}(t_1, x) = i \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed

\[ \Rightarrow \text{multiply } t_1' \text{'s runtime bound } \mathcal{R}(t_1) \text{ with local change } \mathcal{LC}(t_1, x) \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + R(t_1) \cdot LC(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + LC(t, v) \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_1, x) = i \)

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
  - consider how often \( t_1 \) is executed

\[ \Rightarrow \text{multiply } t_1 \text{'s runtime bound } R(t_1) \text{ with local change } LC(t_1, x) \]
Size Bounds

Size bounds:

\[ S(t_0, \nu) = \nu_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + R(t_1) \cdot LC(t_1, x) \]

Computing size bound for variable \( \nu \) after transition \( t \):

\[ S(t, \nu) = S(t', \nu) + R(t) \cdot LC(t, \nu) \]

- \( LC(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed

\[ \Rightarrow \text{multiply } t_1 \text{'s runtime bound } R(t_1) \text{ with local change } LC(t_1, x) \]
Size Bounds

Size bounds
\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot LC(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = S(t', v) + \mathcal{R}(t) \cdot LC(t, v) \]
- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)
- \( \mathcal{R}(t) \): runtime bounds

\[ LC(t_1, x) = i \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed

⇒ multiply \( t_1 \)'s runtime bound \( \mathcal{R}(t_1) \) with local change \( LC(t_1, x) \)
Size Bounds

**Size bounds**

\[ S(t_0, \nu) = \nu_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot \mathcal{LC}(t_1, x) \]

**Computing size bound for variable \( \nu \) after transition \( t \)**

\[ S(t, \nu) = S(t', \nu) + \mathcal{R}(t) \cdot \mathcal{LC}(t, \nu) \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( \mathcal{LC}(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{multiply } t_1 \text{'s runtime bound } \mathcal{R}(t_1) \text{ with local change } \mathcal{LC}(t_1, x) \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0 \cdot LC(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v) \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( LC(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC(t_1, x) \text{ by } LC(t_1, x)[i / \max(S(t_0, i), S(t_1, i))] \]
Size Bounds

Size bounds
\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0 \cdot LC(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v) [u / \max(S(t', u), S(t, u))] \]
- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( LC(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace} \ LC(t_1, x) \text{ by} \ LC(t_1, x) [i / \max(S(t_0, i), S(t_1, i))] \]
Size Bounds

<table>
<thead>
<tr>
<th>Size bounds</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S(t_0, \nu) = \nu_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot LC(t_1, x) )</td>
</tr>
</tbody>
</table>

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = S(t', \nu) + R(t) \cdot LC(t, \nu)[u / \max(S(t', u), S(t, u))] \]

- \( LC(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_1, x) = i \)
- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
  - consider how often \( t_1 \) is executed
  - consider values of \( LC(t_1, x) \)'s variables in full run

⇒ replace \( LC(t_1, x) \) by \( LC(t_1, x)[i / \max(i_0, i_0)] \)
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot \text{LC}(t_1, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot \text{LC}(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( \text{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( \text{LC}(t_1, x) = i \)

For global result:

- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( \text{LC}(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace} \ \text{LC}(t_1, x) \text{ by } \text{LC}(t_1, x)[i / i_0] \]
Size Bounds

Size bounds
\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0 \cdot \text{LC}(t_1, x) \ [i / i_0] \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = S(t', v) + R(t) \cdot \text{LC}(t, v) [u / \max(S(t', u), S(t, u))] \]

- **LC**(\( t, v \)): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \text{LC}(t_1, x) = i \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( \text{LC}(t_1, x) \)'s variables in full run

⇒ replace \( \text{LC}(t_1, x) \) by \( \text{LC}(t_1, x) [i / i_0] \)
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot \frac{i}{i_0} \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot \mathcal{LC}(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( \mathcal{LC}(t_1, x) \)'s variables in full run

⇒ replace \( \mathcal{LC}(t_1, x) \) by \( \mathcal{LC}(t_1, x)[i / i_0] \)
Size Bounds

### Size bounds

\[
S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0 \cdot i_0
\]

### Computing size bound for variable \( v \) after transition \( t \)

\[
S(t, v) = S(t', v) + R(t) \cdot LC(t, v) [u / \max(S(t', u), S(t, u))]
\]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[
LC(t_1, x) = i
\]

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
  - consider how often \( t_1 \) is executed
  - consider values of \( LC(t_1, x) \)'s variables in full run

\[\Rightarrow \text{replace } LC(t_1, x) \text{ by } LC(t_1, x) [i / i_0] \]
Size Bounds

**Size bounds**

\[ S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0^2 \]

**Computing size bound for variable \( v \) after transition \( t \)**

\[ S(t, v) = S(t', v) + R(t) \cdot \mathcal{LC}(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( \mathcal{LC}(t_1, x) = i \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( \mathcal{LC}(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } \mathcal{LC}(t_1, x) \text{ by } \mathcal{LC}(t_1, x)[i / i_0] \]
Size Bounds

**Size bounds**

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2 \]

**Computing size bound for variable \( v \) after transition \( t \)**

\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ LC(t_2, x) = 0 \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_1 \) is executed
- consider values of \( LC(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC(t_1, x) \text{ by } LC(t_1, x)[i / i_0] \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = LC(t_2, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

- \( LC(t_2, x) = 0 \)

- For global result:
  - consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
  - consider how often \( t_1 \) is executed
  - consider values of \( LC(t_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC(t_1, x) \text{ by } LC(t_1, x)[i / i_0] \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \mathcal{LC}(t_2, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + \mathcal{R}(t) \cdot \mathcal{LC}(t, v) \left[ u / \max(S(t', u), S(t, u)) \right] \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \mathcal{LC}(t_2, x) = 0 \]

For global result:

- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{replace } \mathcal{LC}(t_1, x) \text{ by } \mathcal{LC}(t_1, x) [i / i_0] \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = LC(t_2, x) \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v) [u / \max(S(t', u), S(t, u))] \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ LC(t_2, x) = 0 \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_2 \) is executed
- consider values of \( LC(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{replace} \ LC(t_2, x) \text{ by} \ R(t_2) \cdot \ LC(t_2, x) [\ldots] \]
Size Bounds

Size bounds

\[ S(t_0, \nu) = \nu_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \mathcal{LC}(t_2, x) \]

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = S(t', \nu) + \mathcal{R}(t) \cdot \mathcal{LC}(t, \nu)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\( \mathcal{LC}(t_2, x) = 0 \)

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{replace } \mathcal{LC}(t_2, x) \text{ by } 0 \]
### Size Bounds

#### Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \]

#### Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot \mathcal{LC}(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \mathcal{LC}(t_2, x) = 0 \]

For global result:
- consider value of \( x \) before reaching \( t_1 \) (after \( t_0 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{replace } \mathcal{LC}(t_2, x) \text{ by } 0 \]
Size Bounds

**Size bounds**

\[ S(t_0, \nu) = \nu_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0^2, \quad S(t_2, x) = \]

**Computing size bound for variable \( \nu \) after transition \( t \)**

\[ S(t, \nu) = S(t', \nu) + R(t) \cdot \mathcal{LC}(t, \nu)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \mathcal{LC}(t_2, x) = 0 \]

For global result:

- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\( \Rightarrow \) replace \( \mathcal{LC}(t_2, x) \) by 0
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( LC(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ LC(t_2, x) = 0 \]

For global result:
- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( LC(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{add } \max(S(t_0, x), S(t_1, x)) \text{ to } LC(t_2, x) \]
Size Bounds

Size bounds

\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \]

Computing size bound for variable \( v \) after transition \( t \)

\[ S(t, v) = S(t', v) + R(t) \cdot \mathcal{LC}(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\( \mathcal{LC}(t_2, x) = 0 \)

For global result:
- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{add} \ \max(x_0, S(t_1, x)) \ \text{to} \ \mathcal{LC}(t_2, x) \]
Size Bounds

Size bounds
\[ S(t_0, v) = v_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \]

Computing size bound for variable \( v \) after transition \( t \)
\[ S(t, v) = S(t', v) + R(t) \cdot LC(t, v)[u / \max(S(t', u), S(t, u))] \]

- \( LC(t,v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\( LC(t_2, x) = 0 \)

For global result:
- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( LC(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{add } \max(x_0, x_0 + i_0^2) \text{ to } LC(t_2, x) \]
Size Bounds

Size bounds

\[ S(t_0, \nu) = \nu_0, \ S(t_1, i) = i_0, \ S(t_1, x) = x_0 + i_0^2, \ S(t_2, x) = \]

Computing size bound for variable \( \nu \) after transition \( t \)

\[ S(t, \nu) = S(t', \nu) + \mathcal{R}(t) \cdot \mathcal{LC}(t, \nu)[u / \max(S(t', u), S(t, u))] \]

- \( \mathcal{LC}(t, \nu) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[ \mathcal{LC}(t_2, x) = 0 \]

For global result:
- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

\[ \Rightarrow \text{add} \quad x_0 + i_0^2 \quad \text{to} \quad \mathcal{LC}(t_2, x) \]
# Size Bounds

## Size bounds

\[
S(t_0, v) = v_0, \quad S(t_1, i) = i_0, \quad S(t_1, x) = x_0 + i_0^2, \quad S(t_2, x) = x_0 + i_0^2
\]

## Computing size bound for variable \( v \) after transition \( t \)

\[
S(t, v) = S(t', v) + \mathcal{R}(t) \cdot \mathcal{LC}(t, v)[u / \max(S(t', u), S(t, u))]
\]

- \( \mathcal{LC}(t, v) \): local change by one application of \( t \)
- \( t' \): pre-transition of \( t \)

\[
\mathcal{LC}(t_2, x) = 0
\]

For global result:

- consider value of \( x \) before reaching \( t_2 \) (after \( t_0 \) or \( t_1 \))
- consider how often \( t_2 \) is executed
- consider values of \( \mathcal{LC}(t_2, x) \)'s variables in full run

⇒ add \( x_0 + i_0^2 \) to \( \mathcal{LC}(t_2, x) \)
**Expected Size Bounds**

\[ SE(g_0, v) = v_0, \]
\[ SE(g_1, i) = i_0, \]
\[ SE(g_1, x) = SE(g_0, x) + x_0 + RE(g_1) \cdot 2 \cdot i_0 \cdot 2 \]

Computing expected size bound for variable \( v \) after transition \( g \):

\[ SE(g, v) = E(SE(g', v)) + E(RE(g) \cdot LE(g, v) / \max(SE(g', u), ...)) \]

\( LE(g, v) \): expect. local change by \( g' \), \( t' \): pre-transition of \( g \), \( g \):

Expected value not multiplicative!

But:

\[ LE(g_1, x) = i_2 \]

For global result:

consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))

consider how often \( g_1 \) is expected to be executed

consider values of \( LE(g_1, x) \)'s variables in full run

\( \ell_0 \)

\( t_0 \)

\( \ell_1 \)

\( t_1 \)

if \((i > 0)\)

\( x = x + i \)

\( i = i - 1 \)

\( \ell_2 \)

if \((i \leq 0)\)

\( t_2 \)

\( \ell_3 \)

if \((x > 0)\)

\( x = x - 1 \)
Expected Size Bounds

\[ S_E(g_0, v) = v_0, \]
\[ S_E(g_1, i) = i_0 \]
\[ S_E(g_1, x) = S_E(g_0, x) + x_0 + \mathbb{E}(R_E(g_1) \cdot 2 \cdot i_0 / i_0) \]

Computing expected size bound for variable \( v \) after transition \( g \):

\[ S_E(g', v) = \mathbb{E}(S_E(g', v)) + \mathbb{E}(R_E(g) \cdot \mathbb{L}(E(g, v) / \max(S_E(g', u), \ldots))) \]

\[ \mathbb{L}(E)(g, v) : \text{expect. local change by } g, t' : \text{pre-transition of } g \]

\[ \mathbb{E} \text{ not multiplicative!} \]

But:

\[ \mathbb{L}(E) \text{ independent of runtime} \]

\[ \mathbb{L}(E)(g_1, x) = i_2 \]

For global result:

consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))

consider how often \( g_1 \) is expected to be executed

consider values of \( \mathbb{L}(E)(g_1, x) \)'s variables in full run

\[ t_0 \in g_0 \]
\[ \frac{1}{2} : t_1 \in g_1 \]
\[ \text{if}(i > 0) \]
\[ x = x + i \]
\[ i = i - 1 \]

\[ t_2 \in g_2 \]
\[ \text{if}(i \leq 0) \]

\[ t_3 \in g_3 \]
\[ \text{if}(x > 0) \]
\[ x = x - 1 \]

\[ t_4 \in g_1 \]
\[ \frac{1}{2} : t_4 \in g_1 \]
\[ \text{if}(i > 0) \]
**Expected Size Bounds**

*Expected size bounds*

\[ S_E(g_0, v) = v_0, S_E(g_1, i) = i_0 \]
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

**Computing size bound for variable v after transition g**

\[ S(g, v) = S(g', v) + R(g) \cdot LC(g, v)[u/\max(S(g', u), \ldots)] \]

- **LC** \((g, v)\): local change by \(g\)
- **\(g'\)**: pre-transition of \(g\)

---

![Diagram with states and transitions](image)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable v after transition g**

\[ S(g, v) = S(g', v) + R(g) \cdot LC (g, v) [u/ \max(S(g', u), ...)] \]

- \( LC (g, v) \): local change by \( g \)
- \( g' \): pre-transition of \( g \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = \mathbb{E}( S(g', v) + \mathcal{R}(g) \cdot \mathcal{LC} (g, v) [u/ max(S(g', u), ...)]) \]

- \( \mathcal{LC} (g, v) \): local change by \( g \)
- \( g' \): pre-transition of \( g \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = \mathbb{E}(S(g', v)) + \mathbb{E}(R(g) \cdot \mathcal{LC}(g, v)[u/\max(S(g', u), ...)]) \]

- \( \mathcal{LC}(g, v) \): local change by \( g \)
- \( g' \): pre-transition of \( g \)

---

**Diagram**

- \( t_0 \in g_0 \)
- \( t_1 \in g_1 \)
- \( t_2 \in g_2 \)
- \( t_3 \in g_3 \)
- \( t_4 \in g_1 \)

**Rules**

- If \( i > 0 \):
  - \( x = x + i \)
  - \( i = i - 1 \)

- If \( i \leq 0 \):
  - \( x = x - 1 \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + \mathbb{E}( R(g) \cdot LC(g, v) [u/ \max(S(g', u), ...)] ) \]

- \( LC(g, v) \) : local change by \( g \)
- \( g' \) : pre-transition of \( g \)

---

**Diagram**

- From \( l_0 \):
  - If \( i > 0 \), then \( t_0 \in g_1 \) and \( t_1 \in g_1 \) if \( i > 0 \)
    - \( x = x + i \)
    - \( i = i - 1 \)
  - Otherwise, \( t_2 \in g_2 \) if \( i \leq 0 \)

- From \( l_1 \):
  - If \( i > 0 \), then \( t_3 \in g_3 \) if \( x > 0 \)
    - \( x = x - 1 \)

- From \( l_2 \):
  - \( t_2 \in g_2 \) if \( i \leq 0 \)

- From \( l_2 \):
  - \( t_3 \in g_3 \) if \( x > 0 \)
    - \( x = x - 1 \)
Expected Size Bounds

Expected size bounds

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

Computing expected size bound for variable \( v \) after transition \( g \)

\[ S_E(g, v) = S_E(g', v) + \mathbb{E}( R(g) \cdot LC(g, v) [u/ \max(S(g', u), ...)]) \]

- \( LC(g, v) \): local change by \( g \)  
- Expected value not multiplicative!
- \( g' \): pre-transition of \( g \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + \mathbb{E}( R(g) \cdot LC(g, v)[u/ \max(S(g', u), ...)]) \]

- **LC (g, v):** local change by g
- **g':** pre-transition of g

\[ i \in g_0 \]

- \( \frac{1}{2} : t_1 \in g_1 \)
- \( \text{if}(i > 0) \)
  - \( x = x + i \)
  - \( i = i - 1 \)

\[ \frac{1}{2} : t_4 \in g_1 \]
- \( \text{if}(i > 0) \)

\[ t_2 \in g_2 \]
- \( \text{if}(i \leq 0) \)

\[ t_3 \in g_3 \]
- \( \text{if}(x > 0) \)
  - \( x = x - 1 \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0 \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u/ max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

**Expected value not multiplicative!**

**But:** \( LC \) independent of runtime
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0, \ S_E(g_1, x) = \frac{i}{2} \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LCE_E(g, v) [u/ max(S(t', u), ...)] \]

- \( LCE_E(g, v) \): *expected* local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LCE_E(g_1, x) = \frac{i}{2} \]

![Diagram](image)
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0, \ S_E(g_1, x) = \frac{i}{2} \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u/ \max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expected local change by g
- \( g', t' \): pre-transition of g

- \( LC_E(g_1, x) = \frac{i}{2} \)

- For global result:

\[ t_0 \in g_0 \]

\[ \frac{1}{2} : t_1 \in g_1 \]

\[ x = x + i \]

\[ i = i - 1 \]

\[ \frac{1}{2} : t_4 \in g_1 \]

\[ if(i > 0) \]

\[ t_2 \in g_2 \]

\[ if(i \leq 0) \]

\[ t_3 \in g_3 \]

\[ if(x > 0) \]

\[ x = x - 1 \]
Expected Size Bounds

Expected size bounds

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = \frac{1}{2} \]

Computing expected size bound for variable \( v \) after transition \( g \)

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) \left[ u/ \max(S(t', u), \ldots) \right] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{1}{2} \)

- For global result:
  - consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
**Expected Size Bounds**

**Expected size bounds**

\[
S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = \frac{i}{2}
\]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[
S_E(g, v) = S_E(g', v) + \mathcal{R}_E(g) \cdot \mathcal{LC}_E(g, v) [u/ \max(S(t', u), ...)]
\]

- \( \mathcal{LC}_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( \mathcal{LC}_E(g_1, x) = \frac{i}{2} \)

- For global result:
  - consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))

\( \Rightarrow \) add expected size bound \( S_E(g_0, x) \) to \( \mathcal{LC}_E(g_1, x) \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = S_E(g_0, x) + \frac{1}{2} \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v)[u/\max(S(t', u), ...)] \]

- **LC_E(g, v):** expect. local change by g
- **g', t':** pre-transition of g

- **LC_E(g_1, x) = \frac{1}{2}**
- For global result:
  - consider expected value of x before reaching \( g_1 \) (after \( g_0 \))

\[ \Rightarrow \text{add expected size bound } S_E(g_0, x) \text{ to } LC_E(g_1, x) \]
**Expected Size Bounds**

### Expected size bounds

\[
S_E(g_0, \nu) = \nu_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + \frac{i}{2}
\]

### Computing expected size bound for variable \( \nu \) after transition \( g \)

\[
S_E(g, \nu) = S_E(g', \nu) + R_E(g) \cdot LC_E(g, \nu) [u/ \max(S(t', u), ...)]
\]

- \( LC_E(g, \nu) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{i}{2} \)

- For global result:
  - consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))

\[ \Rightarrow \text{add expected size bound } S_E(g_0, x) \text{ to } LC_E(g_1, x) \]
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + \frac{i_1}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v)[u/\max(S(t', u), \ldots)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{i_2}{2} \)
- For global result:
  - consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
  - consider how often \( g_1 \) is expected to be executed

⇒ add expected size bound \( S_E(g_0, x) \) to \( LC_E(g_1, x) \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + \frac{i}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u/ \max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_1, x) = \frac{i}{2} \]

For global result:
- consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed

\[ \Rightarrow \text{multiply } g_1 \text{'s expected runtime bound } R_E(g_1) \text{ with local change } LC_E(g_1, x) \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + R_E(g_1) \cdot \frac{1}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot \mathcal{LC}_E(g, v) \left[ u/\max(S(t', u), \ldots) \right] \]

- \( \mathcal{LC}_E(g, v) \): expected local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( \mathcal{LC}_E(g_1, x) = \frac{1}{2} \)

For global result:
- consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed

\[ \Rightarrow \text{multiply } g_1 \text{'s expected runtime bound } R_E(g_1) \text{ with local change } \mathcal{LC}_E(g_1, x) \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0, \ S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{1}{2} \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u/ \text{max}(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_1, x) = \frac{1}{2} \]

For global result:
- consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed

⇒ multiply \( g_1 \)'s expected runtime bound \( R_E(g_1) \) with local change \( LC_E(g_1, x) \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{i}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) \cdot [u/ \max(S(t', u), \ldots)] \]

- **\( LC_E(g, v) \):** *expected* local change by \( g \)
- **\( g', t' \):** pre-transition of \( g \)

- **\( LC_E(g_1, x) = \frac{i}{2} \)**

- For global result:
  - consider *expected* value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
  - consider how often \( g_1 \) is *expected* to be executed
  - consider values of \( LC_E(g_1, x) \)'s variables in full run

\[ ⇒ \text{ multiply } g_1 \text{'s } \text{expected } \text{runtime bound } R_E(g_1) \text{ with local change } LC_E(g_1, x) \]
**Expected Size Bounds**

### Expected size bounds

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{1}{2} \]

### Computing expected size bound for variable \( v \) after transition \( g \)

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v)[u/ \max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{1}{2} \)

For global result:
- consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed
- consider values of \( LC_E(g_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC_E(g_1, x) \text{ by } LC_E(g_1, x)[i/ \max(S(t_0, i), S(t_1, i), S(t_4, i))] \]
**Expected Size Bounds**

**Expected size bounds**
\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{1}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LCE_E(g, v) \cdot [u/\max(S(t', u), \ldots)] \]

- **\( LCE_E(g, v) \): expect. local change by \( g \)**
- **\( g', t' \): pre-transition of \( g \)**

\[ LCE_E(g_1, x) = \frac{1}{2} \]

- For global result:
  - consider *expected* value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
  - consider how often \( g_1 \) is *expected* to be executed
  - consider values of \( LCE_E(g_1, x) \)'s variables in full run

⇒ replace \( LCE_E(g_1, x) \) by \( LCE_E(g_1, x)[i / \max(0, i_0, 0, 0)] \)
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \; S_E(g_1, i) = i_0, \; S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{1}{2} \]

**Computing expected size bound for variable** \( v \) **after transition** \( g \)

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) \left[ u/\max(S(t', u), \ldots) \right] \]

- \( LC_E(g, v) \): **expected** local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{1}{2} \)

For global result:
- consider **expected** value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is **expected** to be executed
- consider values of \( LC_E(g_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC_E(g_1, x) \text{ by } LC_E(g_1, x)[i/i_0] \]
**(Expected Size Bounds)**

Expected size bounds:

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{1}{2} [i / i_0] \]

Computing *expected* size bound for variable \( v \) after transition \( g \):

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) \cdot [u / \max(S(t', u), ...)] \]

- \( LC_E(g, v) \): *expected* local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LC_E(g_1, x) = \frac{1}{2} \)

For global result:

- consider *expected* value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is *expected* to be executed
- consider values of \( LC_E(g_1, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC_E(g_1, x) \text{ by } LC_E(g_1, x)[i / i_0] \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + 2 \cdot i_0 \cdot \frac{i_0}{2} \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LCE_E(g, v) [u/ \max(S(t', u), \ldots)] \]

- \( LCE_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

**For global result:**
- consider *expected* value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed
- consider values of \( LCE_E(g_1, x) \)'s variables in full run

⇒ replace \( LCE_E(g_1, x) \) by \( LCE_E(g_1, x)[i / i_0] \)
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) \left[ u / \max(S(t', u), ...) \right] \]

- **LC_E(g, v):** expected local change by \( g \)
- **g', t':** pre-transition of \( g \)

**LC_E(g_1, x) = \frac{1}{2}**

For global result:
- consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- consider how often \( g_1 \) is expected to be executed
- consider values of \( LC_E(g_1, x) \)'s variables in full run

\[ t_0 \in g_0 \]

\[ \frac{1}{2} : t_1 \in g_1 \]

\[ \text{if} (i > 0) \]

\[ x = x + i \]

\[ i = i - 1 \]

\[ t_2 \in g_2 \]

\[ \text{if} (i \leq 0) \]

\[ t_3 \in g_3 \]

\[ \text{if} (x > 0) \]

\[ x = x - 1 \]

\[ \Rightarrow \text{replace } LC_E(g_1, x) \text{ by } LC_E(g_1, x)[i / i_0] \]
Expected Size Bounds

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u/ \max(S(t', u), \ldots)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_2, x) = 0 \]

For global result:
- Consider expected value of \( x \) before reaching \( g_1 \) (after \( g_0 \))
- Consider how often \( g_1 \) is expected to be executed
- Consider values of \( LC_E(g_1, x) \)'s variables in full run

⇒ replace \( LC_E(g_1, x) \) by \( LC_E(g_1, x) [i / i_0] \)
Expected Size Bounds

Expected size bounds

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0, \ S_E(g_1, x) = x_0 + i_0^2 \]

Computing expected size bound for variable \( v \) after transition \( g \)

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v)[u/\max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_2, x) = 0 \]

For global result:
- consider expected value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is expected to be executed
- consider values of \( LC_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \text{replace } LC_E(g_1, x) \text{ by } LC_E(g_1, x)[i/i_0] \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot \mathcal{LC}_E(g, v) \left[ u / \max(S(t', u), ...) \right] \]

- \( \mathcal{LC}_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ \mathcal{LC}_E(g_2, x) = 0 \]

For global result:
- consider expected value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is expected to be executed
- consider values of \( \mathcal{LC}_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \text{add } \max(S_E(g_0, x), S_E(g_1, x)) \text{ to } \mathcal{LC}_E(g_2, x) \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i^2 \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v)[u/\max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_2, x) = 0 \]

For global result:
- consider expected value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is expected to be executed
- consider values of \( LC_E(g_2, x) \)'s variables in full run

⇒ add \( \max(x_0, S_E(g_1, x)) \) to \( LC_E(g_2, x) \)
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \ S_E(g_1, i) = i_0, \ S_E(g_1, x) = x_0 + i^2_0 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + \mathcal{R}_E(g) \cdot \mathcal{LC}_E(g, v) \cdot [u / \max(S(t', u), ...)] \]

- \( \mathcal{LC}_E(g, v) \): *expect.* local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( \mathcal{LC}_E(g_2, x) = 0 \)

For global result:
- consider *expected* value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is *expected* to be executed
- consider values of \( \mathcal{LC}_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \text{add} \ \max(x_0, \ x_0 + i^2_0) \ \text{to} \ \mathcal{LC}_E(g_2, x) \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LC_E(g, v) [u / \max(S(t', u), ...)] \]

- \( LC_E(g, v) \): expected local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

\[ LC_E(g_2, x) = 0 \]

For global result:
- consider expected value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is expected to be executed
- consider values of \( LC_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \text{ add } x_0 + i_0^2 \text{ to } LC_E(g_2, x) \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 \]

**Computing expected size bound for variable \( v \) after transition \( g \)**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot LCE_E(g, v) \cdot [u/ \max(S(t', u), \ldots)] \]

- \( LCE_E(g, v) \): expect. local change by \( g \)
- \( g', t' \): pre-transition of \( g \)

- \( LCE_E(g_2, x) = 0 \)

For global result:
- consider expected value of \( x \) before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
- consider how often \( g_2 \) is expected to be executed
- consider values of \( LCE_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \quad \text{add} \quad x_0 + i_0^2 \quad \text{to} \quad 0 \]
**Expected Size Bounds**

**Expected size bounds**

\[ S_E(g_0, v) = v_0, \quad S_E(g_1, i) = i_0, \quad S_E(g_1, x) = x_0 + i_0^2 = S_E(g_2, x) \]

**Computing expected size bound for variable v after transition g**

\[ S_E(g, v) = S_E(g', v) + R_E(g) \cdot \mathcal{LC}_E(g, v) \left[ u / \max(S(t', u), \ldots) \right] \]

- \( \mathcal{LC}_E(g, v) \): expect. local change by g
- \( g', t' \): pre-transition of g

- \( \mathcal{LC}_E(g_2, x) = 0 \)

- For global result:
  - consider expected value of x before reaching \( g_2 \) (after \( g_0 \) or \( g_1 \))
  - consider how often \( g_2 \) is expected to be executed
  - consider values of \( \mathcal{LC}_E(g_2, x) \)'s variables in full run

\[ \Rightarrow \quad \text{add} \quad x_0 + i_0^2 \quad \text{to} \quad 0 \]
Inferring *Expected* Runtimes of *Probabilistic* Programs

Implementation in KoAT
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in *KoAT*
  - all 46 benchmarks from *Absynth*
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in *KoAT*
  - all 46 benchmarks from *Absynth*
  - 29 new benchmarks including examples
    from TPDB enriched with randomization
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in KoAT
  - all 46 benchmarks from Absynth
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in **KoAT**
  - all 46 benchmarks from **Absynth**
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(1))</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>(O(n))</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>(O(n^2))</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>(O(n^2))</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\infty)</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg. Time**
- 4.26 s
- 3.53 s
- 0.93 s

**Success**
- 91%
- 68%
- 77%
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in **KoAT**
  - all 46 benchmarks from **Absynth**
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg. Time**

- 4.26 s
- 3.53 s
- 0.93 s
Inferring *Expected* Runtimes of *Probabilistic* Programs

- Implementation in **KoAT**
  - all 46 benchmarks from **Absynth**
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td><strong>Avg. Time</strong></td>
<td>4.26 s</td>
<td>3.53 s</td>
<td>0.93 s</td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>91 %</td>
<td>68 %</td>
<td>77 %</td>
</tr>
</tbody>
</table>
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected **runtime** and **size** bounds

**Implementation in KoAT**
- all 46 benchmarks from **Absynth**
- 29 new benchmarks including examples from TPDB enriched with randomization
- timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

| **Avg. Time** | 4.26 s | 3.53 s | 0.93 s |
| **Success**   | 91 %   | 68 %   | 77 %   |
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected runtime and size bounds
  - compute size bounds by combining local change bound with runtime bounds

- Implementation in KoAT
  - all 46 benchmarks from Absynth
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td><strong>Avg. Time</strong></td>
<td>4.26 s</td>
<td>3.53 s</td>
<td>0.93 s</td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>91 %</td>
<td>68 %</td>
<td>77 %</td>
</tr>
</tbody>
</table>
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected **runtime** and **size bounds**
  - compute **size bounds** by combining local change bound with **runtime bounds**
  - compute **runtime bounds** for program parts based on **size bounds** for preceding parts

---

**Implementation in KoAT**

- all 46 benchmarks from **Absynth**
- 29 new benchmarks including examples from TPDB enriched with randomization
- timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>(O(1))</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>(O(n))</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>(O(n^2))</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>(O(n^{&lt;2}))</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(\infty)</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td><strong>Avg. Time</strong></td>
<td>4.26 s</td>
<td>3.53 s</td>
<td>0.93 s</td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>91 %</td>
<td>68 %</td>
<td>77 %</td>
</tr>
</tbody>
</table>
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected runtime and size bounds
  - compute size bounds by combining local change bound with runtime bounds
  - compute runtime bounds for program parts based on size bounds for preceding parts
  - based on both expected and non-probabilistic bounds for program parts

- Implementation in *KoAT*
  - all 46 benchmarks from *Absynth*
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>∞</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

| Avg. Time | 4.26 s | 3.53 s | 0.93 s |
| Success   | 91 %   | 68 %   | 77 %   |
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected **runtime** and **size** bounds
  - compute **size bounds** by combining local change bound with **runtime bounds**
  - compute **runtime bounds** for program parts based on **size bounds** for preceding parts
  - based on both *expected* and *non-probabilistic* bounds for program parts

- **modular:** only consider small program parts at a time

Implementation in **KoAT**

- all 46 benchmarks from **Absynth**
- 29 new benchmarks including examples from TPDB enriched with randomization
- timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td><strong>Avg. Time</strong></td>
<td>4.26 s</td>
<td>3.53 s</td>
<td>0.93 s</td>
</tr>
<tr>
<td><strong>Success</strong></td>
<td>91 %</td>
<td>68 %</td>
<td>77 %</td>
</tr>
</tbody>
</table>
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected runtime and size bounds
  - compute size bounds by combining local change bound with runtime bounds
  - compute runtime bounds for program parts based on size bounds for preceding parts
  - based on both expected and non-probabilistic bounds for program parts

- **modular**: only consider small program parts at a time
  - linear probabilistic ranking functions

- Implementation in *KoAT*
  - all 46 benchmarks from *Absynth*
  - 29 new benchmarks including examples from TPDB enriched with randomization
  - timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{O}(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$\mathcal{O}(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$\mathcal{O}(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$\mathcal{O}(n^{&gt;2})$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

| Avg. Time | 4.26 s | 3.53 s | 0.93 s |
| Success  | 91 %   | 68 %   | 77 %   |
Inferring *Expected* Runtimes of *Probabilistic* Programs

- alternate finding expected *runtime* and *size bounds*
  - compute *size bounds* by combining local change bound with *runtime bounds*
  - compute *runtime bounds* for program parts based on *size bounds* for preceding parts
  - based on both *expected* and *non-probabilistic* bounds for program parts

- **modular**: only consider small program parts at a time
  - linear probabilistic ranking functions
  - approach scales to larger programs

Implementation in **KoAT**
- all 46 benchmarks from **Absynth**
- 29 new benchmarks including examples from TPDB enriched with randomization
- timeout of 5 minutes

<table>
<thead>
<tr>
<th>Bound</th>
<th>KoAT</th>
<th>Absynth</th>
<th>eco-imp</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{O}(1)$</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>$\mathcal{O}(n)$</td>
<td>42</td>
<td>35</td>
<td>35</td>
</tr>
<tr>
<td>$\mathcal{O}(n^2)$</td>
<td>15</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>$\mathcal{O}(n^3)$</td>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>EXP</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\infty$</td>
<td>7</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>TO</td>
<td>0</td>
<td>9</td>
<td>3</td>
</tr>
</tbody>
</table>

**Avg. Time**: 4.26 s, 3.53 s, 0.93 s

**Success**: 91%, 68%, 77%