Concurrent Separation Logic

Peter O'Hearn

Queen Mary, University of London

MFPS Tutorial, Genova, 23 May, 2006

Part I

Independence and Separation

The Good Old Days

- *We have stipulated that processes should be loosely connected; by this we mean that apart from the (rare) moments of explicit intercommunication, the individual processes are to be regarded as completely independent of each other.*
  (Cooperating Sequential Processes, EW Dijkstra, 1965)

- *[The problem of] designing a program to control the fantastic number of combinations involved in arbitrary interleaving.*
  (Towards a Theory of Parallel Programming, CAR Hoare, 1972)

Towards a Theory of Parallel Programming, a forgotten (or overlooked) gem

- The disjoint concurrency rule:

$$\frac{\{P\}C\{Q\} \quad \{P'\}C'\{Q'\}}{\{P \wedge P'\}C \parallel C'\{Q \wedge Q'\}}$$

  where C does not modify any variables free in P', C', Q', and conversely.
- Independent, sequential reasoning about processes.
- Beautifully modular rules for process interaction (CCRs, see later), implementing Dijkstra's principle.
- People moved on from TTPP because of certain limitations. But its modularity was not preserved by its successors.

Example: Parallel Mergesort

$\{array(a, i, j)\}$
$\texttt{procedure ms}(a, i, j)$
$\texttt{newvar } m := (i + j)/2;$
$\texttt{if } \; i < j \texttt{ then}$
$\quad \big(\texttt{ms}(a, i, m) \parallel \texttt{ms}(a, m + 1, j)\big);$
$\quad \texttt{merge}(a, i, m + 1, j);$
$\{sorted(a, i, j)\}$

- $\blacktriangleright$

Example: Parallel Mergesort

$\{array(a, i, j)\}$
$\texttt{procedure ms}(a, i, j)$
$\texttt{newvar } m := (i + j)/2;$
$\texttt{if } \; i < j \texttt{ then}$
$\quad (\texttt{ms}(a, i, m) \parallel \texttt{ms}(a, m + 1, j));$
$\quad \texttt{merge}(a, i, m + 1, j);$
$\{sorted(a, i, j)\}$

- Can't prove with disjoint concurrency rule because Hoare logic treats assignment to array component as assignment to whole.

$$\{P[(a \mid i : E)/a]\}\, a[i] := E\, \{P\}$$

The two parallel calls (are judged to) alter the *same* variable, a.

Example: Parallel Mergesort

$\{array(a, i, j)\}$
$\texttt{procedure ms}(a, i, j)$
$\texttt{newvar } m := (i + j)/2;$
$\texttt{if } \; i < j \texttt{ then}$
$\quad (\texttt{ms}(a, i, m) \parallel \texttt{ms}(a, m + 1, j));$
$\quad \texttt{merge}(a, i, m + 1, j);$
$\{sorted(a, i, j)\}$

- Very hard to make sense of in Owicki-Gries logic. The "non-interference with proofs" notion must be married to recursion hypotheses. (uugh)

Example: Parallel Mergesort

$\{array(a, i, j)\}$
$\texttt{procedure ms}(a, i, j)$
$\texttt{newvar } m := (i + j)/2;$
$\texttt{if } \; i < j \texttt{ then}$
$\quad (\texttt{ms}(a, i, m) \parallel \texttt{ms}(a, m + 1, j));$
$\quad \texttt{merge}(a, i, m + 1, j);$
$\{sorted(a, i, j)\}$

- Can prove with Rely/guarantee (Jones), but at the cost of complicating the specification (not just the reasoning)
  - Rely: no one else touches my segment
  - Guarantee: I only touch my own segment (frame axiom)

Separation Logic

- We just use the pre/post spec.
- Main part of proof

$$\{array(a,i,m) * array(a,m+1,j)\}$$
$$\begin{array}{lcl} \{array(a,i,m)\} & & \{array(a,m+1,j)\} \\ \mathtt{ms}(a,i,m) & \| & \mathtt{ms}(a,m+1,j) \\ \{sorted(a,i,m)\} & & \{sorted(a,m+1,j)\} \end{array}$$
$$\{sorted(a,i,m) * sorted(a,m+1,j)\}$$

- Proof rule:

$$\frac{\{P_1\}C_1\{Q_1\} \quad \{P_2\}C_2\{Q_2\}}{\{P_1 * P_2\}C_1 \| C_2\{Q_1 * Q_2\}}$$

Disjoint Concurrency: A Closer Look

$$\frac{\{P_1\}\,C_1\,\{Q_1\} \quad \{P_2\}\,C_2\,\{Q_2\}}{\{P_1 * P_2\}\,C_1 \parallel C_2\,\{Q_1 * Q_2\}}$$

Remember, well-specified programs don't go wrong.

$$\{10 \mapsto -\}[10] := 42\{10 \mapsto 42\}$$

$$\{\texttt{emp}\}[10] := 42]\{??\}$$

Disjoint Concurrency: A Closer Look

$$\frac{\{P_1\}\,C_1\,\{Q_1\} \quad \{P_2\}\,C_2\,\{Q_2\}}{\{P_1 * P_2\}\,C_1 \parallel C_2\,\{Q_1 * Q_2\}}$$

We can't prove racy programs like

$$\{10 \mapsto -\}$$
$$[10] := 42 \;\parallel\; [10] := 6$$
$$\{??\}$$

We cannot send 10 to both processes in their preconditions, since

$$(10 \mapsto -) * (10 \mapsto -)$$

is false. But...

Disjoint Concurrency: A Closer Look

$$\frac{\{P_1\}\,C_1\,\{Q_1\} \quad \{P_2\}\,C_2\,\{Q_2\}}{\{P_1 * P_2\}\,C_1 \parallel C_2\,\{Q_1 * Q_2\}}$$

Preconditions can pick out race-free start-states, when they exist:

$$\begin{array}{lcl} \{x \mapsto 3\} & & \{y \mapsto 3\} \\ [x] := 4 & \parallel & [y] := 5 \\ \{x \mapsto 4\} & & \{y \mapsto 5\} \end{array}$$

Disjoint Concurrency: A Closer Look

$$\frac{\{P_1\}\,C_1\,\{Q_1\} \quad \{P_2\}\,C_2\,\{Q_2\}}{\{P_1 * P_2\}\,C_1 \parallel C_2\,\{Q_1 * Q_2\}}$$

Preconditions can pick out race-free start-states, when they exist:

$$\begin{array}{c} \{x \mapsto 3 * y \mapsto 3\} \\ \{x \mapsto 3\} \qquad\qquad \{y \mapsto 3\} \\ [x]{:=}\,4 \qquad \parallel \qquad [y]{:=}\,5 \\ \{x \mapsto 4\} \qquad\qquad \{y \mapsto 5\} \\ \{x \mapsto 4 * y \mapsto 5\} \end{array}$$

That 'proof figure" is an annotation form for

$$\frac{\{x \mapsto 3\}\,[x]{:=}\,4\,\{x \mapsto 4\} \qquad \{y \mapsto 3\}\,[y]{:=}\,5\,\{y \mapsto 5\}}{\{x \mapsto 3 * y \mapsto 3\}\,[x]{:=}\,4 \parallel [y]{:=}\,5\,\{x \mapsto 4 * y \mapsto 5\}}$$

Part II

Process Interaction

The Programming Model:

Conditional Critical Regions (Hoare, 72)

init;
`resource` r_1(variable list), ..., r_m(variable list)
$C_1 \parallel \cdots \parallel C_n$

`with` r `when` B `do` C `endwith`.

–if a variable belongs to a resource, it cannot appear in a parallel process except in a critical section for that resource

–if a variable is changed in one process, it cannot appear in another unless it belongs to a resource.

The Sequential Processes

$$
\begin{array}{rcl}
C & ::= & x{:=}\,E \mid x{:=}\,[E] \mid [E]{:=}\,F \\
& \mid & x{:=}\,\mathtt{cons}(E_1, \ldots, E_n) \mid \mathtt{dispose}(E) \\
& \mid & \mathtt{skip} \mid C; C \mid \mathtt{if}\ B\ \mathtt{then}\ C\ \mathtt{else}\ C \\
& \mid & \mathtt{while}\ B\ \mathtt{do}\ C \\
& \mid & \mathtt{with}\ r\ \mathtt{when}\ B\ \mathtt{do}\ C\ \mathtt{endwith}
\end{array}
$$

Some sugar: $x.i{:=}\,E$ for $[x + i - 1]{:=}\,E$

$$\frac{\{P\}\mathit{init}\{RI_{r_1} * \cdots * RI_{r_m} * P'\} \qquad \{P'\}C_1 \parallel \cdots \parallel C_n\{Q\}}{}$$

$\{P\}$

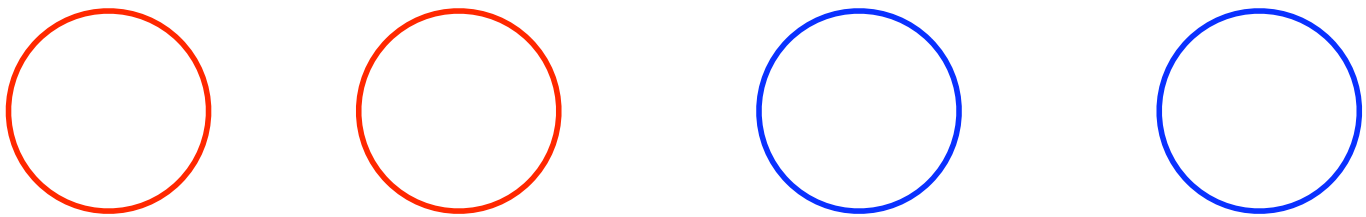
init;

`resource` r_1(variable list), . . . , r_m(variable list)

$C_1 \parallel \cdots \parallel C_n$

$\{RI_{r_1} * \cdots * RI_{r_m} * Q\}$

RI * · · · * RI P'



$$\frac{\{P\}\,init\,\{RI_{r_1} * \cdots * RI_{r_m} * P'\} \qquad \{P'\}\,C_1 \parallel \cdots \parallel C_n\,\{Q\}}{}$$

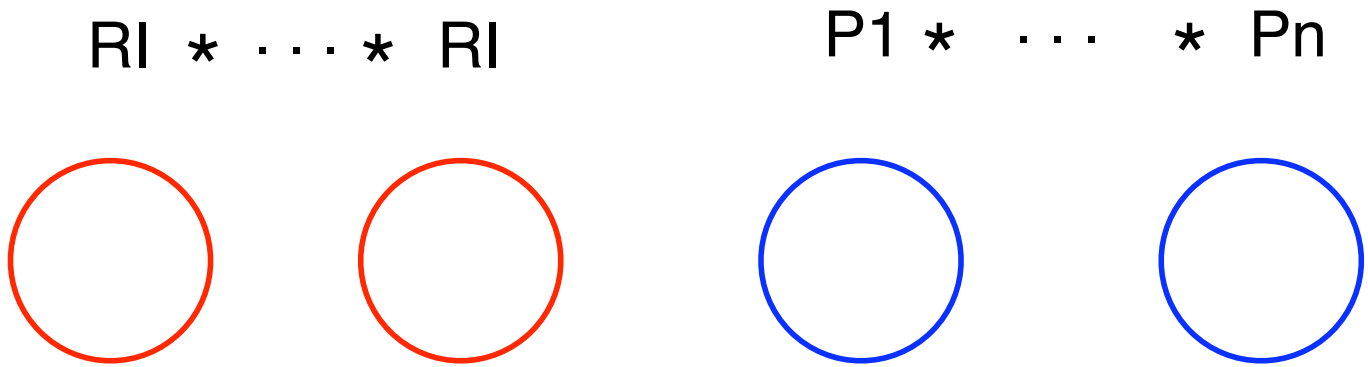
$\{P\}$

init;

`resource` r_1(variable list), . . . , r_m(variable list)

$C_1 \parallel \cdots \parallel C_n$

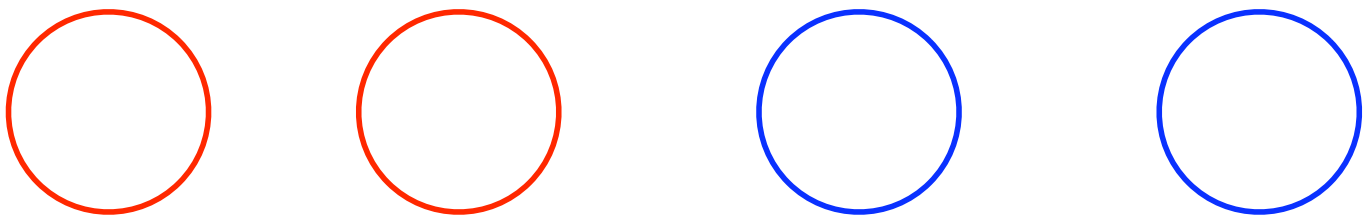
$\{RI_{r_1} * \cdots * RI_{r_m} * Q\}$



$$\frac{\{P_1\}\, C_1\, \{Q_1\} \;\cdots\; \{P_n\}\, C_n\, \{Q_n\}}{\{P_1 * \cdots * P_n\}\, C_1 \parallel \cdots \parallel C_n\, \{Q_1 * \cdots * Q_n\}}$$

RI $*$ $\cdots$ $*$ RI

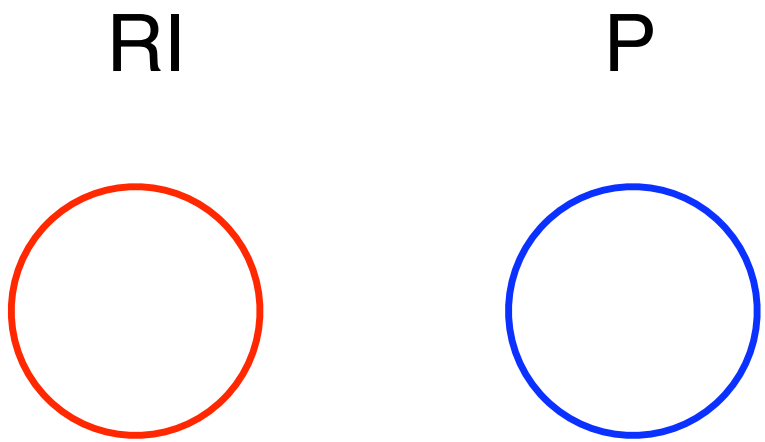
P1 $*$ $\cdots$ $*$ Pn



– no variable free in P_i or Q_i is changed in C_j when $j \neq i$.

– any modified variable free in RI_r must occur only within a critical region for r.

$$\frac{\{(P * RI_r) \wedge B\}\ C\ \{Q * RI_r\}}{\{P\}\ \texttt{with}\ r\ \texttt{when}\ B\ \texttt{do}\ C\ \texttt{endwith}\ \{Q\}}$$



- No other process modifies variables in P or Q

Example: Pointer Transfer

$$\texttt{resource}\ \mathit{buf}(c, \mathit{full} := \texttt{false});$$

$$\begin{array}{l} x := \texttt{cons}(-); \\ \texttt{with}\ \mathit{buf}\ \texttt{when}\ \neg \mathit{full}\ \textbf{do} \\ \quad (c := x;\ \mathit{full} := \texttt{true};) \end{array} \quad \| \quad \begin{array}{l} \texttt{with}\ \mathit{buf}\ \texttt{when}\ \mathit{full}\ \textbf{DO} \\ \quad (y := c;\ \mathit{full} := \texttt{false};) \\ \texttt{dispose}(y); \end{array}$$

Example: Pointer Transfer

$$\texttt{resource}\ \mathit{buf}(c, \mathit{full} := \texttt{false});$$

$$\begin{array}{l} x := \texttt{cons}(-); \\ \texttt{with}\ \mathit{buf}\ \texttt{when}\ \neg \mathit{full}\ \textbf{do} \\ \quad (c := x;\ \mathit{full} := \texttt{true};) \\ \texttt{dispose}(x); \end{array} \quad \| \quad \begin{array}{l} \texttt{with}\ \mathit{buf}\ \texttt{when}\ \mathit{full}\ \textbf{DO} \\ \quad (y := c;\ \mathit{full} := \texttt{false};) \\ \texttt{dispose}(y); \end{array}$$

Example: Pointer Transfer

$$\texttt{resource}\ \mathit{buf}(c, \mathit{full} := \texttt{false});$$

$$\begin{array}{l} x := \texttt{cons}(-); \\ \texttt{with}\ \mathit{buf}\ \texttt{when}\ \neg \mathit{full}\ \textbf{do} \\ \quad (c := x;\ \mathit{full} := \texttt{true};) \end{array} \quad \| \quad \begin{array}{l} \texttt{with}\ \mathit{buf}\ \texttt{when}\ \mathit{full}\ \textbf{DO} \\ \quad (y := c;\ \mathit{full} := \texttt{false};) \\ \texttt{dispose}(y); \end{array}$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} \wedge \mathit{full}\}$

$x := \mathtt{cons}(-);$		$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$	$\parallel$	$y := c;\ \mathit{full} := \mathtt{false}$
$c := x;\ \mathit{full} := \mathtt{true};$		$\mathtt{endwith};$
		$\mathtt{dispose}(y);$
$\mathtt{endwith};$		

$$RI \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * (\mathtt{emp} \wedge \neg \mathit{full})\}$

$x := \mathtt{cons}(-);$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$\quad c := x;\ \mathit{full} := \mathtt{true};$

$\mathtt{endwith};$

$\|$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$\quad y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$RI \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$x := \mathtt{cons}(-);$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$\quad c := x;\ \mathit{full} := \mathtt{true};$

$\mathtt{endwith};$

$\|$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$\quad y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$

$x := \mathtt{cons}(-);$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$c := x;\ \mathit{full} := \mathtt{true};$

$\mathtt{endwith};$

$\|$

$\{\mathtt{emp}\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$

$x := \mathtt{cons}(-);$

$\{x \mapsto -\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$c := x;\ \mathit{full} := \mathtt{true};$

$\mathtt{endwith};$

$\|$

$\{\mathtt{emp}\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$

$x := \mathtt{cons}(-);$

$\{x \mapsto -\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$

$c := x;\ \mathit{full} := \mathtt{true};$

$\mathtt{endwith};$

$\|$

$\{\mathtt{emp}\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$

$x := \mathtt{cons}(-);$

$\{x \mapsto -\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$

$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$

$c := x;\ \mathit{full} := \mathtt{true};$

$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$

$\mathtt{endwith};$

$\|$

$\{\mathtt{emp}\}$

$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$

$y := c;\ \mathit{full} := \mathtt{false}$

$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$		$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$		$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{x \mapsto -\}$		
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$	$\|$	$y := c;\ \mathit{full} := \mathtt{false}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$		
$c := x;\ \mathit{full} := \mathtt{true};$		$\mathtt{endwith};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$		
$\{(c \mapsto - \wedge \mathit{full}\}$		$\mathtt{dispose}(y);$
$\mathtt{endwith};$		

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$	$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$	$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{x \mapsto -\}$	
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$	$y := c;\ \mathit{full} := \mathtt{false}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$	
$c := x;\ \mathit{full} := \mathtt{true};$	$\mathtt{endwith};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$	
$\{\mathit{RI}\}$	$\mathtt{dispose}(y);$
$\mathtt{endwith};$	

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$	$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$	$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{x \mapsto -\}$	
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$	$y := c;\ \mathit{full} := \mathtt{false}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$	
$c := x;\ \mathit{full} := \mathtt{true};$	$\mathtt{endwith};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$	
$\{\mathtt{emp} * \mathit{RI}\}$	$\mathtt{dispose}(y);$
$\mathtt{endwith};$	

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$	$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$	$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{x \mapsto -\}$	
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$	$\Vert \quad y := c;\ \mathit{full} := \mathtt{false}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$	
$c := x;\ \mathit{full} := \mathtt{true};$	$\mathtt{endwith};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$	
$\{\mathtt{emp} * \mathit{RI}\}$	$\mathtt{dispose}(y);$
$\mathtt{endwith};$	
$\{\mathtt{emp}\}$	

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$

$\|$

$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$

$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$

$\|$

$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$
$\{y \mapsto -\}$
$\mathtt{dispose}(y);$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$

$\|$

$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$
$\{y \mapsto -\}$
$\mathtt{dispose}(y);$
$\{\mathtt{emp}\}$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$
$\mathtt{dispose}(x);$

$\|$

$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$
$\{y \mapsto -\}$
$\mathtt{dispose}(y);$
$\{\mathtt{emp}\}$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$
$\mathtt{dispose}(x);$
$\{???\}$

$\|$

$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$
$\{y \mapsto -\}$
$\mathtt{dispose}(y);$
$\{\mathtt{emp}\}$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$\{\mathtt{emp}\}$

$\mathtt{resource}\ \mathit{buf}(c, \mathit{full} := \mathtt{false});$

$\{\mathtt{emp} * \mathtt{emp} * \mathit{RI}\}$

$\{\mathtt{emp}\}$
$x := \mathtt{cons}(-);$
$\{x \mapsto -\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \neg \mathit{full}\ \mathbf{do}$
$\{(x \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full})\}$
$c := x;\ \mathit{full} := \mathtt{true};$
$\{(c \mapsto -) * (\mathtt{emp} \wedge \mathit{full})\}$
$\{\mathtt{emp} * \mathit{RI}\}$
$\mathtt{endwith};$
$\{\mathtt{emp}\}$

$\|$

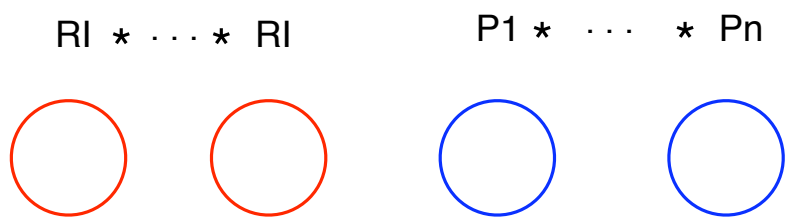
$\{\mathtt{emp}\}$
$\mathtt{with}\ \mathit{buf}\ \mathtt{when}\ \mathit{full}\ \mathbf{do}$
$\{\mathtt{emp} * ((c \mapsto -) \wedge \mathit{full})\}$
$y := c;\ \mathit{full} := \mathtt{false}$
$\{(y \mapsto -) * (\mathtt{emp} \wedge \neg \mathit{full}$
$\mathtt{endwith};$
$\{y \mapsto -\}$
$\mathtt{dispose}(y);$
$\{\mathtt{emp}\}$

$\{\mathit{RI}\}$

$$\mathit{RI} \quad = \quad (\mathtt{emp} \wedge \neg \mathit{full}) \vee (c \mapsto - \wedge \mathit{full})$$

$$\frac{\{P_1\}\, C_1\, \{Q_1\} \;\cdots\; \{P_n\}\, C_n\, \{Q_n\}}{\{P_1 * \cdots * P_n\}\, C_1 \parallel \cdots \parallel C_n\, \{Q_1 * \cdots * Q_n\}}$$

$$\frac{\{(P * RI_r) \wedge B\}\, C\, \{Q * RI_r\}}{\{P\}\, \texttt{with}\ r\ \texttt{when}\ B\ \texttt{do}\ C\ \texttt{endwith}\, \{Q\}}$$



Semaphores

- We can encode semaphores as little critical regions

$$P(s) \quad \equiv \quad \texttt{with}\ s\ \texttt{when}\ s > 0\ \texttt{do}\ s := s - 1$$

$$V(s) \quad \equiv \quad \texttt{with number when true do}\ s := s + 1$$

- So, to reason about a semaphore we must invent an invariant, describing the resource that it "owns".

$$
\begin{array}{lcl}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 & & \\
\vdots & & \vdots \\
P(\mathit{free}); & & P(\mathit{busy}); \\
[10] := m; & \| & n := [10]; \\
V(\mathit{busy}); & & V(\mathit{free}); \\
\vdots & & \vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \qquad\qquad\qquad\qquad\qquad\qquad\quad P(\mathit{busy}); \\
[10] := m; \qquad\qquad \| \qquad\qquad n := [10]; \\
V(\mathit{busy}); \qquad\qquad\qquad\qquad\qquad\quad V(\mathit{free}); \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \qquad\qquad\qquad\qquad\qquad\qquad\quad P(\mathit{busy}); \\
\{10 \mapsto -\} \\
[10] := m; \qquad\qquad \| \qquad\qquad n := [10]; \\
\\
V(\mathit{busy}); \qquad\qquad\qquad\qquad\qquad\quad V(\mathit{free}); \\
\\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \qquad\qquad\qquad\qquad\qquad\qquad\quad P(\mathit{busy}); \\
\{10 \mapsto -\} \\
[10] := m; \qquad\qquad \| \qquad\qquad n := [10]; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \qquad\qquad\qquad\qquad\qquad\quad V(\mathit{free}); \\
\\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \qquad\qquad\qquad\qquad\qquad\qquad\quad P(\mathit{busy}); \\
\{10 \mapsto -\} \\
[10] := m; \qquad\qquad \| \qquad\qquad n := [10]; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \qquad\qquad\qquad\qquad\qquad\qquad\quad V(\mathit{free}); \\
\{\texttt{emp}\} \\
\vdots \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \\
\{10 \mapsto -\} \\
[10] := m; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
\quad \| \quad
\begin{array}{l}
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{busy}); \\
\\
n := [10]; \\
\\
V(\mathit{free}); \\
\\
\vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \\
\{10 \mapsto -\} \\
[10] := m; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
\quad \| \quad
\begin{array}{l}
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{busy}); \\
\{10 \mapsto -\} \\
n := [10]; \\
\\
V(\mathit{free}); \\
\\
\vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \\
\{10 \mapsto -\} \\
[10] := m; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
\quad \| \quad
\begin{array}{l}
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{busy}); \\
\{10 \mapsto -\} \\
n := [10]; \\
\{10 \mapsto -\} \\
V(\mathit{free}); \\
\\
\vdots
\end{array}
$$

Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

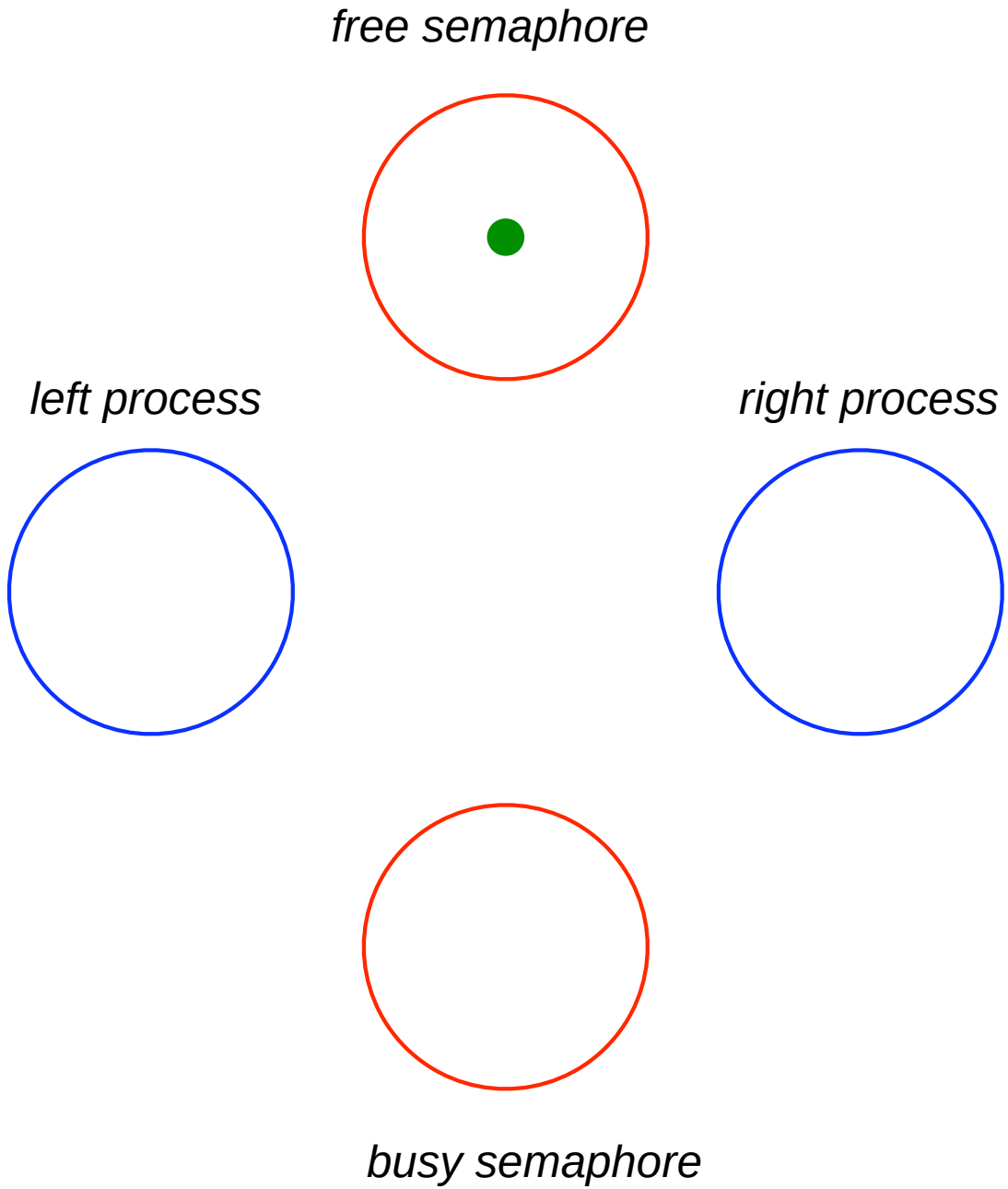
$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \\
\{10 \mapsto -\} \\
[10] := m; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
\quad \| \quad
\begin{array}{l}
\\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{busy}); \\
\{10 \mapsto -\} \\
n := [10]; \\
\{10 \mapsto -\} \\
V(\mathit{free}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
$$

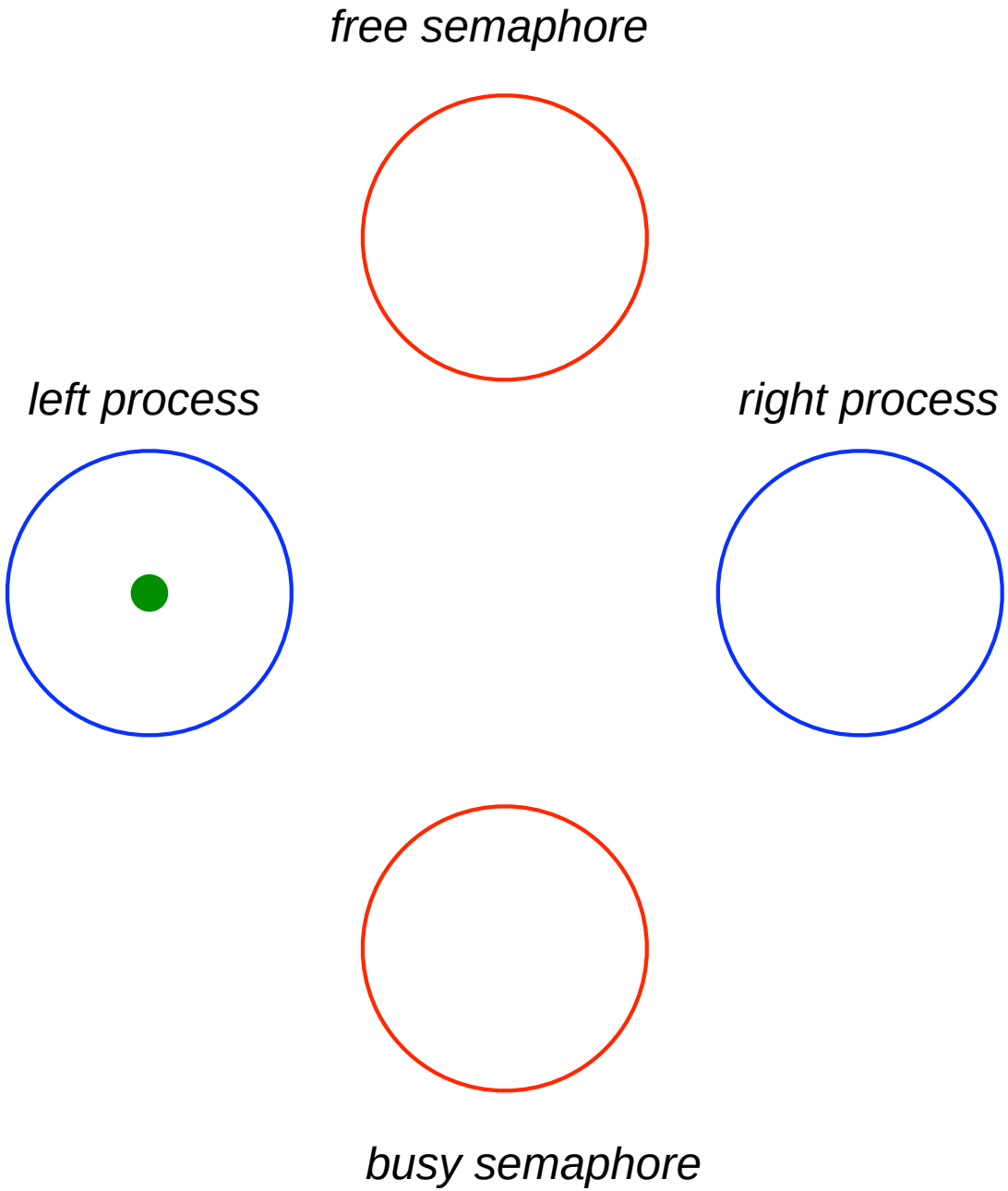
Recall: $V(s)$ bumps s up by 1; $P(s)$ decrements by 1 if > 0, waiting otherwise

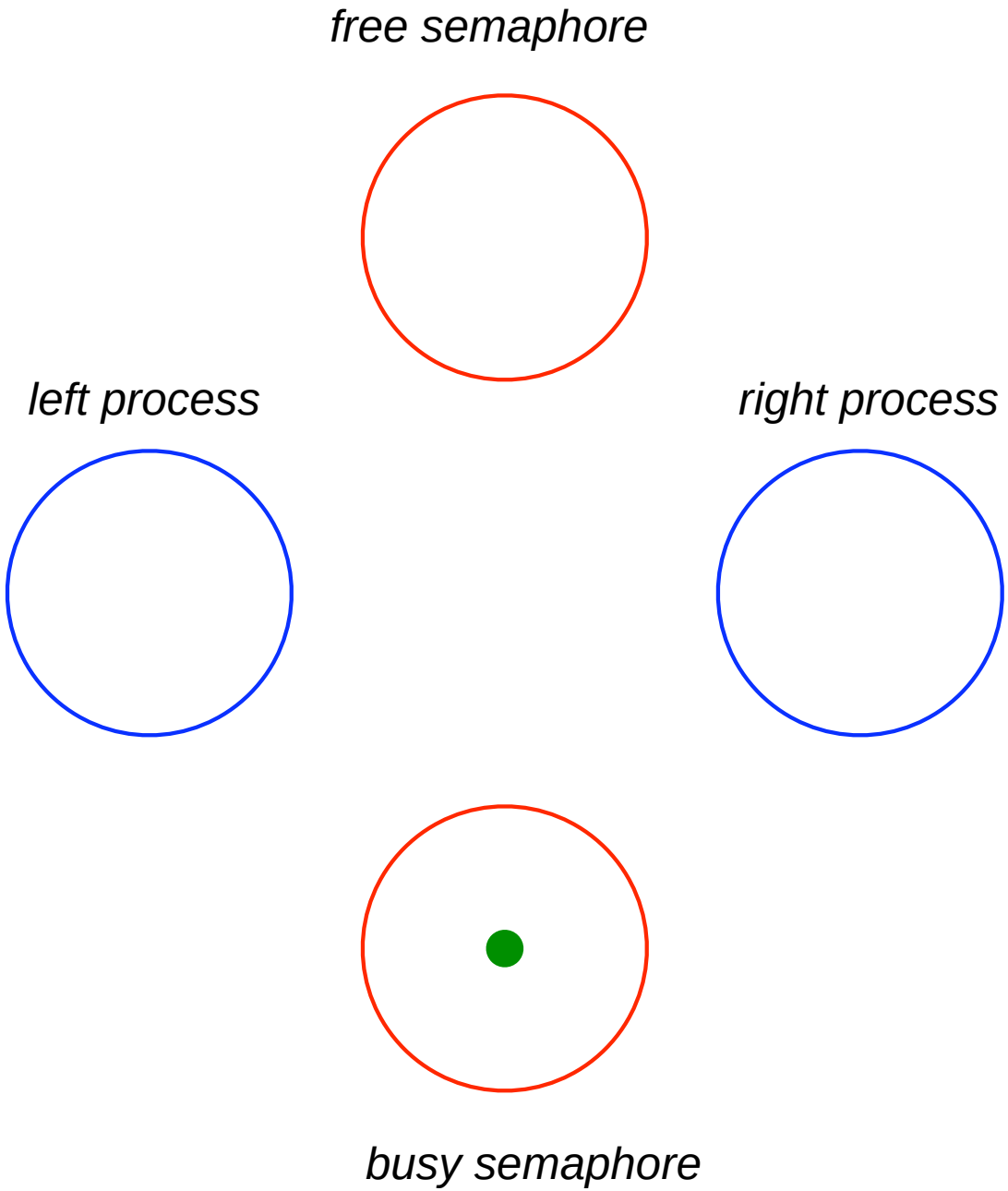
$$
\begin{array}{l}
\texttt{semaphore } \mathit{free} := 1;\ \mathit{busy} := 0 \\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{free}); \\
\{10 \mapsto -\} \\
[10] := m; \\
\{10 \mapsto -\} \\
V(\mathit{busy}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
\quad \| \quad
\begin{array}{l}
\\
\vdots \\
\{\texttt{emp}\} \\
P(\mathit{busy}); \\
\{10 \mapsto -\} \\
n := [10]; \\
\{10 \mapsto -\} \\
V(\mathit{free}); \\
\{\texttt{emp}\} \\
\vdots
\end{array}
$$

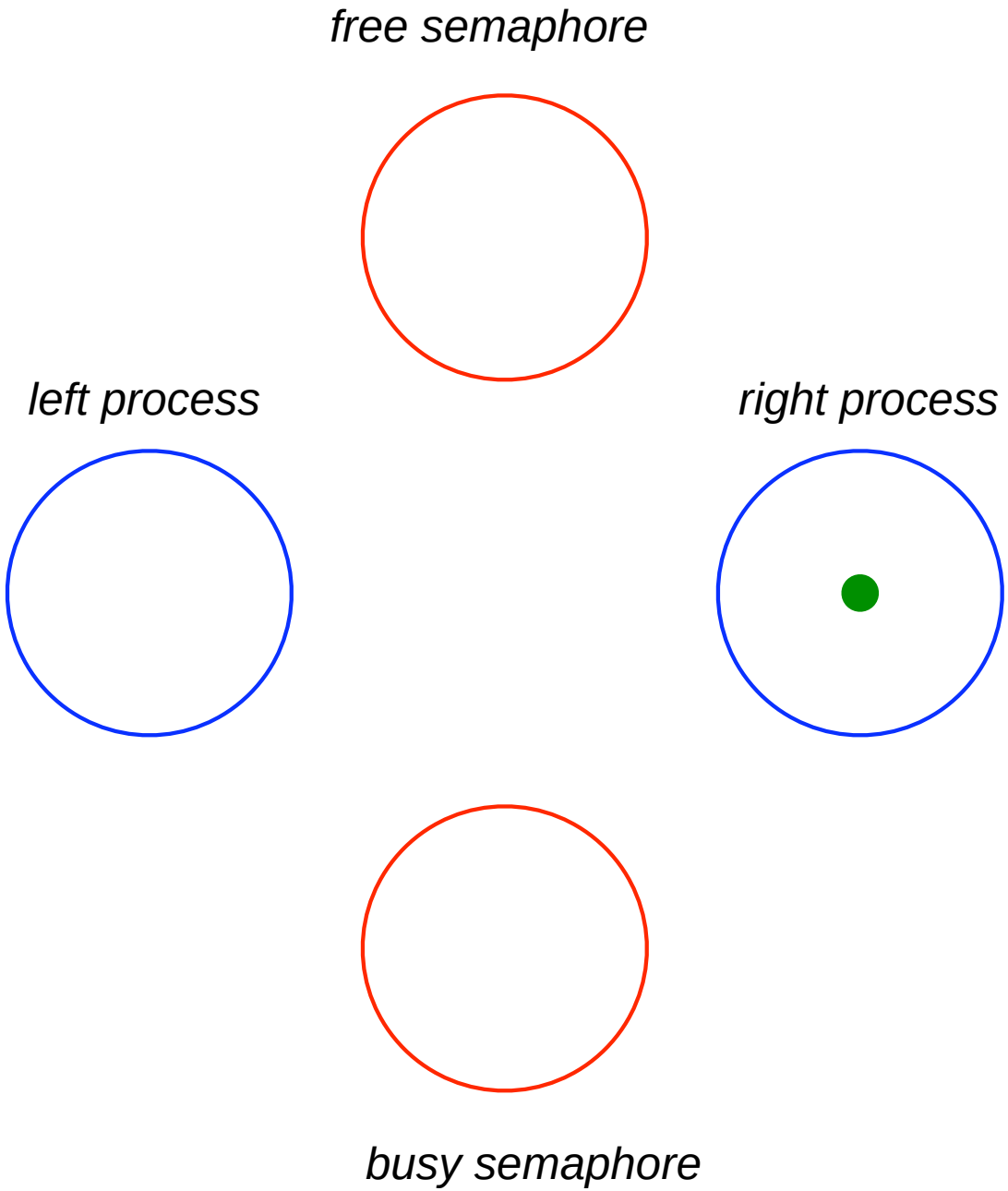
$$RI_{free} \quad \equiv \quad (\mathit{free} = 0 \wedge \texttt{emp}) \vee (\mathit{free} = 1 \wedge 10 \mapsto -)$$

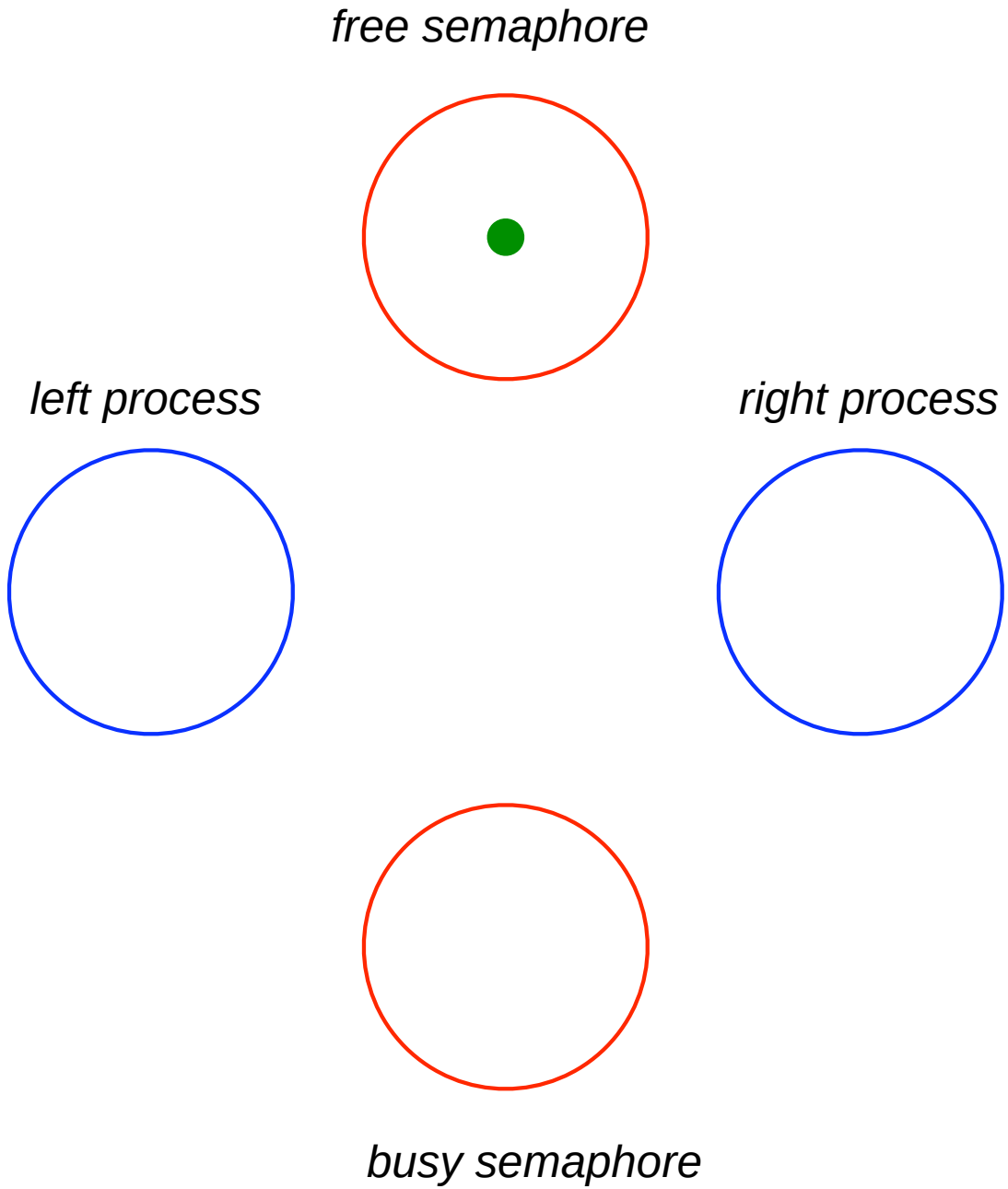
$$RI_{busy} \quad \equiv \quad (\mathit{busy} = 0 \wedge \texttt{emp}) \vee (\mathit{busy} = 1 \wedge 10 \mapsto -)$$


free semaphore
left process
right process
busy semaphore


free semaphore
left process
right process
busy semaphore


free semaphore
left process
right process
busy semaphore


free semaphore
left process
right process
busy semaphore


free semaphore
left process
right process
busy semaphore

$\{10 \mapsto -\}$

$\texttt{semaphore}\ \mathit{free} := 1,\ \mathit{busy} := 0;$

$\{RI_{free} * RI_{busy} * \texttt{emp} * \texttt{emp}\}$

$\{\texttt{emp}\}$		$\{\texttt{emp}\}$
$\texttt{while true do}$		$\texttt{while true do}$
$\mathit{produce}\ m$		$P(\mathit{busy});$
$P(\mathit{free});$		$n := [10];$
$[10] := m;$	$\|$	$V(\mathit{free});$
$V(\mathit{busy});$		$\mathit{consume}\ n$
$\{\texttt{false}\}$		$\{\texttt{false}\}$

$\{\texttt{false}\}$

$$RI_{free} \equiv (\mathit{free} = 0 \wedge \texttt{emp}) \vee (\mathit{free} = 1 \wedge 10 \mapsto -)$$

$$RI_{busy} \equiv (\mathit{busy} = 0 \wedge \texttt{emp}) \vee (\mathit{busy} = 1 \wedge 10 \mapsto -)$$

$\{10 \mapsto -\}$

$\texttt{semaphore}\ \mathit{free} := 1,\ \mathit{busy} := 0;$

$\{RI_{free} * RI_{busy} * \texttt{emp} * \texttt{emp}\}$

$\{\texttt{emp}\}$		$\{\texttt{emp}\}$
$\texttt{while true do}$		$\texttt{while true do}$
$\{\texttt{emp} \wedge \texttt{true}\}$		$\{\texttt{emp} \wedge \texttt{true}\}$
$\mathit{produce}\ m$		$P(\mathit{busy});$
$\{\texttt{emp}\}$		$\{\texttt{emp}\}$
$P(\mathit{free});$		$n := [10];$
$\{10 \mapsto -\}$		$\{10 \mapsto -\}$
$[10] := m;$	$\|$	$V(\mathit{free});$
$\{10 \mapsto -\}$		$\{10 \mapsto -\}$
$V(\mathit{busy});$		$\mathit{consume}\ n$
$\{\texttt{emp}\}$		$\{\texttt{emp}\}$
$\{\texttt{false}\}$		$\{\texttt{false}\}$

$\{\texttt{false}\}$

$$RI_{free} \equiv (\mathit{free} = 0 \wedge \texttt{emp}) \vee (\mathit{free} = 1 \wedge 10 \mapsto -)$$

$$RI_{busy} \equiv (\mathit{busy} = 0 \wedge \texttt{emp}) \vee (\mathit{busy} = 1 \wedge 10 \mapsto -)$$

Remarks

- Each semaphore invariant talks only about itself, not about the other semaphore or the processes.

Remarks

- Each semaphore invariant talks only about itself, not about the other semaphore or the processes.
- Each assertion within a process talks about only its own state, not the state of the other process or even the semaphores.

Remarks

- Each semaphore invariant talks only about itself, not about the other semaphore or the processes.
- Each assertion within a process talks about only its own state, not the state of the other process or even the semaphores.
- We don't maintain $0 \leq \mathit{free} + \mathit{busy} \leq 1$ as a global invariant.

Remarks

- Each semaphore invariant talks only about itself, not about the other semaphore or the processes.
- Each assertion within a process talks about only its own state, not the state of the other process or even the semaphores.
- We don't maintain $0 \leq \mathit{free} + \mathit{busy} \leq 1$ as a global invariant.
- Semaphores are "logically attached" to resource. P and V are ownership transformers.

Dynamic Partitioning Idioms

-
  - Memory Managers, Thread Pools, Connection Pools
  - Efficient Message Passing (copy avoiding)
  - Double-buffered I/O
  - Essentially all semaphore programs

Dynamic Partitioning Idioms

-
  - Memory Managers, Thread Pools, Connection Pools
  - Efficient Message Passing (copy avoiding)
  - Double-buffered I/O
  - Essentially all semaphore programs
- These idioms underlie much *fundamental code*: Microkernel OS designs; web servers; network packet processing...

Dynamic Partitioning Idioms

-
  - Memory Managers, Thread Pools, Connection Pools
  - Efficient Message Passing (copy avoiding)
  - Double-buffered I/O
  - Essentially all semaphore programs
- These idioms underlie much *fundamental code*: Microkernel OS designs; web servers; network packet processing...
- Old program design ideas. Reflected in concurrent separation logic

Dynamic Partitioning Idioms

-
  - Memory Managers, Thread Pools, Connection Pools
  - Efficient Message Passing (copy avoiding)
  - Double-buffered I/O
  - Essentially all semaphore programs
- These idioms underlie much *fundamental code*: Microkernel OS designs; web servers; network packet processing...
- Old program design ideas. Reflected in concurrent separation logic
- **Wait a minute, not so fast!**

The Reynolds Counterexample

$\texttt{resource}\ r()\qquad \textit{invariant}\ RI_r = \texttt{true}$

From

$$\dfrac{\dfrac{\{\texttt{true}\}\texttt{skip}\{\texttt{true}\}}{\{(\texttt{emp} \vee \texttt{one}) * \texttt{true}\}\texttt{skip}\{\texttt{emp} * \texttt{true}\}}}{\{\texttt{emp} \vee \texttt{one}\}\texttt{with}\ r\ \texttt{when}\ \texttt{true}\ \textbf{do}\ \texttt{skip}\ \{\texttt{emp}\}}$$

We obtain an inconsistency

$$\dfrac{\dfrac{\dfrac{\dfrac{\{\texttt{emp} \vee \texttt{one}\}\texttt{with}\cdots\ \{\texttt{emp}\}}{\{\texttt{emp}\}\texttt{with}\cdots\ \{\texttt{emp}\}}}{\{\texttt{emp} * \texttt{one}\}\texttt{with}\cdots\ \{\texttt{emp} * \texttt{one}\}}}{\{\texttt{one}\}\texttt{with}\cdots\{\texttt{one}\}} \qquad \dfrac{\{\texttt{emp} \vee \texttt{one}\}\texttt{with}\ \cdots\ \{\texttt{emp}\}}{\{\texttt{one}\}\texttt{with}\ \cdots\ \{\texttt{emp}\}}}{\{\texttt{one} \wedge \texttt{one}\}\texttt{with}\ r\ \texttt{when}\ \texttt{true}\ \textbf{do}\ \texttt{skip}\ \{\texttt{emp} \wedge \texttt{one}\}}}{\{\texttt{one}\}\texttt{with}\ r\ \texttt{when}\ \texttt{true}\ \textbf{do}\ \texttt{skip}\ \{\texttt{false}\}}$$

What the Reynolds Counterexample Implies

Trouble, if you have all of

$$\frac{\{P_1\}C\{Q_1\} \quad \{P_2\}C\{Q_2\}}{\{P_1 \wedge P_2\}C\{Q_1 \wedge Q_2\}}$$

$$\frac{\{(P * RI_r) \wedge B\}\ C\ \{Q * RI_r\}}{\{P\}\ \texttt{with}\ r\ \texttt{when}\ B\ \texttt{do}\ C\ \texttt{endwith}\ \{Q\}}$$

$$\frac{\{P\}C\{Q\}}{\{P * R\}C\{Q * R\}}$$

Diagnosis

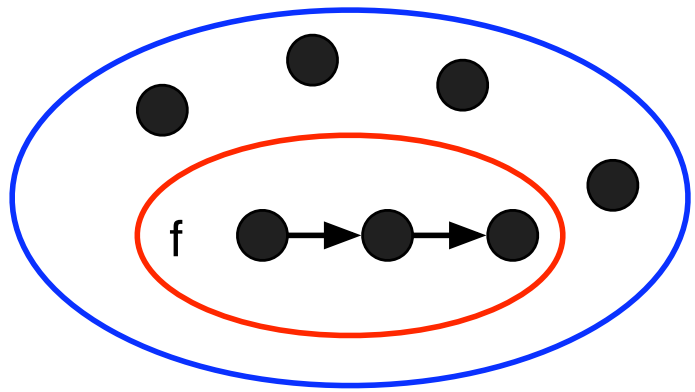
- The semantics of $*$ is nondeterministic

  $$s, h \models P*Q \;\Leftrightarrow\; \exists h_P, h_Q.\, h = h_P \cdot h_Q \wedge s, h_P \models P \wedge s, h_Q \models Q$$

- The resource invariant `true` does not precisely nail down the storage owned by the resource; it is ambiguous. $*$ can be satisfied with different splittings
- If we can nail down the storage owned more precisely, perhaps we can get around this problem.

Precise Predicates

- A predicate is **precise** if, for every state, there is at most one substate that satisfies it.



- For all s, h, exists at most one $h' \sqsubseteq h$ where $s, h' \models P$.

Precise Predicates, II

- Examples of Imprecise Predicates
  1. $\texttt{true}$
  2. $10 \mapsto - \ \vee \ 11 \mapsto -$
  3. $ls(x, y) \iff (x = y \wedge \texttt{emp}) \vee \exists x'.x \mapsto x' * ls(x', y)$

Precise Predicates, II

- Examples of Imprecise Predicates
  1. $\texttt{true}$
  2. $10 \mapsto - \ \vee \ 11 \mapsto -$
  3. $ls(x, y) \iff (x = y \wedge \texttt{emp}) \vee \exists x'.x \mapsto x' * ls(x', y)$
- Examples of Precise Predicates
  1. $\texttt{emp}$
  2. $10 \mapsto -$
  3. $(\texttt{emp} \wedge \neg full) \vee (c \mapsto - \wedge full)$
  4. $ls(x, y) \iff \texttt{if } (x = y) \texttt{ then emp else } \exists x'.x \mapsto x' * ls(x', y)$

Precise Predicates, II

- Examples of Imprecise Predicates
  1. $\texttt{true}$
  2. $10 \mapsto - \ \vee \ 11 \mapsto -$
  3. $ls(x, y) \iff (x = y \wedge \texttt{emp}) \vee \exists x'.x \mapsto x' * ls(x', y)$
- Examples of Precise Predicates
  1. $\texttt{emp}$
  2. $10 \mapsto -$
  3. $(\texttt{emp} \wedge \neg full) \vee (c \mapsto - \wedge full)$
  4. $ls(x, y) \iff \texttt{if } (x = y) \texttt{ then emp else } \exists x'.x \mapsto x' * ls(x', y)$
- If P_1 and P_2 are precise then so are
  - $P_1 * P_2$
  - $P_1 \wedge Q$
  - $\texttt{if } B \texttt{ then } P_1 \texttt{ else } P_2$

Brookes's Semantic Analysis

- **Soundness Theorem**. The proof rules are sound, as long as resource invariants are *precise*.
- Reynolds's counterexample shows unsoundness for imprecise invariants.
- Brookes has further results about no races.
- His semantics reflects Dijkstra's principle, and the slogan

  Well-specified processes mind their own business.

The Permissions Idea

- Remember

$$\{10 \mapsto -\}$$
$$[10] := 42 \;\|\; [10] := 6$$
$$\{??\}$$

- Unfortunately, also

$$\{10 \mapsto -\}$$
$$x := [10] \;\|\; y := [10]$$
$$\{??\}$$

The Permissions Idea

- Boyland's idea: $\frac{1}{2} + \frac{1}{2} = 1$.
- $E \overset{p}{\mapsto} F \iff (E \overset{p/2}{\mapsto} F) * (E \overset{p/2}{\mapsto} F)$
- You can write/dispose only when $p = 1$. You can read when $0 < p \leq 1$ a rational.

$$\{10 \overset{1}{\mapsto} 4\}$$

$$\begin{array}{ccc} \{10 \overset{0.5}{\mapsto} 4\} & & \{10 \overset{0.5}{\mapsto} 4\} \\ x{:=}[10] & \| & y{:=}[10] \\ \{x = 4 \wedge 10 \overset{0.5}{\mapsto} 4\} & & \{y = 4 \wedge 10 \overset{0.5}{\mapsto} 4\} \end{array}$$

$$\{x = 4 \wedge y - 4 \wedge 10 \overset{1}{\mapsto} 4\}$$

- Bornat: Permissions are a strong fertilizer for novel ideas. Existence permissions, infinitesmal permissions, counting permissions...

Further Work

- Grainless semantics. Reynolds. Brookes.
- True concurrency model. Hayman-Winskel (LICS'06)
- Permissions, shared read access. Bornat et. al. (POPL'05)
- Variables as resources. Bornat et. al. (MFPS'05), Parkinson et.al. (LICS'06)
- Raciness and non-blocking. Parkinson-Bornat
- Smallfoot Assertion Checker. Berdine-Calcagno (and O'Hearn)