Class Notes for CS 819 C - Spring 2002

7. Semantics of Concurrency

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

revised May 17, 2002

Structural Operational Semantics Revisited

An execution (of a command) is a sequence

$$\gamma_0 \leadsto \gamma_1 \leadsto \cdots$$

of *configurations* γ_i related by the *transition relation* $\leadsto$.

A configuration is either:

- *nonterminal*: A command-state pair $\langle c, \sigma \rangle$, where $\sigma = (s, h)$ and $\mathrm{FV}(c) \subseteq \mathrm{dom}\, s$.
- *terminal*: A state σ or **abort**.

We write:

- $\gamma \leadsto^* \gamma'$ iff there is a finite execution from γ to γ'.
- $\gamma \uparrow$ iff there is an infinite execution beginning with γ.

Our semantics is *total* (Proposition 2):

- For every nonterminal configuration γ, there is at least one configuration γ' such that $\gamma \leadsto \gamma'$, i.e., executions do not get *stuck*.
- Every execution sequence can be extended to a sequence that is either infinite or ends with a terminal configuration.

Moreover, each transition preserves the domain of the store.

The transition relation is defined by inference rules called *transition rules*.

Concurrent Composition

$$\langle \text{comm} \rangle ::= \langle \text{comm} \rangle \parallel \langle \text{comm} \rangle$$

Transition Rules

$$\frac{\langle c_0, \sigma \rangle \rightsquigarrow \langle c_0', \sigma' \rangle}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \langle c_0' \parallel c_1, \sigma' \rangle} \qquad \frac{\langle c_1, \sigma \rangle \rightsquigarrow \langle c_1', \sigma' \rangle}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \langle c_0 \parallel c_1', \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightsquigarrow \sigma'}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \langle c_1, \sigma' \rangle} \qquad \frac{\langle c_1, \sigma \rangle \rightsquigarrow \sigma'}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \langle c_0, \sigma' \rangle}$$

$$\frac{\langle c_0, \sigma \rangle \rightsquigarrow \textbf{abort}}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \textbf{abort}} \qquad \frac{\langle c_1, \sigma \rangle \rightsquigarrow \textbf{abort}}{\langle c_0 \parallel c_1, \sigma \rangle \rightsquigarrow \textbf{abort}}.$$

Proposition 10 *Suppose*

$$\mathrm{FV}(c_0) \cap \mathrm{FA}(c_1) = \{\} \qquad \mathrm{FV}(c_1) \cap \mathrm{FA}(c_0) = \{\}$$
$$h = h_0 \cdot h_1 \ \text{ where } \ h_0 \perp h_1.$$

For any store s *such that* $\mathrm{FV}(c_0) \cup \mathrm{FV}(c_1) \subseteq \mathrm{dom}\, s$, *let*

$$s_0 = s\rceil(\mathrm{dom}\, s - \mathrm{FA}(c_1)) \qquad s_1 = s\rceil(\mathrm{dom}\, s - \mathrm{FA}(c_0)),$$

so that

$$s = s_0 \cup s_1 \qquad \mathrm{FV}(c_0) \subseteq \mathrm{dom}\, s_0 \qquad \mathrm{FV}(c_1) \subseteq \mathrm{dom}\, s_1.$$

If $\langle c_0 \parallel c_1, (s, h)\rangle \rightsquigarrow^* \textbf{abort}$ *then*

1. $\langle c_0, (s_0, h_0)\rangle \rightsquigarrow^* \textbf{abort}$

or 2. $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^* \textbf{abort}$.

If $\langle c_0 \parallel c_1, (s, h)\rangle \rightsquigarrow^* (s', h')$ *then*

1. $\langle c_0, (s_0, h_0)\rangle \rightsquigarrow^* \textbf{abort}$

or 2. $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^* \textbf{abort}$

or 3. $\langle c_0, (s_0, h_0)\rangle \rightsquigarrow^* (s_0', h_0')$ *and* $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^* (s_1', h_1')$,
where $s' = s_0' \cup s_1'$ *and* $h' = h_0' \cdot h_1'$ *and* $h_0' \perp h_1'$.

Proof Suppose $\langle c_0 \parallel c_1, (s,h)\rangle \leadsto^* \tau'$, where τ' is terminal, i.e. **abort** or (s', h'). The proof is by induction on the length of this execution. Since $\langle c_0 \parallel c_1, (s,h)\rangle$ is nonterminal, there must be at least one transition, i.e., $\langle c_0 \parallel c_1, (s,h)\rangle \leadsto \tau'' \leadsto^* \tau'$. The first transition must be inferred from one of the six rules for concurrent composition:

- $$\frac{\langle c_0, (s,h)\rangle \leadsto \langle c_0'', (s'', h'')\rangle}{\langle c_0 \parallel c_1, (s,h)\rangle \leadsto \langle c_0'' \parallel c_1, (s'', h'')\rangle}$$

  Since $\langle c_0, (s,h)\rangle \leadsto \langle c_0'', (s'', h'')\rangle$, and $\mathrm{FV}(c_0) \subseteq \mathrm{dom}\, s_0$, we have $\langle c_0, (s_0,h)\rangle \leadsto \langle c_0'', (s_0'', h'')\rangle$. Then, by Proposition 5, either $\langle c_0, (s_0,h_0)\rangle \leadsto$ **abort**, so that $\langle c_0, (s_0,h_0)\rangle \leadsto^*$ **abort**, or $\langle c_0, (s_0,h_0)\rangle \leadsto \langle c_0'', (s_0'', h_0'')\rangle$, where $h_0'' \perp h_1$ and $h'' = h_0'' \cdot h_1$ and, since c_0 does not assign to any variable in $\mathrm{dom}\, s_1$, $s_1'' = s_1$.

  In the latter case, since $\tau'' = \langle c_0'' \parallel c_1, (s'', h'')\rangle \leadsto^* \tau'$, the induction hypothesis gives either

  1. $\langle c_0'', (s_0'', h_0'')\rangle \leadsto^*$ **abort**,
     in which case $\langle c_0, (s_0, h_0)\rangle \leadsto^*$ **abort**,

  or 2. $\langle c_1, (s_1, h_1)\rangle \leadsto^*$ **abort**,

  or 3. (providing $\tau' = (s', h')$)

  3a. $\langle c_0'', (s_0'', h_0'')\rangle \leadsto^* (s_0', h_0')$,
  in which case $\langle c_0, (s_0, h_0)\rangle \leadsto^* (s_0', h_0')$,

  and 3b. $\langle c_1, (s_1, h_1)\rangle \leadsto^* (s_1', h_1')$,

  where $s' = s_0' \cup s_1'$ and $h' = h_0' \cdot h_1'$ and $h_0' \perp h_1'$.

Proof (continued)

- $$\frac{\langle c_0, (s,h)\rangle \rightsquigarrow (s'', h'')}{\langle c_0 \parallel c_1, (s,h)\rangle \rightsquigarrow \langle c_1, (s'', h'')\rangle}$$

  Since $\langle c_0, (s,h)\rangle \rightsquigarrow (s'', h'')$, and $\mathrm{FV}(c_0) \subseteq \operatorname{dom} s_0$, we have $\langle c_0, (s_0,h)\rangle \rightsquigarrow (s''_0, h'')$. By Proposition 5, either $\langle c_0, (s_0,h_0)\rangle \rightsquigarrow$ **abort**, so that $\langle c_0, (s_0,h_0)\rangle \rightsquigarrow^*$ **abort**, or $\langle c_0, (s_0,h_0)\rangle \rightsquigarrow (s''_0, h''_0)$, where $h''_0 \perp h_1$ and $h'' = h''_0 \cdot h_1$ and, since c_0 does not assign to any variable in $\operatorname{dom} s_1$, $s''_1 = s_1$.

  In the latter case, if $\tau' =$ **abort**, then $\tau'' = \langle c_1, (s'', h'')\rangle \rightsquigarrow^*$ **abort**, and since $\mathrm{FV}(c_1) \subseteq \operatorname{dom} s_1$, $\langle c_1, (s_1, h'')\rangle \rightsquigarrow^*$ **abort**. Then Proposition 5 gives $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^*$ **abort**.

  On the other hand, if $\tau' = (s', h')$, then $\tau'' = \langle c_1, (s'', h'')\rangle \rightsquigarrow^* (s', h')$, and since $\mathrm{FV}(c_1) \subseteq \operatorname{dom} s_1$, $\langle c_1, (s_1, h'')\rangle \rightsquigarrow^* (s'_1, h')$. Then Proposition 5 gives either $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^*$ **abort** or $\langle c_1, (s_1, h_1)\rangle \rightsquigarrow^* (s'_1, h'_1)$, where $h''_0 \perp h'_1$ and $h' = h''_0 \cdot h'_1$. Moreover, since c_1 does not assign to any variable in $\operatorname{dom} s_0$, $s''_0 = s'_0$.

- $$\frac{\langle c_0, (s,h)\rangle \rightsquigarrow \textbf{abort}}{\langle c_0 \parallel c_1, (s,h)\rangle \rightsquigarrow \textbf{abort}}$$

  Since $\mathrm{FV}(c_0) \subseteq \operatorname{dom} s_0$, we have $\langle c_0, (s_0,h)\rangle \rightsquigarrow$ **abort**, and, by Proposition 5, $\langle c_0, (s_0,h_0)\rangle \rightsquigarrow$ **abort**.

The remaining three cases are obtained from those above by interchaging the subscripts 0 and 1. END OF PROOF

A Rule for Noninterfering Concurrency

$$\frac{\{p_0\}\ c_0\ \{q_0\} \qquad \{p_1\}\ c_1\ \{q_1\}}{\{p_0 \ast p_1\}\ c_0 \parallel c_1\ \{q_0 \ast q_1\}}$$

when $\mathrm{FV}(c_0) \cap \mathrm{FA}(c_1)$ and $\mathrm{FV}(c_1) \cap \mathrm{FA}(c_0)$ are empty.

Soundness

Suppose the premisses $\{p_0\}\ c_0\ \{q_0\}$ and $\{p_1\}\ c_1\ \{q_1\}$ are valid, and $[\![p_0 \ast p_1]\!]_{\text{assert}} s\, h$. Then there are h_0 and h_1 such that

$$h_0 \perp h_1 \qquad h = h_0 \cdot h_1 \qquad [\![p_0]\!]_{\text{assert}} s_0\, h_0 \qquad [\![p_1]\!]_{\text{assert}} s_1\, h_1.$$

Since the premisses are valid, $\neg\, \langle c_0, (s_0, h_0)\rangle \leadsto^* \textbf{abort}$ and $\neg\, \langle c_1, (s_1, h_1)\rangle \leadsto^* \textbf{abort}$, so that Proposition 10 gives

$$\neg\, \langle c_0 \parallel c_1, (s, h)\rangle \leadsto^* \textbf{abort}.$$

Now suppose $\langle c_0 \parallel c_1, (s, h)\rangle \leadsto^* (s', h')$. By Proposition 10,

$$\langle c_0, (s_0, h_0)\rangle \leadsto^* (s_0', h_0') \qquad \langle c_1, (s_1, h_1)\rangle \leadsto^* (s_1', h_1'),$$

where $s' = s_0' \cup s_1'$ and $h' = h_0' \cdot h_1'$ and $h_0' \perp h_1'$. Then, since the premisses are valid, $[\![q_0]\!]_{\text{assert}} s_0'\, h_0'$ and $[\![q_1]\!]_{\text{assert}} s_1'\, h_1'$. Thus $[\![q_0 \ast q_1]\!]_{\text{assert}} s'\, h'$.

Adding Conditional Critical Regions

$$\begin{aligned}\langle\text{comm}\rangle ::= &\ \textbf{resource}\ \langle\text{resname}\rangle\ \textbf{in}\ \langle\text{comm}\rangle \\ &\mid \textbf{with}\ \langle\text{resname}\rangle\ \textbf{when}\ \langle\text{boolexp}\rangle\ \textbf{do}\ \langle\text{comm}\rangle\end{aligned}$$

In a resource declaration **resource** r **in** c, the binder r has scope c. We write $\mathrm{FR}(c)$ for the set of resource names occurring free in c.

Extending States

$$\begin{aligned}\text{ResourceStatus}_R &= R \to \{\mathbf{free}, \mathbf{busy}\} \\ \text{Stores}_V &= V \to \mathbf{Z} \\ \text{Heaps} &= \bigcup_{A \overset{\text{fin}}{\subseteq} \text{Locations}} (A \to \mathbf{Z}) \\ \text{States}_{VR} &= \text{ResourceStatus}_R \times \text{Stores}_V \times \text{Heaps},\end{aligned}$$

where $V \overset{\text{fin}}{\subseteq} \langle\text{var}\rangle$ and $R \overset{\text{fin}}{\subseteq} \langle\text{resname}\rangle$.

Transition Rules

- Resource Declaration

$$\frac{}{\langle \textbf{resource}\ r\ \textbf{in}\ c, (\rho, s, h)\rangle \leadsto \langle c\,;\textbf{remove}\ r, ([\,\rho \mid r{:}\,\textbf{free}\,], s, h)\rangle}$$

- Removal

$$\frac{}{\langle \textbf{remove}\ r, (\rho, s, h)\rangle \leadsto (\rho\rceil(\mathrm{dom}\,\rho - \{r\}), s, h)}$$

We must prohibit more than one declaration of the same resource name, e.g.,

$$(\textbf{resource}\ \mathsf{r}\ \textbf{in}\ c) \parallel (\textbf{resource}\ \mathsf{r}\ \textbf{in}\ c')$$

$$\textbf{resource}\ \mathsf{r}\ \textbf{in}\ ((\textbf{resource}\ \mathsf{r}\ \textbf{in}\ c) \parallel c').$$

- Conditional Critical Region

  When $\rho\, r = \textbf{free}$ and $[\![b]\!]_{\text{boolexp}} s = \textbf{true}$:

$$\frac{}{\langle \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c, (\rho, s, h)\rangle \leadsto \langle c\,;\textbf{release}\ r, ([\,\rho \mid r{:}\,\textbf{busy}\,], s, h)\rangle}$$

  When $\rho\, r = \textbf{busy}$ or $[\![b]\!]_{\text{boolexp}} s = \textbf{false}$:

$$\frac{}{\langle \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c, (\rho, s, h)\rangle \leadsto \langle \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c, (\rho, s, h)\rangle}$$

- Release

$$\frac{}{\langle \textbf{release}\ r, (\rho, s, h)\rangle \leadsto ([\,\rho \mid r{:}\,\textbf{free}\,], s, h)}$$

Examples of Deadlock

Inevitable

$$\textbf{with } \mathsf{r} \textbf{ when false do } c$$

$$\textbf{with } \mathsf{r} \textbf{ when true do with } \mathsf{r} \textbf{ when true do } c$$

$$\mathsf{x} := 0 \,;\, ((\textbf{with } \mathsf{r} \textbf{ when } \mathsf{x} = 1 \textbf{ do } c) \parallel (\textbf{while } \mathsf{x} < 10 \textbf{ do } \mathsf{x} := \mathsf{x} + 2))$$

Possible

$$\mathsf{x} := 0 \,;\, ((\textbf{with } \mathsf{r} \textbf{ when } \mathsf{x} = 1 \textbf{ do } c) \parallel (\mathsf{x} := \mathsf{x} + 1 \,;\, \mathsf{x} := \mathsf{x} + 1))$$

$$\begin{aligned}&(\textbf{with } \mathsf{r}_1 \textbf{ when true do with } \mathsf{r}_2 \textbf{ when true do } c) \parallel \\ &\qquad (\textbf{with } \mathsf{r}_2 \textbf{ when true do with } \mathsf{r}_1 \textbf{ when true do } c')\end{aligned}$$

Fairness

Consider a program that searches through all nonnegative integers for a k such that $\mathsf{f(k)} \geq 0$:

$$\begin{array}{l} \mathsf{k} := -1; \\ \big((\mathsf{i} := 0\,;\ \textbf{while}\ \mathsf{k} = -1\ \textbf{do} \\ \quad\ \textbf{if}\ \mathsf{f(i)} \geq 0\ \textbf{then}\ \mathsf{k} := \mathsf{i}\ \textbf{else}\ \mathsf{i} := \mathsf{i} + 2) \\ \ \|\ (\mathsf{j} := 1\,;\ \textbf{while}\ \mathsf{k} = -1\ \textbf{do} \\ \quad\ \textbf{if}\ \mathsf{f(j)} \geq 0\ \textbf{then}\ \mathsf{k} := \mathsf{j}\ \textbf{else}\ \mathsf{j} := \mathsf{j} + 2)\big). \end{array}$$

If $\mathsf{f(k)} \geq 0$ never holds for even k, then, according to our transition semantics, there is an infinite execution that, after the initialization step, has the form

$$\langle c_0 \parallel c', \sigma_0\rangle \rightsquigarrow \langle c_1 \parallel c', \sigma_1\rangle \rightsquigarrow \langle c_2 \parallel c', \sigma_2\rangle \rightsquigarrow \cdots,$$

where the second process never executes at all and the function f is never tested for odd arguments. Such an execution is said to be *unfair*.

Even worse is an unfair execution of

$$\mathsf{x} := 0\,;\ ((\textbf{with}\ \mathsf{r}\ \textbf{when}\ \mathsf{x} = 1\ \textbf{do}\ c) \parallel \mathsf{x} := \mathsf{x} + 1)$$

where the first process does “busy waiting” forever, and the second process never executes.

We will impose fairness when we develop a *trace* semantics.

Transition Traces

When the execution of a command c_0 can be interrupted by other processes, the sequence of transitions representing the execution of c_0 will have one of the forms

$$\begin{aligned}&\langle c_0, \sigma_0\rangle \rightsquigarrow \langle c_1, \sigma_0'\rangle \Longrightarrow \langle c_1, \sigma_1\rangle \rightsquigarrow \langle c_2, \sigma_1'\rangle \Longrightarrow \cdots \\ &\qquad\qquad \Longrightarrow \langle c_{n-1}, \sigma_{n-1}\rangle \rightsquigarrow \sigma_{n-1}'\end{aligned}$$

$$\begin{aligned}&\langle c_0, \sigma_0\rangle \rightsquigarrow \langle c_1, \sigma_0'\rangle \Longrightarrow \langle c_1, \sigma_1\rangle \rightsquigarrow \langle c_2, \sigma_1'\rangle \Longrightarrow \cdots \\ &\qquad\qquad \Longrightarrow \langle c_{n-1}, \sigma_{n-1}\rangle \rightsquigarrow \mathbf{abort}\end{aligned}$$

$$\langle c_0, \sigma_0\rangle \rightsquigarrow \langle c_1, \sigma_0'\rangle \Longrightarrow \langle c_1, \sigma_1\rangle \rightsquigarrow \langle c_2, \sigma_1'\rangle \Longrightarrow \cdots$$

Here the occurrences of $\Longrightarrow$ denote interruptions by other processes. Notice that, across each $\Longrightarrow$,

- The state, which can be affected by the other processes, changes from σ_{i-1}' to σ_i.
- The command c_i, which is internal to the workings of c_0, does not change.

As far as the rest of the program is concerned, the commands $c_0, c_1, \ldots$ are invisible, and all that matters is the sequence of before-and-after state pairs, which will have one of the forms

$$\langle \sigma_0, \sigma_0'\rangle, \langle \sigma_1, \sigma_1'\rangle, \ \ldots \ , \langle \sigma_{n-1}, \sigma_{n-1}'\rangle$$

$$\langle \sigma_0, \sigma_0'\rangle, \langle \sigma_1, \sigma_1'\rangle, \ \ldots \ , \langle \sigma_{n-1}, \mathbf{abort}\rangle$$

$$\langle \sigma_0, \sigma_0'\rangle, \langle \sigma_1, \sigma_1'\rangle, \ \ldots$$

Such a sequence is called a *transition trace*. When each $\sigma_{i-1}' = \sigma_i$, a transition trace is said to be *interference-free*.

Some Notation for Sets of Sequences

$$
\begin{aligned}
S\,;S' &\stackrel{\text{def}}{=} \{\, \alpha \cdot \beta \mid \alpha \in S \text{ and } \beta \in S' \,\} \\
S^0 &\stackrel{\text{def}}{=} \{\langle\rangle\} \\
S^{n+1} &\stackrel{\text{def}}{=} S\,;S^n \\
S^* &\stackrel{\text{def}}{=} \bigcup_{n=0}^{\infty} S^n \\
S^+ &\stackrel{\text{def}}{=} \bigcup_{n=1}^{\infty} S^n \\
S^\omega &\stackrel{\text{def}}{=} \{\, \alpha_0 \cdot \alpha_1 \cdot \cdots \mid \forall i \geq 0.\ \alpha_i \in S \,\} \\
S^\infty &\stackrel{\text{def}}{=} S^* \cup S^\omega.
\end{aligned}
$$

Note that, if $\alpha_0, \ldots, \alpha_{i-1}$ are finite sequences and α_i is an infinite sequence, then

$$
\begin{aligned}
\alpha_0 \cdot \cdots \cdot \alpha_i, \cdot \cdots \cdot \alpha_{n-1} &= \alpha_0 \cdot \cdots \cdot \alpha_i \\
\alpha_0 \cdot \alpha_1 \cdot \cdots\ &= \alpha_0 \cdot \cdots \cdot \alpha_i.
\end{aligned}
$$

Trace Semantics

When $\mathrm{FV}(c) \subseteq V$ and $\mathrm{FR}(c) \subseteq R$, the trace semantics $\mathcal{T}[\![c]\!]_{VR}$ of c is the set of traces describing possible executions of c:

$$
\begin{aligned}
\mathcal{T}[\![c]\!]_{VR} \subseteq\ & (\mathrm{States}_{VR} \times \mathrm{States}_{VR})^+ \\
& \cup (\mathrm{States}_{VR} \times \mathrm{States}_{VR})^*\,; (\mathrm{States}_{VR} \times \{\mathbf{abort}\}) \\
& \cup (\mathrm{States}_{VR} \times \mathrm{States}_{VR})^\omega.
\end{aligned}
$$

Trace Semantics (continued)

Trace semantics is defined by structural induction on c:

- Assignment

$$\mathcal{T}[\![v := e]\!]_{VR} = \\ \left\{ \langle(\rho, s, h), (\rho, [\, s \mid v\colon [\![e]\!]_{\text{exp}} s\,], h)\rangle \;\middle|\; (\rho, s, h) \in \text{States}_{VR} \right\}.$$

- **skip**

$$\mathcal{T}[\![\textbf{skip}]\!]_{VR} = \left\{ \langle\sigma, \sigma\rangle \;\middle|\; \sigma \in \text{States}_{VR} \right\}.$$

- Sequential Composition

$$\mathcal{T}[\![c_0\,;c_1]\!]_{VR} = \mathcal{T}[\![c_0]\!]_{VR}\,;\mathcal{T}[\![c_1]\!]_{VR}.$$

- Conditional

$$\mathcal{T}[\![\textbf{if}\ b\ \textbf{then}\ c_0\ \textbf{else}\ c_1]\!]_{VR} = \mathcal{B}[\![b]\!]\,;\mathcal{T}[\![c_0]\!] \cup \mathcal{B}[\![\neg\, b]\!]\,;\mathcal{T}[\![c_1]\!].$$

- **while**

$$\mathcal{T}[\![\textbf{while}\ b\ \textbf{do}\ c]\!]_{VR} = \\ \left(\mathcal{B}[\![b]\!]_{VR}\,;\mathcal{T}[\![c]\!]_{VR}\right)^{*};\mathcal{B}[\![\neg\, b]\!]_{VR} \cup \left(\mathcal{B}[\![b]\!]_{VR}\,;\mathcal{T}[\![c]\!]_{VR}\right)^{\omega}.$$

where

$$\mathcal{B}[\![b]\!]_{VR} = \\ \left\{ \langle(\rho, s, h), (\rho, s, h)\rangle \;\middle|\; (\rho, s, h) \in \text{States}_{VR} \text{ and } [\![b]\!]_{\text{boolexp}} s = \textbf{true} \right\}.$$

Trace Semantics of the Heap Operations

- Allocation

$$\begin{aligned}
&\mathcal{T}[\![v := \mathbf{cons}(e_0, \ldots, e_{n-1})]\!]_{VR} = \\
&\quad \{\langle (\rho, s, h), \\
&\qquad\quad (\rho, [\, s \mid v\colon \ell\,], [\, h \mid \ell\colon [\![e_0]\!]_{\mathrm{exp}} s \mid \ldots \mid \ell + n - 1\colon [\![e_{n-1}]\!]_{\mathrm{exp}} s\,])\rangle \\
&\qquad \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and} \\
&\qquad\quad \ell, \ldots, \ell + n - 1 \in \mathrm{Locations} - \mathrm{dom}\, h\}.
\end{aligned}$$

- Lookup

$$\begin{aligned}
&\mathcal{T}[\![v := [e]]\!]_{VR} = \\
&\quad \{\langle (\rho, s, h), (\rho, [\, s \mid v\colon h([\![e]\!]_{\mathrm{exp}} s)\,], h)\rangle \\
&\qquad\quad \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \in \mathrm{dom}\, h\} \\
&\quad \cup \{\langle (\rho, s, h), \mathbf{abort}\rangle \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \notin \mathrm{dom}\, h\}.
\end{aligned}$$

- Mutation

$$\begin{aligned}
&\mathcal{T}[\![[e] := e']\!]_{VR} = \\
&\quad \{\langle (\rho, s, h), (\rho, s, [\, h \mid [\![e]\!]_{\mathrm{exp}} s\colon [\![e']\!]_{\mathrm{exp}} s\,])\rangle \\
&\qquad\quad \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \in \mathrm{dom}\, h\} \\
&\quad \cup \{\langle (\rho, s, h), \mathbf{abort}\rangle \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \notin \mathrm{dom}\, h\}.
\end{aligned}$$

- Disposal

$$\begin{aligned}
&\mathcal{T}[\![\mathbf{dispose}\ e]\!]_{VR} = \\
&\quad \{\langle (\rho, s, h), (\rho, s, h\rceil(\mathrm{dom}\, h - \{[\![e]\!]_{\mathrm{exp}} s\}))\rangle \\
&\qquad\quad \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \in \mathrm{dom}\, h\} \\
&\quad \cup \{\langle (\rho, s, h), \mathbf{abort}\rangle \mid (\rho, s, h) \in \mathrm{States}_{VR} \text{ and } [\![e]\!]_{\mathrm{exp}} s \notin \mathrm{dom}\, h\}.
\end{aligned}$$

Trace Semantics of Concurrency

- Concurrent Composition

$$\mathcal{T}[\![c_0 \parallel c_1]\!]_{VR} = \\ \Big\{ \operatorname{trn}\gamma \;\Big|\; \exists\alpha \in \mathcal{T}[\![c_0]\!]_{VR}.\ \exists\beta \in \mathcal{T}[\![c_1]\!]_{VR}.\ \text{fair-merge}(\alpha,\beta,\gamma) \Big\},$$

where

- fair-merge(α, β, γ) holds when there are functions $f \in \operatorname{dom}\alpha \to \operatorname{dom}\gamma$ and $g \in \operatorname{dom}\beta \to \operatorname{dom}\gamma$ such that:
  - f and g are monotone injections:

$$fi < fj \text{ and } gi < gj \text{ when } i < j.$$

  - The ranges of f and g partition the domain of γ:

$$\operatorname{ran} f \cap \operatorname{ran} g = \{\} \text{ and } \operatorname{ran} f \cup \operatorname{ran} g = \operatorname{dom}\gamma.$$

  - f (or g) maps each position of a state-pair in α (or in β) to a position of the same state-pair in γ:

$$\gamma_{fi} = \alpha_i \text{ and } \gamma_{gi} = \beta_i.$$

- $\operatorname{trn}\gamma$ is defined by:
  - If γ contains a pair of the form $\langle\sigma, \mathbf{abort}\rangle$, then $\operatorname{trn}\gamma$ is the shortest prefix of γ ending in such a pair.
  - Otherwise, $\operatorname{trn}\gamma = \gamma$.

Trace Semantics of Concurrency (continued)

- Conditional Critical Regions

$$
\begin{aligned}
&\mathcal{T}[\![\textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c]\!]_{VR} = \\
&\quad \{\langle(\rho,s,h),(\rho,s,h)\rangle \\
&\qquad \mid (\rho,s,h) \in \mathrm{STATES}_{VR} \text{ and } (\rho\, r = \textbf{busy} \text{ or } \neg[\![b]\!]_{\text{boolexp}} s)\}^* \,; \\
&\quad \{\langle(\rho,s,h),([\,\rho \mid r\colon \textbf{busy}\,],s,h)\rangle \\
&\qquad \mid (\rho,s,h) \in \mathrm{STATES}_{VR} \text{ and } (\rho\, r = \textbf{free} \text{ and } [\![b]\!]_{\text{boolexp}} s)\} \,; \\
&\quad \mathcal{T}[\![c]\!]_{VR} \,; \\
&\quad \{\langle(\rho,s,h),([\,\rho \mid r\colon \textbf{free}\,],s,h)\rangle \mid (\rho,s,h) \in \mathrm{STATES}_{VR}\} \\
&\ \cup \{\langle(\rho,s,h),(\rho,s,h)\rangle \\
&\qquad \mid (\rho,s,h) \in \mathrm{STATES}_{VR} \text{ and } (\rho\, r = \textbf{busy} \text{ or } \neg[\![b]\!]_{\text{boolexp}} s)\}^\omega
\end{aligned}
$$

Trace Semantics of Declarations

- Variable Declaration

$$\mathcal{T}[\![\mathbf{newvar}\ v\ \mathbf{in}\ c]\!]_{VR} = \\ \left\{\ \hat{\alpha}\ \middle|\ \exists\alpha \in \mathcal{T}[\![c]\!]_{V\cup\{v\},R}.\ \text{local-global-var}_{VR}(v,\alpha,\hat{\alpha})\ \right\}.$$

where, if

$$\alpha = \langle(\rho_0, s_0, h_0), (\rho'_0, s'_0, h'_0)\rangle, \langle(\rho_1, s_1, h_1), (\rho'_1, s'_1, h'_1)\rangle, \ldots$$

$$\hat{\alpha} = \langle(\hat{\rho}_0, \hat{s}_0, \hat{h}_0), (\hat{\rho}'_0, \hat{s}'_0, \hat{h}'_0)\rangle, \langle(\hat{\rho}_1, \hat{s}_1, \hat{h}_1), (\hat{\rho}'_1, \hat{s}'_1, \hat{h}'_1)\rangle, \ldots$$

then local-global-var$_{VR}(v,\alpha,\hat{\alpha})$ holds iff

- $\hat{\alpha}$ has the same length as α,
- Corresponding states give the same value to all variables except v:

$$\begin{array}{lll} \hat{s}_i w = s_i w \text{ when } w \neq v & \hat{\rho}_i = \rho_i & \hat{h}_i = h_i \\ \hat{s}'_i w = s'_i w \text{ when } w \neq v & \hat{\rho}'_i = \rho'_i & \hat{h}'_i = h'_i, \end{array}$$

- The value of v in the states of α is never changed externally:

$$s_{i+1}v = s'_i v,$$

- If $v \in V$ then the value of v in the states of $\hat{\alpha}$ is never changed internally:

$$\hat{s}'_i v = \hat{s}_i v.$$

Trace Semantics of Declarations (continued)

- Resource Declaration

$$\mathcal{T}[\![\textbf{resource } r \textbf{ in } c]\!]_{VR} =$$
$$\left\{\, \hat{\alpha} \;\middle|\; \exists \alpha \in \mathcal{T}[\![c]\!]_{V,R\cup r}.\ \text{local-global-res}_{VR}(r, \alpha, \hat{\alpha}) \,\right\}.$$

where, if

$$\alpha = \langle(\rho_0, s_0, h_0), (\rho'_0, s'_0, h'_0)\rangle, \langle(\rho_1, s_1, h_1), (\rho'_1, s'_1, h'_1)\rangle, \ldots$$

$$\hat{\alpha} = \langle(\hat{\rho}_0, \hat{s}_0, \hat{h}_0), (\hat{\rho}'_0, \hat{s}'_0, \hat{h}'_0)\rangle, \langle(\hat{\rho}_1, \hat{s}_1, \hat{h}_1), (\hat{\rho}'_1, \hat{s}'_1, \hat{h}'_1)\rangle, \ldots$$

then local-global-res$_{VR}(r, \alpha, \hat{\alpha})$ holds iff

- $\hat{\alpha}$ has the same length as α,
- Corresponding states give the same value to all resource names except r:

$$\hat{\rho}_i q = \rho_i q \text{ when } q \neq r \qquad \hat{s}_i = s_i \qquad \hat{h}_i = h_i$$
$$\hat{\rho}'_i q = \rho'_i q \text{ when } q \neq r \qquad \hat{s}'_i = s'_i \qquad \hat{h}'_i = h'_i,$$

- The value of r in the states of α is never changed externally:

$$\rho_{i+1} r = \rho'_i r,$$

- If $r \in R$ then the value of r in the states of $\hat{\alpha}$ is never changed internally:

$$\hat{\rho}'_i r = \hat{\rho}_i r.$$

Resource-Dependent Hoare Triples

A *resource-dependent Hoare triple* has the form

$$\Gamma \vdash_{V_0} \{p\}\ c\ \{q\},$$

where Γ, called the *resource context*, is

$$r_1(V_1) \colon R_1, \ldots, r_n(V_n) \colon R_n,$$

and

- $r_1, \ldots, r_n$ are distinct resource names.
- $V_0, V_1, \ldots, V_n$ are disjoint finite sets of variables.
- $R_1, \ldots, R_n$ are assertions, called the *resource invariants*.
- $\mathrm{FV}(p) \subseteq V_0$, $\mathrm{FV}(q) \subseteq V_0$, and $\mathrm{FV}(c) \subseteq V_0 \cup V_1 \cup \cdots \cup V_n$.

We write V for $V_0 \cup V_1 \cup \cdots \cup V_n$, and R for $\{r_1, \ldots, r_n\}$.

Enriched Traces

Let

$$\alpha = \langle(\rho_0, s_0, h_0), (\rho'_0, s'_0, h'_0)\rangle \cdot \langle(\rho_1, s_1, h_1), (\rho'_1, s'_1, h'_1)\rangle \cdot \cdots$$

be a (finite or infinite) transition trace, and let

$$\hat{\rho}_0 = \begin{cases} \textbf{free} & \text{if } \rho_0 = \textbf{free} \\ \textbf{busy}^- & \text{if } \rho_0 = \textbf{busy} \end{cases}$$

$$\hat{\rho}'_i = \begin{cases} \textbf{free} & \text{if } \rho'_i = \textbf{free} \\ \textbf{busy}^+ & \text{if } \rho'_i = \textbf{busy} \text{ and } \hat{\rho}_i = \textbf{free} \\ \textbf{busy}^- & \text{if } \rho'_i = \textbf{busy} \text{ and } \hat{\rho}_i = \textbf{busy}^- \\ \textbf{busy}^+ & \text{if } \rho'_i = \textbf{busy} \text{ and } \hat{\rho}_i = \textbf{busy}^+ \end{cases}$$

$$\hat{\rho}_{i+1} = \begin{cases} \textbf{free} & \text{if } \rho_{i+1} = \textbf{free} \\ \textbf{busy}^- & \text{if } \rho_{i+1} = \textbf{busy} \text{ and } \hat{\rho}'_i = \textbf{free} \\ \textbf{busy}^- & \text{if } \rho_{i+1} = \textbf{busy} \text{ and } \hat{\rho}'_i = \textbf{busy}^- \\ \textbf{busy}^+ & \text{if } \rho_{i+1} = \textbf{busy} \text{ and } \hat{\rho}'_i = \textbf{busy}^+. \end{cases}$$

Then the $\hat{\rho}_i$ and $\hat{\rho}'_i$ are called enriched resource statuses, and the sequence

$$\hat{\alpha} = \langle(\hat{\rho}_0, s_0, h_0), (\hat{\rho}'_0, s'_0, h'_0)\rangle \cdot \langle(\hat{\rho}_1, s_1, h_1), (\hat{\rho}'_1, s'_1, h'_1)\rangle \cdot \cdots$$

obtained from α by replacing each ρ_i by $\hat{\rho}_i$ and each ρ'_i by $\hat{\rho}'_i$ is called the *enrichment* of α.

A resource-dependent Hoare triple $\Gamma \vdash_{V_0} \{p\}\ c\ \{q\}$ is valid iff the enrichment of every trace in $\mathcal{T}[\![c]\!]_{VR}$ is *acceptable* for Γ, V_0, p, and q.

Decomposition of States

Suppose $\sigma = (\rho, s, h)$ is a state such that $\operatorname{dom} s = V$. Then we write

$$\rho, \delta :: \sigma$$

to indicate that there are $s_0, \ldots, s_n$ and $h_0, \ldots, h_n$ such that

- Each $s_i = s \rceil V_i$, so that $s = s_0 \cdot \cdots \cdot s_n$ and the s_i have disjoint domains that partition V.
- $h = h_0 \cdot \cdots \cdot h_n$, where the h_i have disjoint domains that partition the domain of h.
- $\delta = \langle \delta_0, \ldots, \delta_n \rangle$, where each $\delta_i = (s_i, h_i)$.

If σ is **abort**, then $\rho, \delta :: \sigma$ is false.

Then we write

$$\forall \rho, \delta :: \sigma. \qquad \text{for} \qquad \forall \rho, \delta \text{ such that } \rho, \delta :: \sigma,$$

and

$$\exists \rho, \delta :: \sigma. \qquad \text{for} \qquad \exists \rho, \delta \text{ such that } \rho, \delta :: \sigma.$$

We also write ρ_i for $\rho\, r_i$.

Acceptable Enriched Traces

The enriched trace $\langle\sigma,\sigma'\rangle\cdot\alpha$ is *acceptable* for Γ, V_0, p, and q when $\mu_k(\langle\sigma,\sigma'\rangle\cdot\alpha)$ is true for all $k \geq 0$, where

$$
\begin{aligned}
&\mu_k(\langle\sigma,\sigma'\rangle\cdot\alpha) \stackrel{\text{def}}{=}\\
&\quad \forall\rho,\delta :: \sigma.\ [\![p]\!]_{\text{assert}}\delta_0\\
&\qquad\quad \text{and } \Big(\forall_{i=1}^{n}\ \rho_i \neq \mathbf{busy}^{+} \text{ and } (\rho_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}}\delta_i)\Big)\\
&\qquad \text{implies } \phi_k(\rho,\delta,\sigma',\alpha)
\end{aligned}
$$

$$
\begin{aligned}
&\phi_k(\rho,\delta,\sigma',\alpha) \stackrel{\text{def}}{=}\\
&\quad \exists\rho',\delta' :: \sigma'.\ \Big(\forall_{i=1}^{n}\ \mathbf{if}\ \rho_i = \mathbf{busy}^{-}\ \mathbf{then}\ \rho'_i = \mathbf{busy}^{-} \text{ and } \delta'_i = \delta_i\\
&\qquad\qquad \mathbf{else}\ \rho'_i \neq \mathbf{busy}^{-} \text{ and } (\rho'_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}}\delta'_i)\Big)\\
&\qquad \text{and } \psi_k(\rho',\delta',\alpha)
\end{aligned}
$$

$$\psi_0(\rho',\delta',\alpha) \stackrel{\text{def}}{=} \mathbf{true}$$

$$\psi_{k+1}(\rho',\delta',\epsilon) \stackrel{\text{def}}{=} [\![q]\!]_{\text{assert}}\delta'_0 \text{ and } \forall_{i=1}^{n}\ \rho'_i \neq \mathbf{busy}^{+}$$

$$
\begin{aligned}
&\psi_{k+1}(\rho',\delta',\langle\sigma,\sigma''\rangle\cdot\alpha) \stackrel{\text{def}}{=}\\
&\quad \forall\rho,\delta :: \sigma.\ \delta_0 = \delta'_0\\
&\qquad\quad \text{and } \Big(\forall_{i=1}^{n}\ \mathbf{if}\ \rho'_i = \mathbf{busy}^{+}\ \mathbf{then}\ \rho_i = \mathbf{busy}^{+} \text{ and } \delta_i = \delta'_i\\
&\qquad\qquad \mathbf{else}\ \rho_i \neq \mathbf{busy}^{+} \text{ and } (\rho_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}}\delta_i)\Big)\\
&\qquad \text{implies } \phi_k(\rho,\delta,\sigma',\alpha).
\end{aligned}
$$

An Example

The enriched trace $\langle \sigma, \sigma' \rangle \cdot \langle \sigma'', \sigma''' \rangle$ is acceptable for Γ, V_0, p, and q when

$$
\begin{array}{l}
\forall \rho, \delta :: \sigma.\ [\![p]\!]_{\text{assert}} \delta_0 \\
\qquad \text{and } \Big(\forall_{i=1}^{n}\ \rho_i \neq \mathbf{busy}^+ \text{ and } (\rho_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}} \delta_i) \Big) \\
\text{implies} \\
\exists \rho', \delta' :: \sigma'.\ \Big(\forall_{i=1}^{n}\ \mathbf{if}\ \rho_i = \mathbf{busy}^-\ \mathbf{then}\ \rho'_i = \mathbf{busy}^- \text{ and } \delta'_i = \delta_i \\
\qquad \mathbf{else}\ \rho'_i \neq \mathbf{busy}^- \text{ and } (\rho'_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}} \delta'_i) \Big) \\
\text{and} \\
\forall \rho'', \delta'' :: \sigma''.\ \delta''_0 = \delta'_0 \\
\qquad \text{and } \Big(\forall_{i=1}^{n}\ \mathbf{if}\ \rho'_i = \mathbf{busy}^+\ \mathbf{then}\ \rho''_i = \mathbf{busy}^+ \text{ and } \delta''_i = \delta'_i \\
\qquad\quad \mathbf{else}\ \rho''_i \neq \mathbf{busy}^+ \text{ and } (\rho''_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}} \delta''_i) \Big) \\
\text{implies} \\
\exists \rho''', \delta''' :: \sigma'''.\ \Big(\forall_{i=1}^{n}\ \mathbf{if}\ \rho''_i = \mathbf{busy}^-\ \mathbf{then}\ \rho'''_i = \mathbf{busy}^- \text{ and } \delta'''_i = \delta''_i \\
\qquad \mathbf{else}\ \rho'''_i \neq \mathbf{busy}^- \text{ and } (\rho'''_i = \mathbf{free} \text{ implies } [\![R_i]\!]_{\text{assert}} \delta'''_i) \Big) \\
\text{and} \\
[\![q]\!]_{\text{assert}} \delta'''_0 \text{ and } \forall_{i=1}^{n}\ \rho'''_i \neq \mathbf{busy}^+.
\end{array}
$$