# Rely-Guarantee

Lecturer: John Wickerson

# Lecture plan

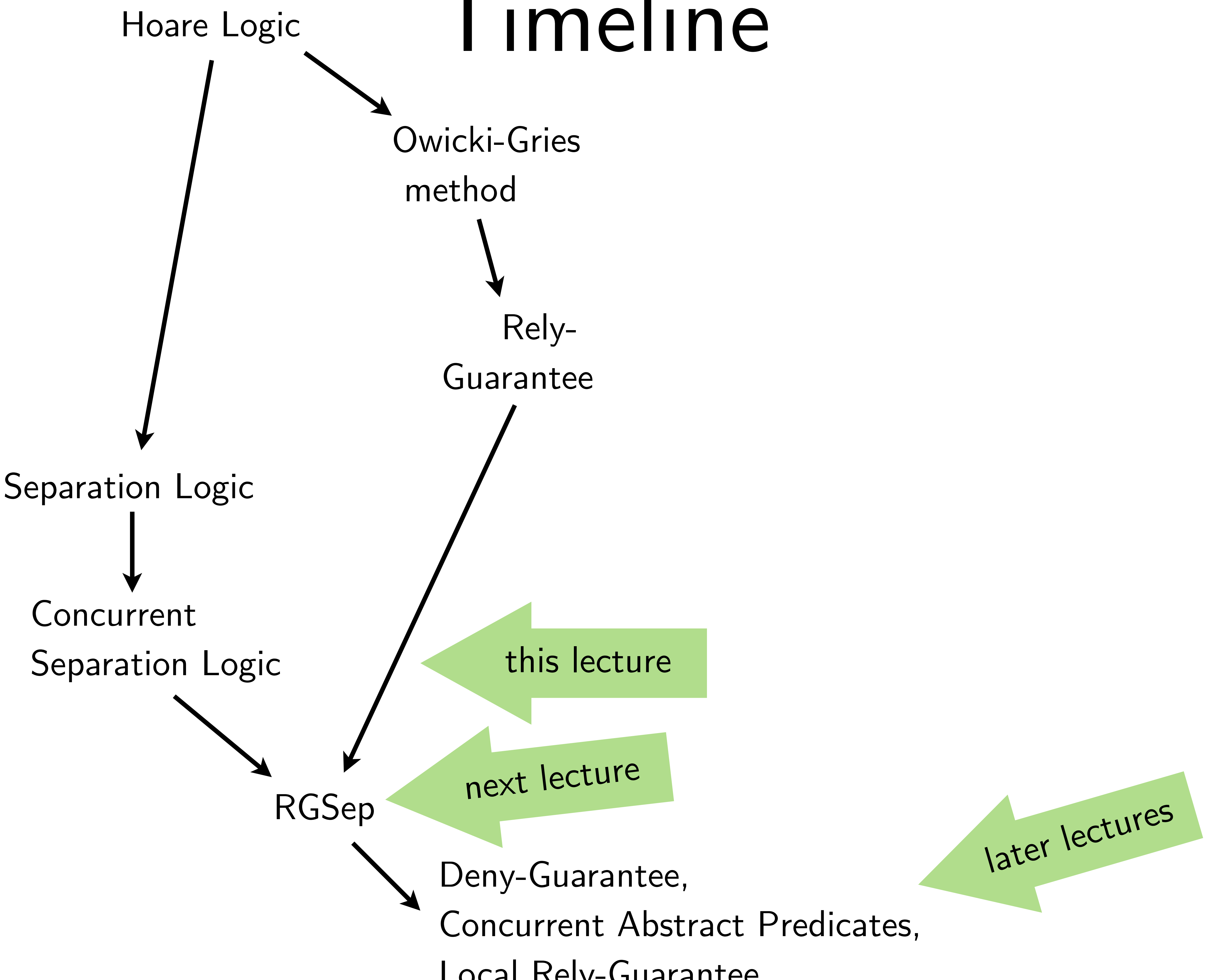
1. Setting the stage

2. Introducing Rely-Guarantee

3. Rely-Guarantee vs. CSL

4. Limitations of Rely-Guarantee

# Timeline


Hoare Logic
Owicki-Gries method
Rely-Guarantee
Separation Logic
Concurrent Separation Logic
this lecture
next lecture
RGSep
later lectures
Deny-Guarantee,
Concurrent Abstract Predicates,
Local Rely-Guarantee...

# Parallel Rule

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

# 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|;
{P1 ∧ P2}
```

```
{P1}
while i<min(x,y) do {P1 ∧ i<x ∧ i<|A|}
  if A[i]>0 then {P1 ∧ i<x ∧ i<|A| ∧ A[i]>0}
    x:=i {P1}
  else {P1 ∧ i<x ∧ i<|A| ∧ A[i]≤0}
    i:=i+2 {P1}
  end if {P1}
end while {P1 ∧ i≥min(x,y)}
```

||

```
{P2}
while j<min(x,y) do {P2 ∧ j<y ∧ j<|A|}
  if A[j]>0 then {P2 ∧ j<y ∧ j<|A| ∧ A[j]>0}
    y:=j {P2}
  else {P2 ∧ j<y ∧ j<|A| ∧ A[j]≤0}
    j:=j+2 {P2}
  end if {P2}
end while {P2 ∧ j≥min(x,y)}
```

```
{P1 ∧ P2 ∧ i≥min(x,y) ∧ j≥min(x,y)}
r := min(x,y)
{r≤|A| ∧ (∀k. 0≤k<r ⇒ A[k]≤0) ∧ (r<|A| ⇒ A[r]>0)}
```

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

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

# Compositionality

$$\frac{\vdash \{P\}\ C_1\ \{Q\} \qquad \vdash \{Q\}\ C_2\ \{R\}}{\vdash \{P\}\ C_1;C_2\ \{R\}}$$

$$\frac{\vdash \{P\}\ C\ \{Q\} \qquad \mathsf{mods}(C) \cap \mathsf{fv}(R) = \emptyset}{\vdash \{P * R\}\ C\ \{Q * R\}}$$

$$\frac{\vdash \{P\}\ C_1\ \{Q\}[X_1] \qquad \vdash \{Q\}\ C_2\ \{R\}[X_2]}{\vdash \{P\}\ C_1;C_2\ \{R\}[X_1 \cup X_2]}$$

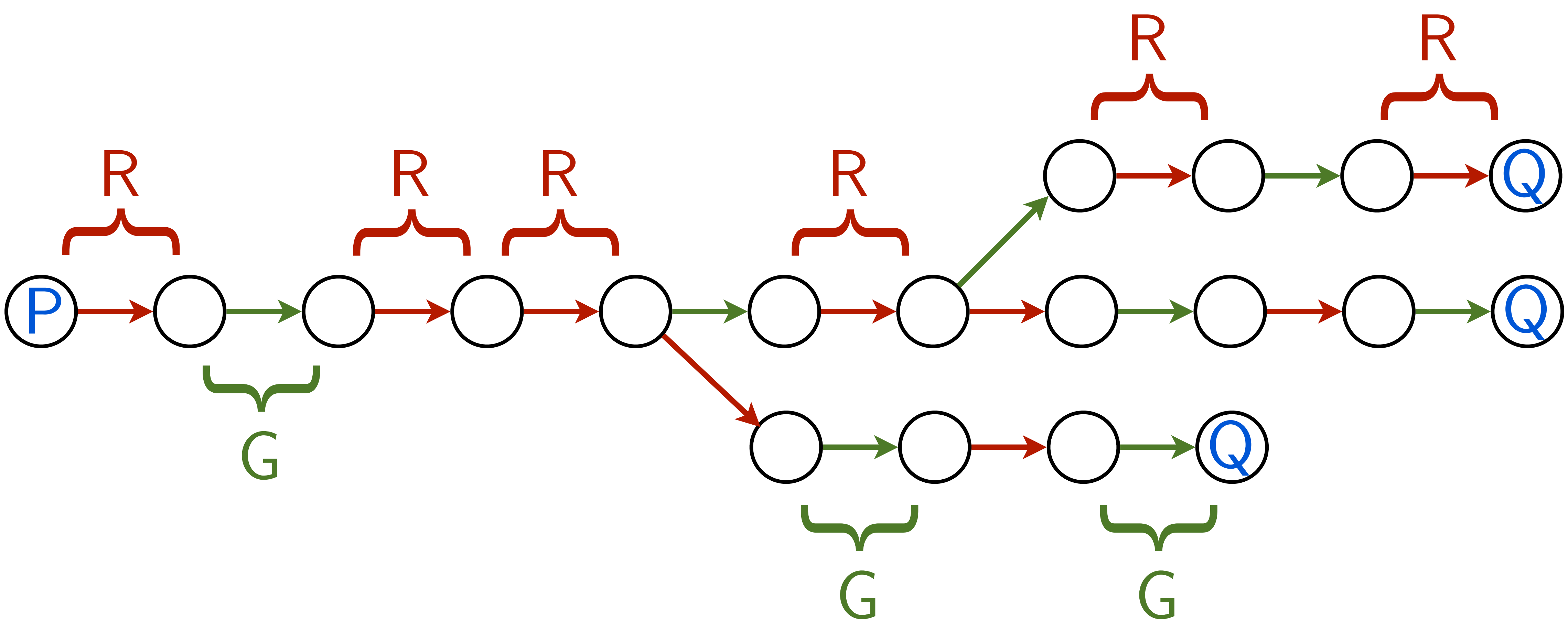
$$\frac{\vdash \{P\}\ C\ \{Q\}[X] \qquad X \cap \mathsf{fv}(R) = \emptyset}{\vdash \{P * R\}\ C\ \{Q * R\}[X]}$$

# Lecture plan

1. Setting the stage
2. Introducing Rely-Guarantee
3. Rely-Guarantee vs. CSL
4. Limitations of Rely-Guarantee

# 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

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

# Rule of Consequence

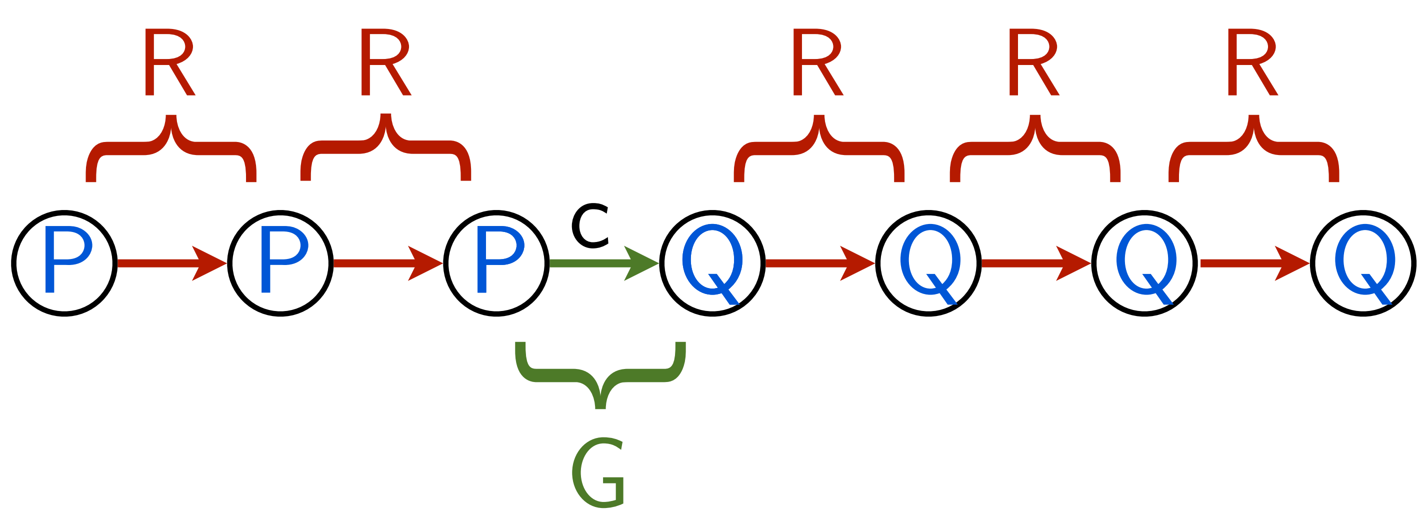
$$\frac{R \subseteq R' \qquad R', G' \vdash \{P\}\ C\ \{Q\} \qquad G' \subseteq G}{R, G \vdash \{P\}\ C\ \{Q\}}$$

# Basic commands

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

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

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



# Making assertions stable

$x{=}n \rightsquigarrow x{=}n\text{-}1$ $\quad$ $x{=}n \rightsquigarrow x{=}n{+}1$

$\{(\sigma,\sigma') \mid \exists n.\ \sigma(x) = n \wedge \sigma'{=}\sigma[x{\mapsto}n\text{-}1]\}$

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

$x := x{+}1$

$\{x{=}3\}$

# Making assertions stable

$x=n \rightsquigarrow x=n\text{-}1$  $x=n \rightsquigarrow x=n+1$

$\{(\sigma,\sigma') \mid \exists n.\ \sigma(x) = n \land \sigma'=\sigma[x\mapsto n\text{-}1]\}$

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

$x := x+1$

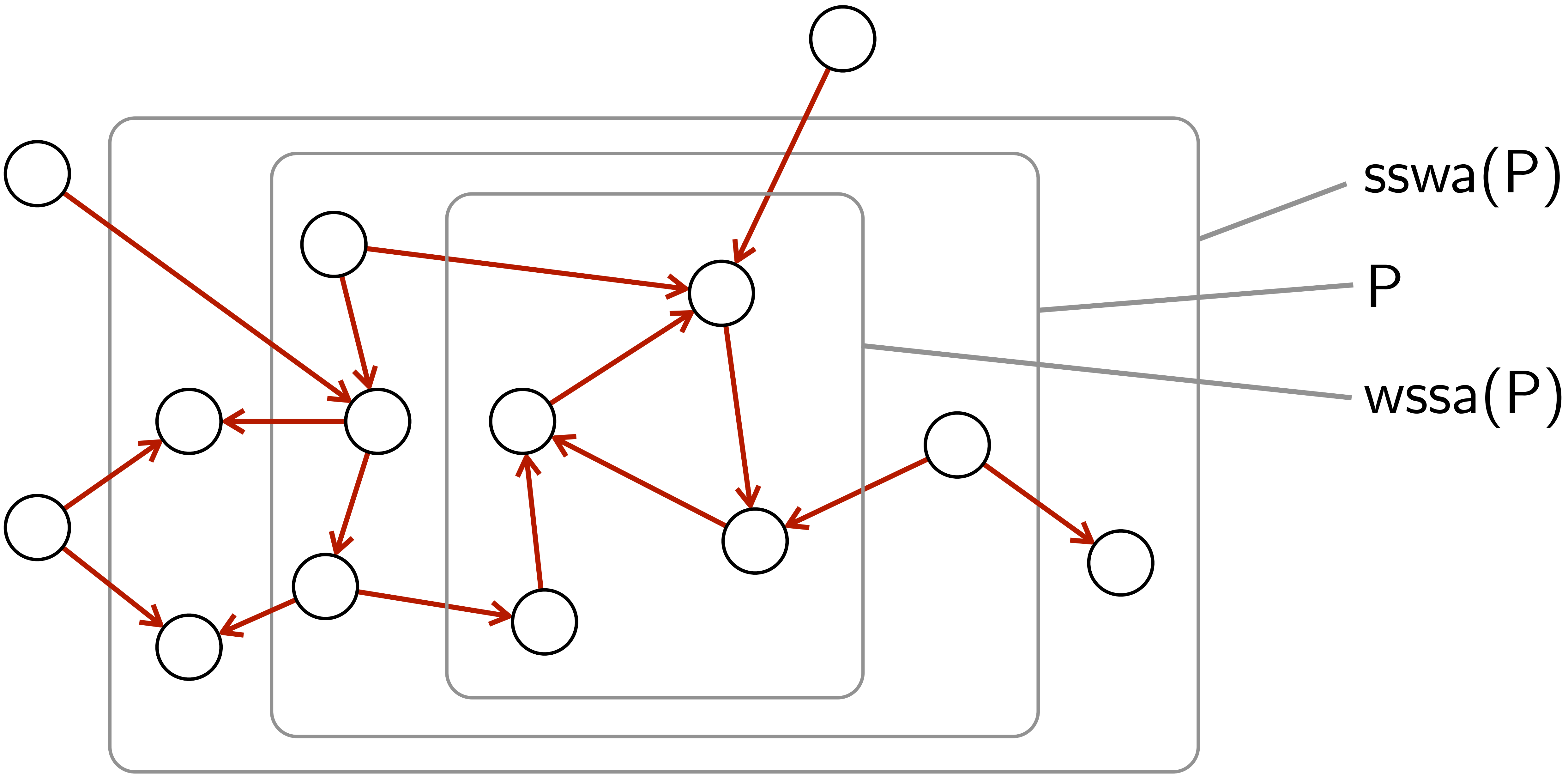
$\{x\leq 3\}$

# Quiz

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

| | Strongest stable weaker assertion | Weakest stable stronger assertion |
|---|---|---|
| x=0 | x≥0 | false |
| x≠0 | true | x>0 |

# Stabilisation



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

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

$G_2$, $G_1$ ⊦

$\{P_1\}$

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

‖

$G_1$, $G_2$ ⊦

$\{P_2\}$

**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. Setting the stage
2. Introducing Rely-Guarantee
3. Rely-Guarantee vs. CSL
4. Limitations of Rely-Guarantee

# Comparison

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

# Verify this...

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

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

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

# Try CSL...

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

# Try CSL...

$\{x \mapsto 0\}$

| | | |
|---|---|---|
| $\{emp\}$ | ‖ | $\{emp\}$ |
| atomic ( | ‖ | atomic ( |
| $\{\exists n \geq 0.\ x \mapsto n\}$ | ‖ | $\{\exists n \geq 0.\ x \mapsto n\}$ |
| [x] := [x]+1 | ‖ | [x] := [x]+2 |
| $\{\exists n \geq 0.\ x \mapsto n\}$ | ‖ | $\{\exists n \geq 0.\ x \mapsto n\}$ |
| ) | ‖ | ) |
| $\{emp\}$ | ‖ | $\{emp\}$ |

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

# Try CSL + auxiliary state...

$\{x \mapsto 0\}$

a := 0; b := 0;

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

| | | |
|---|---|---|
| $\{a \doteq 0\}$ | ‖ | $\{b \doteq 0\}$ |
| atomic ( | ‖ | atomic ( |
| $\{x \mapsto a+b * a \doteq 0\}$ | ‖ | $\{x \mapsto a+b * b \doteq 0\}$ |
| [x] := [x]+1; a := 1 | ‖ | [x] := [x]+2; b := 2 |
| $\{x \mapsto a+b * a \doteq 1\}$ | ‖ | $\{x \mapsto a+b * b \doteq 2\}$ |
| ) | ‖ | ) |
| $\{a \doteq 1\}$ | ‖ | $\{b \doteq 2\}$ |

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

# Try Rely-Guarantee...

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

$$G_2, G_1 \vdash \{x \mapsto 0 \lor x \mapsto 2\}\ \text{atomic}\ (\ [x] := [x]{+}1\ )\ \{x \mapsto 1 \lor x \mapsto 3\} \quad \| \quad G_1, G_2 \vdash \{x \mapsto 0 \lor x \mapsto 1\}\ \text{atomic}\ (\ [x] := [x]{+}2\ )\ \{x \mapsto 2 \lor x \mapsto 3\}$$

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

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

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

# Lecture plan

1. Setting the stage
2. Introducing Rely-Guarantee
3. Rely-Guarantee vs. CSL
4. Limitations of Rely-Guarantee

# Verify this...

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

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

$$\{x \mapsto 2\}$$

# Try Rely-Guarantee...

not stable

$\{x \mapsto 0\}$

$G_2, G_1 \vdash \{x \mapsto 0 \lor x \mapsto 1\}$

atomic (

  [x] := [x]+1

)

$\{x \mapsto 1 \lor x \mapsto 2\}$

$\|$

$G_1, G_2 \vdash \{x \mapsto 0 \lor x \mapsto 1\}$

atomic (

  [x] := [x]+1

)

$\{x \mapsto 1 \lor x \mapsto 2\}$

$\{x \mapsto 1 \lor x \mapsto 2\}$

where $G_1, G_2 \stackrel{\text{def}}{=} (x \mapsto 0 \rightsquigarrow x \mapsto 1) \cup (x \mapsto 1 \rightsquigarrow x \mapsto 2)$

# Try Rely-Guarantee...

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

| | | |
|---|---|---|
| $G_2, G_1 \vdash \{\exists n \geq 0.\ x \mapsto n\}$<br>atomic (<br>[x] := [x]+1<br>)<br>$\{\exists n \geq 1.\ x \mapsto n\}$ | $\|$ | $G_1, G_2 \vdash \{\exists n \geq 0.\ x \mapsto n\}$<br>atomic (<br>[x] := [x]+1<br>)<br>$\{\exists n \geq 1.\ x \mapsto n\}$ |

$$\{\exists n \geq 1.\ x \mapsto n\}$$

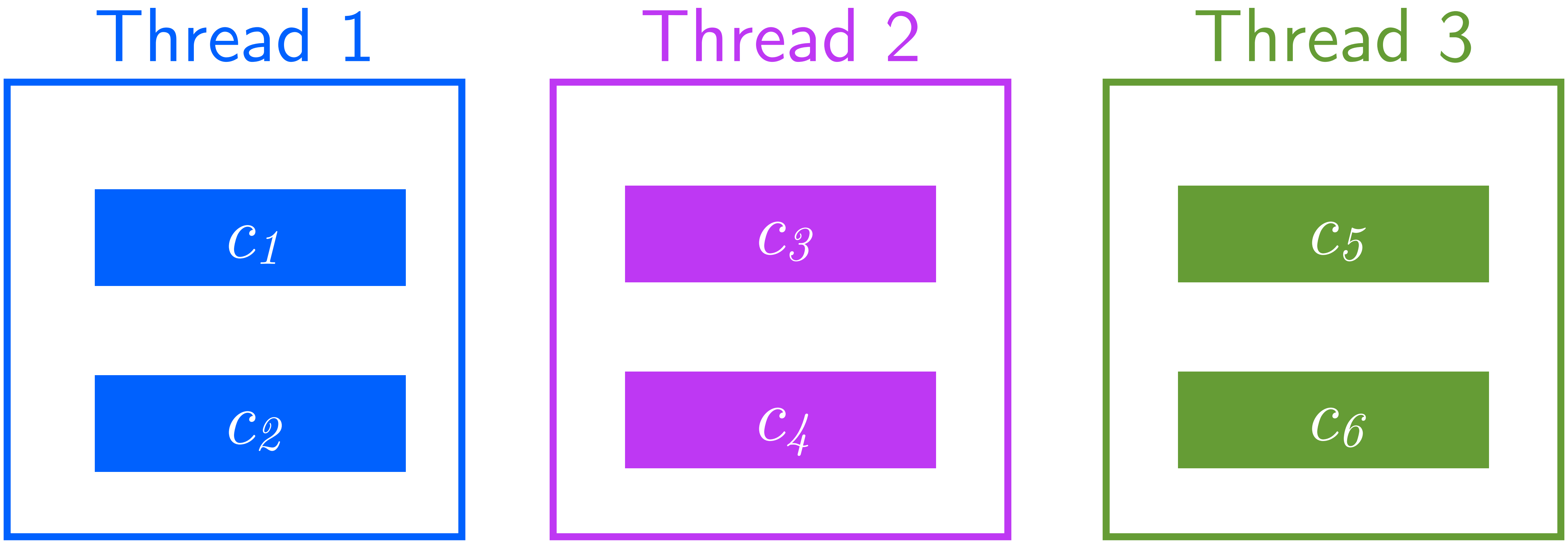
where $G_1, G_2 \stackrel{\text{def}}{=} (x \mapsto n \rightsquigarrow x \mapsto n+1)$

# Abstracting the environment

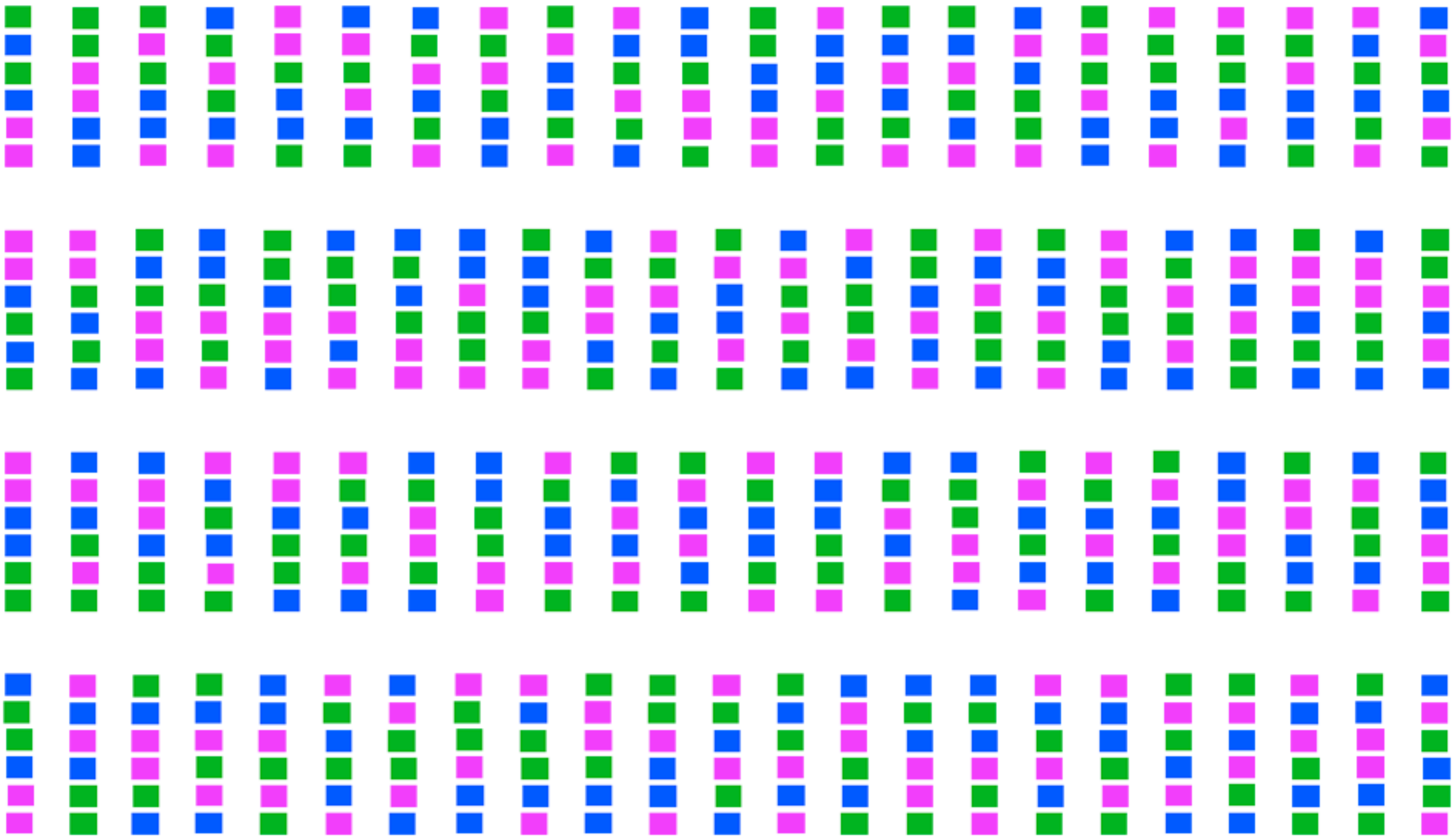
By encoding the environment as a rely, we forget:

- the order in which actions can happen;
- which thread performs which action; and
- how many times each action is performed.

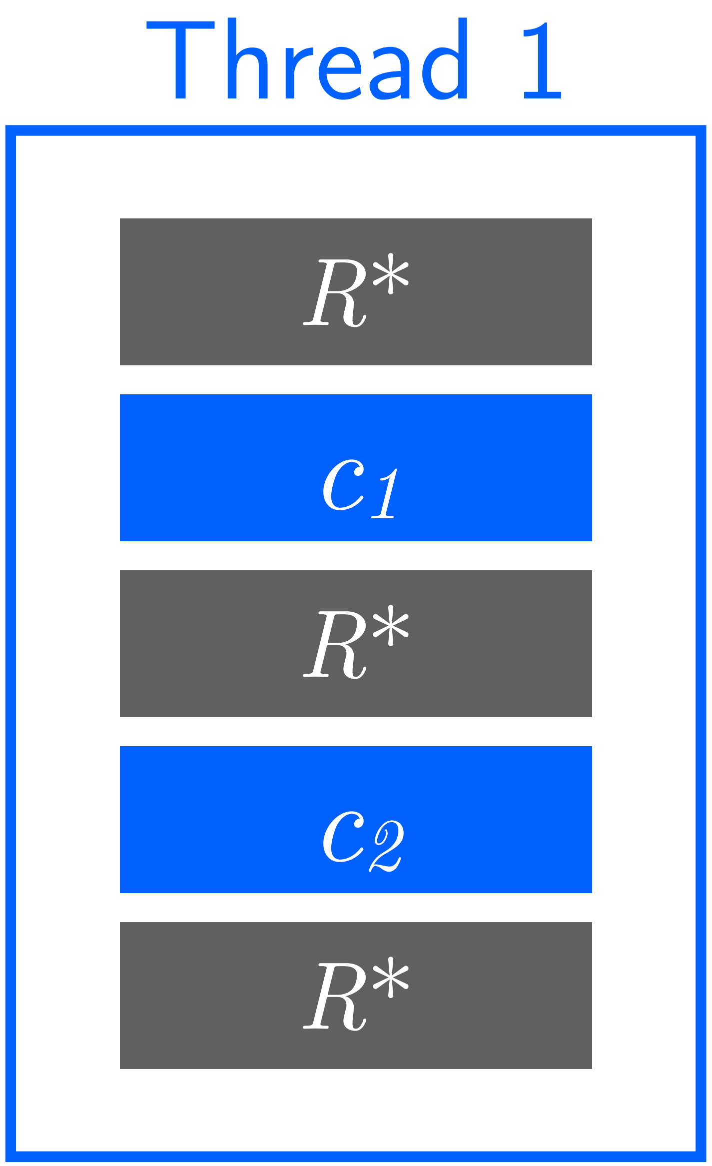
# Abstracting the environment



# Abstracting the environment



# Abstracting the environment




p stable under R

$$\Longleftrightarrow$$

$\vdash \{p\}\ R^*\ \{p\}$

# References


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

- 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

- 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


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

Contains proof of `FindFirstPositive` in RG.

Clear and comprehensive introduction to Rely-Guarantee

# Summary

- Owicki-Gries method
- Rely-Guarantee
- Stability
- Relating Rely-Guarantee with CSL
- Limitations of Rely-Guarantee
- NEXT LECTURE: RGSep...