Operational Semantics, Soundness, Laziness

Albert Y. C. Lai

Department of Computer Science
University of Toronto

June 10, 2008

1. Operational Semantics
2. Soundness of Refinements
3. Lazy Execution
4. Calculating Lazy Timing
5. Conclusions

Programming Constructs

- $\sigma := e$ or $x, y := a, b$
- $P \,.\, Q$
- $\textbf{if}\ b\ \textbf{then}\ P\ \textbf{else}\ Q$
- specification S, provided refinement $S \Leftarrow P$
  recursion allowed

Operational Semantics

Operational semantics as rewrite rules:

$$
\begin{array}{rcl}
\sigma{:=}k\,.\,\sigma{:=}e & \rightarrow & \sigma{:=}\langle\sigma\!\rightarrow e\rangle\, k \\
\sigma{:=}k\,.\,\textbf{if}\ b\ \textbf{then}\ P\ \textbf{else}\ Q & \rightarrow & \textbf{if}\ \langle\sigma\!\rightarrow b\rangle\, k\ \textbf{then}\ (\sigma{:=}k\,.\,P)\ \textbf{else}\ (\sigma{:=}k\,.\,Q) \\
\textbf{if}\ \top\ \textbf{then}\ P\ \textbf{else}\ Q & \rightarrow & P \\
\textbf{if}\ \bot\ \textbf{then}\ P\ \textbf{else}\ Q & \rightarrow & Q \\
S & \xrightarrow{1} & P\ \text{(provided } S \Longleftarrow P\text{)}
\end{array}
$$

Note: no order specified yet. This talk will describe two.

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\ .\ Q\,(n{+}1))$$

Execution Example

$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\ .\ Q\,(n{+}1))$

$L{:=}[0;1;2]\ .\ s{:=}0\ .\ Q\,0$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\ .\ Q\,(n{+}1))$$

$$
\begin{array}{ll}
 & L{:=}[0;1;2]\ .\ s{:=}0\ .\ Q\,0 \\
\rightarrow & s, L{:=}0, [0;1;2]\ .\ Q\,0
\end{array}
$$

Execution Example

$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \,.\, Q\,(n{+}1))$

$$
\begin{array}{ll}
 & L{:=}[0;1;2]\,.\,s{:=}0\,.\,Q\,0 \\
\rightarrow & s, L{:=}0, [0;1;2]\,.\,Q\,0 \\
\rightarrow & s, L{:=}0, [0;1;2]\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0 \,.\, Q\,1)
\end{array}
$$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \,.\, Q\,(n{+}1))$$

$$\begin{array}{ll} & L{:=}[0;1;2] \,.\, s{:=}0 \,.\, Q\,0 \\ \to & s, L{:=}0, [0;1;2] \,.\, Q\,0 \\ \to & s, L{:=}0, [0;1;2] \,.\, \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0 \,.\, Q\,1) \\ \to & s, L{:=}0, [0;1;2] \,.\, s := s + L\,0 \,.\, Q\,1 \end{array}$$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s+L\,n\ .\ Q\,(n{+}1))$$

$$\begin{array}{ll}
 & L{:=}[0;1;2]\ .\ s{:=}0\ .\ Q\,0 \\
\to & s, L{:=}0, [0;1;2]\ .\ Q\,0 \\
\to & s, L{:=}0, [0;1;2]\ .\ \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s+L\,0\ .\ Q\,1) \\
\to & s, L{:=}0, [0;1;2]\ .\ s := s+L\,0\ .\ Q\,1 \\
\to & s, L{:=}0, [0;1;2]\ .\ Q\,1
\end{array}$$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \,.\, Q\,(n{+}1))$$

$$
\begin{array}{ll}
 & L{:=}[0;\,1;\,2] \,.\, s{:=}0 \,.\, Q\,0 \\
\to & s, L{:=}0, [0;\,1;\,2] \,.\, Q\,0 \\
\to & s, L{:=}0, [0;\,1;\,2] \,.\, \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0 \,.\, Q\,1) \\
\to & s, L{:=}0, [0;\,1;\,2] \,.\, s := s + L\,0 \,.\, Q\,1 \\
\to & s, L{:=}0, [0;\,1;\,2] \,.\, Q\,1 \\
\to & s, L{:=}1, [0;\,1;\,2] \,.\, Q\,2
\end{array}
$$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \;.\; Q\,(n{+}1))$$

$$\begin{array}{ll}
 & L{:=}[0;1;2] \,.\, s{:=}0 \,.\, Q\,0 \\
\to & s, L{:=}0, [0;1;2] \,.\, Q\,0 \\
\to & s, L{:=}0, [0;1;2] \,.\, \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0 \;.\; Q\,1) \\
\to & s, L{:=}0, [0;1;2] \;.\; s := s + L\,0 \;.\; Q\,1 \\
\to & s, L{:=}0, [0;1;2] \,.\, Q\,1 \\
\to & s, L{:=}1, [0;1;2] \,.\, Q\,2 \\
\to & s, L{:=}1, [0;1;2] \,.\, ok
\end{array}$$

Execution Example

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s+L\,n \,.\, Q\,(n{+}1))$$

$$\begin{array}{ll}
 & L{:=}[0;1;2] \,.\, s{:=}0 \,.\, Q\,0 \\
\to & s, L{:=}0, [0;1;2] \,.\, Q\,0 \\
\to & s, L{:=}0, [0;1;2] \,.\, \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s+L\,0 \,.\, Q\,1) \\
\to & s, L{:=}0, [0;1;2] \,.\, s := s+L\,0 \,.\, Q\,1 \\
\to & s, L{:=}0, [0;1;2] \,.\, Q\,1 \\
\to & s, L{:=}1, [0;1;2] \,.\, Q\,2 \\
\to & s, L{:=}1, [0;1;2] \,.\, ok \\
\to & s, L{:=}1, [0;1;2]
\end{array}$$

Eager Execution Defined

1. To run S with initial value j: Start with $\sigma := j \,.\, S$.

Eager Execution Defined

1. To run S with initial value j: Start with $\sigma{:=}j\,.\,S$.
2. At each step, use the first one applicable to the left:
   - looks like $\sigma{:=}k\,.\,S$: use

$$S \quad \xrightarrow{1} \quad \text{what refines } S$$

   - use the (unique) matching rule from

$$\begin{array}{rcl}
\sigma{:=}k\,.\,\sigma{:=}e & \to & \sigma{:=}\langle\sigma\to e\rangle\,k \\
\sigma{:=}k\,.\,\mathbf{if}\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q & \to & \ldots \\
\mathbf{if}\ \top\ \mathbf{then}\ P\ \mathbf{else}\ Q & \to & P \\
\mathbf{if}\ \bot\ \mathbf{then}\ P\ \mathbf{else}\ Q & \to & Q \\
S & \xrightarrow{1} & \text{what refines } S
\end{array}$$

Eager Execution Defined

1. To run S with initial value j: Start with $\sigma := j \,.\, S$.
2. At each step, use the first one applicable to the left:
   - looks like $\sigma := k \,.\, S$: use

$$S \xrightarrow{1} \text{what refines } S$$

   - use the (unique) matching rule from

$$\begin{array}{rcl} \sigma := k \,.\, \sigma := e & \to & \sigma := \langle \sigma \to e \rangle\, k \\ \sigma := k \,.\, \textbf{if } b \textbf{ then } P \textbf{ else } Q & \to & \ldots \\ \textbf{if } \top \textbf{ then } P \textbf{ else } Q & \to & P \\ \textbf{if } \bot \textbf{ then } P \textbf{ else } Q & \to & Q \\ S & \xrightarrow{1} & \text{what refines } S \end{array}$$

3. Stop when the whole program is just $\sigma := k$.

Operational Semantics, Soundness, Laziness

Eager Soundness

Can prove: refinement $\Rightarrow$ execution

Eager Soundness

Can prove: refinement $\Rightarrow$ execution

- Safety:

$$\forall n, j, k \cdot (\sigma{:=}j\,.\,S \xrightarrow{n} \sigma{:=}k) \Rightarrow \langle \sigma, \sigma' \rightarrow S \rangle\; j\; k$$

Eager Soundness

Can prove: refinement $\Rightarrow$ execution

- Safety:

$$\forall n, j, k \cdot (\sigma := j \,.\, S \xrightarrow{n} \sigma := k) \Rightarrow \langle \sigma, \sigma' \rightarrow S \rangle\; j\; k$$

- Liveness:

$$(\text{“}t := t+1\text{” inserted}) \wedge \forall \sigma \cdot \exists \sigma' \cdot S$$

$$\Rightarrow \quad \forall n, j \cdot (\forall \sigma' \cdot \langle \sigma \rightarrow S \rangle\; j \Rightarrow t' \leq t+n) \Rightarrow (\exists k \cdot \sigma := j \,.\, S \xrightarrow{\leq n} \sigma := k)$$

$$\Rightarrow \quad (\forall \sigma, \sigma' \cdot S \Rightarrow t' \leq t + f\, \sigma) \Rightarrow (\forall j \cdot \exists k \cdot \sigma := j \,.\, S \xrightarrow{\leq f\, j} \sigma := k)$$

Exact Precondition for Termination

139. Define "exact precondition for termination" as

$$\exists n \cdot \forall \sigma' \cdot S \Rightarrow t' \le t+n$$

Comment on whether it is reasonable.

Exact Precondition for Termination

139. Define "exact precondition for termination" as

$$\exists n\cdot \forall\sigma'\cdot S \Rightarrow t' \le t{+}n$$

Comment on whether it is reasonable.

§ $\quad \forall n\cdot (\forall\sigma'\cdot S \Rightarrow t' \le t{+}n) \Rightarrow (\exists k\cdot S \xrightarrow{\le n} \sigma{:=}k)$

$= \quad \forall n\cdot (\forall\sigma'\cdot S \Rightarrow t' \le t{+}n) \Rightarrow S$ terminates

$= \quad (\exists n\cdot \forall\sigma'\cdot S \Rightarrow t' \le t{+}n) \Rightarrow S$ terminates

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.

Proving Implementability

- Prove $\forall \sigma \cdot \exists \sigma' \cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.
- Is the refinement sound? Prove $\forall \sigma \cdot \exists \sigma' \cdot S$ ...

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.
- Is the refinement sound? Prove $\forall\sigma\cdot\exists\sigma'\cdot S$ ...

Don't despair. *Bootstrap*!

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.
- Is the refinement sound? Prove $\forall\sigma\cdot\exists\sigma'\cdot S$ ...

Don't despair. *Bootstrap*!

1. $S \Leftarrow \textbf{if } g \textbf{ then } B\,.\,S \textbf{ else } ok$
   $t' \le t + f\,\sigma \Leftarrow \textbf{if } g \textbf{ then } B\,.\,t' \le t + f\,\sigma \textbf{ else } ok$

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.
- Is the refinement sound? Prove $\forall\sigma\cdot\exists\sigma'\cdot S$ ...

Don't despair. *Bootstrap*!

1. $S \Leftarrow \textbf{if } g \textbf{ then } B\,.\,S \textbf{ else } ok$
   $t' \le t + f\,\sigma \Leftarrow \textbf{if } g \textbf{ then } B\,.\,t' \le t + f\,\sigma \textbf{ else } ok$
2. $\forall\sigma\cdot\exists\sigma'\cdot t' \le t + f\,\sigma$ clearly yes.
   liveness theorem $\Rightarrow$ $\sigma{:=}j\,.\,t' \le t + f\,\sigma \xrightarrow{\le f\,j} \sigma{:=}k$

Proving Implementability

- Prove $\forall\sigma\cdot\exists\sigma'\cdot S$? Write an algorithm to find σ'.
- Is the algorithm correct? Prove the refinement.
- Is the refinement sound? Prove $\forall\sigma\cdot\exists\sigma'\cdot S$ ...

Don't despair. *Bootstrap*!

1. $S \Leftarrow \textbf{if } g \textbf{ then } B\,.\,S \textbf{ else } ok$
   $t' \le t + f\,\sigma \Leftarrow \textbf{if } g \textbf{ then } B\,.\,t' \le t + f\,\sigma \textbf{ else } ok$
2. $\forall\sigma\cdot\exists\sigma'\cdot t' \le t + f\,\sigma$ clearly yes.
   liveness theorem $\Rightarrow$ $\sigma{:=}j\,.\,t' \le t + f\,\sigma \xrightarrow{\le f\,j} \sigma{:=}k$
3. Re-label execution trace: $\sigma{:=}j\,.\,S \xrightarrow{\le f\,j} \sigma{:=}k$
   safety theorem $\Rightarrow$ k is a witness $\Rightarrow$ $\exists\sigma'\cdot S$

Operational Semantics

Operational semantics as rewrite rules:

$$
\begin{array}{rcl}
\sigma{:=}k \,.\, \sigma{:=}e & \rightarrow & \sigma{:=}\langle \sigma \rightarrow e \rangle\, k \\
\sigma{:=}k \,.\, \textbf{if } b \textbf{ then } P \textbf{ else } Q & \rightarrow & \textbf{if } \langle \sigma \rightarrow b \rangle\, k \textbf{ then } (\sigma{:=}k \,.\, P) \textbf{ else } (\sigma{:=}k \,.\, Q) \\
\textbf{if } \top \textbf{ then } P \textbf{ else } Q & \rightarrow & P \\
\textbf{if } \bot \textbf{ then } P \textbf{ else } Q & \rightarrow & Q \\
S & \xrightarrow{1} & P \text{ (provided } S \Leftarrow P\text{)}
\end{array}
$$

New order coming up.

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$P\,0$$

Execution Example

$$P\,m \Leftarrow P\,(m+1)\,.\,L{:=}[m]^{+}L$$

$$P\,0$$

$$\rightarrow \quad P\,1\,.\,L{:=}[0]^{+}L$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$\begin{array}{ll} & P\,0 \\ \rightarrow & P\,1\,.\,L{:=}[0]^{+}L \\ \rightarrow & P\,2\,.\,L{:=}[1]^{+}L\,.\,L{:=}[0]^{+}L \end{array}$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$\begin{array}{ll} & P\,0 \\ \to & P\,1\,.\,L{:=}[0]^{+}L \\ \to & P\,2\,.\,L{:=}[1]^{+}L\,.\,L{:=}[0]^{+}L \\ \to & P\,2\,.\,L{:=}[0;\,1]^{+}L \end{array}$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$
\begin{array}{ll}
 & P\,0 \\
\to & P\,1\,.\,L{:=}[0]^{+}L \\
\to & P\,2\,.\,L{:=}[1]^{+}L\,.\,L{:=}[0]^{+}L \\
\to & P\,2\,.\,L{:=}[0;\,1]^{+}L \\
\to & P\,3\,.\,L{:=}[2]^{+}L\,.\,L{:=}[0;\,1]^{+}L
\end{array}
$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$\begin{array}{ll}
 & P\,0 \\
\to & P\,1\,.\,L{:=}[0]^{+}L \\
\to & P\,2\,.\,L{:=}[1]^{+}L\,.\,L{:=}[0]^{+}L \\
\to & P\,2\,.\,L{:=}[0;\,1]^{+}L \\
\to & P\,3\,.\,L{:=}[2]^{+}L\,.\,L{:=}[0;\,1]^{+}L \\
\to & P\,3\,.\,L{:=}[0;\,1;\,2]^{+}L
\end{array}$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\ .\ Q\,(n{+}1))$$

Execution Example

$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$

$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s+L\,n\ .\ Q\,(n{+}1))$

$P\,0\,.\,s{:=}0\,.\,Q\,0$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$
$$Q\,n \Leftarrow \mathbf{if}\ n{=}2\ \mathbf{then}\ ok\ \mathbf{else}\ (s := s + L\,n\ .\ Q\,(n{+}1))$$

$$P\,0\,.\,s{:=}0\,.\,Q\,0$$
$$\rightarrow \quad P\,0\,.\,s{:=}0\,.\,\mathbf{if}\ 0{=}2\ \mathbf{then}\ ok\ \mathbf{else}\ (s := s + L\,0\ .\ Q\,1)$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,Q\,(n{+}1))$$

$$P\,0\,.\,s{:=}0\,.\,Q\,0$$

$$\rightarrow \quad P\,0\,.\,s{:=}0\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0\,.\,Q\,1)$$

$$\rightarrow \quad P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,Q\,1$$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$
$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,Q\,(n{+}1))$$

$$P\,0\,.\,s{:=}0\,.\,Q\,0$$
$$\to\quad P\,0\,.\,s{:=}0\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0\,.\,Q\,1)$$
$$\to\quad P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,Q\,1$$

many iterations

$$\to\quad P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,s := s + L\,1\,.\,ok$$

Execution Example

$$P\,m \Longleftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Longleftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,Q\,(n{+}1))$$

$$
\begin{array}{ll}
 & P\,0\,.\,s{:=}0\,.\,Q\,0 \\
\to & P\,0\,.\,s{:=}0\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0\,.\,Q\,1) \\
\to & P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,Q\,1 \\
\to & P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,s := s + L\,1\,.\,ok \\
\to & P\,0\,.\,s := L\,0 + L\,1
\end{array}
$$

many iterations

Execution Example

$$P\,m \Leftarrow P\,(m+1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,Q\,(n+1))$$

$P\,0\,.\,s{:=}0\,.\,Q\,0$

$\rightarrow\quad P\,0\,.\,s{:=}0\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0\,.\,Q\,1)$

$\rightarrow\quad P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,Q\,1$ many iterations

$\rightarrow\quad P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,s := s + L\,1\,.\,ok$

$\rightarrow\quad P\,0\,.\,s := L\,0 + L\,1$

$\rightarrow\quad P\,1\,.\,L{:=}[0]^{+}L\,.\,s := L\,0 + L\,1$

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Leftarrow \textbf{if}\ n{=}2\ \textbf{then}\ ok\ \textbf{else}\ (s := s + L\,n\ .\ Q\,(n{+}1))$$

$$
\begin{array}{ll}
 & P\,0\,.\,s{:=}0\,.\,Q\,0 \\
\to & P\,0\,.\,s{:=}0\,.\,\textbf{if}\ 0{=}2\ \textbf{then}\ ok\ \textbf{else}\ (s := s + L\,0\ .\ Q\,1) \\
\to & P\,0\ .\ s{:=}0\ .\ s := s + L\,0\ .\ Q\,1 \\
\to & P\,0\ .\ s{:=}0\ .\ s := s + L\,0\ .\ s := s + L\,1\ .\ ok \\
\to & P\,0\ .\ s := L\,0 + L\,1 \\
\to & P\,1\ .\ L{:=}[0]^{+}L\ .\ s := L\,0 + L\,1 \\
\to & P\,1\ .\ s, L := 0 + L\,1\,,\ [0]^{+}L
\end{array}
$$

many iterations

Execution Example

$$P\,m \Leftarrow P\,(m+1) \,.\, L{:=}[m]^{+}L$$
$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \,.\, Q\,(n+1))$$

$$\begin{array}{ll}
 & P\,0 \,.\, s{:=}0 \,.\, Q\,0 \\
\to & P\,0 \,.\, s{:=}0 \,.\, \textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0 \,.\, Q\,1) \\
\to & P\,0 \,.\, s{:=}0 \,.\, s := s + L\,0 \,.\, Q\,1 \\
\to & P\,0 \,.\, s{:=}0 \,.\, s := s + L\,0 \,.\, s := s + L\,1 \,.\, ok \\
\to & P\,0 \,.\, s := L\,0 + L\,1 \\
\to & P\,1 \,.\, L{:=}[0]^{+}L \,.\, s := L\,0 + L\,1 \\
\to & P\,1 \,.\, s, L := 0 + L\,1,\ [0]^{+}L \\
\to & P\,2 \,.\, L{:=}[1]^{+}L \,.\, s, L := 0 + L\,1,\ [0]^{+}L
\end{array}$$

many iterations

Execution Example

$$P\,m \Leftarrow P\,(m{+}1)\,.\,L{:=}[m]^{+}L$$

$$Q\,n \Leftarrow \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,Q\,(n{+}1))$$

$$
\begin{array}{ll}
 & P\,0\,.\,s{:=}0\,.\,Q\,0 \\
\to & P\,0\,.\,s{:=}0\,.\,\textbf{if } 0{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,0\,.\,Q\,1) \\
\to & P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,Q\,1 \\
\to & P\,0\,.\,s{:=}0\,.\,s := s + L\,0\,.\,s := s + L\,1\,.\,ok \\
\to & P\,0\,.\,s := L\,0 + L\,1 \\
\to & P\,1\,.\,L{:=}[0]^{+}L\,.\,s := L\,0 + L\,1 \\
\to & P\,1\,.\,s, L := 0 + L\,1,\ [0]^{+}L \\
\to & P\,2\,.\,L{:=}[1]^{+}L\,.\,s, L := 0 + L\,1,\ [0]^{+}L \\
\to & P\,2\,.\,s, L := 1,\ [0;\,1]^{+}L
\end{array}
$$

many iterations

Lazy Execution Defined

1. Decide which variables you want.

Lazy Execution Defined

1. Decide which variables you want.
2. With initial value j, start with $\sigma := j \,.\, S$

Lazy Execution Defined

1. Decide which variables you want.
2. With initial value j, start with $\sigma{:=}j\,.\,S$
3. Stop when the rightmost looks like $\sigma{:=}e$
   and
   it assigns constants to variables you want.

Lazy Execution Defined

1. Decide which variables you want.
2. With initial value j, start with $\sigma{:=}j\,.\,S$
3. Stop when the rightmost looks like $\sigma{:=}e$
   and
   it assigns constants to variables you want.
4. Otherwise, rewrite the rightmost, *unless*. . .

Lazy Execution Defined

1. Decide which variables you want.
2. With initial value j, start with $\sigma{:=}j\,.\,S$
3. Stop when the rightmost looks like $\sigma{:=}e$
   and
   it assigns constants to variables you want.
4. Otherwise, rewrite the rightmost, *unless*. . .
5. Rightmost is conditional: $\sigma{:=}j\,.\,R\,.\,\mathbf{if}\ b\ \mathbf{then}\ P\ \mathbf{else}\ Q$
   Suspend. You want variables in b. Execute $\sigma{:=}j\,.\,R$
   Resolve conditional. Resume.

Applications & Difficulty of Lazy Execution

- Programming application: producer-consumer.

Applications & Difficulty of Lazy Execution

- Programming application: producer-consumer.
- Theoretic application: most terminating.

Applications & Difficulty of Lazy Execution

- Programming application: producer-consumer.
- Theoretic application: most terminating.
- Difficulty: execution time less obvious.
  I will show you how to do it.

Applications & Difficulty of Lazy Execution

- Programming application: producer-consumer.
- Theoretic application: most terminating.
- Difficulty: execution time less obvious.
  I will show you how to do it.

Remark: *Not* a difficulty: I/O order. Lazy execution for internal computation between two I/O actions *only*.

Operational Semantics, Soundness, Laziness

Usage Variables

- For each state variable v,v', a usage variable u_v,u'_v (boolean).
  u_v = “v is used”
  u'_v = “v' is used”
  Array needs usage array.

Usage Variables

- For each state variable v, v', a usage variable u_v, u'_v (boolean).
  $u_v \;\boldsymbol{=}\;$ "v is used"
  $u'_v \;\boldsymbol{=}\;$ "v' is used"
  Array needs usage array.
- $x'=x \wedge y'=y \wedge t'=t \wedge u_x=e' \wedge u_y=u'_y \;\boldsymbol{=}\; u_x{=}{:}e'$

Usage Variables

- For each state variable v, v', a usage variable u_v, u'_v (boolean).
  u_v **=** "v is used"
  u'_v **=** "v' is used"
  Array needs usage array.
- $x'=x \wedge y'=y \wedge t'=t \wedge u_x=e' \wedge u_y=u'_y$ **=** $u_x=:e'$
- Annotate assignment statements:
  $x:=y+1 \;.\; u_x, u_y =: \bot, u'_x \vee u'_y$
  $x:=x+y \;.\; u_y =: u'_x \vee u'_y$

Usage Variables

- For each state variable v,v', a usage variable u_v,u'_v (boolean).
  u_v = "v is used"
  u'_v = "v' is used"
  Array needs usage array.
- $x'=x \wedge y'=y \wedge t'=t \wedge u_x=e' \wedge u_y=u'_y$ = $u_x=:e'$
- Annotate assignment statements:
  $x:=y+1 \;.\; u_x, u_y =: \bot, u'_x \vee u'_y$
  $x:=x+y \;.\; u_y=:u'_x \vee u'_y$
- Specifications and recursive time:
  $x'>x \wedge y'=y \wedge t'=t+u'_x \times x \;.\; t:=t+u'_x$
  (abuse: treat u'_x as 0 or 1)

Annotate if-then-else

$\textbf{if } y{=}0 \textbf{ then } (x{:=}1 \,.\, u_x{=:}\bot) \textbf{ else } (x{:=}2 \,.\, u_x{=:}\bot)$ isn't right.

Annotate if-then-else

$\textbf{if } y{=}0 \textbf{ then } \exists v_y \cdot u_y{=}(\quad \vee v_y) \wedge \langle u_y \rightarrow x{:=}1 \,.\, u_x{=}{:}\bot \rangle\, v_y \textbf{ else } \ldots$

- copy out u_y from statement

Annotate if-then-else

if $y{=}0$ **then** $\exists v_y \cdot u_y{=}(u'_x \vee v_y) \wedge \langle u_y \rightarrow\ x{:=}1\ .\ u_x{=:}\bot \rangle\ v_y$ **else** . . .

- copy out u_y from statement
- amend by u'_x because statement sets x'

Annotate if-then-else

if $y{=}0$ **then** $\exists v_y \cdot u_y{=}(u'_x \vee v_y) \wedge \langle u_y \rightarrow\ x{:=}1 \,.\, u_x{=}{:}\bot \rangle\, v_y$ **else** ...

- copy out u_y from statement
- amend by u'_x because statement sets x'

All variables in test need amendment.

Annotate if-then-else

if $y{=}0$ **then** $\exists v_y \cdot u_y{=}(u'_x \vee v_y) \wedge \langle u_y \rightarrow\ x{:=}1 \,.\, u_x{=:}\bot \rangle\, v_y$ **else** ...

- copy out u_y from statement
- amend by u'_x because statement sets x'

All variables in test need amendment.
Result:
if $y{=}0$ **then** $x{:=}1 \,.\, u_x, u_y {=:} \bot, (u'_x \vee u'_y)$ **else** ...

Example of Lazy Timing Refinements

Consumer (entry):

$$\neg u_s \wedge u_L = (u'_L\,[0;..2] \vee u'_s)^+\, u'_L\,[2;..\infty] \wedge t' = t + u'_s \times 2$$

$$\Leftarrow \quad s{:=}0\,.\,u_s{=}{:}\bot\,.\,Q\,0$$

Example of Lazy Timing Refinements

Consumer (entry):

$\neg u_s \wedge u_L = (u'_L\,[0;..2] \vee u'_s)^+\, u'_L\,[2;..\infty] \wedge t' = t + u'_s \times 2$

$\Leftarrow \quad s{:=}0 \,.\, u_s{=:}\bot \,.\, Q\,0$

Consumer (loop):

$Q\,m$

$= \quad u_s{=}u'_s \wedge u_L = u'_L\,[0;..m]^+\,(u'_L\,[m;..2] \vee u'_s)^+\, u'_L\,[2;..\infty] \wedge t' = t + u'_s \times (2{-}m)$

$\Leftarrow \quad \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n \,.\, u_L\,n =: u'_L\,n \vee u'_s \,.\, Q\,(m{+}1) \,.\, t{:=}t{+}u'_s)$

Example of Lazy Timing Refinements

Consumer (entry):

$\neg u_s \wedge u_L = (u'_L\,[0;..2] \vee u'_s)^+\, u'_L\,[2;..\infty] \wedge t' = t + u'_s \times 2$

$\Leftarrow \quad s{:=}0\,.\,u_s{=:}\bot\,.\,Q\,0$

Consumer (loop):

$Q\,m$

$= \quad u_s{=}u'_s \wedge u_L = u'_L\,[0;..m]^+\,(u'_L\,[m;..2] \vee u'_s)^+\, u'_L\,[2;..\infty] \wedge t' = t + u'_s \times (2{-}m)$

$\Leftarrow \quad \textbf{if } n{=}2 \textbf{ then } ok \textbf{ else } (s := s + L\,n\,.\,u_L\,n =: u'_L\,n \vee u'_s\,.\,Q\,(m{+}1)\,.\,t{:=}t{+}u'_s)$

Producer:

$t' = t + (MAX\,i \mid u'_L\,i \cdot i{+}1{-}m)$

$\Leftarrow \quad t' = t + (MAX\,i \mid u'_L\,i \cdot i{-}m)\,.\,t{:=}t{+}(\exists i\cdot\, i{\geq}m \wedge u'_L\,i)\,.\,L{:=}[m]^+L\,.\,u_L{=:}[\bot]^+u'_L$

Conclusions on Lazy Execution

- Lazy timing harder but tractible. *Can analyze locally*.

Conclusions on Lazy Execution

- Lazy timing harder but tractible. *Can analyze locally*.
- Restricting accessible variables helps.

Conclusions on Lazy Execution

- Lazy timing harder but tractible. *Can analyze locally*.
- Restricting accessible variables helps.
- Refinement independent of termination
  $\Rightarrow$ Choose execution, choose timing.

Future Work

- Prove soundness of lazy timing.
- Array u_L clumsy, can abstract to nat.
- Lazy spacing.
- More execution orders.