Class Notes for CS 819 C - Spring 2002

6. Shared Variable Concurrency

John C. Reynolds
Department of Computer Science
Carnegie Mellon University

revised March 15, 2002

New Forms of Commands

$$\begin{aligned}\langle\text{comm}\rangle ::= {} & \textbf{resource}\ \langle\text{resname}\rangle(\langle\text{var}\rangle, \ldots, \langle\text{var}\rangle)\ \textbf{in}\ \langle\text{comm}\rangle \\ & \mid \textbf{with}\ \langle\text{resname}\rangle\ \textbf{when}\ \langle\text{boolexp}\rangle\ \textbf{do}\ \langle\text{comm}\rangle \\ & \mid \langle\text{comm}\rangle \parallel \cdots \parallel \langle\text{comm}\rangle\end{aligned}$$

In a resource declaration **resource** $r(x_1, \ldots, x_n)$ **in** C, the binder r has scope C, and the variables $x_1, \ldots, x_n$ must be distinct.

When resource declarations nest, i.e,

$$\textbf{resource}\ r(x_1, \ldots, x_n)\ \textbf{in}\ \ldots\ \textbf{resource}\ r'(x'_1, \ldots, x'_{n'})\ \textbf{in}\ \ldots,$$

r and r' must be distinct, and $x_1, \ldots, x_n, x'_1, \ldots, x'_{n'}$ must be distinct.

In the scope of **resource** $r(x_1, \ldots, x_n)$ **in** $\ldots$, each x_i is said to *belong* to the resource r, and may occur only in B or C of critical regions of the form **with** r **when** B **do** C.

In the concurrent composition $C_1 \parallel \cdots \parallel C_n$, if a variable does not belong to a resource and is assigned in some C_i, then it must not occur in any C_j when $j \neq i$.

Note that O'Hearn writes **region** r **when** B **do** C **endregion** instead of **with** r **when** B **do** C.

New Proof Rules

We assume that, associated with each resource r_i, there is an assertion RI_i, called the *resouce invariant* for r_i, which contains only variables belonging to r_i. Then we have

- Critical Regions

$$\frac{\{(P \ast RI_i) \wedge B\}\ C\ \{Q \ast RI_i\}}{\{P\}\ \textbf{with}\ r_i\ \textbf{when}\ B\ \textbf{do}\ C\ \{Q\},}$$

  where no other process modifies variables free in P or Q.

- Concurrent Composition

$$\frac{\{P_1\}\ C_1\ \{Q_1\} \qquad \cdots \qquad \{P_n\}\ C_n\ \{Q_n\}}{\{P_1 \ast \cdots \ast P_n\}\ C_1 \parallel \cdots \parallel C_n\ \{Q_1 \ast \cdots \ast Q_n\},}$$

  where no variable free in P_i or Q_i is assigned by C_j when $i \neq j$.

- Resource Declaration

$$\frac{\{P\}\ C\ \{Q\}}{\{RI_i \ast P\}\ \textbf{resource}\ r_i(x_1, \ldots, x_n)\ \textbf{in}\ C\ \{RI_i \ast Q\}.}$$

A Concurrent Cyclic Buffer

We assume that an n-element array has been allocated at location l. These variables will be used throughout our program but never assigned to. We will also use the variables

f	number of free elements
b	number of busy elements
$\mathsf{i}, \mathsf{i}_f, \mathsf{i}_b$	pointers to oldest free element
$\mathsf{j}, \mathsf{j}_f, \mathsf{j}_b$	pointers to oldest busy element
x, y	values to be placed in or removed from buffers

Without mentioning it explicitly in our assertions, we assume that the variables $\mathsf{i}, \mathsf{i}_f, \mathsf{i}_b, \mathsf{j}, \mathsf{j}_f, \mathsf{j}_b$ will range over the integers between l and $\mathsf{l} + \mathsf{n} - 1$ inclusive.

We write $x =_{\mathsf{n}} y$ when x and y are equal modulo n. We write $\oplus$ for the modular addition operator such that $x \oplus y =_{\mathsf{n}} x + y$ and $\mathsf{l} \leq x \oplus y < \mathsf{l} + \mathsf{n}$.

We introduce **resource** $\mathsf{free}(\mathsf{f}, \mathsf{i}_f, \mathsf{j}_f)$, with the invariant

$$RI_{\mathsf{free}} \equiv 0 \leq \mathsf{f} \leq \mathsf{n} \wedge \mathsf{j}_f =_{\mathsf{n}} \mathsf{i}_f + \mathsf{f} \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{f}-1}(\mathsf{i}_f \oplus \mathsf{k}) \mapsto -.$$

Then

$$\begin{array}{l}
\{((\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f) \;*\; RI_{\mathsf{free}}) \wedge \mathsf{f} > 0\} \\
\{((\mathsf{i} \mapsto -) \wedge \mathsf{i} = \mathsf{i}_f) \;*\; \\
\quad (0 \leq \mathsf{f} - 1 \leq \mathsf{n} \wedge \mathsf{j}_f =_{\mathsf{n}} \mathsf{i}_f \oplus 1 + \mathsf{f} - 1 \wedge \bigodot_{\mathsf{k}=1}^{\mathsf{f}-1}(\mathsf{i}_f \oplus \mathsf{k}) \mapsto -)\} \\
\mathsf{f} := \mathsf{f} - 1 \,;\, \mathsf{i}_f := \mathsf{i}_f \oplus 1 \\
\{((\mathsf{i} \mapsto -) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f) \;*\; RI_{\mathsf{free}}\},
\end{array}$$

so that

$$\begin{array}{rcl}
 & & \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f\} \\
\mathsf{getfree} & \equiv & \mathbf{with}\ \mathsf{free}\ \mathbf{when}\ \mathsf{f} > 0\ \mathbf{do} \\
 & & \quad (\mathsf{f} := \mathsf{f} - 1 \,;\, \mathsf{i}_f := \mathsf{i}_f \oplus 1) \\
 & & \{(\mathsf{i} \mapsto -) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f\}.
\end{array}$$

Also,

$$\begin{array}{l}
\{((\mathsf{j} \mapsto -) \wedge \mathsf{j} = \mathsf{j}_f) \;*\; RI_{\mathsf{free}}\} \\
\{(\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_f) \;*\; \\
\quad (0 \leq \mathsf{f} + 1 \leq \mathsf{n} \wedge \mathsf{j}_f \oplus 1 =_{\mathsf{n}} \mathsf{i}_f + \mathsf{f} + 1 \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{f}}(\mathsf{i}_f \oplus \mathsf{k}) \mapsto -)\} \\
\mathsf{f} := \mathsf{f} + 1 \,;\, \mathsf{j}_f := \mathsf{j}_f \oplus 1 \\
\{(\mathbf{emp} \wedge \mathsf{j} \oplus 1 = \mathsf{j}_f) \;*\; RI_{\mathsf{free}}\},
\end{array}$$

so that

$$\begin{array}{rcl}
 & & \{(\mathsf{j} \mapsto -) \wedge \mathsf{j} = \mathsf{j}_f\} \\
\mathsf{putfree} & \equiv & \mathbf{with}\ \mathsf{free}\ \mathbf{when}\ \mathbf{true}\ \mathbf{do} \\
 & & \quad (\mathsf{f} := \mathsf{f} + 1 \,;\, \mathsf{j}_f := \mathsf{j}_f \oplus 1) \\
 & & \{\mathbf{emp} \wedge \mathsf{j} \oplus 1 = \mathsf{j}_f\}.
\end{array}$$

Similarly, we introduce **resource** $\mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b)$, with invariant

$$RI_{\mathsf{busy}} \equiv 0 \leq \mathsf{b} \leq \mathsf{n} \wedge \mathsf{i}_b =_{\mathsf{n}} \mathsf{j}_b + \mathsf{b} \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{b}-1} (\mathsf{j}_b \oplus \mathsf{k}) \mapsto -.$$

Then

$$\begin{array}{l}
\{((\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b) \ * \ RI_{\mathsf{busy}}) \wedge \mathsf{b} > 0\} \\
\{((\mathsf{j} \mapsto -) \wedge \mathsf{j} = \mathsf{j}_b) \ * \\
\quad (0 \leq \mathsf{b} - 1 \leq \mathsf{n} \wedge \mathsf{i}_b =_{\mathsf{n}} \mathsf{j}_b \oplus 1 + \mathsf{b} - 1 \wedge \bigodot_{\mathsf{k}=1}^{\mathsf{b}-1} (\mathsf{j}_b \oplus \mathsf{k}) \mapsto -)\} \\
\mathsf{b} := \mathsf{b} - 1 \,;\, \mathsf{j}_b := \mathsf{j}_b \oplus 1 \\
\{((\mathsf{j} \mapsto -) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b) \ * \ RI_{\mathsf{busy}}\},
\end{array}$$

so that

$$\begin{array}{rcl}
& & \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b\} \\
\mathsf{getbusy} & \equiv & \mathbf{with}\ \mathsf{busy}\ \mathbf{when}\ \mathsf{b} > 0\ \mathbf{do} \\
& & \quad (\mathsf{b} := \mathsf{b} - 1 \,;\, \mathsf{j}_b := \mathsf{j}_b \oplus 1) \\
& & \{(\mathsf{j} \mapsto -) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b\}.
\end{array}$$

Also,

$$\begin{array}{l}
\{((\mathsf{i} \mapsto -) \wedge \mathsf{i} = \mathsf{i}_b) \ * \ RI_{\mathsf{busy}}\} \\
\{(\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_b) \ * \\
\quad (0 \leq \mathsf{b} + 1 \leq \mathsf{n} \wedge \mathsf{i}_b \oplus 1 =_{\mathsf{n}} \mathsf{j}_b + \mathsf{b} + 1 \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{b}} (\mathsf{j}_b \oplus \mathsf{k}) \mapsto -)\} \\
\mathsf{b} := \mathsf{b} + 1 \,;\, \mathsf{i}_b := \mathsf{i}_b \oplus 1 \\
\{(\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b) \ * \ RI_{\mathsf{busy}}\},
\end{array}$$

so that

$$\begin{array}{rcl}
& & \{(\mathsf{i} \mapsto -) \wedge \mathsf{i} = \mathsf{i}_b\} \\
\mathsf{putbusy} & \equiv & \mathbf{with}\ \mathsf{busy}\ \mathbf{when}\ \mathbf{true}\ \mathbf{do} \\
& & \quad (\mathsf{b} := \mathsf{b} + 1 \,;\, \mathsf{i}_b := \mathsf{i}_b \oplus 1) \\
& & \{\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b\}.
\end{array}$$

Note the symmetry: The second resource, its invariant, and its critical regions can be obtained from the first (or vice-versa) by interchanging free and busy (both as region names and as subscripts), interchanging f and b (both as variables and as subscripts), and interchanging i and j (as possibly subscripted variables).

Next, our first process is

$$
\begin{array}{rl}
 & \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
\mathsf{p}_1 \;\equiv & \mathbf{while}\ \text{“test}_1\text{”}\ \mathbf{do} \\
 & \quad \big(\{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
 & \quad \text{“Produce”}\ \mathsf{x}\,; \\
 & \quad \mathsf{getfree}\,; \\
 & \quad \{(\mathsf{i} \mapsto -) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
 & \quad [\mathsf{i}] := \mathsf{x}\,; \\
 & \quad \{(\mathsf{i} \mapsto -) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
 & \quad \mathsf{putbusy}\,; \\
 & \quad \{\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b\} \\
 & \quad \mathsf{i} := \mathsf{i} \oplus 1 \\
 & \quad \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\}\big) \\
 & \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b \wedge \neg\,\text{“test}_1\text{”}\}.
\end{array}
$$

Here we can use the resource variables i_f and i_b in assertions, since they will not be set by the second process p_2, which will only execute the critical regions $\mathsf{getbusy}$ and $\mathsf{putfree}$.

Similarly, in the second process, we will be able to use the resource variables j_f and j_b, since they are not set by the first process, which only executes the critical regions $\mathsf{getfree}$ and $\mathsf{putbusy}$:

$$
\begin{array}{rcl}
& & \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\} \\
\mathsf{p}_2 & \equiv & \mathbf{while}\ \text{“test}_2\text{”}\ \mathbf{do} \\
& & \quad \big(\{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\} \\
& & \quad \mathsf{getbusy}\ ; \\
& & \quad \{(\mathsf{j} \mapsto -) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\} \\
& & \quad \mathsf{y} := [\mathsf{j}]\ ; \\
& & \quad \{(\mathsf{j} \mapsto -) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\} \\
& & \quad \mathsf{putfree}\ ; \\
& & \quad \{\mathbf{emp} \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} \oplus 1 = \mathsf{j}_f\} \\
& & \quad \mathsf{j} := \mathsf{j} \oplus 1\ ; \\
& & \quad \text{“Consume”}\ \mathsf{y} \\
& & \quad \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\big) \\
& & \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f \wedge \neg\ \text{“test}_2\text{”}\}.
\end{array}
$$

Except for the occurrence of “Produce” in one and “Consume” in the other, the two processed are symmetric in the same sense as the two resources.

Finally, we have the main program

$$
\begin{array}{l}
\{\bigodot_{\mathsf{k}=0}^{\mathsf{n}-1}(\mathsf{l}+\mathsf{k}) \mapsto -\} \\
\mathsf{f} := \mathsf{n} \,;\, \mathsf{b} := 0 \,; \\
\mathsf{i} := \mathsf{l} \,;\, \mathsf{i}_f := \mathsf{l} \,;\, \mathsf{i}_b := \mathsf{l} \,;\, \mathsf{j} := \mathsf{l} \,;\, \mathsf{j}_f := \mathsf{l} \,;\, \mathsf{j}_b := \mathsf{l} \,; \\
\{RI_{\mathsf{free}} \ * \ RI_{\mathsf{busy}} \ * \ (\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b) \ * \\
\qquad (\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f)\} \\
\mathbf{resource}\ \mathsf{free}(\mathsf{f}, \mathsf{i}_f, \mathsf{j}_f)\ \mathbf{in}\ \mathbf{resource}\ \mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b)\ \mathbf{in}\ (\mathsf{p}_1 \parallel \mathsf{p}_2) \\
\{RI_{\mathsf{free}} \ * \ RI_{\mathsf{busy}} \ * \ (\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b \wedge \neg \text{“test}_1\text{”}) \ * \\
\qquad (\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f \wedge \neg \text{“test}_1\text{”})\}.
\end{array}
$$

Deleting assertions and putting everything together, we get

$$
\begin{array}{l}
\mathsf{f} := \mathsf{n}\ ;\ \mathsf{b} := 0\ ; \\
\mathsf{i} := \mathsf{l}\ ;\ \mathsf{i}_f := \mathsf{l}\ ;\ \mathsf{i}_b := \mathsf{l}\ ;\ \mathsf{j} := \mathsf{l}\ ;\ \mathsf{j}_f := \mathsf{l}\ ;\ \mathsf{j}_b := \mathsf{l}\ ; \\
\textbf{resource}\ \mathsf{free}(\mathsf{f}, \mathsf{i}_f, \mathsf{j}_f)\ \textbf{in resource}\ \mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b)\ \textbf{in} \\
\quad \big(\textbf{while}\ \text{“test}_1\text{”}\ \textbf{do} \\
\qquad\quad \big(\text{“Produce”}\ \mathsf{x}\ ; \\
\qquad\quad \textbf{with}\ \mathsf{free}\ \textbf{when}\ \mathsf{f} > 0\ \textbf{do} \\
\qquad\qquad (\mathsf{f} := \mathsf{f} - 1\ ;\ \mathsf{i}_f := \mathsf{i}_f \oplus 1)\ ; \\
\qquad\quad [\mathsf{i}] := \mathsf{x}\ ; \\
\qquad\quad \textbf{with}\ \mathsf{busy}\ \textbf{when true do} \\
\qquad\qquad (\mathsf{b} := \mathsf{b} + 1\ ;\ \mathsf{i}_b := \mathsf{i}_b \oplus 1)\ ; \\
\qquad\quad \mathsf{i} := \mathsf{i} \oplus 1\big) \\
\quad \parallel \textbf{while}\ \text{“test}_2\text{”}\ \textbf{do} \\
\qquad\quad \big(\textbf{with}\ \mathsf{busy}\ \textbf{when}\ \mathsf{b} > 0\ \textbf{do} \\
\qquad\qquad (\mathsf{b} := \mathsf{b} - 1\ ;\ \mathsf{j}_b := \mathsf{j}_b \oplus 1)\ ; \\
\qquad\quad \mathsf{y} := [\mathsf{j}]\ ; \\
\qquad\quad \textbf{with}\ \mathsf{free}\ \textbf{when true do} \\
\qquad\qquad (\mathsf{f} := \mathsf{f} + 1\ ;\ \mathsf{j}_f := \mathsf{j}_f \oplus 1)\ ; \\
\qquad\quad \mathsf{j} := \mathsf{j} \oplus 1\ ; \\
\qquad\quad \text{“Consume”}\ \mathsf{y}\big)\big)
\end{array}
$$

It is easy to see that $\mathsf{i}_f, \mathsf{i}_b, \mathsf{j}_f, \mathsf{j}_b$ are all auxilliary. Removing the assignments to these variables gives:

$$
\begin{array}{l}
\mathsf{f} := \mathsf{n}\,;\, \mathsf{b} := 0\,;\, \mathsf{i} := \mathsf{l}\,;\, \mathsf{j} := \mathsf{l}\,;\\
\textbf{resource}\ \mathsf{free}(\mathsf{f})\ \textbf{in resource}\ \mathsf{busy}(\mathsf{b})\ \textbf{in}\\
\quad \big(\textbf{while}\ \text{“test}_1\text{”}\ \textbf{do}\\
\qquad \big(\text{“Produce”}\ \mathsf{x}\,;\\
\qquad \textbf{with}\ \mathsf{free}\ \textbf{when}\ \mathsf{f} > 0\ \textbf{do}\ \mathsf{f} := \mathsf{f} - 1\,;\\
\qquad [\mathsf{i}] := \mathsf{x}\,;\\
\qquad \textbf{with}\ \mathsf{busy}\ \textbf{when true do}\ \mathsf{b} := \mathsf{b} + 1\,;\\
\qquad \mathsf{i} := \mathsf{i} \oplus 1\big)\\
\quad \|\ \textbf{while}\ \text{“test}_2\text{”}\ \textbf{do}\\
\qquad \big(\textbf{with}\ \mathsf{busy}\ \textbf{when}\ \mathsf{b} > 0\ \textbf{do}\ \mathsf{b} := \mathsf{b} - 1\,;\\
\qquad \mathsf{y} := [\mathsf{j}]\,;\\
\qquad \textbf{with}\ \mathsf{free}\ \textbf{when true do}\ \mathsf{f} := \mathsf{f} + 1\,;\\
\qquad \mathsf{j} := \mathsf{j} \oplus 1\,;\\
\qquad \text{“Consume”}\ \mathsf{y}\big)\big)
\end{array}
$$

Keeping Track of the Buffer Contents

We introduce the following sequence variables:

α the sequence of values that have been produced
β the sequence of values that have been consumed
γ the sequence of values currently stored in the buffer

Then we add these variables to **resource** $\mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b, \alpha, \beta, \gamma)$, and modify its invariant:

$$RI_{\mathsf{busy}} \equiv 0 \leq \mathsf{b} \leq \mathsf{n} \wedge \mathsf{i}_b =_{\mathsf{n}} \mathsf{j}_b + \mathsf{b} \wedge \\ \alpha = \beta{\cdot}\gamma \wedge \#\gamma = \mathsf{b} \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{b}-1} (\mathsf{j}_b \oplus \mathsf{k}) \mapsto \gamma_{\mathsf{k}}.$$

Next, we modify the critical regions for busy. We have

$$\begin{array}{l}
\{((\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b) \ast RI_{\mathsf{busy}}) \wedge \mathsf{b} > 0\} \\
\{((\mathsf{j} \mapsto \mathsf{last}(\beta{\cdot}\mathsf{first}(\gamma))) \wedge \mathsf{j} = \mathsf{j}_b) \ast \\
\quad (0 \leq \mathsf{b} - 1 \leq \mathsf{n} \wedge \mathsf{i}_b =_{\mathsf{n}} \mathsf{j}_b \oplus 1 + \mathsf{b} - 1 \wedge \\
\qquad \alpha = \beta{\cdot}\mathsf{first}(\gamma){\cdot}\mathsf{rest}(\gamma) \wedge \#\mathsf{rest}(\gamma) = \mathsf{b} - 1 \wedge \\
\qquad \bigodot_{\mathsf{k}=0}^{\mathsf{b}-2} (\mathsf{j}_b \oplus 1 \oplus \mathsf{k}) \mapsto (\mathsf{rest}(\gamma))_{\mathsf{k}})\} \\
\mathsf{b} := \mathsf{b} - 1 \,;\, \mathsf{j}_b := \mathsf{j}_b \oplus 1 \,;\, \beta := \beta{\cdot}\mathsf{first}(\gamma) \,;\, \gamma := \mathsf{rest}(\gamma) \\
\{((\mathsf{j} \mapsto \mathsf{last}(\beta)) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b) \ast RI_{\mathsf{busy}}\},
\end{array}$$

so that

$$\begin{array}{rcl}
 & & \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b\} \\
\mathsf{getbusy} & \equiv & \mathbf{with}\ \mathsf{busy}\ \mathbf{when}\ \mathsf{b} > 0\ \mathbf{do} \\
 & & \quad (\mathsf{b} := \mathsf{b} - 1 \,;\, \mathsf{j}_b := \mathsf{j}_b \oplus 1 \,;\, \beta := \beta{\cdot}\mathsf{first}(\gamma) \,;\, \gamma := \mathsf{rest}(\gamma)) \\
 & & \{(\mathsf{j} \mapsto \mathsf{last}(\beta)) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b\}.
\end{array}$$

Also,

$$
\begin{array}{l}
\{((\mathsf{i} \mapsto \mathsf{x}) \wedge \mathsf{i} = \mathsf{i}_b) \ * \ RI_{\mathsf{busy}}\} \\
\{(\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_b) \ * \\
\quad (0 \leq \mathsf{b} + 1 \leq \mathsf{n} \wedge \mathsf{i}_b \oplus 1 =_{\mathsf{n}} \mathsf{j}_b + \mathsf{b} + 1 \wedge \\
\qquad \alpha{\cdot}\mathsf{x} = \beta{\cdot}\gamma{\cdot}\mathsf{x} \wedge \#\gamma{\cdot}\mathsf{x} = \mathsf{b} + 1 \wedge \bigodot_{\mathsf{k}=0}^{\mathsf{b}} (\mathsf{j}_b \oplus \mathsf{k}) \mapsto (\gamma{\cdot}\mathsf{x})_{\mathsf{k}})\} \\
\mathsf{b} := \mathsf{b} + 1 \ ; \mathsf{i}_b := \mathsf{i}_b \oplus 1 \ ; \alpha := \alpha{\cdot}\mathsf{x} \ ; \gamma := \gamma{\cdot}\mathsf{x} \\
\{(\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b) \ * \ RI_{\mathsf{busy}}\},
\end{array}
$$

so that

$$
\begin{array}{rcl}
 & & \{(\mathsf{i} \mapsto \mathsf{x}) \wedge \mathsf{i} = \mathsf{i}_b\} \\
\mathsf{putbusy} & \equiv & \textbf{with } \mathsf{busy} \textbf{ when true do} \\
 & & \quad (\mathsf{b} := \mathsf{b} + 1 \ ; \mathsf{i}_b := \mathsf{i}_b \oplus 1 \ ; \alpha := \alpha{\cdot}\mathsf{x} \ ; \gamma := \gamma{\cdot}\mathsf{x}) \\
 & & \{\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b\}.
\end{array}
$$

The first process becomes

$$
\begin{array}{rcl}
& & \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
\mathsf{p}_1 & \equiv & \mathbf{while}\ \text{“test}_1\text{”}\ \mathbf{do} \\
& & \quad \big(\{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
& & \quad \text{“Produce”}\ \mathsf{x}\,; \\
& & \quad \mathsf{getfree}\,; \\
& & \quad \{(\mathsf{i} \mapsto -) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
& & \quad [\mathsf{i}] := \mathsf{x}\,; \\
& & \quad \{(\mathsf{i} \mapsto \mathsf{x}) \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\} \\
& & \quad \mathsf{putbusy}\,; \\
& & \quad \{\mathbf{emp} \wedge \mathsf{i} \oplus 1 = \mathsf{i}_f \wedge \mathsf{i} \oplus 1 = \mathsf{i}_b\} \\
& & \quad \mathsf{i} := \mathsf{i} \oplus 1 \\
& & \quad \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b\}\big) \\
& & \{\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b \wedge \neg\, \text{“test}_1\text{”}\},
\end{array}
$$

Note that p_1 does not assign to β, since $\mathsf{putbusy}$ does not assign to β. Thus we may use β in the assertions of the second process.

The second process becomes

$$
\begin{aligned}
&\{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\\
\mathsf{p}_2 \;\equiv\; &\textbf{while } \text{“test}_2\text{” } \textbf{do}\\
&\quad \big(\{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\\
&\quad \mathsf{getbusy}\;;\\
&\quad \{(\mathsf{j} \mapsto \mathsf{last}(\beta)) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\\
&\quad \mathsf{y} := [\mathsf{j}]\;;\\
&\quad \{(\mathsf{j} \mapsto -) \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\\
&\quad \mathsf{putfree}\;;\\
&\quad \{\mathbf{emp} \wedge \mathsf{j} \oplus 1 = \mathsf{j}_b \wedge \mathsf{j} \oplus 1 = \mathsf{j}_f\}\\
&\quad \mathsf{j} := \mathsf{j} \oplus 1\;;\\
&\quad \text{“Consume” } \mathsf{y}\\
&\quad \{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f\}\big)\\
&\{\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f \wedge \neg \text{“test}_2\text{”}\}.
\end{aligned}
$$

Finally, we have

$$
\begin{aligned}
&\{\textstyle\bigodot_{\mathsf{k}=0}^{\mathsf{n}-1}(\mathsf{l} + \mathsf{k}) \mapsto -\}\\
&\mathsf{f} := \mathsf{n}\;;\; \mathsf{b} := 0\;;\; \alpha := \epsilon\;;\; \beta := \epsilon\;;\; \gamma := \epsilon\;;\\
&\mathsf{i} := \mathsf{l}\;;\; \mathsf{i}_f := \mathsf{l}\;;\; \mathsf{i}_b := \mathsf{l}\;;\; \mathsf{j} := \mathsf{l}\;;\; \mathsf{j}_f := \mathsf{l}\;;\; \mathsf{j}_b := \mathsf{l}\;;\\
&\{RI_{\mathsf{free}} \,*\, RI_{\mathsf{busy}} \,*\, (\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b) \,*\\
&\qquad (\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f)\}\\
&\textbf{resource } \mathsf{free}(\mathsf{f}, \mathsf{i}_f, \mathsf{j}_f) \textbf{ in resource } \mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b, \alpha, \beta, \gamma) \textbf{ in } (\mathsf{p}_1 \parallel \mathsf{p}_2)\\
&\{RI_{\mathsf{free}} \,*\, RI_{\mathsf{busy}} \,*\, (\mathbf{emp} \wedge \mathsf{i} = \mathsf{i}_f \wedge \mathsf{i} = \mathsf{i}_b \wedge \neg \text{“test}_1\text{”}) \,*\\
&\qquad (\mathbf{emp} \wedge \mathsf{j} = \mathsf{j}_b \wedge \mathsf{j} = \mathsf{j}_f \wedge \neg \text{“test}_1\text{”})\}.
\end{aligned}
$$

Deleting most assertions and putting everything together, we get

$$
\begin{array}{l}
\mathsf{f} := \mathsf{n}\,;\, \mathsf{b} := 0\,;\, \alpha := \epsilon\,;\, \beta := \epsilon\,;\, \gamma := \epsilon\,;\\
\mathsf{i} := \mathsf{l}\,;\, \mathsf{i}_f := \mathsf{l}\,;\, \mathsf{i}_b := \mathsf{l}\,;\, \mathsf{j} := \mathsf{l}\,;\, \mathsf{j}_f := \mathsf{l}\,;\, \mathsf{j}_b := \mathsf{l}\,;\\
\textbf{resource}\ \mathsf{free}(\mathsf{f}, \mathsf{i}_f, \mathsf{j}_f)\ \textbf{in resource}\ \mathsf{busy}(\mathsf{b}, \mathsf{j}_b, \mathsf{i}_b, \alpha, \beta, \gamma)\ \textbf{in}\\
\quad \big(\textbf{while}\ \text{“test}_1\text{”}\ \textbf{do}\\
\qquad \big(\text{“Produce”}\ \mathsf{x}\,;\\
\qquad \textbf{with}\ \mathsf{free}\ \textbf{when}\ \mathsf{f} > 0\ \textbf{do}\\
\qquad\quad (\mathsf{f} := \mathsf{f} - 1\,;\, \mathsf{i}_f := \mathsf{i}_f \oplus 1)\,;\\
\qquad [\mathsf{i}] := \mathsf{x}\,;\\
\qquad \textbf{with}\ \mathsf{busy}\ \textbf{when true do}\\
\qquad\quad (\mathsf{b} := \mathsf{b} + 1\,;\, \mathsf{i}_b := \mathsf{i}_b \oplus 1\,;\, \alpha := \alpha{\cdot}\mathsf{x}\,;\, \gamma := \gamma{\cdot}\mathsf{x})\,;\\
\qquad \mathsf{i} := \mathsf{i} \oplus 1\big)\\
\quad \|\ \textbf{while}\ \text{“test}_2\text{”}\ \textbf{do}\\
\qquad \big(\textbf{with}\ \mathsf{busy}\ \textbf{when}\ \mathsf{b} > 0\ \textbf{do}\\
\qquad\quad (\mathsf{b} := \mathsf{b} - 1\,;\, \mathsf{j}_b := \mathsf{j}_b \oplus 1\,;\, \beta := \beta{\cdot}\mathsf{first}(\gamma)\,;\, \gamma := \mathsf{rest}(\gamma))\,;\\
\qquad \{(\mathsf{j} \mapsto \mathsf{last}(\beta))\}\\
\qquad \mathsf{y} := [\mathsf{j}]\,;\\
\qquad \textbf{with}\ \mathsf{free}\ \textbf{when true do}\\
\qquad\quad (\mathsf{f} := \mathsf{f} + 1\,;\, \mathsf{j}_f := \mathsf{j}_f \oplus 1)\,;\\
\qquad \mathsf{j} := \mathsf{j} \oplus 1\,;\\
\qquad \text{“Consume”}\ \mathsf{y}\big)\big)
\end{array}
$$