Verifying concurrent programs using Rely/Guarantee and Separation Logic

A lecture by John Wickerson,
as part of the
Software Reliability series

6 March 2014

Family Tree

Hoare Logic

Tony Hoare

1969

Separation Logic

Peter O'Hearn

John Reynolds

2001

Concurrent Separation Logic

Peter O'Hearn

2004

Owicki-Gries

Susan Owicki

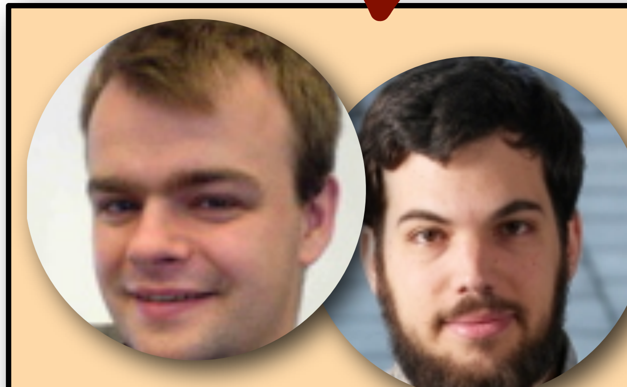
David Gries

1976

Rely/Guarantee

Cliff Jones

1981



RGSep

Matthew Parkinson

Viktor Vafeiadis

2008

Lecture plan

1. The Owicki-Gries method
2. The Rely/Guarantee method
3. Concurrent Separation Logic
4. Towards RGSep

Parallel Rule (Owicki-Gries)

$$\frac{\begin{array}{c}\vdash \{P_1\}\ C_1\ \{Q_1\} \\ \vdash \{P_2\}\ C_2\ \{Q_2\} \\ C_1 \text{ doesn't affect } C_2\text{'s proof} \\ C_2 \text{ doesn't affect } C_1\text{'s proof}\end{array}}{\vdash \{P_1 \wedge P_2\}\ C_1 \parallel C_2\ \{Q_1 \wedge Q_2\}}$$

FindFirstPositive

i := 0; j := 1; x := |A|; y := |A|;

```
while i<min(x,y) do          ||  while j<min(x,y) do
  if A[i]>0 then             ||    if A[j]>0 then
    x:=i                     ||      y:=j
  else                       ||    else
    i:=i+2                   ||      j:=j+2
  end if                     ||    end if
end while                    ||  end while
```

r := min(x,y)

$i := 0;\ j := 1;\ x := |A|;\ y := |A|;$

$\{P_1 \wedge P_2\}$

$\{P_1\}$
while $i<\min(x,y)$ **do** $\{P_1 \wedge i<x \wedge i<|A|\}$
  if $A[i]>0$ **then** $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]>0\}$
    $x:=i$ $\{P_1\}$
  else $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]\leq 0\}$
    $i:=i+2$ $\{P_1\}$
  end if $\{P_1\}$
end while $\{P_1 \wedge i\geq\min(x,y)\}$

$\parallel$

$\{P_2\}$
while $j<\min(x,y)$ **do** $\{P_2 \wedge j<y \wedge j<|A|\}$
  if $A[j]>0$ **then** $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]>0\}$
    $y:=j$ $\{P_2\}$
  else $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]\leq 0\}$
    $j:=j+2$ $\{P_2\}$
  end if $\{P_2\}$
end while $\{P_2 \wedge j\geq\min(x,y)\}$

$\{P_1 \wedge P_2 \wedge i\geq\min(x,y) \wedge j\geq\min(x,y)\}$

$r := \min(x,y)$

$\{r\leq|A| \wedge (\forall k.\ 0\leq k<r \Rightarrow A[k]\leq 0) \wedge (r<|A| \Rightarrow A[r]>0)\}$

where $P_1 \stackrel{\text{def}}{=} x\leq|A| \wedge (\forall k.\ 0\leq k<i \wedge k \text{ even} \Rightarrow A[k]\leq 0) \wedge i \text{ even} \wedge (x<|A| \Rightarrow A[x]>0)$

and $P_2 \stackrel{\text{def}}{=} y\leq|A| \wedge (\forall k.\ 0\leq k<j \wedge k \text{ odd} \Rightarrow A[k]\leq 0) \wedge j \text{ odd} \wedge (y<|A| \Rightarrow A[y]>0)$

Parallel Rule (Owicki-Gries)

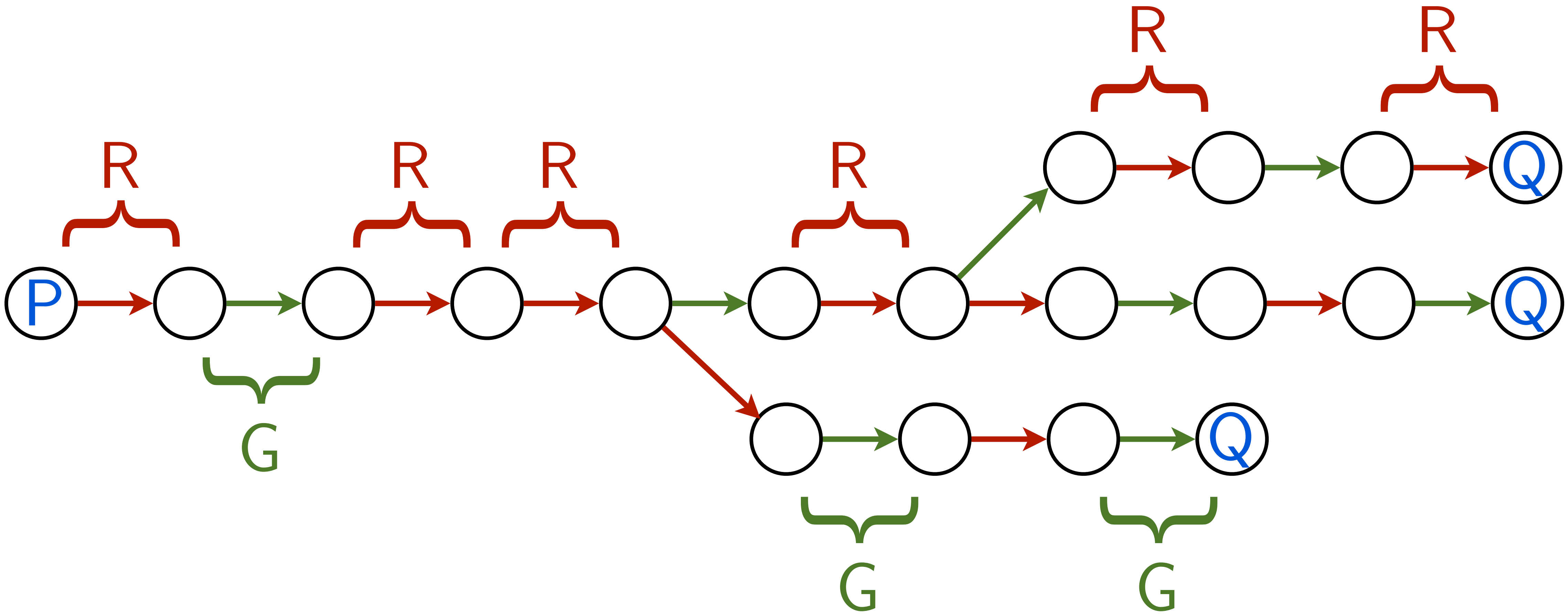
$$\frac{\begin{array}{c}\vdash \{P_1\}\ C_1\ \{Q_1\} \\ \vdash \{P_2\}\ C_2\ \{Q_2\} \\ C_1 \text{ doesn't affect } C_2\text{'s proof} \\ C_2 \text{ doesn't affect } C_1\text{'s proof}\end{array}}{\vdash \{P_1 \wedge P_2\}\ C_1 \parallel C_2\ \{Q_1 \wedge Q_2\}}$$

Lecture plan

1. The Owicki-Gries method
2. The Rely/Guarantee method
3. Concurrent Separation Logic
4. Towards RGSep

Rely/Guarantee

$R, G \vdash \{P\}\ C\ \{Q\}$



Rely/Guarantee

$$R, G \vdash \{P\}\ C\ \{Q\}$$

IF:

(1) the initial state satisfies P, and

(2) every state change by another thread is in R,

THEN:

(1) every final state satisfies Q, and

(2) every state change by C is in G

Parallel Rule (Rely/Guarantee)

$$\frac{R \cup G_2, G_1 \vdash \{P_1\}\ C_1\ \{Q_1\} \qquad R \cup G_1, G_2 \vdash \{P_2\}\ C_2\ \{Q_2\}}{R,\ G_1 \cup G_2 \vdash \{P_1 \land P_2\}\ C_1 \parallel C_2\ \{Q_1 \land Q_2\}}$$

Rule of Consequence

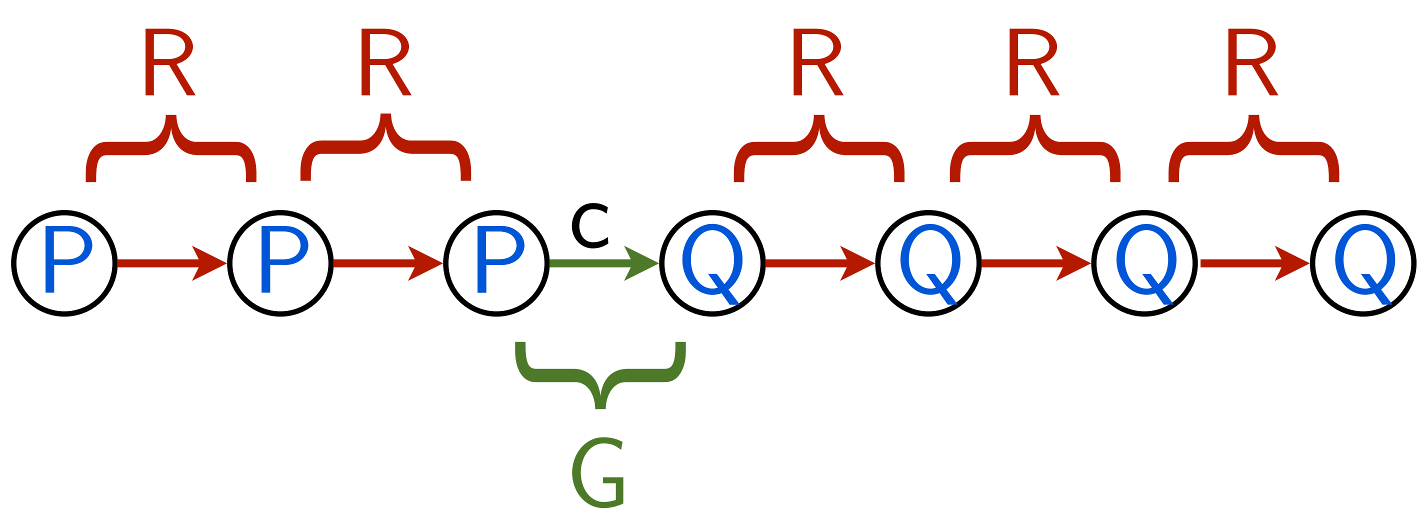
$$\frac{R \subseteq R' \qquad R', G' \vdash \{P\}\ C\ \{Q\} \qquad G' \subseteq G}{R, G \vdash \{P\}\ C\ \{Q\}}$$

Basic commands

$\forall\sigma,\sigma'.\ P(\sigma) \land (\sigma,\sigma')\in[\![c]\!] \Rightarrow G(\sigma,\sigma')$

$\forall\sigma,\sigma'.\ P(\sigma) \land R(\sigma,\sigma') \Rightarrow P(\sigma')$

$$\frac{\begin{array}{c} P \text{ is stable under } R \\ Q \text{ is stable under } R \\ \text{the effect of } c \text{ is contained in } G \\ \vdash \{P\}\ c\ \{Q\} \end{array}}{R,G \vdash \{P\}\ c\ \{Q\}}$$



Making assertions stable

$\{x=2\}$
$x := x+1$
$\{x=3\}$

Making assertions stable

$x=n \rightsquigarrow x=n-1$ $\quad$ $x=n \rightsquigarrow x=n+1$

$$R,G \vdash \{x=2\}$$

$$x := x+1$$

$$\{x=3\}$$

Making assertions stable

$x=n \rightsquigarrow x=n-1$ $x=n \rightsquigarrow x=n+1$

$R, G \vdash \{x \leq 2\}$

$x := x+1$

$\{x \leq 3\}$

Quiz

$R \stackrel{\text{def}}{=} x{=}n \rightsquigarrow x{=}n{+}1$

	Strongest stable weaker assertion	Weakest stable stronger assertion
$x{=}0$		
$x{\neq}0$		

i := 0; j := 1; x := |A|; y := |A|;

$\{P_1 \wedge P_2\}$

$G_2, G_1 \vdash$

$\{P_1\}$

while i<min(x,y) **do** $\{P_1 \wedge i<x \wedge i<|A|\}$

  if A[i]>0 **then** $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]>0\}$

    x:=i $\{P_1\}$

  else $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]\leq 0\}$

    i:=i+2 $\{P_1\}$

  end if $\{P_1\}$

end while $\{P_1 \wedge i\geq \min(x,y)\}$

$\|$

$G_1, G_2 \vdash$

$\{P_2\}$

while j<min(x,y) **do** $\{P_2 \wedge j<y \wedge j<|A|\}$

  if A[j]>0 **then** $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]>0\}$

    y:=j $\{P_2\}$

  else $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]\leq 0\}$

    j:=j+2 $\{P_2\}$

  end if $\{P_2\}$

end while $\{P_2 \wedge j\geq \min(x,y)\}$

$\{P_1 \wedge P_2 \wedge i\geq \min(x,y) \wedge j\geq \min(x,y)\}$

r := min(x,y)

$\{r\leq |A| \wedge (\forall k.\ 0\leq k<r \Rightarrow A[k]\leq 0) \wedge (r<|A| \Rightarrow A[r]>0)\}$

where $P_1 \stackrel{\text{def}}{=} x\leq |A| \wedge (\forall k.\ 0\leq k<i \wedge k \text{ even} \Rightarrow A[k]\leq 0) \wedge i \text{ even} \wedge (x<|A| \Rightarrow A[x]>0)$

and $P_2 \stackrel{\text{def}}{=} y\leq |A| \wedge (\forall k.\ 0\leq k<j \wedge k \text{ odd} \Rightarrow A[k]\leq 0) \wedge j \text{ odd} \wedge (y<|A| \Rightarrow A[y]>0)$

$$i := 0;\ j := 1;\ x := |A|;\ y := |A|;$$

$$\{P_1 \wedge P_2\}$$

G_2, G_1 $\vdash$

$\{P_1\}$

while $i<\min(x,y)$ **do** $\{P_1 \wedge i<x \wedge i<|A|\}$

  if $A[i]>0$ **then** $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]>0\}$

    $x:=i$ $\{P_1\}$

  else $\{P_1 \wedge i<x \wedge i<|A| \wedge A[i]\leq 0\}$

    $i:=i+2$ $\{P_1\}$

  end if $\{P_1\}$

end while $\{P_1 \wedge i\geq\min(x,y)\}$

$\|$

G_1, G_2 $\vdash$

$\{P_2\}$

while $j<\min(x,y)$ **do** $\{P_2 \wedge j<y \wedge j<|A|\}$

  if $A[j]>0$ **then** $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]>0\}$

    $y:=j$ $\{P_2\}$

  else $\{P_2 \wedge j<y \wedge j<|A| \wedge A[j]\leq 0\}$

    $j:=j+2$ $\{P_2\}$

  end if $\{P_2\}$

end while $\{P_2 \wedge j\geq\min(x,y)\}$

$$\{P_1 \wedge P_2 \wedge i\geq\min(x,y) \wedge j\geq\min(x,y)\}$$

$$r := \min(x,y)$$

$$\{r\leq|A| \wedge (\forall k.\ 0\leq k<r \Rightarrow A[k]\leq 0) \wedge (r<|A| \Rightarrow A[r]>0)\}$$

where $P_1 \stackrel{\text{def}}{=} x\leq|A| \wedge (\forall k.\ 0\leq k<i \wedge k \text{ even} \Rightarrow A[k]\leq 0) \wedge i \text{ even} \wedge (x<|A| \Rightarrow A[x]>0)$

and $P_2 \stackrel{\text{def}}{=} y\leq|A| \wedge (\forall k.\ 0\leq k<j \wedge k \text{ odd} \Rightarrow A[k]\leq 0) \wedge j \text{ odd} \wedge (y<|A| \Rightarrow A[y]>0)$

and $G_1 \stackrel{\text{def}}{=} \{(\sigma,\sigma') \mid \sigma'(y) = \sigma(y) \wedge \sigma'(j) = \sigma(j) \wedge \sigma'(x) \leq \sigma(x)\}$

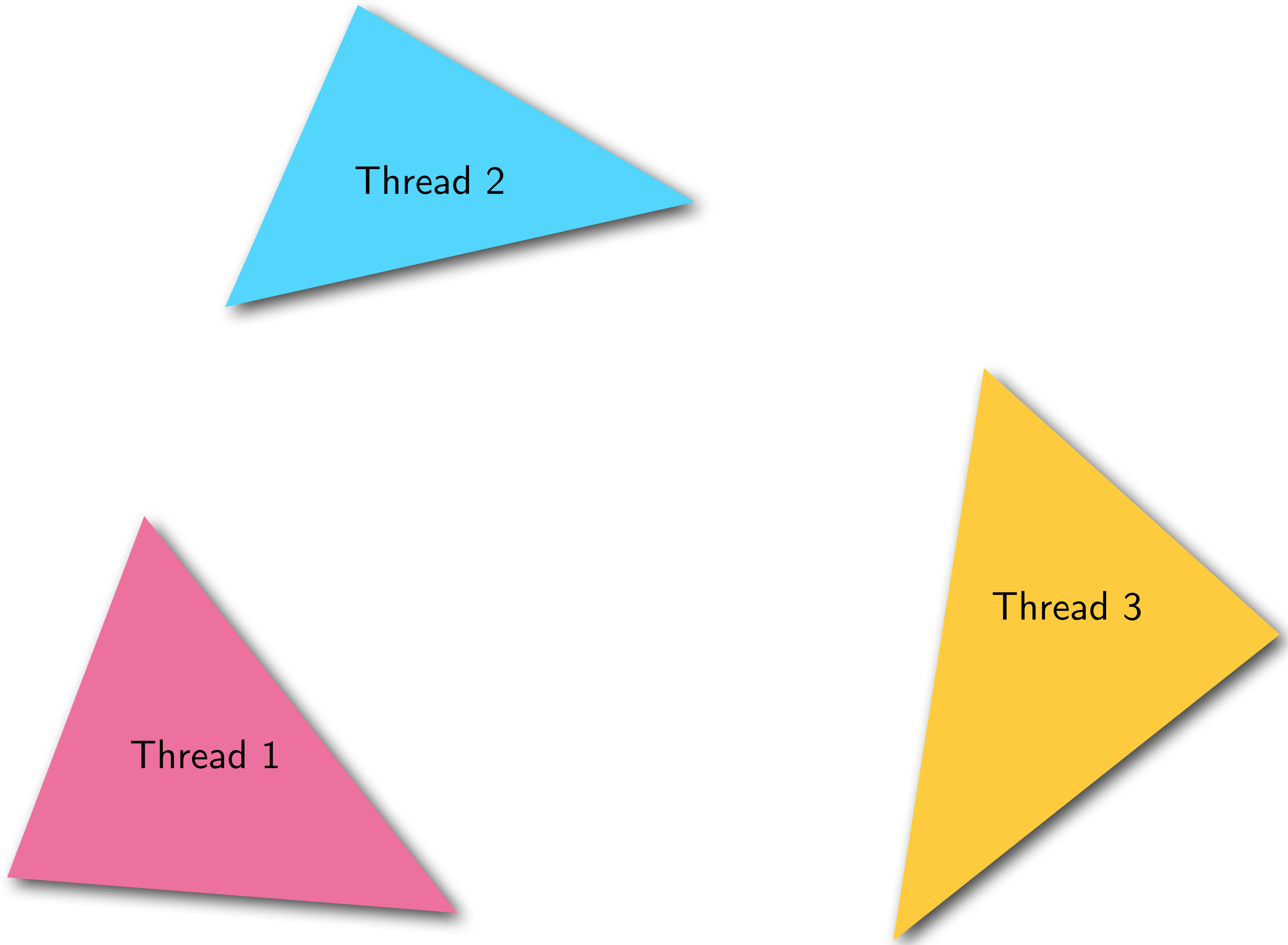
and $G_2 \stackrel{\text{def}}{=} \{(\sigma,\sigma') \mid \sigma'(x) = \sigma(x) \wedge \sigma'(i) = \sigma(i) \wedge \sigma'(y) \leq \sigma(y)\}$

Lecture plan

1. The Owicki-Gries method
2. The Rely/Guarantee method
3. Concurrent Separation Logic
4. Towards RGSep

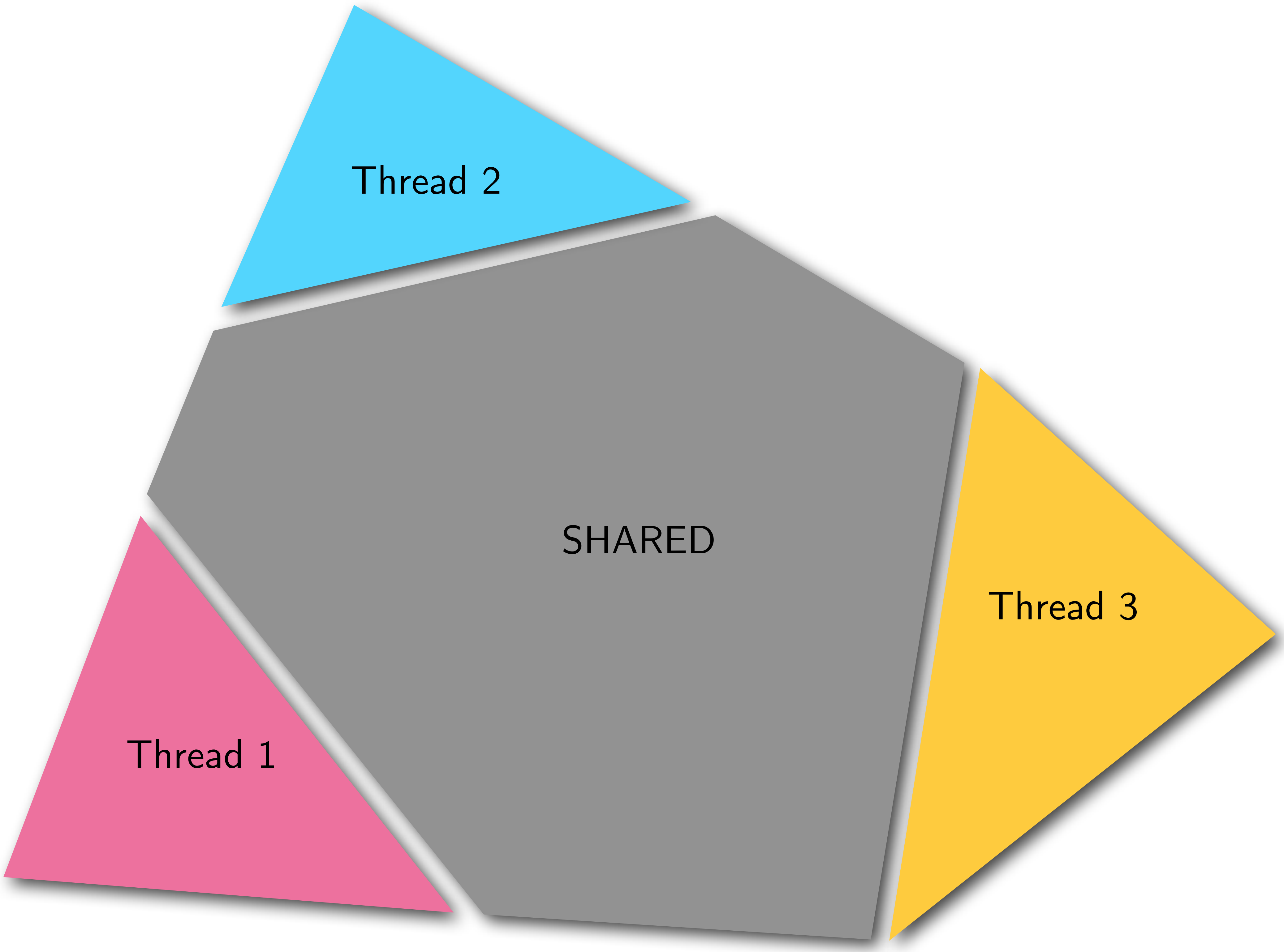
Parallel Rule (Separation Logic)

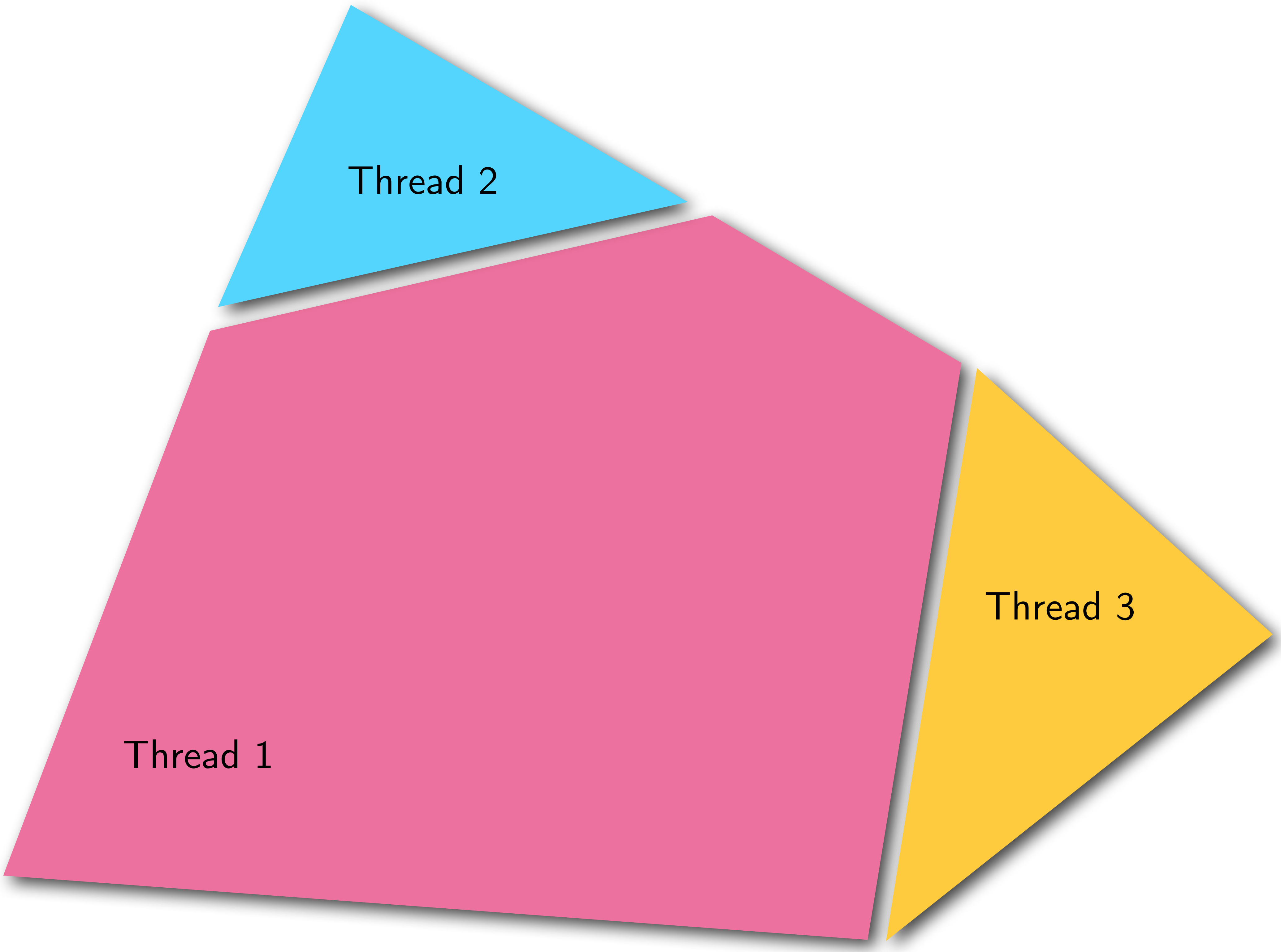
$$\frac{\vdash \{P_1\}\ C_1\ \{Q_1\} \qquad \vdash \{P_2\}\ C_2\ \{Q_2\}}{\vdash \{P_1 * P_2\}\ C_1 \parallel C_2\ \{Q_1 * Q_2\}}$$

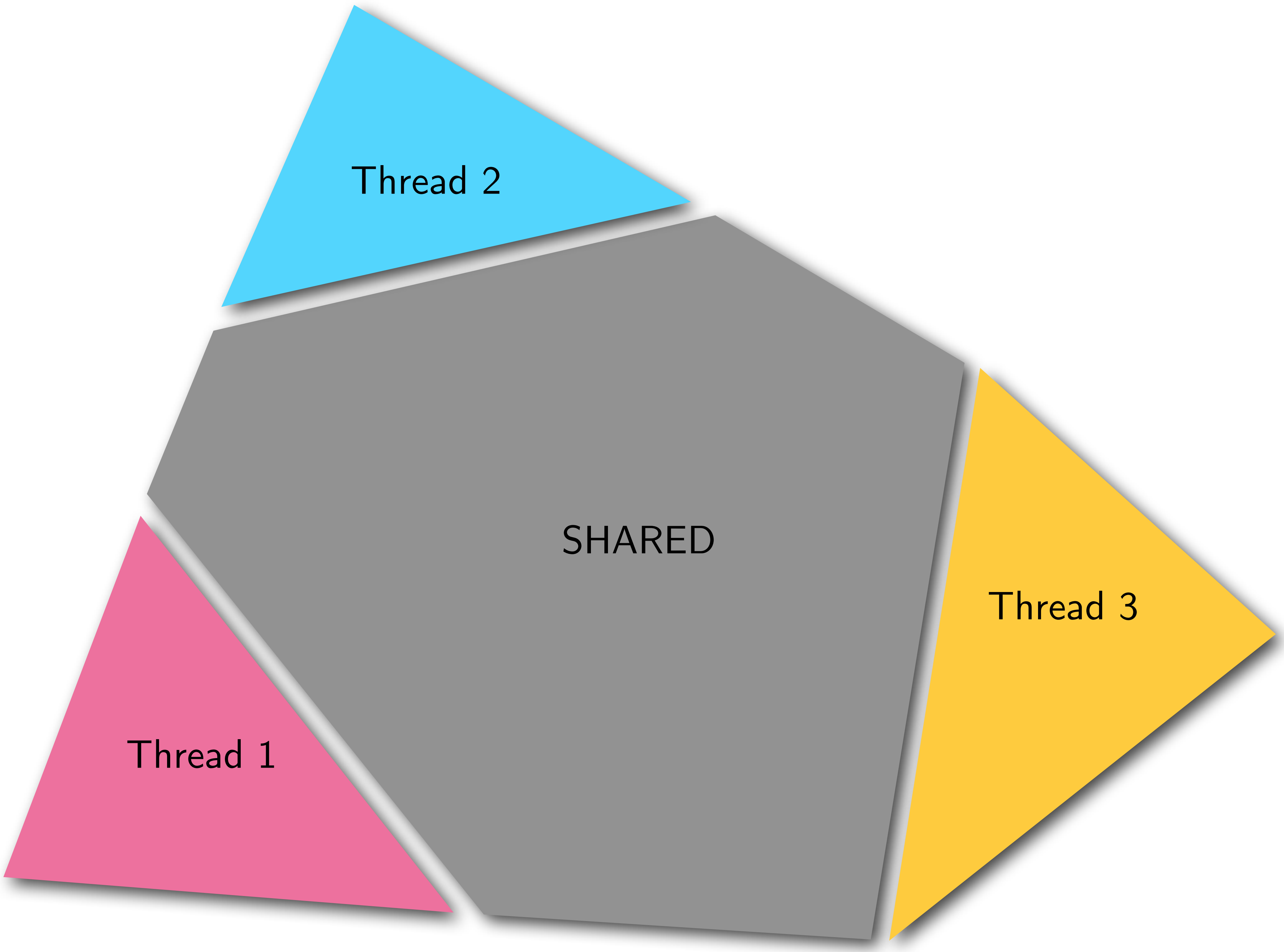

Thread 2
Thread 3
Thread 1

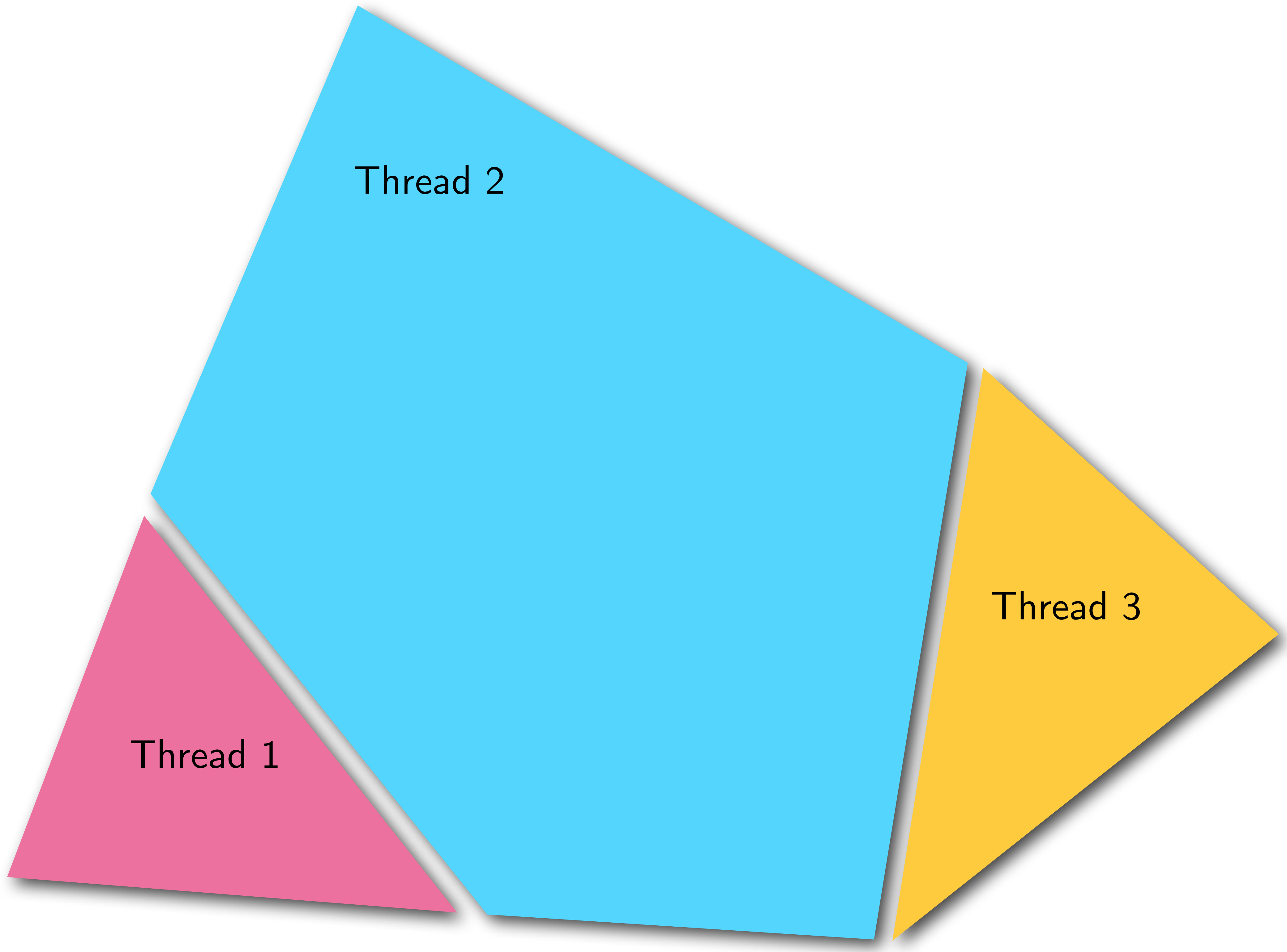
Single-cell buffer

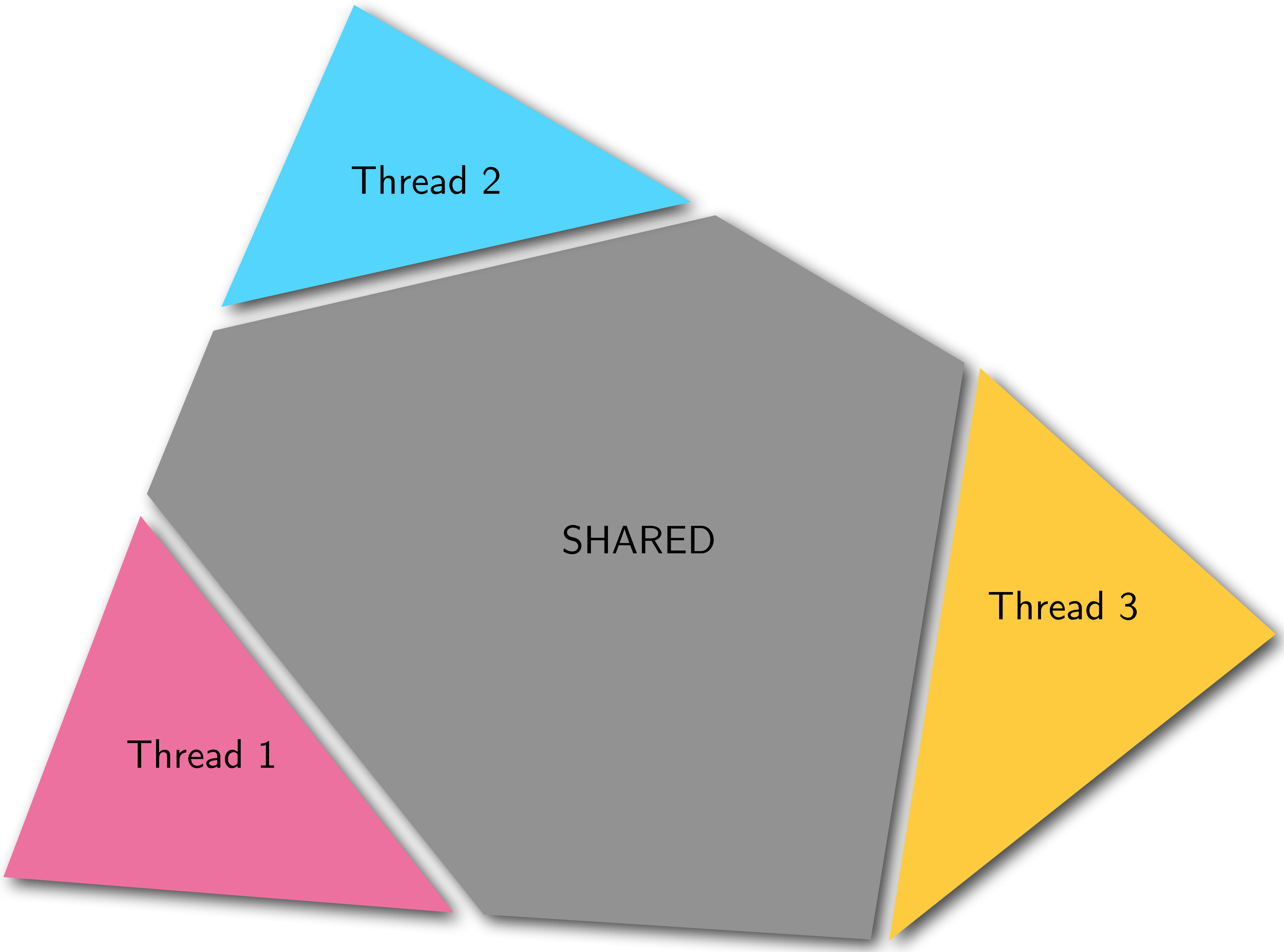
$$
\begin{array}{l||l}
\textbf{while } \text{true } \textbf{do} & \textbf{while } \text{true } \textbf{do} \\
\quad x := \text{cons}(); & \quad \textbf{when } \text{full } \textbf{atomic} \\
\quad \textbf{when } \neg\text{full } \textbf{atomic} & \quad\quad y := c; \\
\quad\quad c := x; & \quad\quad \text{full} := \text{false}; \\
\quad\quad \text{full} := \text{true}; & \quad \textbf{end atomic} \\
\quad \textbf{end atomic} & \quad \text{dispose}(y); \\
\textbf{end while} & \textbf{end while}
\end{array}
$$


Thread 2
SHARED
Thread 3
Thread 1


Thread 2
Thread 3
Thread 1


Thread 2
SHARED
Thread 3
Thread 1


Thread 2
Thread 3
Thread 1


Thread 2
SHARED
Thread 3
Thread 1

Rule for critical regions

$$\frac{\vdash \{(P \land b) * J\}\ C\ \{Q * J\}}{J \vdash \{P\}\ \text{when } b \text{ atomic } C\ \{Q\}}$$

```
{emp}
while true do {emp}
  x := cons(); {x ↦ _}
  when ¬full atomic
    {x ↦ _ ∗ (J ∧ !full)}
    {x ↦ _}
    c := x; {c ↦ _}
    full := true; {c ↦ _ ∧ full}
    {J}
  end atomic {emp}
end while {emp}
```

∥

```
{emp}
while true do {emp}
  when full atomic
    {J ∧ full}
    {c ↦ _}
    y := c; {y ↦ _}
    full := false; {y ↦ _ ∧ ¬full}
    {y ↦ _ ∗ J}
  end atomic {y ↦ _}
  dispose(y); {emp}
end while {emp}
```

where $J = (c \mapsto \_ \wedge \mathrm{full}) \vee (\mathrm{emp} \wedge \neg\mathrm{full})$

Lecture plan

1. The Owicki-Gries method
2. The Rely/Guarantee method
3. Concurrent Separation Logic
4. Towards RGSep

Comparison

Concurrent Separation Logic	**Rely-Guarantee**
$J \vDash \{P\}\ C\ \{Q\}$	$R, G \vDash \{P\}\ C\ \{Q\}$
• initial state satisfies P, and • every state change by another thread preserves J, $\Downarrow$ • C doesn't fault, and • final states satisfy Q, and • every state change by C preserves J	• initial state satisfies P, and • every state change by another thread is in R, $\Downarrow$ • C doesn't fault, and • final states satisfy Q, and • every state change by C is in G

Verify this...

$$\{x \mapsto 0\}$$

$$\textbf{atomic}\ (\ [x] := [x]+1\ ) \;\Big\|\; \textbf{atomic}\ (\ [x] := [x]+2\ )$$

$$\{x \mapsto 3\}$$

Try CSL...

$$
\begin{array}{l|l}
\{\mathsf{emp}\} & \{\mathsf{emp}\} \\
\textbf{atomic}\ (& \textbf{atomic}\ (\\
\quad \{J\} & \quad \{J\} \\
\quad [x] := [x]+1 & \quad [x] := [x]+2 \\
\quad \{J\} & \quad \{J\} \\
) &) \\
\{\mathsf{emp}\} & \{\mathsf{emp}\}
\end{array}
$$

Try CSL...

$\{x \mapsto 0\}$

$$
\begin{array}{l|l}
\{emp\} & \{emp\} \\
\textbf{atomic } (& \textbf{atomic } (\\
\quad \{\exists n \geq 0.\ x \mapsto n\} & \quad \{\exists n \geq 0.\ x \mapsto n\} \\
\quad [x] := [x]+1 & \quad [x] := [x]+2 \\
\quad \{\exists n \geq 0.\ x \mapsto n\} & \quad \{\exists n \geq 0.\ x \mapsto n\} \\
) &) \\
\{emp\} & \{emp\}
\end{array}
$$

$\{\exists n \geq 0.\ x \mapsto n\}$

Try CSL + auxiliary state...

$\{x \mapsto 0\}$

a := 0; b := 0;

$\{x \mapsto a+b * a \doteq 0 * b \doteq 0\}$

$\{a \doteq 0\}$
atomic (
  $\{x \mapsto a+b * a \doteq 0\}$
  [x] := [x]+1; a := 1
  $\{x \mapsto a+b * a \doteq 1\}$
)
$\{a \doteq 1\}$

$\|$

$\{b \doteq 0\}$
atomic (
  $\{x \mapsto a+b * b \doteq 0\}$
  [x] := [x]+2; b := 2
  $\{x \mapsto a+b * b \doteq 2\}$
)
$\{b \doteq 2\}$

$\{x \mapsto a+b * a \doteq 1 * b \doteq 2\}$

Try Rely-Guarantee...

$$\{x \mapsto 0\}$$

$$G_2, G_1 \vdash \{x \mapsto 0\} \; \textbf{atomic} \; (\; [x] := [x]+1 \;) \quad \Big\| \quad G_1, G_2 \vdash \textbf{atomic} \; (\; [x] := [x]+2 \;)$$

where $G_1 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 1) \cup (x \mapsto 2 \rightsquigarrow x \mapsto 3)$

and $G_2 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 2) \cup (x \mapsto 1 \rightsquigarrow x \mapsto 3)$

Try Rely-Guarantee...

$$\{x \mapsto 0\}$$

$$
\begin{array}{l|l}
G_2, G_1 \vdash \{x \mapsto 0 \lor x \mapsto 2\} & G_1, G_2 \vdash \{x \mapsto 0\} \\
\quad \textbf{atomic}\ (& \quad \textbf{atomic}\ (\\
\quad\quad [x] := [x]+1 & \quad\quad [x] := [x]+2 \\
\quad) & \quad) \\
\{x \mapsto 1 \lor x \mapsto 3\} &
\end{array}
$$

where $G_1 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 1) \cup (x \mapsto 2 \rightsquigarrow x \mapsto 3)$

and $G_2 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 2) \cup (x \mapsto 1 \rightsquigarrow x \mapsto 3)$

Try Rely-Guarantee...

$$\{x \mapsto 0\}$$

$$
\begin{array}{l|l}
G_2, G_1 \vdash \{x \mapsto 0 \lor x \mapsto 2\} & G_1, G_2 \vdash \{x \mapsto 0 \lor x \mapsto 1\} \\
\quad \textbf{atomic}\ (& \quad \textbf{atomic}\ (\\
\quad\quad [x] := [x]+1 & \quad\quad [x] := [x]+2 \\
\quad) & \quad) \\
\quad \{x \mapsto 1 \lor x \mapsto 3\} & \quad \{x \mapsto 2 \lor x \mapsto 3\}
\end{array}
$$

$$\{x \mapsto 3\}$$

where $G_1 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 1) \cup (x \mapsto 2 \rightsquigarrow x \mapsto 3)$

and $G_2 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 2) \cup (x \mapsto 1 \rightsquigarrow x \mapsto 3)$

Further reading

- Susan Owicki and David Gries. *An Axiomatic Proof Technique for Parallel Programs.* Acta Informatica, 1976. Available from SpringerLink.

Contains proof of `FindFirstPositive` using Owicki-Gries method.

- Joey Coleman and Cliff Jones. *A structural proof of the soundness of rely/guarantee rules.* Journal of Logic and Computation, 2007. Available from: http://homepages.cs.ncl.ac.uk/j.w.coleman/papers/colemanjones-rg-soundness.pdf

Contains proof of `FindFirstPositive` in RG.

- Viktor Vafeiadis. *Modular fine-grained concurrency verification.* PhD thesis, University of Cambridge, 2007. Available from: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-726.html

Clear and comprehensive introduction to Rely-Guarantee