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
      - Concurrent Separation Logic
        - Concurrent Separation Logic
          - RGSep
            - Deny-Guarantee, Concurrent Abstract Predicates, Local Rely-Guarantee...

Next lecture

This lecture

Later lectures
Parallel Rule

\[ \vdash \{P_1\} C_1 \{Q_1\} \]
\[ \vdash \{P_2\} C_2 \{Q_2\} \]

$C_1$ doesn't affect $C_2$'s proof

$C_2$ doesn't affect $C_1$'s proof

\[ \vdash \{P_1 \land P_2\} C_1 \parallel C_2 \{Q_1 \land Q_2\} \]
FindFirstPositive

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

\[
\text{while } i < \min(x, y) \text{ do }
\begin{align*}
\text{if } A[i] > 0 \text{ then} & \quad \text{while } j < \min(x, y) \text{ do } \\
\quad x := i & \quad \text{if } A[j] > 0 \text{ then } y := j \\
\quad \text{else} & \quad \text{else} \\
\quad i := i + 2 & \quad j := j + 2 \\
\end{align*}
\]

\text{end if}
\text{end while}
\text{end while}

\[ r := \min(x, y) \]
\begin{align*}
i &:= 0; \quad j := 1; \quad x := |A|; \quad y := |A|; \\
\{P_1 \land P_2\}
\end{align*}

\{P_1\}
\begin{align*}
\textbf{while} & \ i<\min(x,y) \ \textbf{do} \ \{P_1 \land i<x \land i<|A|\} \\
\textbf{if} & \ A[i]>0 \ \textbf{then} \ \{P_1 \land i<x \land i<|A| \land A[i]>0\} \\
& \quad x := i \ \{P_1\} \\
\textbf{else} & \ \{P_1 \land i<x \land i<|A| \land A[i] \leq 0\} \\
& \quad i := i+2 \ \{P_1\} \\
\textbf{end if} \ \{P_1\}
\end{align*}

\textbf{end while} \ \{P_1 \land i \geq \min(x,y)\}

\{P_2\}
\begin{align*}
\textbf{while} & \ j<\min(x,y) \ \textbf{do} \ \{P_2 \land j<y \land j<|A|\} \\
\textbf{if} & \ A[j]>0 \ \textbf{then} \ \{P_2 \land j<y \land j<|A| \land A[j]>0\} \\
& \quad y := j \ \{P_2\} \\
\textbf{else} & \ \{P_2 \land j<y \land j<|A| \land A[j] \leq 0\} \\
& \quad j := j+2 \ \{P_2\} \\
\textbf{end if} \ \{P_2\}
\end{align*}

\textbf{end while} \ \{P_2 \land j \geq \min(x,y)\}

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

\begin{align*}
r &:= \min(x,y) \\
\{r \leq |A| \land (\forall k. \ 0 \leq k<r \Rightarrow A[k] \leq 0) \land (r<|A| \Rightarrow A[r]>0)\}
\end{align*}

where
\begin{align*}
P_1 &\equiv x \leq |A| \land (\forall k. \ 0 \leq k<i \land k \text{ even } \Rightarrow A[k] \leq 0) \land i \text{ even } \land (x<|A| \Rightarrow A[x]>0) \\
\text{and} \ P_2 &\equiv y \leq |A| \land (\forall k. \ 0 \leq k<j \land k \text{ odd } \Rightarrow A[k] \leq 0) \land j \text{ odd } \land (y<|A| \Rightarrow A[y]>0)
\end{align*}
Compositionality

\[
\begin{align*}
\vdash \{P\} C_1 \{Q\} \\
\vdash \{Q\} C_2 \{R\} \\
\therefore \vdash \{P\} C_1;C_2 \{R\}
\end{align*}
\]

\[
\begin{align*}
\vdash \{P\} C_1 \{Q\}[X_1] \\
\vdash \{Q\} C_2 \{R\}[X_2] \\
\therefore \vdash \{P\} C_1;C_2 \{R\}[X_1 \cup X_2]
\end{align*}
\]

\[
\begin{align*}
\text{mods}(C) \cap \text{fv}(R) = \emptyset \\
\vdash \{P \ast R\} C \{Q \ast R\}
\end{align*}
\]

\[
\begin{align*}
\vdash \{P\} C \{Q\}[X] \\
X \cap \text{fv}(R) = \emptyset \\
\therefore \vdash \{P \ast R\} C \{Q \ast R\}[X]
\end{align*}
\]
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

\[\begin{align*}
R \cup G_2, G_1 &\vdash \{P_1\} \quad C_1 \quad \{Q_1\} \\
R \cup G_1, G_2 &\vdash \{P_2\} \quad C_2 \quad \{Q_2\} \\
R, G_1 \cup G_2 &\vdash \{P_1 \land P_2\} \quad C_1 \quad \parallel \quad C_2 \quad \{Q_1 \land Q_2\}
\end{align*}\]
Rule of Consequence

\[
\text{If } R \subseteq R' \text{ and } R', G \vdash \{P\} \text{ and } G' \subseteq G \text{, then } R, G \vdash \{P\} \text{ and } \{Q\}
\]
Basic commands

∀σ,σ'. P(σ) ∧ (σ,σ')∈⟦c⟧ ⇒ G(σ,σ')

P is stable under R
Q is stable under R
the effect of c is contained in G

R, G ⊨ {P} c {Q}

∀σ,σ'. P(σ) ∧ R(σ,σ') ⇒ P(σ')
Making assertions stable

\[ x = n \iff x = n - 1 \quad x = n \iff x = n + 1 \]

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

\[ R, G \vdash \{ x = 2 \} \]

\[ x := x + 1 \]

\[ \{ x = 3 \} \]
Making assertions stable

\[
\begin{align*}
x &= n \iff x &= n - 1 \\
x &= n \iff x &= n + 1
\end{align*}
\]

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

\[
R, G \vdash \{x \leq 2\} \\
x := x + 1 \\
\{x \leq 3\}
\]


## Quiz

\[ R \overset{\text{def}}{=} x=n \iff x=n+1 \]

<table>
<thead>
<tr>
<th></th>
<th>Strongest stable weaker assertion</th>
<th>Weakest stable stronger assertion</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x=0 )</td>
<td>( x \geq 0 )</td>
<td>false</td>
</tr>
<tr>
<td>( x \neq 0 )</td>
<td>true</td>
<td>( x &gt; 0 )</td>
</tr>
</tbody>
</table>
Stabilisation

\[ \text{sswa}(P) \]

\[ P \]

\[ \text{wssa}(P) \]
\[
i := 0; j := 1; x := |A|; y := |A|;
\{P_1 \land P_2\}
\]

\[G_2, G_1 \vdash \{P_1\}\]
\[
\text{while } i < \min(x,y) \text{ do } \{P_1 \land i < x \land i < |A|\}
\]
\[
\quad \text{if } A[i] > 0 \text{ then } \{P_1 \land i < x \land i < |A| \land A[i] > 0\}
\quad x := i \{P_1\}
\]
\[
\quad \text{else } \{P_1 \land i < x \land i < |A| \land A[i] \leq 0\}
\quad i := i + 2 \{P_1\}
\]
\[
\text{end if } \{P_1\}
\]
\[
\text{end while } \{P_1 \land i \geq \min(x,y)\}
\]

\[G_1, G_2 \vdash \{P_2\}\]
\[
\text{while } j < \min(x,y) \text{ do } \{P_2 \land j < y \land j < |A|\}
\]
\[
\quad \text{if } A[j] > 0 \text{ then } \{P_2 \land j < y \land j < |A| \land A[j] > 0\}
\quad y := j \{P_2\}
\]
\[
\quad \text{else } \{P_2 \land j < y \land j < |A| \land A[j] \leq 0\}
\quad j := j + 2 \{P_2\}
\]
\[
\text{end if } \{P_2\}
\]
\[
\text{end while } \{P_2 \land j \geq \min(x,y)\}
\]

\[\{P_1 \land P_2 \land i \geq \min(x,y) \land j \geq \min(x,y)\}\]
\[
r := \min(x,y)
\]
\[
\{r \leq |A| \land (\forall k. 0 \leq k < r \Rightarrow A[k] \leq 0) \land (r < |A| \Rightarrow A[r] > 0)\}\]

where \[P_1 \equiv x \leq |A| \land (\forall k. 0 \leq k < i \land k \text{ even } \Rightarrow A[k] \leq 0) \land i \text{ even } \land (x < |A| \Rightarrow A[x] > 0)\]

and \[P_2 \equiv y \leq |A| \land (\forall k. 0 \leq k < j \land k \text{ odd } \Rightarrow A[k] \leq 0) \land j \text{ odd } \land (y < |A| \Rightarrow A[y] > 0)\]

and \[G_1 \equiv \{ (\sigma, \sigma') \mid \sigma'(y) = \sigma(y) \land \sigma'(j) = \sigma(j) \land \sigma'(x) \leq \sigma(x) \}\]

and \[G_2 \equiv \{ (\sigma, \sigma') \mid \sigma'(x) = \sigma(x) \land \sigma'(i) = \sigma(i) \land \sigma'(y) \leq \sigma(y) \}\]
Lecture plan

1. Setting the stage

2. Introducing Rely-Guarantee

3. Rely-Guarantee vs. CSL

4. Limitations of Rely-Guarantee
## Comparison

### Concurrent Separation Logic

\[ J \models \{P\} \ C \ \{Q\} \]

- 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 \( J \)

### Rely-Guarantee

\[ R, G \models \{P\} \ C \ \{Q\} \]

- 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\}

\text{atomic}\ ( [x] := [x]+1 ) \parallel \text{atomic}\ ( [x] := [x]+2 )

\{x \mapsto 3\}
Try CSL...

{\text{emp}}\quad\text{atomic (}
  \{J\}
  [x] := [x]+1
  \{J\}
\)}

{\text{emp}}\quad{\text{emp}}\quad{\text{emp}}\quad{\text{emp}}

{\text{emp}}\quad{\text{emp}}\quad{\text{emp}}\quad{\text{emp}}
Try CSL...

\{x \mapsto 0\}

\{\text{emp}\}

\begin{align*}
\text{atomic (} & \\
\{\exists n \geq 0. \ x \mapsto n\} & \\
[x] := [x] + 1 & \\
\{\exists n \geq 0. \ x \mapsto n\} & \\
\) & \\
\{\text{emp}\} &
\end{align*}

\{\text{emp}\}

\begin{align*}
\text{atomic (} & \\
\{\exists n \geq 0. \ x \mapsto n\} & \\
[x] := [x] + 2 & \\
\{\exists n \geq 0. \ x \mapsto n\} & \\
\) & \\
\{\text{emp}\} &
\end{align*}

\{\exists n \geq 0. \ x \mapsto n\}
Try CSL + auxiliary state...

\{x \mapsto 0\}
\begin{align*}
a & := 0; 
b & := 0; 
\{x \mapsto a + b \ * \ a \neq 0 \ * \ b \neq 0\}
\end{align*}

\{a \neq 0\}
\begin{align*}
\text{atomic (}
\{x \mapsto a + b \ * \ a \neq 0\}
[x] & := [x] + 1; 
a & := 1
\{x \mapsto a + b \ * \ a \neq 1\}
\text{)}
\{a \neq 1\}
\end{align*}

\{b \neq 0\}
\begin{align*}
\text{atomic (}
\{x \mapsto a + b \ * \ b \neq 0\}
[x] & := [x] + 2; 
b & := 2
\{x \mapsto a + b \ * \ b \neq 2\}
\text{)}
\{b \neq 2\}
\end{align*}

\{x \mapsto a + b \ * \ a \neq 1 \ * \ b \neq 2\}
Try Rely-Guarantee...

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

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

where \[ G_1 \overset{\text{def}}{=} (x \mapsto 0 \leadsto x \mapsto 1) \cup (x \mapsto 2 \leadsto x \mapsto 3) \]
and \[ G_2 \overset{\text{def}}{=} (x \mapsto 0 \leadsto x \mapsto 2) \cup (x \mapsto 1 \leadsto x \mapsto 3) \]
Lecture plan

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

\[ \{x\mapsto 0\} \]

atomic ( \([x] := [x]+1\) ) \parallel atomic ( \([x] := [x]+1\) )

\[\{x\mapsto 2\}\]
Try Rely-Guarantee...

not stable

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

atomic ( \[
[x] := [x] + 1
\]
)

\[ \{x \mapsto 1 \lor x \mapsto 2\} \]

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

atomic ( \[
[x] := [x] + 1
\]
)

\[ \{x \mapsto 1 \lor x \mapsto 2\} \]

where \( G_1, G_2 \overset{\text{def}}{=} (x \mapsto 0 \Rightarrow x \mapsto 1) \cup (x \mapsto 1 \Rightarrow x \mapsto 2) \)
Try Rely-Guarantee...

G₂, G₁ ⊢ \{∃n ≥ 0. x ↦ n\}  
\text{atomic (}  
\[x\] := \[x\]+1  
\text{)}  
\{∃n ≥ 1. x ↦ n\}  
\{∃n ≥ 1. x ↦ n\}  
\{∃n ≥ 1. x ↦ n\}  

where G₁,G₂ ≝ (x ↦ n ↭ x ↦ 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

Thread 1
- $C_1$
- $C_2$

Thread 2
- $C_3$
- $C_4$

Thread 3
- $C_5$
- $C_6$
Abstracting the environment
Abstracting the environment

Thread 1

\[ R^* \]
\[ C_1 \]
\[ R^* \]
\[ C_2 \]
\[ R^* \]

\( p \) stable under \( R \)

\( \vdash \{ p \} R^* \{ p \} \)
References


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...