CoLoSL

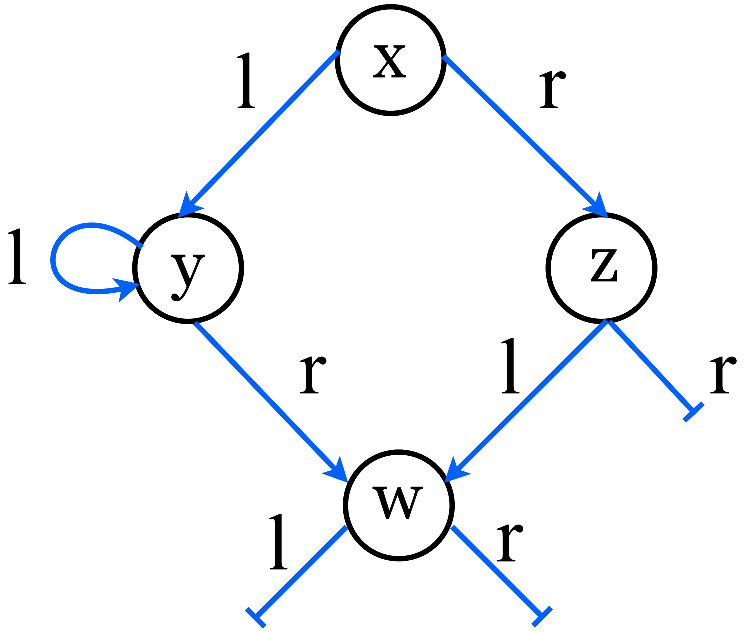
Concurrent **Lo**cal **S**ubjective **L**ogic

Philippa Gardner **Azalea Raad** Jules Villard

Imperial College London

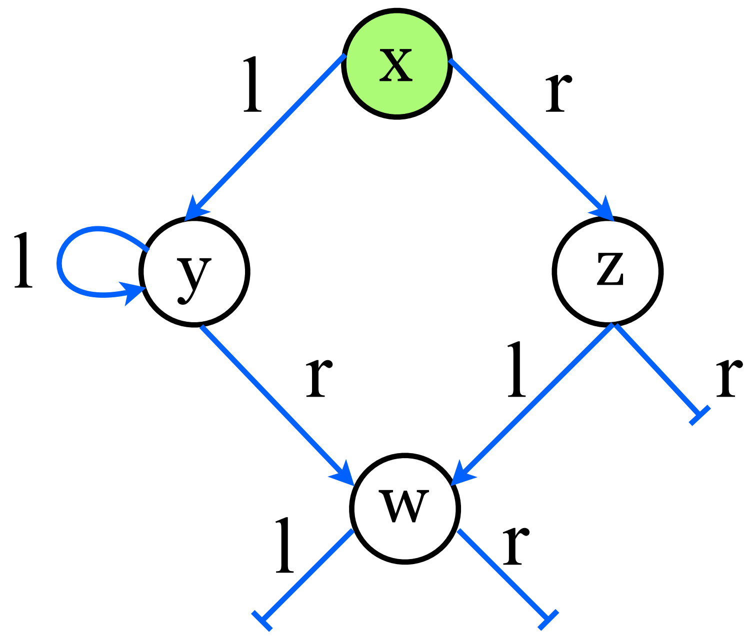
York Concurrency Workshop
28 April 2014

Example - Spanning Tree



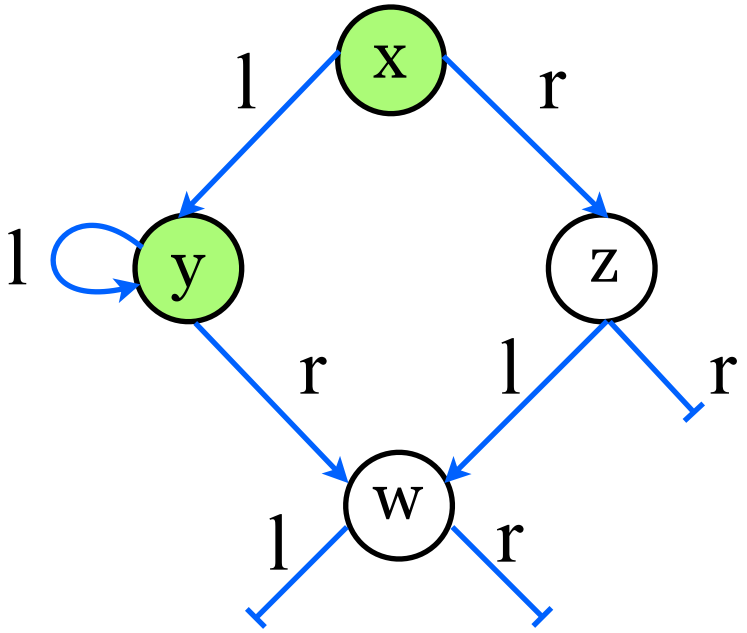
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



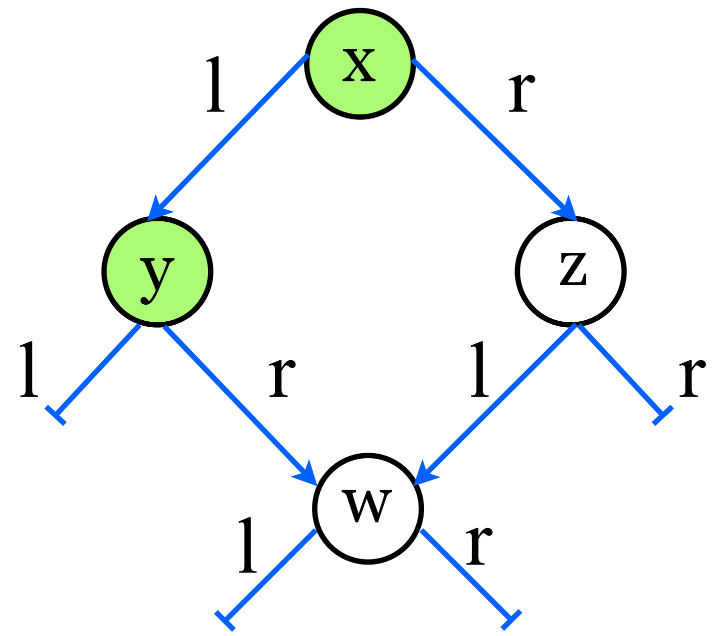
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



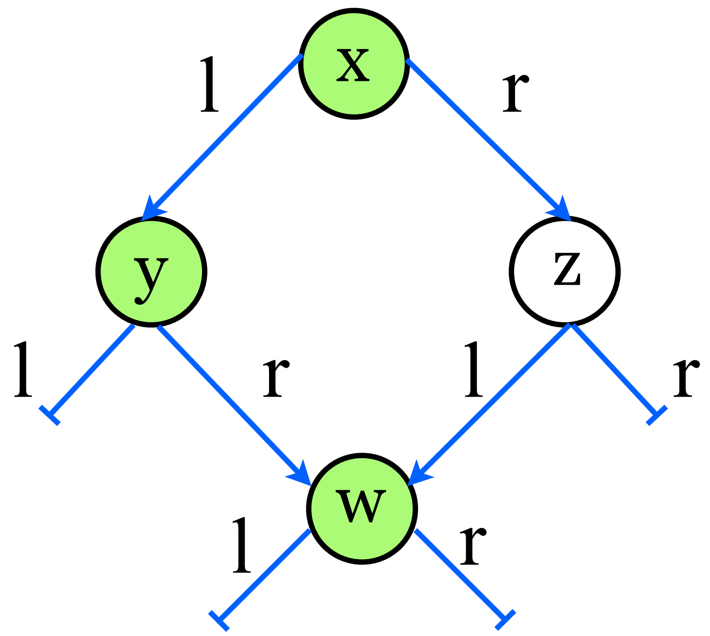
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



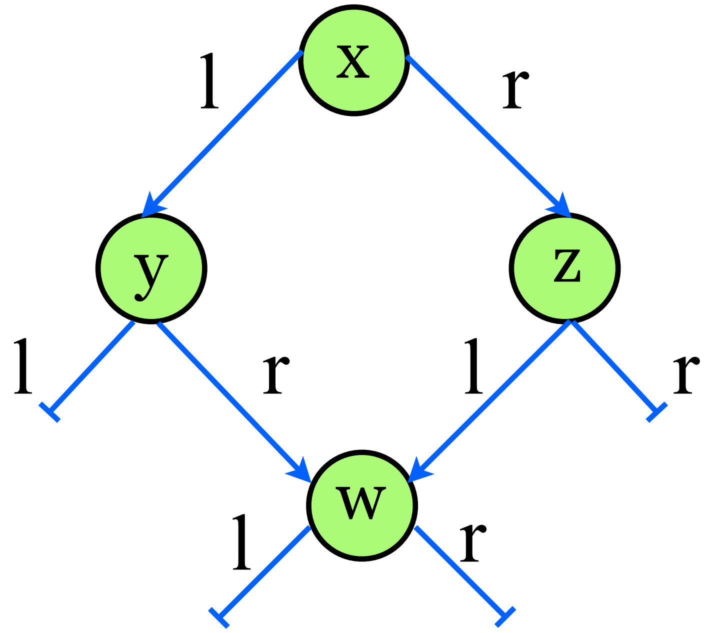
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



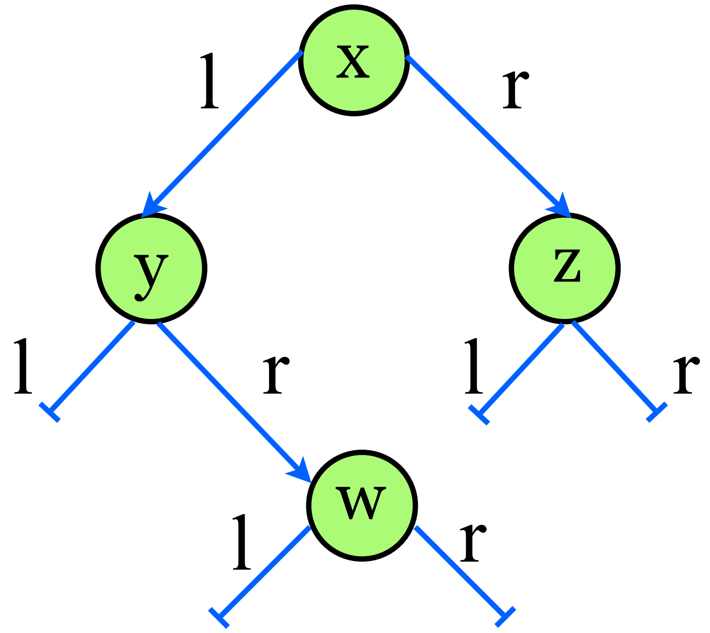
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



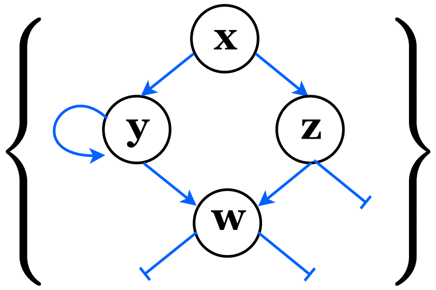
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

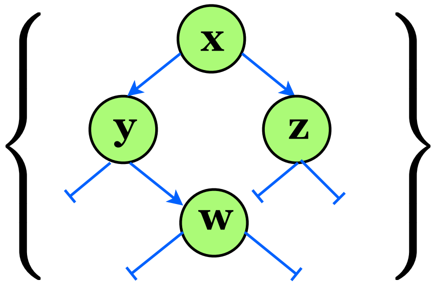


```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

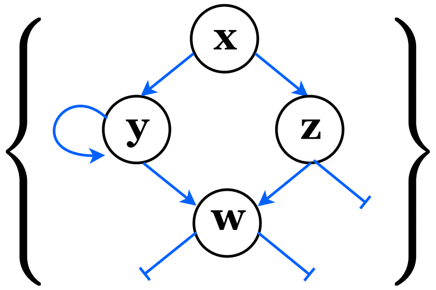
Example - Spanning Tree



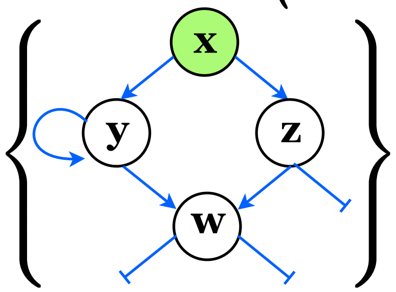
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```



Example - Spanning Tree

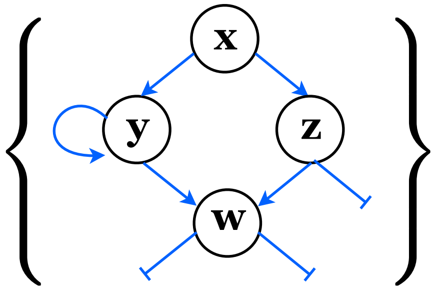


```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

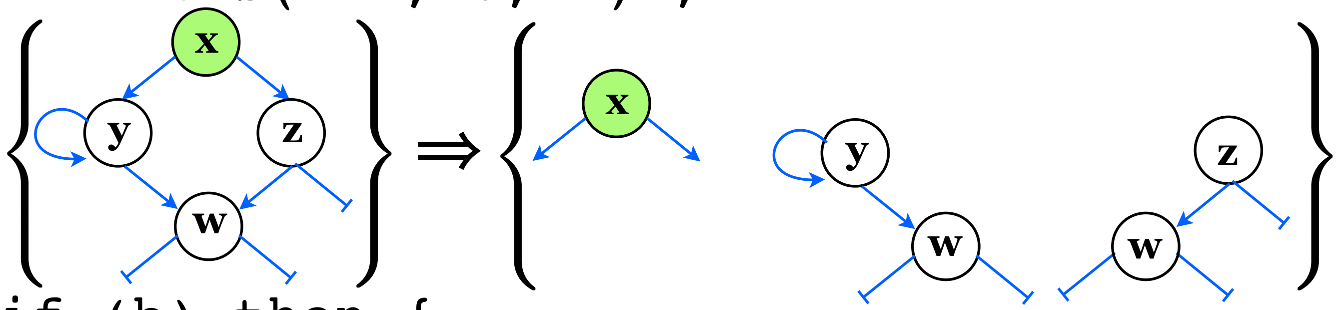


```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

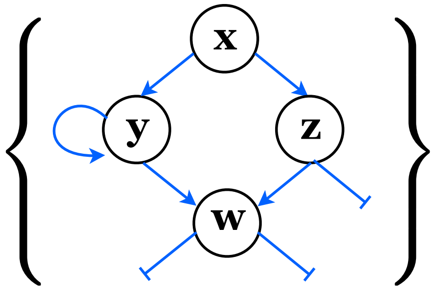


```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

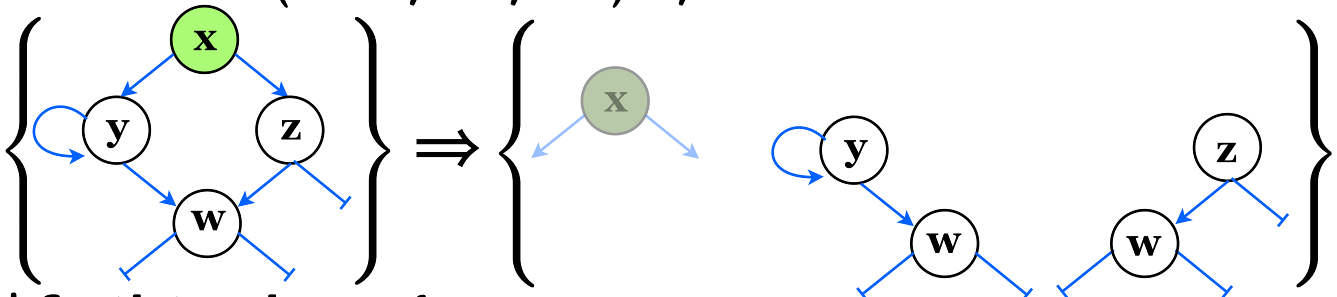


```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

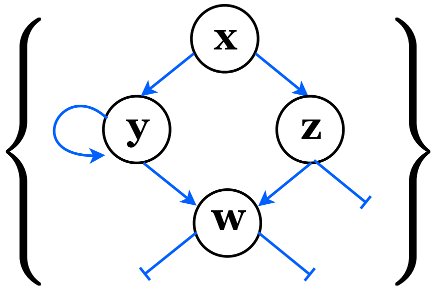


```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

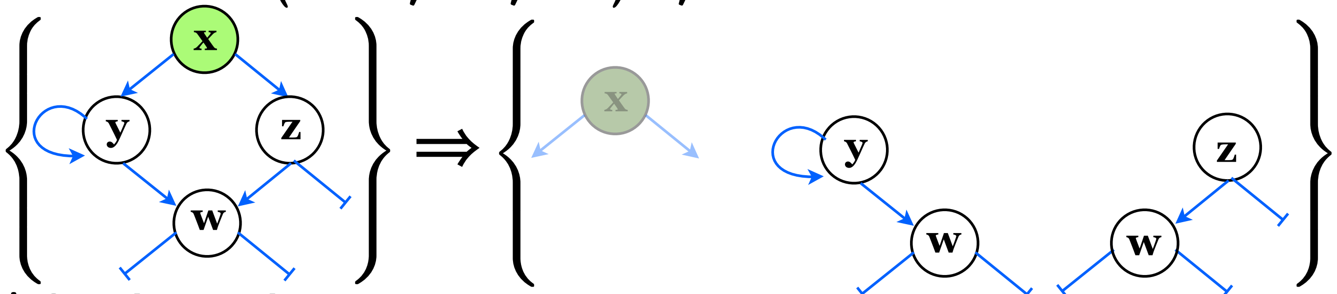


```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

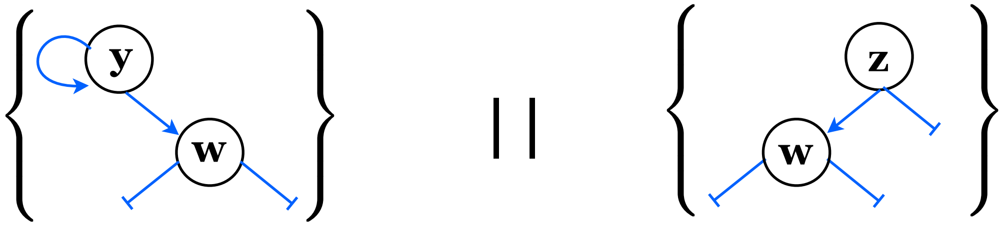
Example - Spanning Tree



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

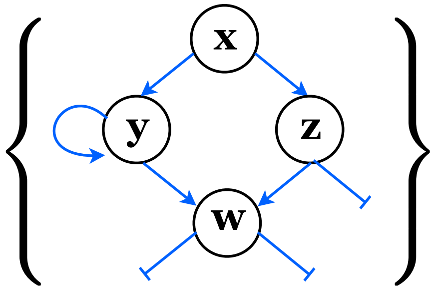


```
  if (b) then {
```

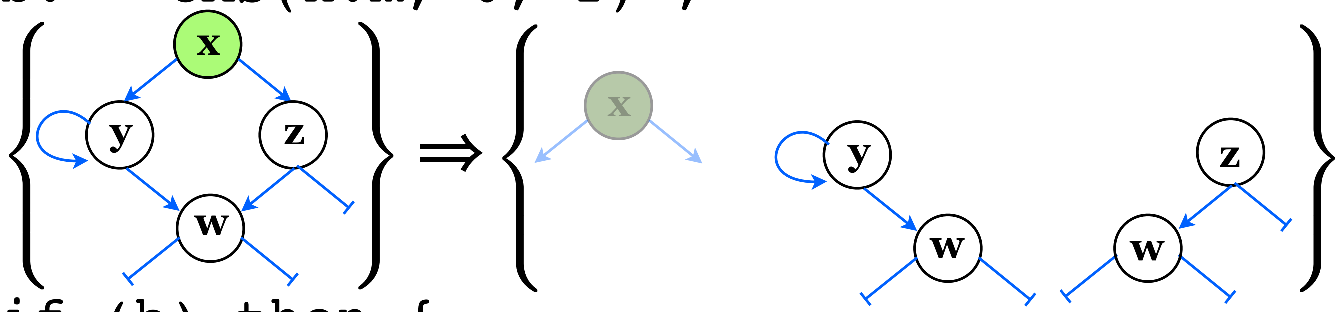


```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

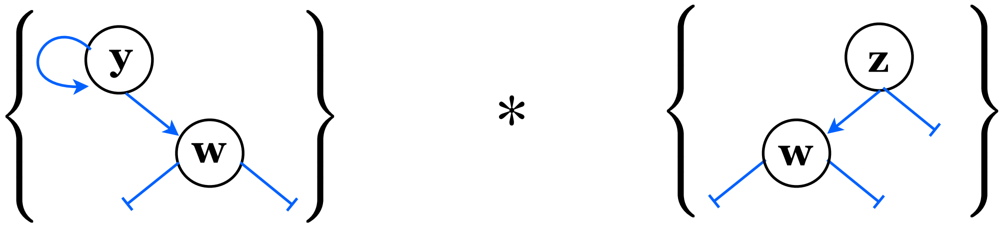
Example - Spanning Tree



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

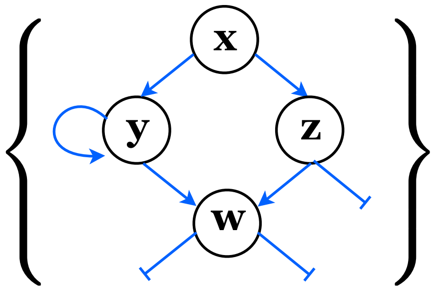


```
  if (b) then {
```

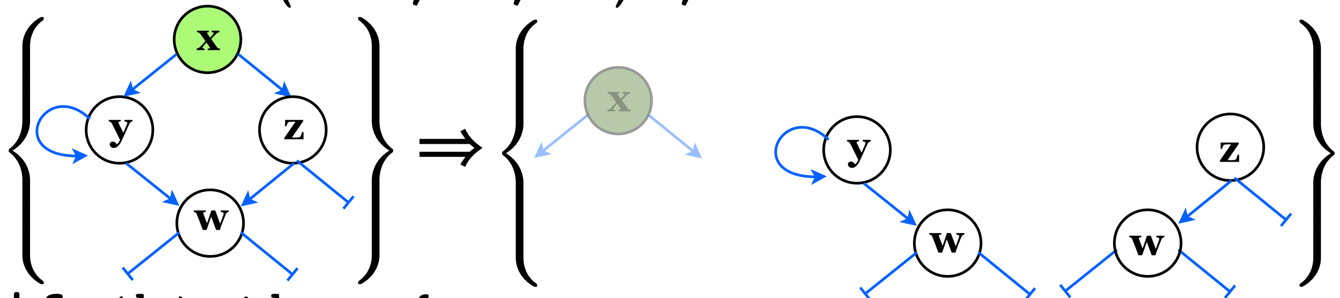


```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

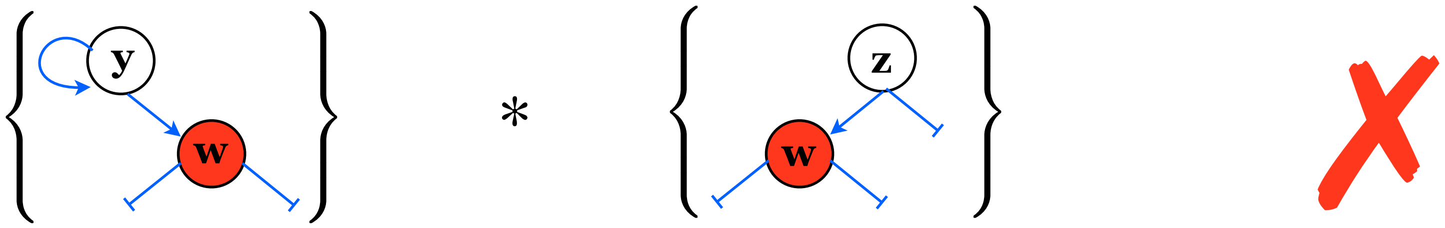
Example - Spanning Tree



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

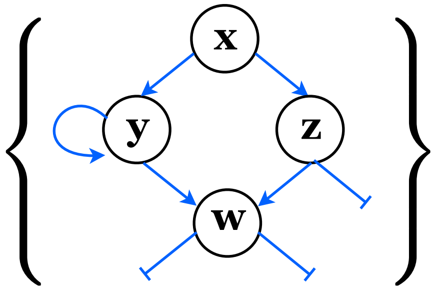


```
  if (b) then {
```

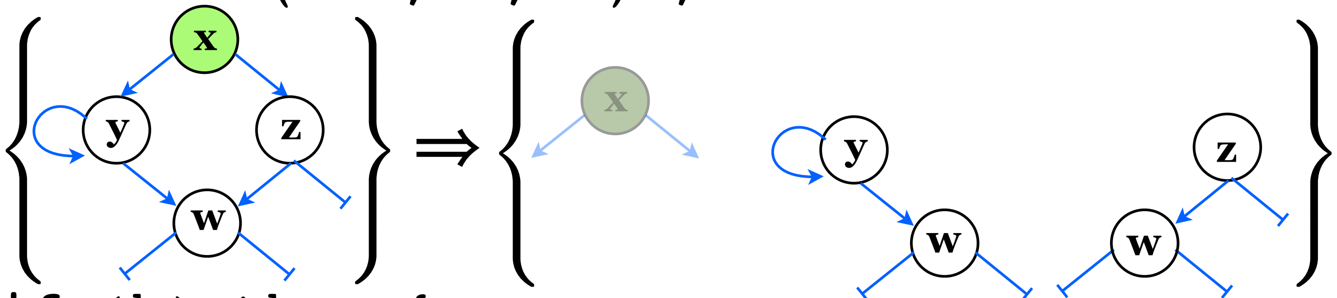


```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

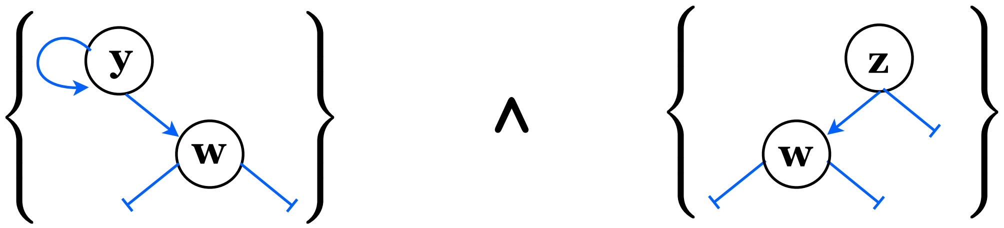
Example - Spanning Tree



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

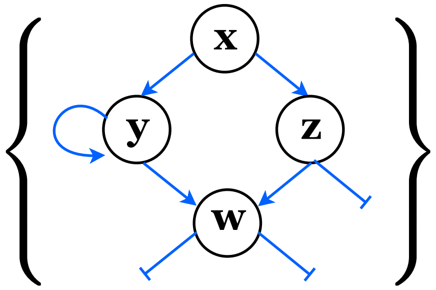


```
  if (b) then {
```

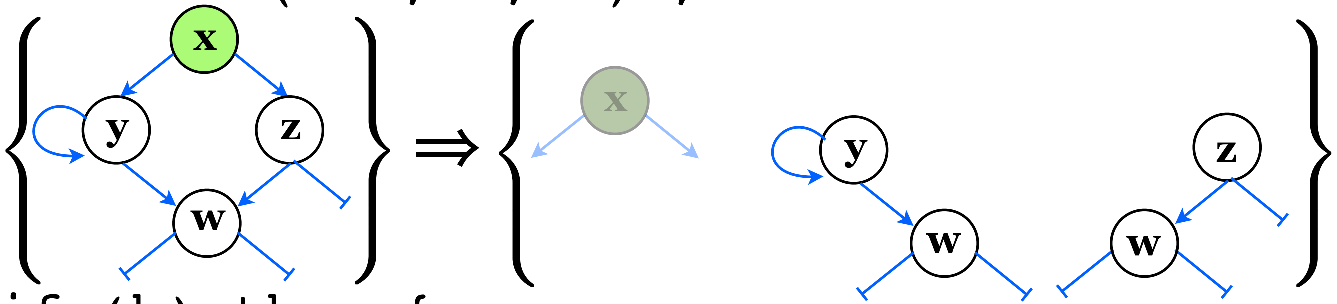


```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

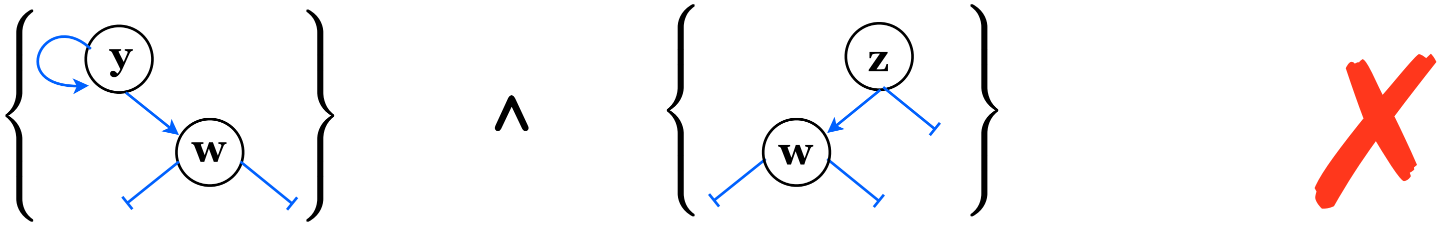
Example - Spanning Tree



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

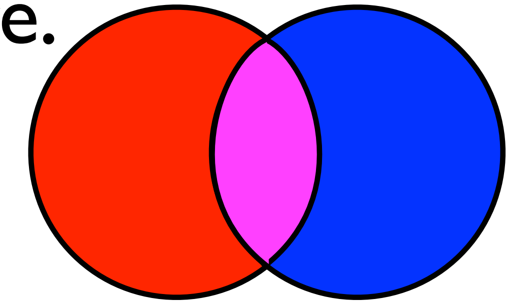


```
  if (b) then {
```



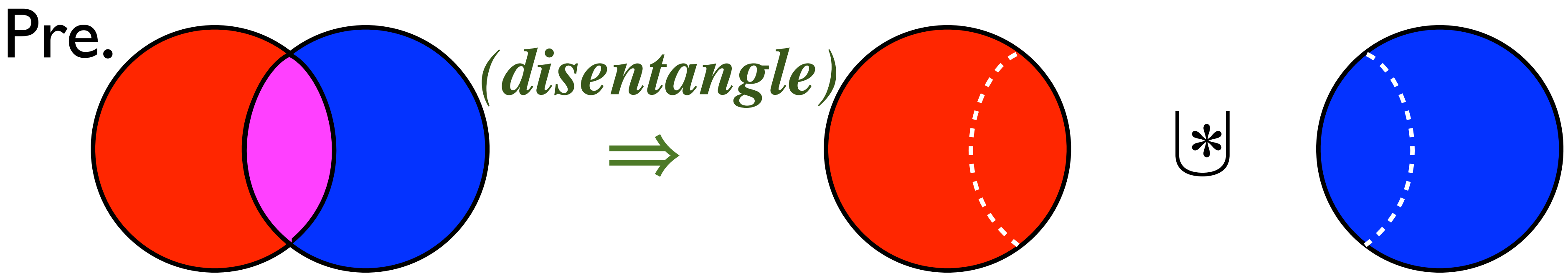
```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Entangled Shared Resources



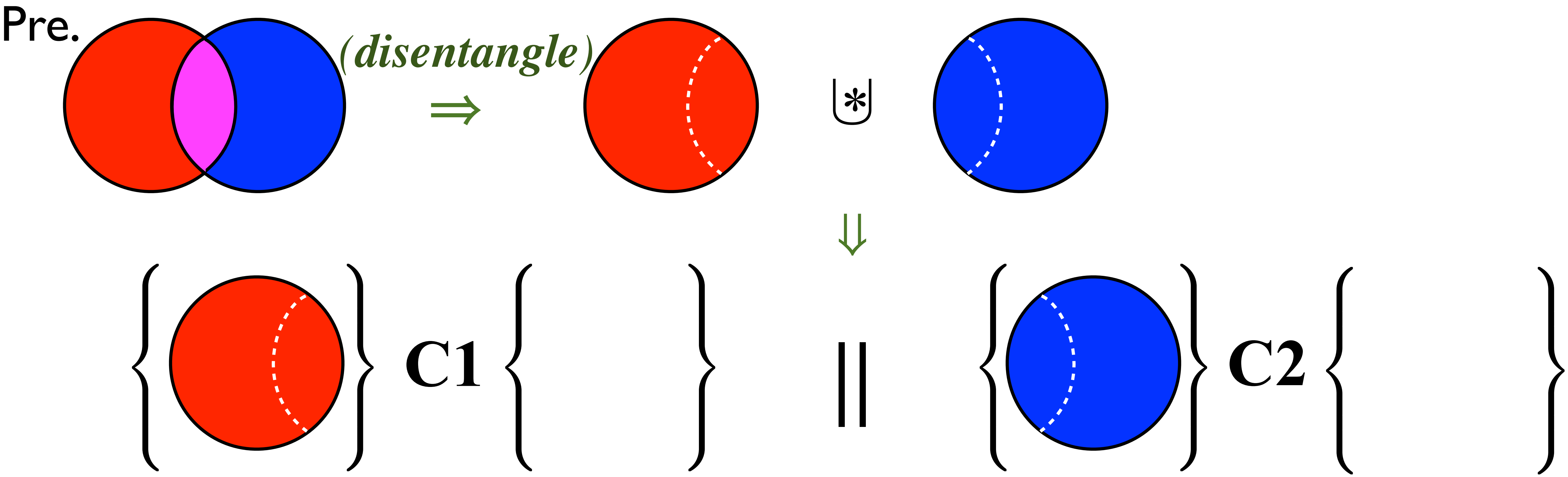
$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\}\ C1\ \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1 \mathbin{\uplus\!\!\!\!\ast} P2}\right\}\ C1 \parallel C2\ \left\{\boxed{Q1 \mathbin{\uplus\!\!\!\!\ast} Q2}\right\}}$$

Entangled Shared Resources



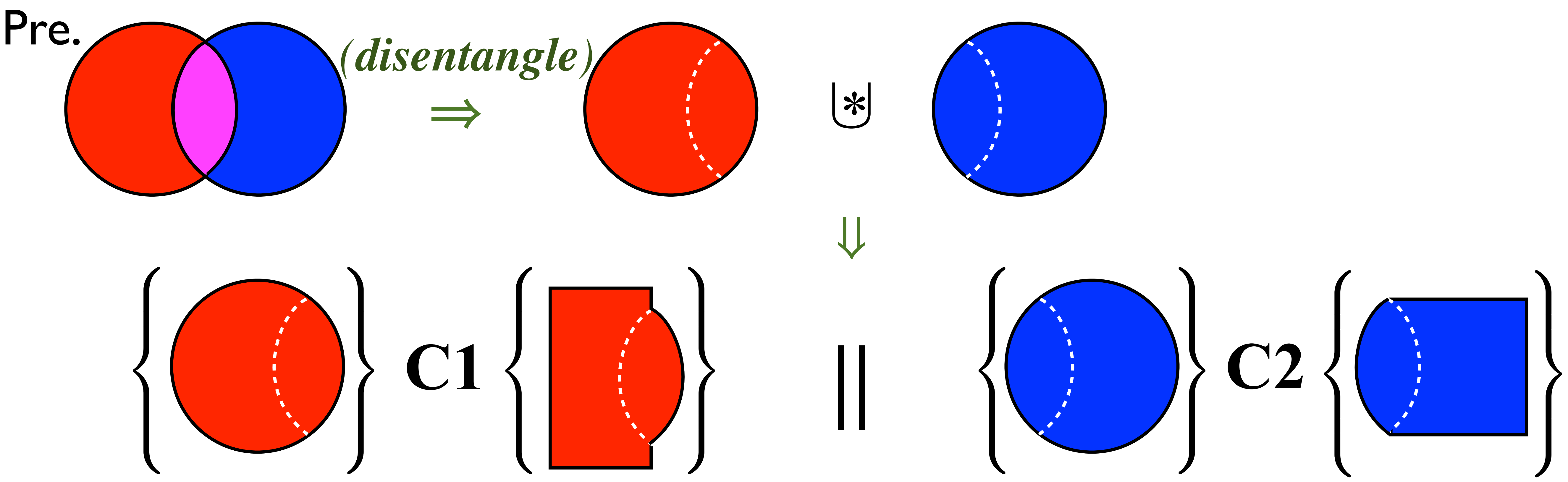
$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\}\ C1\ \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1 \mathbin{\uplus\!\!\!\!\ast} P2}\right\}\ C1 \parallel C2\ \left\{\boxed{Q1 \mathbin{\uplus\!\!\!\!\ast} Q2}\right\}}$$

Entangled Shared Resources



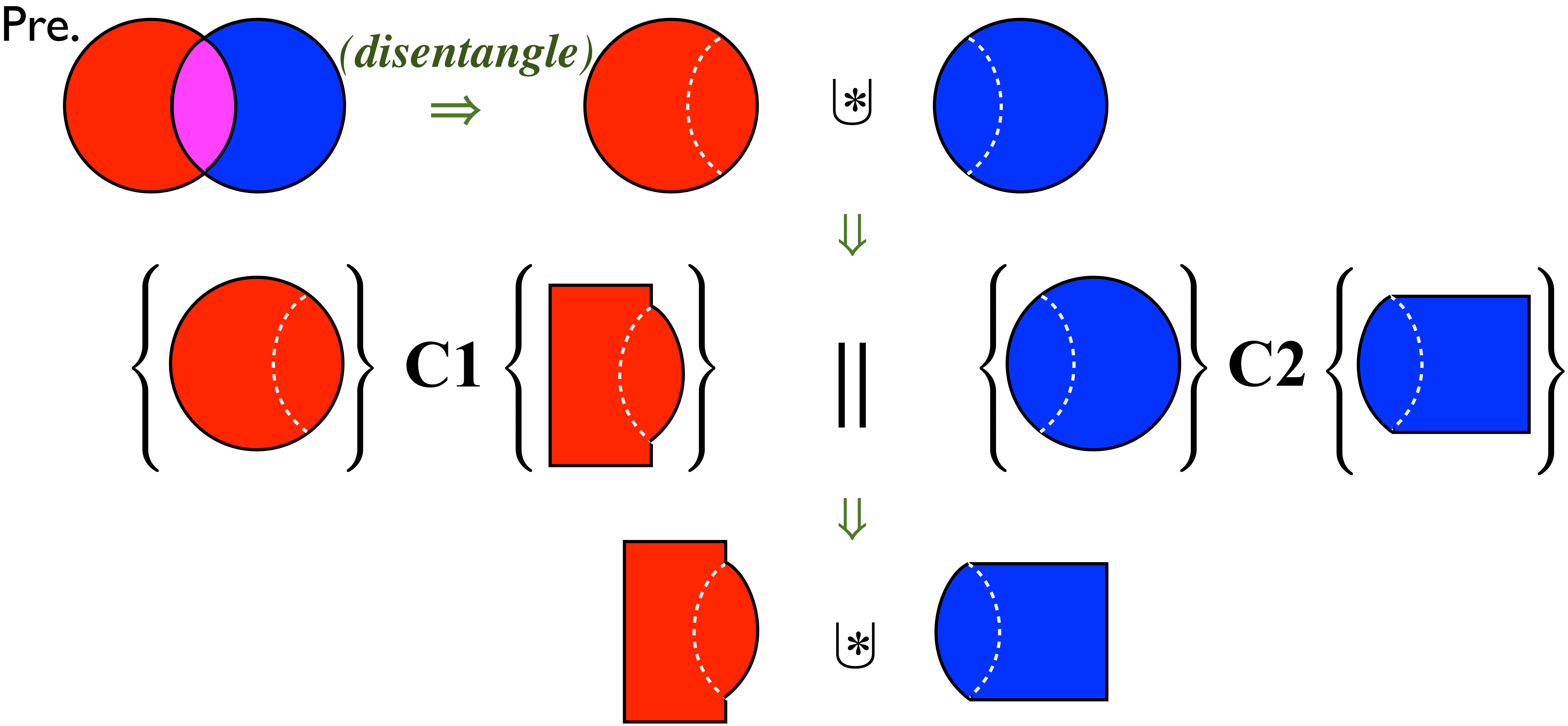
$$\frac{\{\boxed{P1}\}\ C1\ \{\boxed{Q1}\} \qquad \{\boxed{P2}\}\ C1\ \{\boxed{Q2}\}}{\{\boxed{P1 \mathbin{\uplus\!\!\!\!\ast} P2}\}\ C1 \parallel C2\ \{\boxed{Q1 \mathbin{\uplus\!\!\!\!\ast} Q2}\}}$$

Entangled Shared Resources



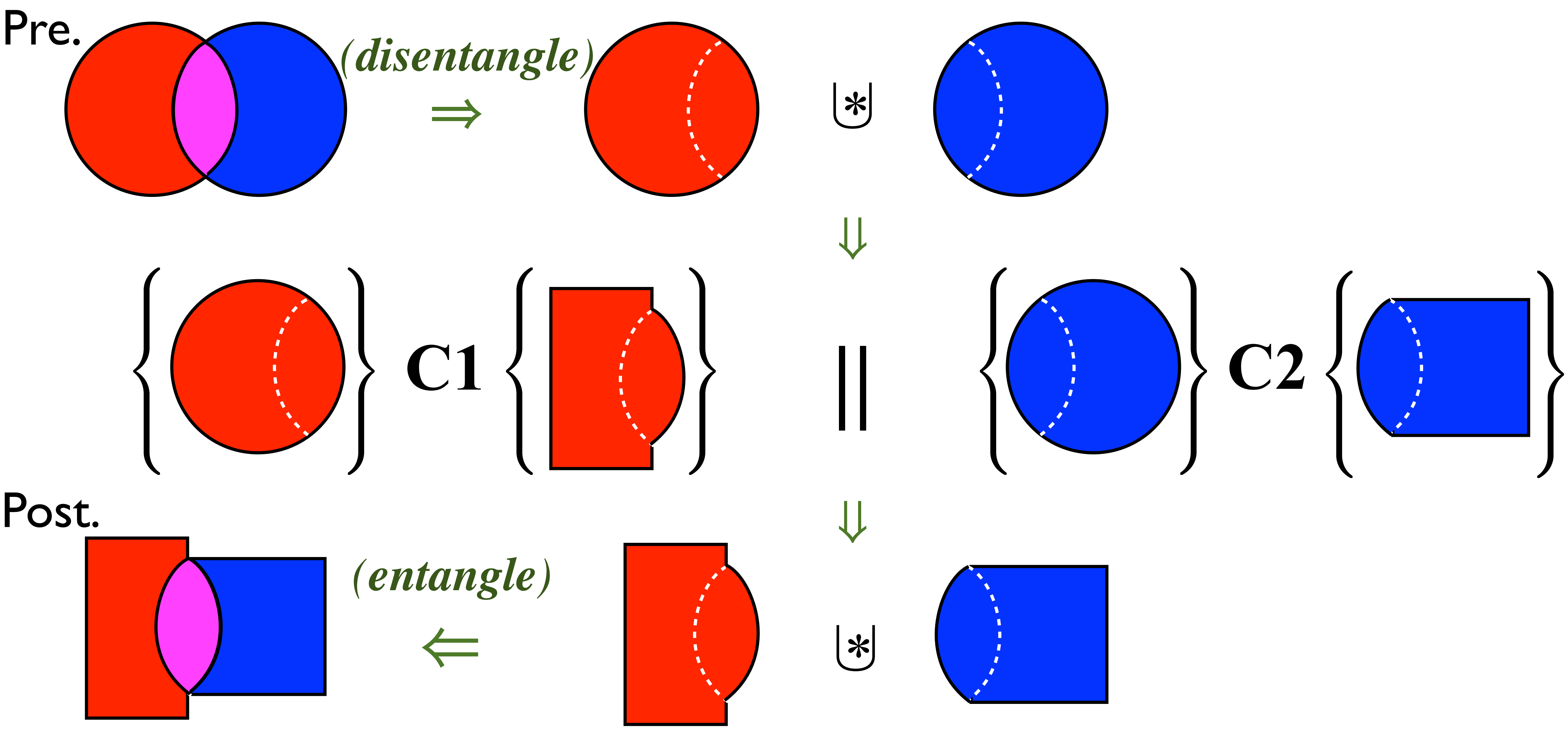
$$\frac{\left\{\boxed{\text{P1}}\right\}\ \text{C1}\ \left\{\boxed{\text{Q1}}\right\} \qquad \left\{\boxed{\text{P2}}\right\}\ \text{C1}\ \left\{\boxed{\text{Q2}}\right\}}{\left\{\boxed{\text{P1} \mathbin{\cup\!\!\!\!{\scriptstyle *}} \text{P2}}\right\}\ \text{C1} \parallel \text{C2}\ \left\{\boxed{\text{Q1} \mathbin{\cup\!\!\!\!{\scriptstyle *}} \text{Q2}}\right\}}$$

Entangled Shared Resources



$$\frac{\{\boxed{\mathrm{P1}}\}\ \mathrm{C1}\ \{\boxed{\mathrm{Q1}}\} \qquad \{\boxed{\mathrm{P2}}\}\ \mathrm{C1}\ \{\boxed{\mathrm{Q2}}\}}{\{\boxed{\mathrm{P1} \mathbin{\uplus\!\!\!*} \mathrm{P2}}\}\ \mathrm{C1} \parallel \mathrm{C2}\ \{\boxed{\mathrm{Q1} \mathbin{\uplus\!\!\!*} \mathrm{Q2}}\}}$$

Entangled Shared Resources

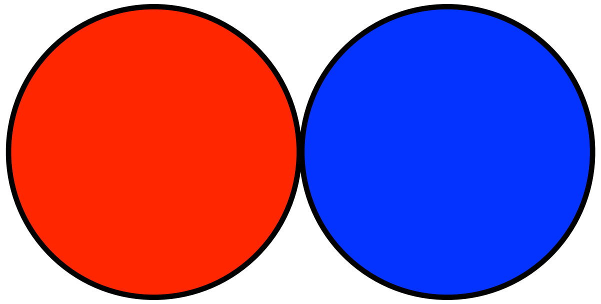


$$\frac{\{\boxed{P1}\}\ C1\ \{\boxed{Q1}\} \qquad \{\boxed{P2}\}\ C1\ \{\boxed{Q2}\}}{\{\boxed{P1 \uplus\!\!\!\!\!\ast\ P2}\}\ C1 \parallel C2\ \{\boxed{Q1 \uplus\!\!\!\!\!\ast\ Q2}\}}$$

CoLoSL

Disjoint Shared Resources

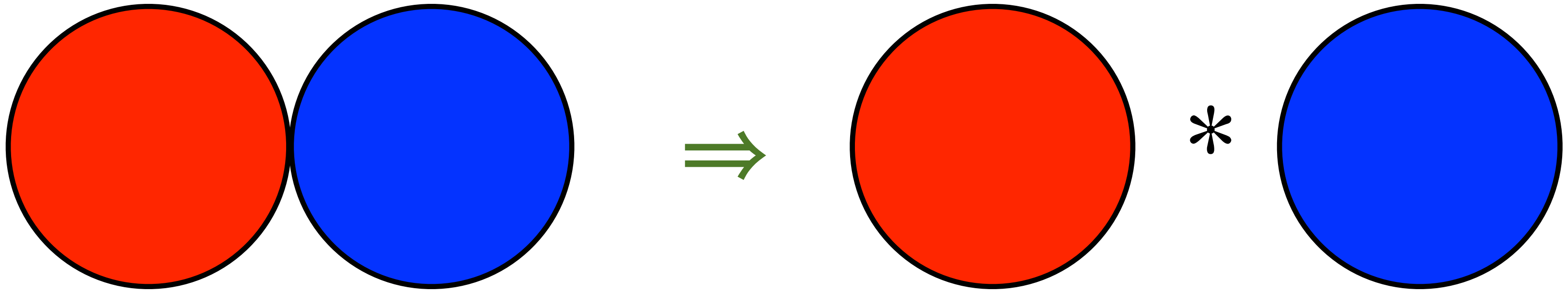
Pre.



$$\frac{\left\{\boxed{\mathrm{P1}}\right\}\ \mathrm{C1}\ \left\{\boxed{\mathrm{Q1}}\right\} \qquad \left\{\boxed{\mathrm{P2}}\right\}\ \mathrm{C1}\ \left\{\boxed{\mathrm{Q2}}\right\}}{\left\{\boxed{\mathrm{P1} \;*\; \mathrm{P2}}\right\}\ \mathrm{C1} \parallel \mathrm{C2}\ \left\{\boxed{\mathrm{Q1} \;*\; \mathrm{Q2}}\right\}}$$

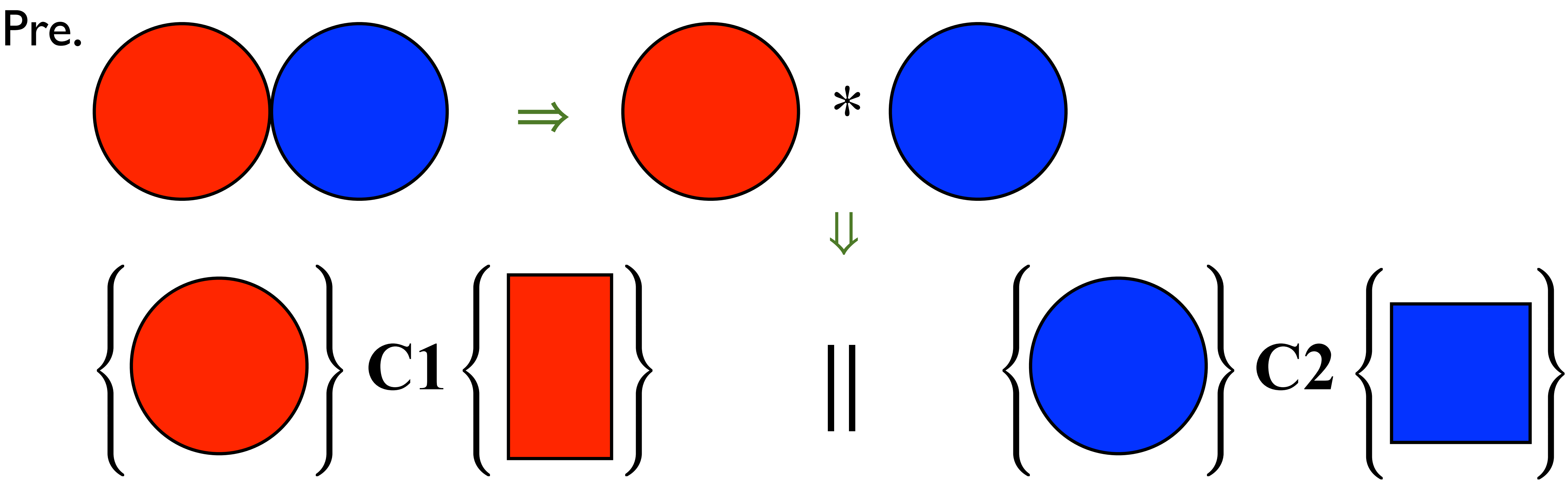
Disjoint Shared Resources

Pre.



$$\frac{\left\{\boxed{P1}\right\} C1 \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\} C1 \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1 \ * \ P2}\right\} C1 \parallel C2 \left\{\boxed{Q1 \ * \ Q2}\right\}}$$

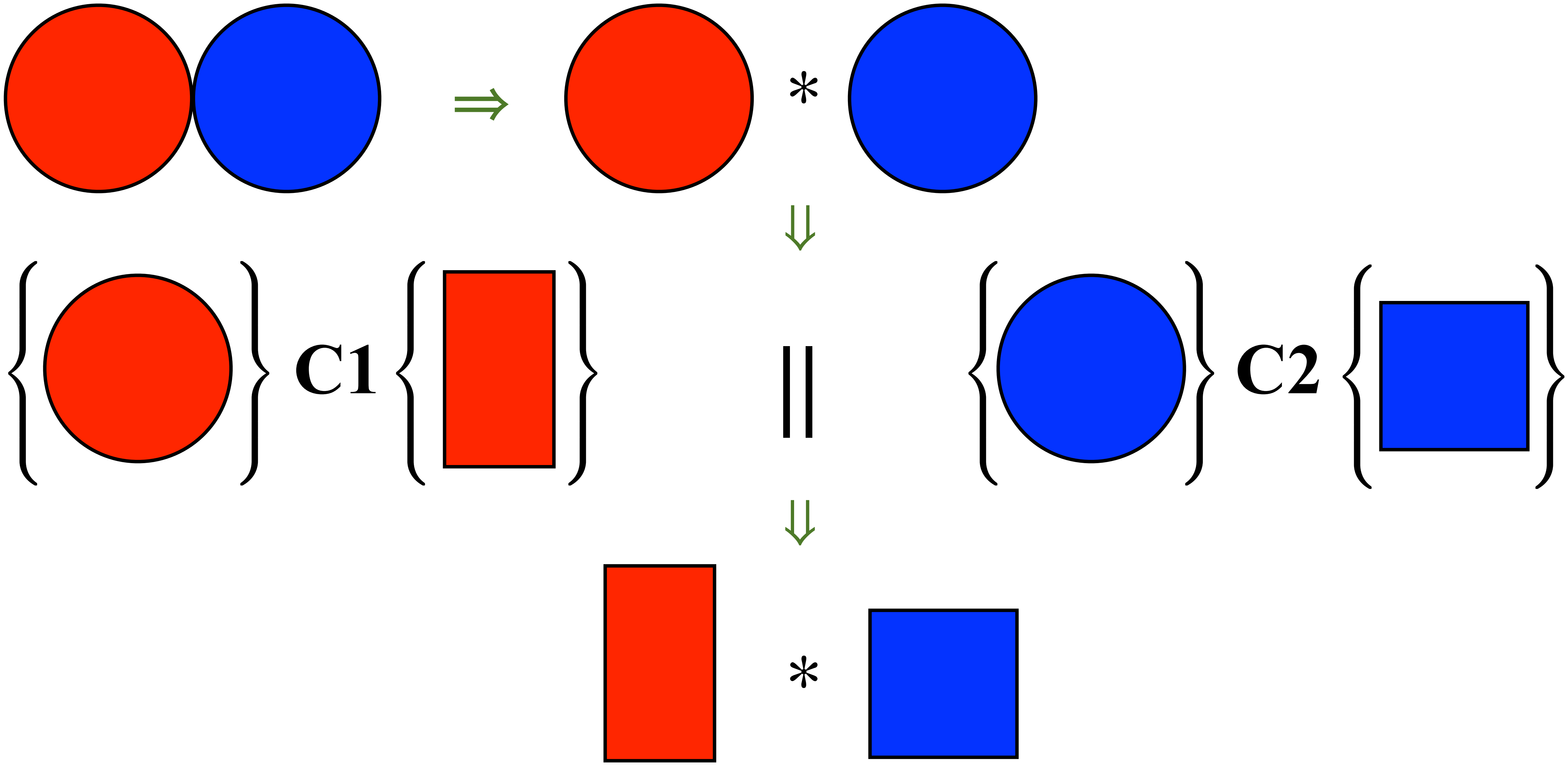
Disjoint Shared Resources



$$\frac{\{\boxed{P1}\}\ C1\ \{\boxed{Q1}\} \qquad \{\boxed{P2}\}\ C1\ \{\boxed{Q2}\}}{\{\boxed{P1\ *\ P2}\}\ C1 \parallel C2\ \{\boxed{Q1\ *\ Q2}\}}$$

Disjoint Shared Resources

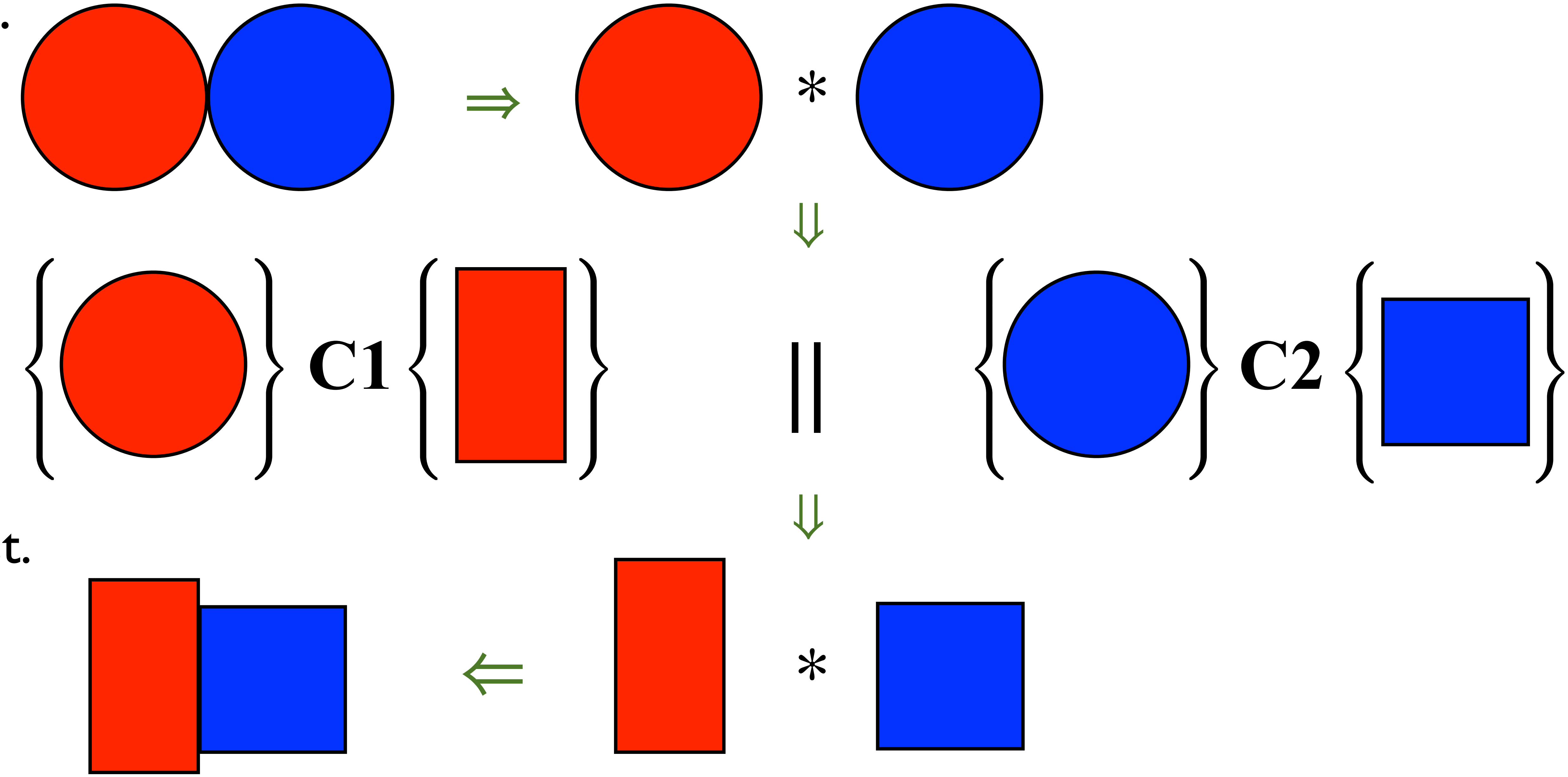
Pre.



$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\}\ C1\ \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1\ *\ P2}\right\}\ C1 \parallel C2\ \left\{\boxed{Q1\ *\ Q2}\right\}}$$

Disjoint Shared Resources

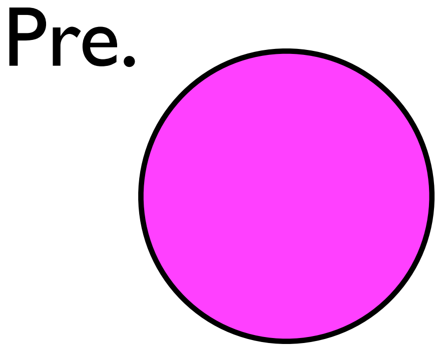
Pre.



Post.

$$\frac{\left\{\boxed{P1}\right\} C1 \left\{\boxed{Q1}\right\} \qquad \left\{\boxed{P2}\right\} C1 \left\{\boxed{Q2}\right\}}{\left\{\boxed{P1 \ * \ P2}\right\} C1 \parallel C2 \left\{\boxed{Q1 \ * \ Q2}\right\}}$$

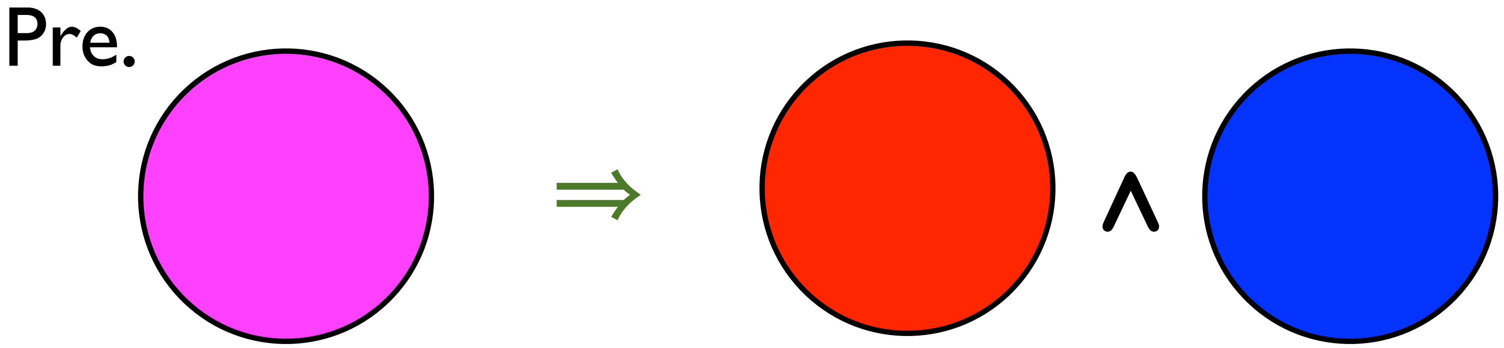
Global Shared Resources



Rely-Guarantee, RGSep

$$\frac{\left\{\boxed{P1}\right\} C1 \left\{\boxed{P1}\right\} \qquad \left\{\boxed{P2}\right\} C1 \left\{\boxed{P2}\right\}}{\left\{\boxed{P1 \wedge P2}\right\} C1 \parallel C2 \left\{\boxed{P1 \wedge P2}\right\}}$$

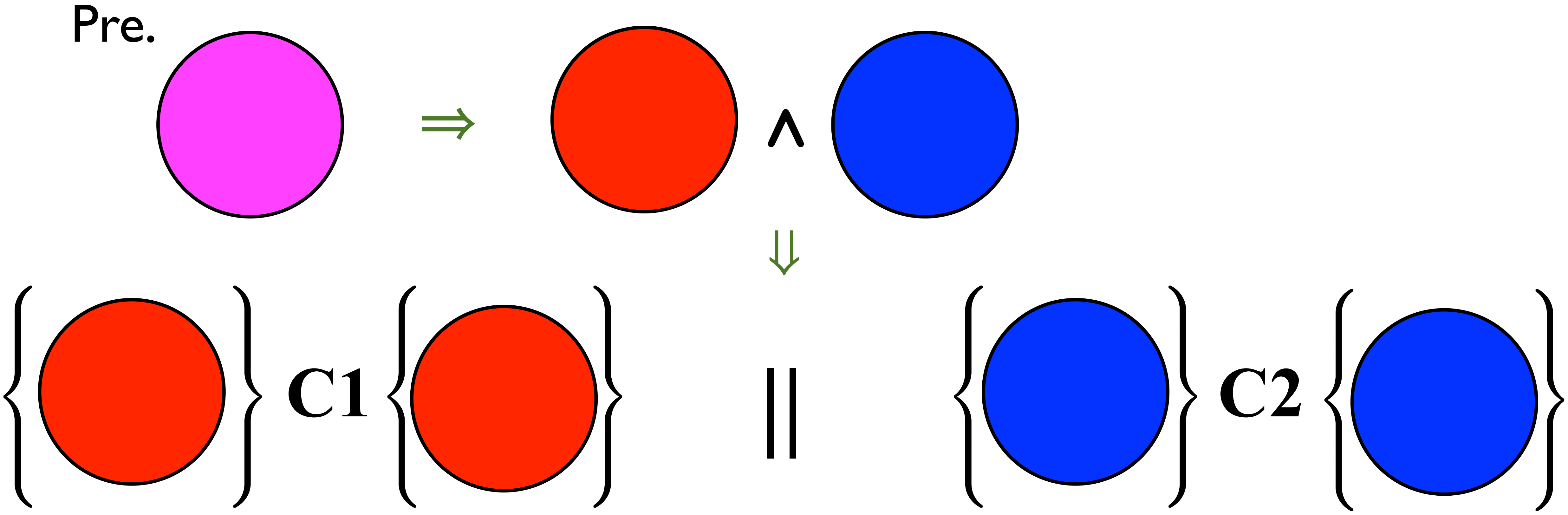
Global Shared Resources



Rely-Guarantee, RGSep

$$\frac{\left\{\boxed{P1}\right\} C1 \left\{\boxed{P1}\right\} \qquad \left\{\boxed{P2}\right\} C1 \left\{\boxed{P2}\right\}}{\left\{\boxed{P1 \land P2}\right\} C1 \parallel C2 \left\{\boxed{P1 \land P2}\right\}}$$

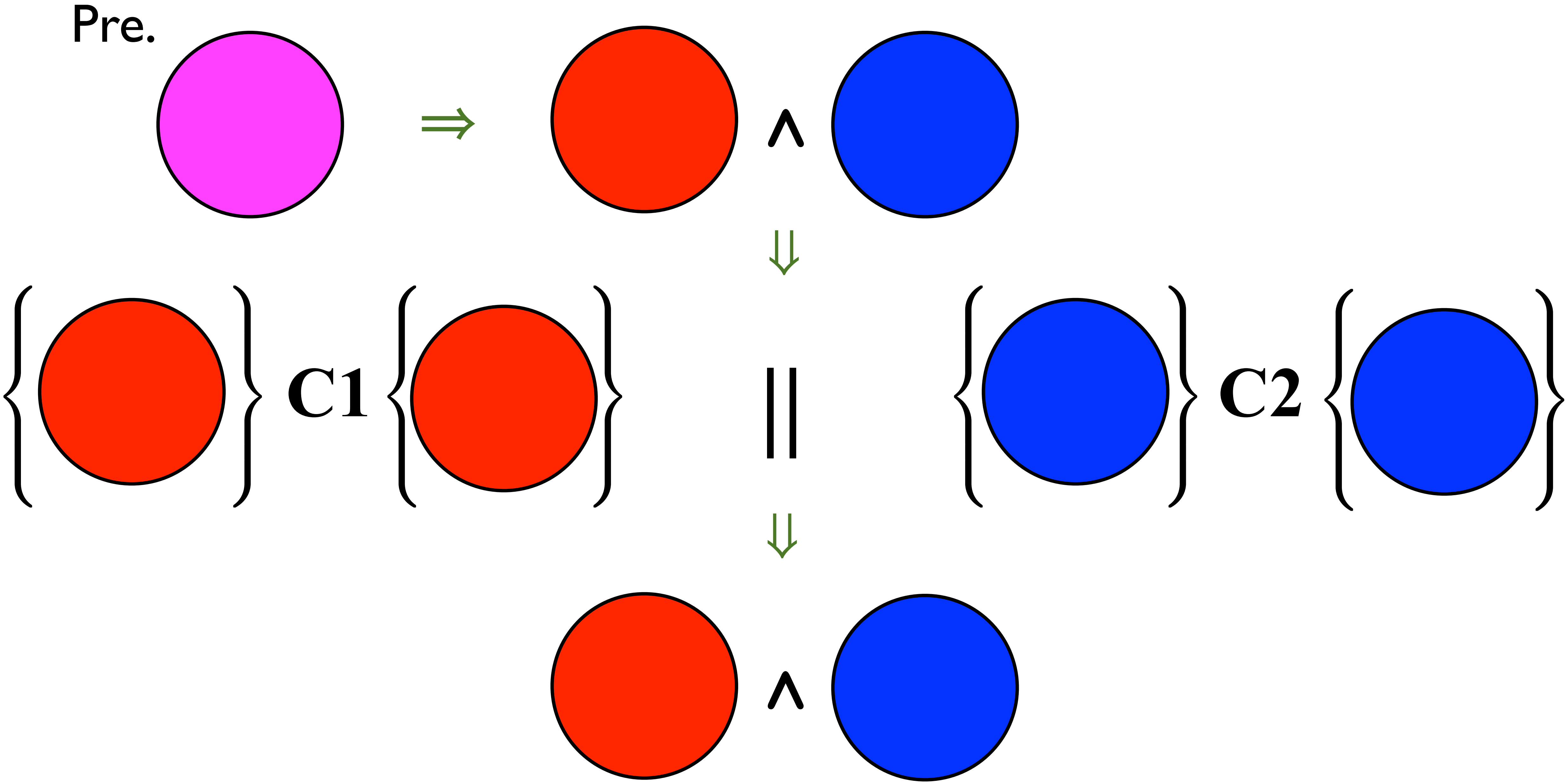
Global Shared Resources



$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{P1}\right\} \qquad \left\{\boxed{P2}\right\}\ C1\ \left\{\boxed{P2}\right\}}{\left\{\boxed{P1\ \land\ P2}\right\}\ C1 \parallel C2\ \left\{\boxed{P1\ \land\ P2}\right\}}$$

Rely-Guarantee, RGSep

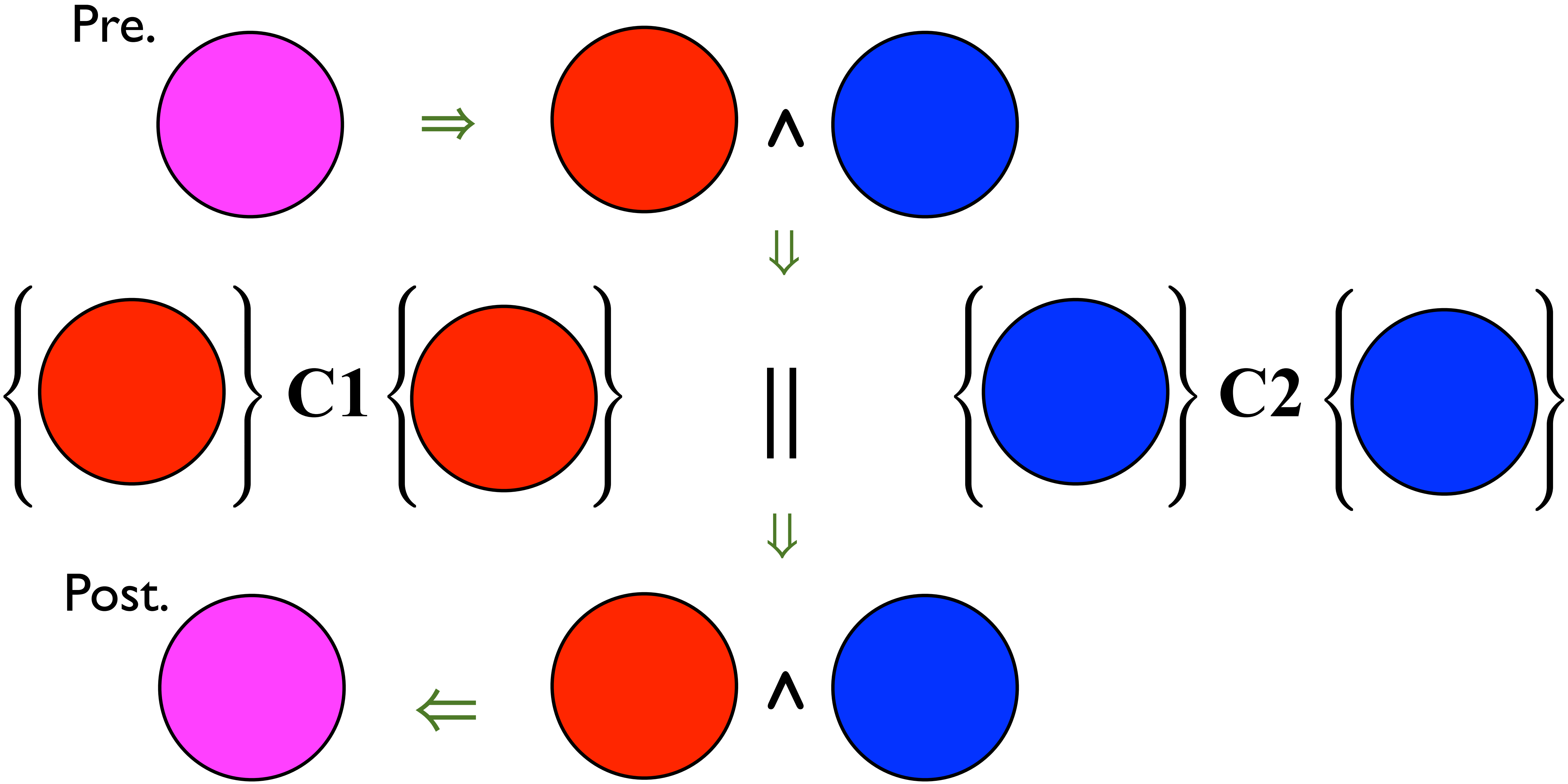
Global Shared Resources



$$\frac{\left\{\boxed{P1}\right\}\ C1\ \left\{\boxed{P1}\right\} \qquad \left\{\boxed{P2}\right\}\ C1\ \left\{\boxed{P2}\right\}}{\left\{\boxed{P1\ \wedge\ P2}\right\}\ C1 \,\|\, C2\ \left\{\boxed{P1\ \wedge\ P2}\right\}}$$

Rely-Guarantee, RGSep

Global Shared Resources



$$\frac{\{\boxed{\mathrm{P1}}\}\ \mathrm{C1}\ \{\boxed{\mathrm{P1}}\} \qquad \{\boxed{\mathrm{P2}}\}\ \mathrm{C1}\ \{\boxed{\mathrm{P2}}\}}{\{\boxed{\mathrm{P1} \wedge \mathrm{P2}}\}\ \mathrm{C1} \parallel \mathrm{C2}\ \{\boxed{\mathrm{P1} \wedge \mathrm{P2}}\}}$$

Rely-Guarantee, RGSep

Contributions

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
    - Program Logic for reasoning about *concurrent* programs

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Program Logic for reasoning about *concurrent* programs
  - Compositional reasoning achieved through (dis)entanglement mechanism

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Program Logic for reasoning about *concurrent* programs
  - Compositional reasoning achieved through (dis)entanglement mechanism
    - *Frame* irrelevant parts of the shared state to allow for more *local* reasoning

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Program Logic for reasoning about *concurrent* programs
  - Compositional reasoning achieved through (dis)entanglement mechanism
    - *Frame* irrelevant parts of the shared state to allow for more *local* reasoning
    - (de)Composition of *overlapping* resources

Contributions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Program Logic for reasoning about *concurrent* programs
  - Compositional reasoning achieved through (dis)entanglement mechanism
    - *Frame* irrelevant parts of the shared state to allow for more *local* reasoning
    - (de)Composition of *overlapping* resources
    - Threads have *different* yet *compatible* views resulting in *subjective* views of the shared state

Example - Lock Step Increment

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 100) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 100) {
  if (z < y) then
    z++;
}
```

x	y	z
0	0	0

Example - Lock Step Increment

```
while (x < 100) {     ||  while (y < 100) {     ||  while (z < 100) {
  if (x==z) then      ||    if (y < x) then     ||    if (z < y) then
    x++;              ||      y++;              ||      z++;
}                     ||  }                     ||  }
```

x	y	z
1	0	0

Example - Lock Step Increment

```
while (x < 100) {   ||  while (y < 100) {   ||  while (z < 100) {
  if (x==z) then    ||    if (y < x) then   ||    if (z < y) then
    x++;            ||      y++;            ||      z++;
}                   ||  }                   ||  }
```

x	y	z
1	1	0

Example - Lock Step Increment

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 100) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 100) {
  if (z < y) then
    z++;
}
```

x	y	z
1	1	1

Example - Lock Step Increment

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 100) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 100) {
  if (z < y) then
    z++;
}
```

x	y	z
100	100	100

Example - Lock Step Increment

```
while (x < 100) {
   if (x==z) then
      x++;
}
```

$\|$

```
while (y < 100) {
   if (y < x) then
      y++;
}
```

$\|$

```
while (z < 100) {
   if (z < y) then
      z++;
}
```

$$\boxed{\begin{array}{lllll} \exists\, v. & & & & \\ & \mathtt{x} \mapsto v & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v{+}1 & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v{+}1 & * \ \mathtt{y} \mapsto v{+}1 & * \ \mathtt{z} \mapsto v \end{array}}\ I$$

Example - Lock Step Increment

```
while (x < 100) {   ||  while (y < 100) {   ||  while (z < 100) {
   if (x==z) then   ||     if (y < x) then  ||     if (z < y) then
      x++;          ||        y++;          ||        z++;
}                   ||  }                   ||  }
```

$$\begin{array}{ll}
\boxed{\begin{array}{l}
\exists\, v. \\
\quad\ \ \mathtt{x} \mapsto v \quad\ * \ \mathtt{y} \mapsto v \quad\ * \ \mathtt{z} \mapsto v \\
\ \lor\ \mathtt{x} \mapsto v+1 \ * \ \mathtt{y} \mapsto v \quad\ * \ \mathtt{z} \mapsto v \\
\ \lor\ \mathtt{x} \mapsto v+1 \ * \ \mathtt{y} \mapsto v+1 \ * \ \mathtt{z} \mapsto v
\end{array}}
&
\begin{array}{l}
\\
\}\!\!-\!\!-\ \mathrm{S0}(v) \\
\}\!\!-\!\!-\ \mathrm{S1}(v) \\
\}\!\!-\!\!-\ \mathrm{S2}(v)
\end{array}
\\
\hfill I &
\end{array}$$

Example - Lock Step Increment

```
while (x < 100) {    ||   while (y < 100) {    ||   while (z < 100) {
  if (x==z) then     ||     if (y < x) then    ||     if (z < y) then
    x++;             ||       y++;             ||       z++;
}                    ||   }                    ||   }
```

$$\boxed{\begin{array}{lllll} \exists\, v. & & & & \\ & \mathtt{x} \mapsto v & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v & \} \text{—}\ \mathrm{S0}(v) \\ \vee & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v & \} \text{—}\ \mathrm{S1}(v) \\ \vee & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v+1 & * \ \mathtt{z} \mapsto v & \} \text{—}\ \mathrm{S2}(v) \end{array}}_{I}$$

$$I = \left\{ \begin{array}{llll} \mathrm{X} : & \mathrm{S0}(v) & \rightsquigarrow & \mathrm{S1}(v) \\ \mathrm{Y} : & \mathrm{S1}(v) & \rightsquigarrow & \mathrm{S2}(v) \\ \mathrm{Z} : & \mathrm{S2}(v) & \rightsquigarrow & \mathrm{S0}(v+1) \end{array} \right.$$

Example - Lock Step Increment

`T1` =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\underset{I}{\boxed{\begin{array}{llll} \exists v. & & & \\ & \mathtt{x} \mapsto v & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \lor & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \lor & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v+1 & * \ \mathtt{z} \mapsto v \end{array}}} \begin{array}{l} \\ \}\!\!-\!\!- \ \mathrm{S0}(v) \\ \}\!\!-\!\!- \ \mathrm{S1}(v) \\ \}\!\!-\!\!- \ \mathrm{S2}(v) \end{array}$$

$$I = \left\{ \begin{array}{l} \mathrm{X} : \mathrm{S0}(v) \rightsquigarrow \mathrm{S1}(v) \\ \mathrm{Y} : \mathrm{S1}(v) \rightsquigarrow \mathrm{S2}(v) \\ \mathrm{Z} : \mathrm{S2}(v) \rightsquigarrow \mathrm{S0}(v+1) \end{array} \right.$$

Example - Lock Step Increment

T1 =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\begin{array}{l|l} \exists v. & \\ \quad\ \ \mathtt{x} \mapsto v \ * \ \mathtt{y} \mapsto v \ * \ \mathtt{z} \mapsto v & \}\!\!- \mathsf{S0}(v) \\ \lor \ \mathtt{x} \mapsto v+1 \ * \ \mathtt{y} \mapsto v \ * \ \mathtt{z} \mapsto v & \}\!\!- \mathsf{S1}(v) \\ \lor \ \mathtt{x} \mapsto v+1 \ * \ \mathtt{y} \mapsto v+1 \ * \ \mathtt{z} \mapsto v & \}\!\!- \mathsf{S2}(v) \end{array}$$

I

$$I = \left\{ \begin{array}{l} \mathsf{X} : \mathsf{S0}(v) \rightsquigarrow \mathsf{S1}(v) \\ \mathsf{Y} : \mathsf{S1}(v) \rightsquigarrow \mathsf{S2}(v) \\ \mathsf{Z} : \mathsf{S2}(v) \rightsquigarrow \mathsf{S0}(v+1) \end{array} \right.$$

Example - Lock Step Increment

T1 =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\underset{I}{\boxed{\begin{array}{llll} \exists v. & & & \\ & \mathtt{x} \mapsto v & * \; \mathtt{y} \mapsto v & * \; \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v{+}1 & * \; \mathtt{y} \mapsto v & * \; \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v{+}1 & * \; \mathtt{y} \mapsto v{+}1 & * \; \mathtt{z} \mapsto v \end{array}}} \begin{array}{l} \\ \}\!\!- S0(v) \\ \}\!\!- S1(v) \\ \}\!\!- S2(v) \end{array}$$

$$I = \left\{ \begin{array}{l} X : S0(v) \rightsquigarrow S1(v) \\ Y : S1(v) \rightsquigarrow S2(v) \\ Z : S2(v) \rightsquigarrow S0(v+1) \end{array} \right.$$

Example - Lock Step Increment

`T1` =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\boxed{\begin{array}{llll} \exists v. & & & \\ & \mathtt{x} \mapsto v & * & \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v+1 & * & \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v+1 & * & \mathtt{z} \mapsto v \end{array}}\ I_1$$

Example - Lock Step Increment

T1 =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\boxed{\begin{array}{lllll} \exists v. & & & & \\ & \mathtt{x} \mapsto v & * & \mathtt{z} \mapsto v & \\ \lor & \mathtt{x} \mapsto v+1 & * & \mathtt{z} \mapsto v & \\ \lor & \mathtt{x} \mapsto v+1 & * & \mathtt{z} \mapsto v & \end{array}} \; I_1$$

Example - Lock Step Increment

`T1` =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\boxed{\begin{array}{l} \exists v. \\ \qquad \mathtt{x} \mapsto v \quad * \ \mathtt{z} \mapsto v \\ \ \ \lor \ \mathtt{x} \mapsto v+1 * \ \mathtt{z} \mapsto v \end{array}}_{\,I_1}$$

Example - Lock Step Increment

T1 =

```
while (x < 100) {
  if (x==z) then
    x++;
}
```

$$\boxed{\begin{array}{l} \exists\, v. \\ \quad\;\; \mathtt{x} \mapsto v \quad\;\; * \; \mathtt{z} \mapsto v \\ \quad \vee\; \mathtt{x} \mapsto v+1 \; * \; \mathtt{z} \mapsto v \end{array}}\; I_1$$

$$I_1 = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v \quad\;\; * \; \mathtt{z} \mapsto v \;\rightsquigarrow\; \mathtt{x} \mapsto v+1 \; * \; \mathtt{z} \mapsto v \\ \mathsf{Z} : \mathtt{x} \mapsto v+1 \; * \; \mathtt{z} \mapsto v \;\rightsquigarrow\; \mathtt{x} \mapsto v+1 \; * \; \mathtt{z} \mapsto v+1 \end{cases}$$

Example - Lock Step Increment

$$[\mathsf{X}] * [\mathsf{Y}] * [\mathsf{Z}] * \boxed{\begin{array}{lllll} \exists v. & & & & \\ & \mathtt{x} \mapsto v & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v & * \ \mathtt{z} \mapsto v \\ \vee & \mathtt{x} \mapsto v+1 & * \ \mathtt{y} \mapsto v+1 & * \ \mathtt{z} \mapsto v \end{array}}_{I}$$

Example - Lock Step Increment

$$[\mathsf{X}] * [\mathsf{Y}] * [\mathsf{Z}] * \boxed{\begin{array}{l} \exists\, v. \\ \quad \mathtt{x} \mapsto v \quad\; * \; \mathtt{y} \mapsto v \quad\; * \; \mathtt{z} \mapsto v \\ \vee\; \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \quad\; * \; \mathtt{z} \mapsto v \\ \vee\; \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I}$$

// Disentangle the resources

$$[\mathsf{X}] * \boxed{\begin{array}{l} \exists\, v. \\ \quad \mathtt{x} \mapsto v \quad\; * \; \mathtt{z} \mapsto v \\ \vee\; \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_1} \qquad [\mathsf{Y}] * \boxed{\begin{array}{l} \exists\, v. \\ \quad \mathtt{x} \mapsto v \quad\; * \; \mathtt{y} \mapsto v \\ \vee\; \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \end{array}}_{I_2} \qquad [\mathsf{Z}] * \boxed{\begin{array}{l} \exists\, v. \\ \quad \mathtt{y} \mapsto v \quad\; * \; \mathtt{z} \mapsto v \\ \vee\; \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_3}$$

Example - Lock Step Increment

$$[X] * [Y] * [Z] * \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \ \mathtt{y} \mapsto v \quad * \ \mathtt{z} \mapsto v \\ \lor \ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \quad * \ \mathtt{z} \mapsto v \\ \lor \ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I}$$

// Disentangle the resources

$$[X]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \ \mathtt{z} \mapsto v \\ \lor \ \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_1}$$

`T1`

$$[Y]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \ \mathtt{y} \mapsto v \\ \lor \ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \end{array}}_{I_2}$$

`T2`

$$[Z]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{y} \mapsto v \quad * \ \mathtt{z} \mapsto v \\ \lor \ \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_3}$$

`T3`

Example - Lock Step Increment

$$[X] * [Y] * [Z] * \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \; \mathtt{y} \mapsto v \quad * \; \mathtt{z} \mapsto v \\ \vee \; \mathtt{x} \mapsto v+1 * \mathtt{y} \mapsto v \quad * \; \mathtt{z} \mapsto v \\ \vee \; \mathtt{x} \mapsto v+1 * \mathtt{y} \mapsto v+1 * \mathtt{z} \mapsto v \end{array}}_{I}$$

// Disentangle the resources

$$[X]*\boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \; \mathtt{z} \mapsto v \\ \vee \; \mathtt{x} \mapsto v+1 * \mathtt{z} \mapsto v \end{array}}_{I_1} \quad [Y]*\boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad * \; \mathtt{y} \mapsto v \\ \vee \; \mathtt{x} \mapsto v+1 * \mathtt{y} \mapsto v \end{array}}_{I_2} \quad [Z]*\boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{y} \mapsto v \quad * \; \mathtt{z} \mapsto v \\ \vee \; \mathtt{y} \mapsto v+1 * \mathtt{z} \mapsto v \end{array}}_{I_3}$$

T1 **T2** **T3**

$$[X]*\boxed{\begin{array}{l} \quad \mathtt{x} \mapsto 100 * \mathtt{z} \mapsto 100 \\ \vee \; \mathtt{x} \mapsto 100 * \mathtt{z} \mapsto 99 \end{array}}_{I_1} \quad [Y]*\boxed{\begin{array}{l} \quad \mathtt{x} \mapsto 100 * \mathtt{y} \mapsto 100 \\ \vee \; \mathtt{x} \mapsto 101 * \mathtt{y} \mapsto 100 \end{array}}_{I_2} \quad [Z]*\boxed{\begin{array}{l} \quad \mathtt{y} \mapsto 100 * \mathtt{z} \mapsto 100 \\ \vee \; \mathtt{y} \mapsto 101 * \mathtt{z} \mapsto 100 \end{array}}_{I_3}$$

Example - Lock Step Increment

$$[\mathrm{X}] * [\mathrm{Y}] * [\mathrm{Z}] * \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad\;\; * \ \mathtt{y} \mapsto v \quad\;\; * \ \mathtt{z} \mapsto v \\ \vee\ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \quad\;\; * \ \mathtt{z} \mapsto v \\ \vee\ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I}$$

// Disentangle the resources

$$[\mathrm{X}]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad\;\; * \ \mathtt{z} \mapsto v \\ \vee\ \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_1}$$

`T1`

$$[\mathrm{X}]* \boxed{\begin{array}{l} \quad \mathtt{x} \mapsto 100 * \mathtt{z} \mapsto 100 \\ \vee\ \mathtt{x} \mapsto 100 * \mathtt{z} \mapsto 99 \end{array}}_{I_1}$$

$$[\mathrm{Y}]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{x} \mapsto v \quad\;\; * \ \mathtt{y} \mapsto v \\ \vee\ \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \end{array}}_{I_2}$$

`T2`

$$[\mathrm{Y}]* \boxed{\begin{array}{l} \quad \mathtt{x} \mapsto 100 * \mathtt{y} \mapsto 100 \\ \vee\ \mathtt{x} \mapsto 101 * \mathtt{y} \mapsto 100 \end{array}}_{I_2}$$

$$[\mathrm{Z}]* \boxed{\begin{array}{l} \exists v. \\ \quad \mathtt{y} \mapsto v \quad\;\; * \ \mathtt{z} \mapsto v \\ \vee\ \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \end{array}}_{I_3}$$

`T3`

$$[\mathrm{Z}]* \boxed{\begin{array}{l} \quad \mathtt{y} \mapsto 100 * \mathtt{z} \mapsto 100 \\ \vee\ \mathtt{y} \mapsto 101 * \mathtt{z} \mapsto 100 \end{array}}_{I_3}$$

// Entangle the resources

$$[\mathrm{X}] * [\mathrm{Y}] * [\mathrm{Z}] * \boxed{\mathtt{x} \mapsto 100 \ * \ \mathtt{y} \mapsto 100 \ * \ \mathtt{z} \mapsto 100}_{I}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \wedge \\ I = (I_1 , P) \bowtie (I_2, Q) \end{array} \iff \boxed{P \mathbin{\cup\!\!\!\!\!\ast} Q}_{I}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} \; * \; \boxed{Q}_{I_2} \\ \wedge \\ I = (I_1, P) \bowtie (I_2, Q) \end{array} \quad \Longleftrightarrow \quad \boxed{P \cup\!\!\!\!\!\!\ast \;\; Q}_{I}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \wedge \\ I = (I_1, P) \bowtie (I_2, Q) \end{array} \iff \boxed{P \mathbin{\cup\!\!\!\!\ast} Q}_{I}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \wedge \\ I = (I_1, P) \bowtie (I_2, Q) \end{array} \iff \boxed{P \mathbin{\cup\!\!\!\!\ast} Q}_{I}$$

$$\boxed{P \mathbin{\cup\!\!\!\!\ast} Q}_{I} \Rightarrow \boxed{P}_{I_1} * \boxed{Q}_{I_2}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \wedge \\ I = (I_1, P) \bowtie (I_2, Q) \end{array} \iff \boxed{P \mathbin{\cup\!\!\!\!\!\ast} Q}_{I}$$

$$\boxed{P \mathbin{\cup\!\!\!\!\!\ast} Q}_{I} \Rightarrow \boxed{P}_{I_1} * \boxed{Q}_{I_2}$$

$$\Downarrow$$

$$\boxed{P}_{I_1} * \boxed{Q}_{I_2} \quad \mathrm{C1} \parallel \mathrm{C2} \quad \boxed{P'}_{I_1} * \boxed{Q'}_{I_2}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \land \\ I = (I_1, P) \bowtie (I_2, Q) \end{array} \iff \boxed{P \mathbin{\cup\!\!\!\!*\,} Q}_{I}$$

$$\boxed{P \mathbin{\cup\!\!\!\!*\,} Q}_{I} \Rightarrow \boxed{P}_{I_1} * \boxed{Q}_{I_2}$$

$$\Downarrow$$

$$\boxed{P}_{I_1} * \boxed{Q}_{I_2} \quad \mathrm{C1} \parallel \mathrm{C2} \quad \boxed{P'}_{I_1} * \boxed{Q'}_{I_2}$$

$$\Downarrow$$

$$\boxed{P'}_{I_1} * \boxed{Q'}_{I_2}$$

State (Dis)entanglement

$$\begin{array}{c} \boxed{P}_{I_1} * \boxed{Q}_{I_2} \\ \wedge \\ I = (\, I_1 ,\, P\,) \bowtie (\, I_2 ,\, Q\,) \end{array} \iff \boxed{P \mathbin{\cup\mkern-12mu*} Q}_{I}$$

$$\boxed{P \mathbin{\cup\mkern-12mu*} Q}_{I} \quad \Rightarrow \quad \boxed{P}_{I_1} * \boxed{Q}_{I_2}$$

$$\Downarrow$$

$$\boxed{P}_{I_1} * \boxed{Q}_{I_2} \quad \mathrm{C1} \parallel \mathrm{C2} \quad \boxed{P'}_{I_1} * \boxed{Q'}_{I_2}$$

$$\Downarrow$$

$$\boxed{P'}_{I_1} * \boxed{Q'}_{I_2} \quad \Rightarrow \quad \boxed{P' \mathbin{\cup\mkern-12mu*} Q'}_{I}$$

Interference (Dis)entanglement

$$I = (I_1, P) \bowtie (I_2, Q) \quad \Longleftrightarrow \quad \begin{array}{c} I = I_1 \cup I_2 \\ \wedge \\ I_1 \rhd (I_2, Q) \\ \wedge \\ I_2 \rhd (I_1, P) \end{array}$$

Interference (Dis)entanglement

$$I = (I_1, P) \bowtie (I_2, Q) \iff \begin{array}{c} I = I_1 \cup I_2 \\ \wedge \\ I_1 \rhd (I_2, Q) \\ \wedge \\ I_2 \rhd (I_1, P) \end{array}$$

$I_2 \rhd (I_1, P)$:

Interference (Dis)entanglement

$$I = (I_1, P) \bowtie (I_2, Q) \iff \begin{array}{c} I = I_1 \cup I_2 \\ \wedge \\ I_1 \triangleright (I_2, Q) \\ \wedge \\ I_2 \triangleright (I_1, P) \end{array}$$

$I_2 \triangleright (I_1, P)$: Given action A in I_2 and unknown to I_1:

A does not mutate P

or future states reachable from P.

Interference (Dis)entanglement

$$I = (I_1, P) \bowtie (I_2, Q) \iff \begin{array}{c} I = I_1 \cup I_2 \\ \wedge \\ I_1 \triangleright (I_2, Q) \\ \wedge \\ I_2 \triangleright (I_1, P) \end{array}$$

$I_2 \triangleright (I_1, P)$: Given action A in I_2 and unknown to I_1:

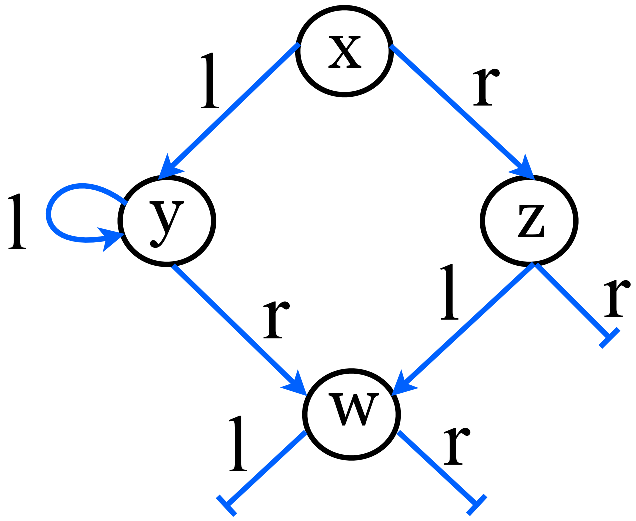
A does not mutate P

or future states reachable from P.

Given action A in both I_1 and I_2:

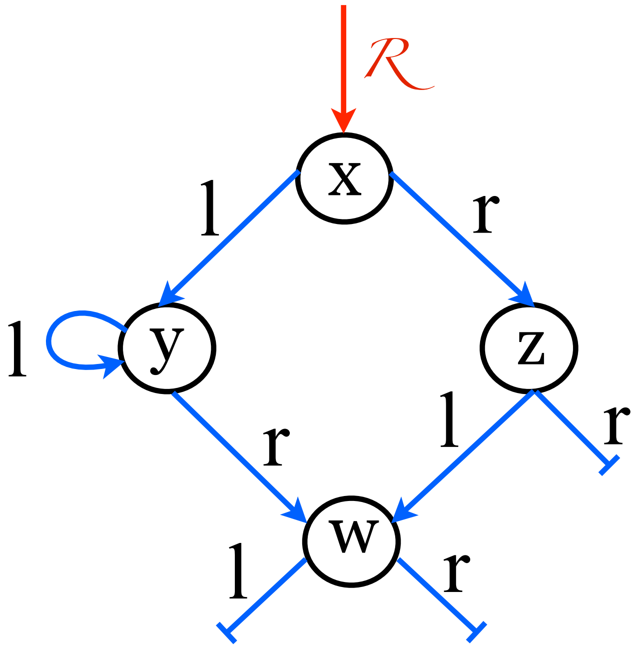
the *effect* of A is the same in I_1 and I_2.

Example - Spanning Tree



$$\text{graph}(x) = x \doteq \texttt{null} \;\vee$$
$$\exists\, m, l, r.\;\; x \mapsto m, l, r \;\; \cup\!\!\!\!\ast \;\; \text{graph}(l) \;\; \cup\!\!\!\!\ast \;\; \text{graph}(r)$$

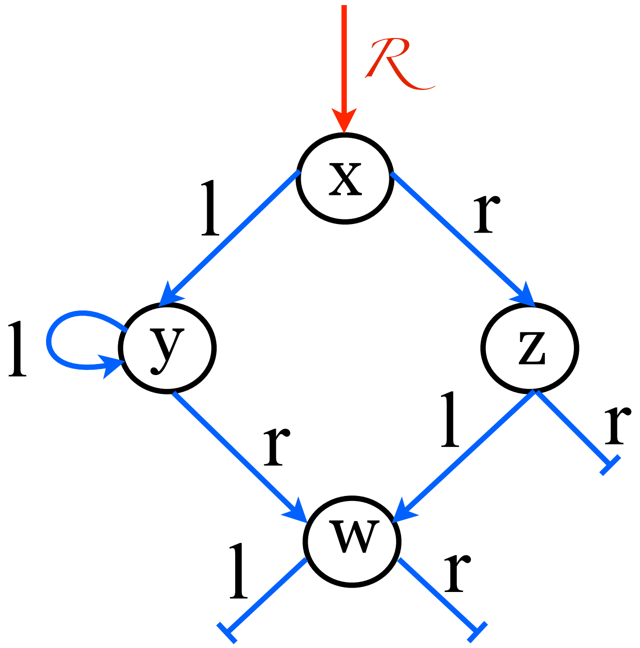
Example - Spanning Tree



$$\text{graph}(x) = x \doteq \texttt{null} \;\vee$$
$$\exists\, m, l, r.\;\; x \mapsto m, l, r \;\, \cup\!\!\!\!\!{\scriptstyle *}\;\, \text{graph}(l) \;\, \cup\!\!\!\!\!{\scriptstyle *}\;\, \text{graph}(r)$$

$$\text{graph}(x) = G(x, \mathcal{R})$$

Example - Spanning Tree

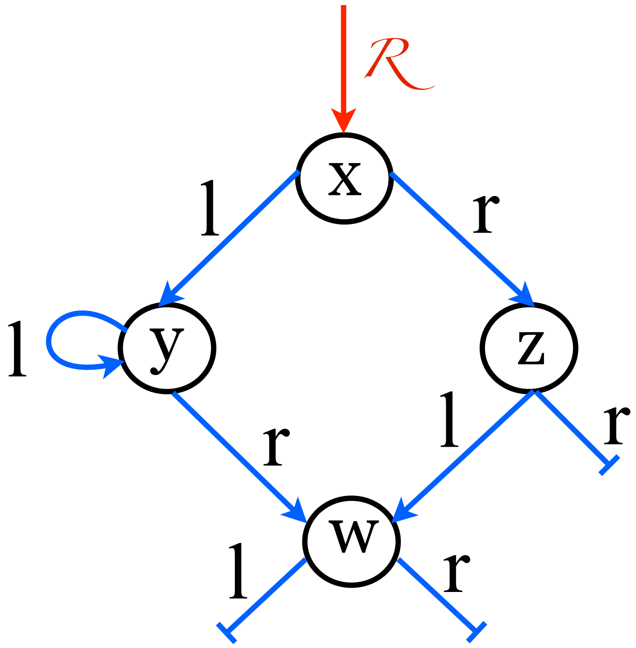


$$\mathrm{graph}(x) = x \doteq \texttt{null} \lor$$
$$\exists\, m, l, r.\ \ x \mapsto m, l, r \ \uplus\!\!\!\!\ast\ \mathrm{graph}(l) \ \uplus\!\!\!\!\ast\ \mathrm{graph}(r)$$

$$\mathrm{graph}(x) = G(x, \mathcal{R})$$

$$G(x, p) = [P(x\ , p)] * \big(x \doteq \texttt{null} \ \lor \exists\, l, r.\ \boxed{U(x, l, r) \ \lor\ M(x)}_{I(x)}\ \big)$$

Example - Spanning Tree



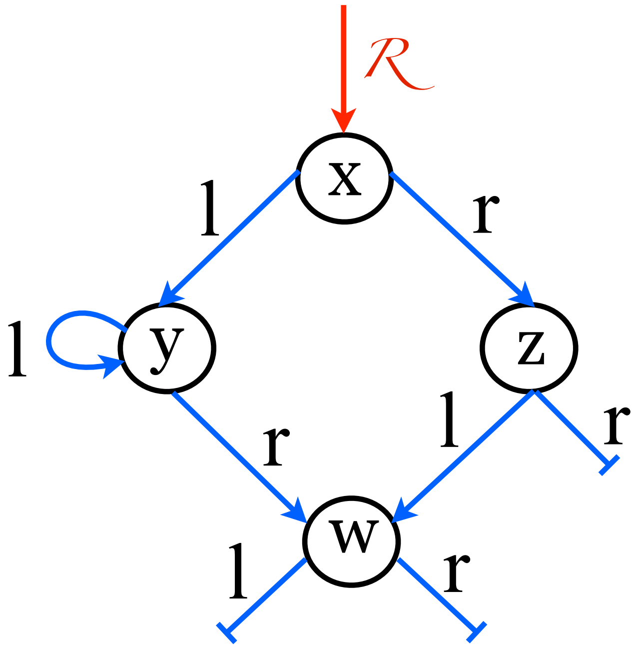
$$\mathrm{graph}(x) = x \doteq \texttt{null} \lor$$
$$\exists\, m, l, r.\;\; x \mapsto m, l, r \;\cup\!\!\!\!\ast\; \mathrm{graph}(l) \;\cup\!\!\!\!\ast\; \mathrm{graph}(r)$$

$$\mathrm{graph}(x) = \mathsf{G}(x, \mathcal{R})$$

$$\mathsf{G}(x, p) = [\mathsf{P}(x\,, p)] * \Big(x \doteq \texttt{null} \;\lor \exists\, l, r.\; \boxed{\mathsf{U}(x, l, r) \;\lor\; \mathsf{M}(x)}\;\Big)_{I(x)}$$

$$\mathsf{U}(x, l, r) = x \mapsto 0, l, r * \mathsf{G}(l, x.l) * \mathsf{G}(r, x.r)$$

Example - Spanning Tree



$$\mathsf{graph}(x) = x \doteq \mathtt{null} \;\vee\; \exists\, m, l, r.\;\; x \mapsto m, l, r \;\mathbin{\cup\!\!\!\!\ast}\; \mathsf{graph}(l) \;\mathbin{\cup\!\!\!\!\ast}\; \mathsf{graph}(r)$$

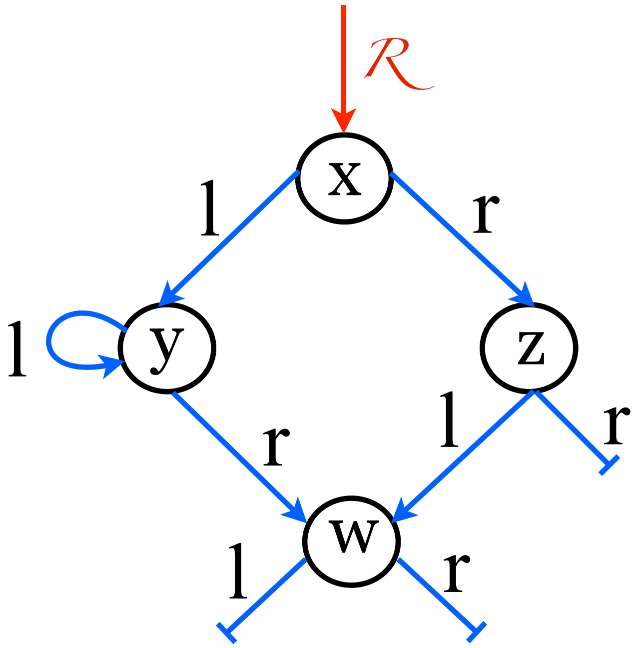
$$\mathsf{graph}(x) = \mathsf{G}(x, \mathcal{R})$$

$$\mathsf{G}(x, p) = [\mathsf{P}(x\,, p)] * \Big(x \doteq \mathtt{null} \;\vee\; \exists\, l, r.\; \boxed{\mathsf{U}(x, l, r) \;\vee\; \mathsf{M}(x)}_{I(x)}\;\Big)$$

$$\mathsf{U}(x, l, r) = x \mapsto 0, l, r * \mathsf{G}(l, x.l) * \mathsf{G}(r, x.r)$$

$$\mathsf{M}(x) = x \mapsto 1$$

Example - Spanning Tree



$$\mathrm{graph}(x) = x \doteq \texttt{null} \;\vee$$
$$\exists\, m, l, r.\;\; x \mapsto m, l, r \;\mathbin{\cup\!\!\!\!*}\; \mathrm{graph}(l) \;\mathbin{\cup\!\!\!\!*}\; \mathrm{graph}(r)$$

$$\mathrm{graph}(x) = \mathsf{G}(x, \mathcal{R})$$

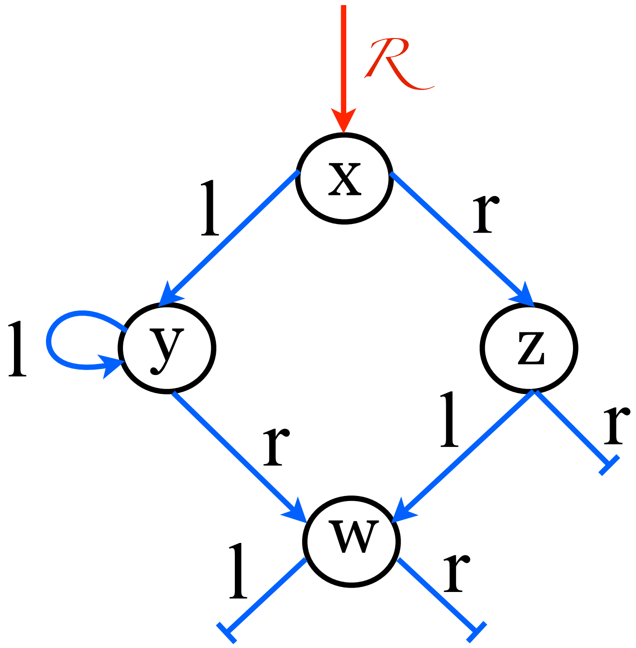
$$\mathsf{G}(x, p) = [\mathsf{P}(x\,, p)] * \Big(x \doteq \texttt{null} \;\vee \exists\, l, r.\; \underset{I(x)}{\boxed{\mathsf{U}(x, l, r) \;\vee\; \mathsf{M}(x)}}\;\Big)$$

$$\mathsf{U}(x, l, r) = x \mapsto 0, l, r * \mathsf{G}(l, x.l) * \mathsf{G}(r, x.r)$$

$$\mathsf{M}(x) = x \mapsto 1$$

$$I\,(x) = \Big\{\; \mathsf{P}(x\,, p) :\; \mathsf{U}(x, l, r) \;\rightsquigarrow\; \mathsf{M}(x)$$

Example - Spanning Tree



$$\mathrm{graph}(x) = x \doteq \texttt{null} \;\vee\; \exists\, m, l, r.\;\; x \mapsto m, l, r \;\; \cup\!\!\!\!\!\ast \;\; \mathrm{graph}(l) \;\; \cup\!\!\!\!\!\ast \;\; \mathrm{graph}(r)$$

$$\mathrm{graph}(x) = \mathsf{G}(x, \mathcal{R})$$

$$\mathsf{G}(x, p) = [\mathsf{P}(x\,, p)] * \Big(x \doteq \texttt{null} \;\vee\; \exists\, l, r.\; \boxed{\mathsf{U}(x, l, r) \;\vee\; \mathsf{M}(x)}_{I(x)}\Big)$$

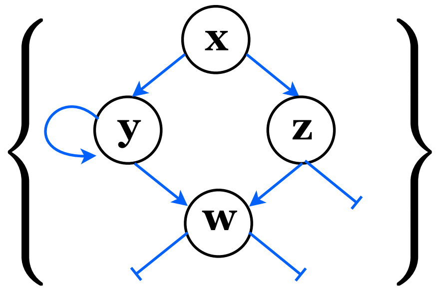
$$\mathsf{U}(x, l, r) = x \mapsto 0, l, r * \mathsf{G}(l, x.l) * \mathsf{G}(r, x.r)$$

$$\mathsf{M}(x) = x \mapsto 1$$

$$I\,(x) = \big\{\; \mathsf{P}(x\,, p) : \mathsf{U}(x, l, r) \;\rightsquigarrow\; \mathsf{M}(x)$$

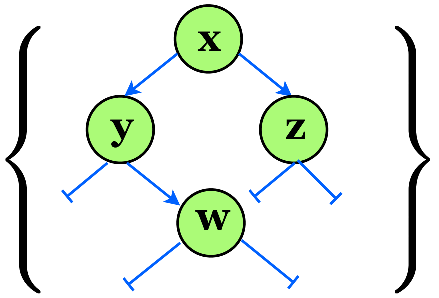
$$\mathsf{T}(x, p) = [\mathsf{P}(x\,, p)] * \left(\begin{array}{l} x \doteq \texttt{null} \;\vee \\ \exists\, l, r.\;\; x.l \mapsto l * x.r \mapsto r * \boxed{\mathsf{M}(x)}_{I(x)} \\ \qquad\quad * \; \mathsf{T}(l, x.l) * \mathsf{T}(r, x.r) \end{array}\right)$$

Example - Spanning Tree



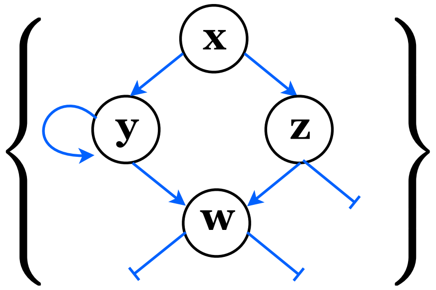
$= G(x, \mathcal{R})$

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```



$= T(x, \mathcal{R})$

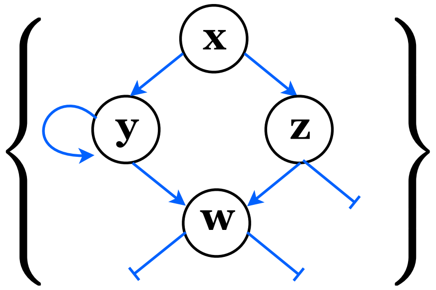
Example - Spanning Tree



$= G(x, \mathcal{R})$

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

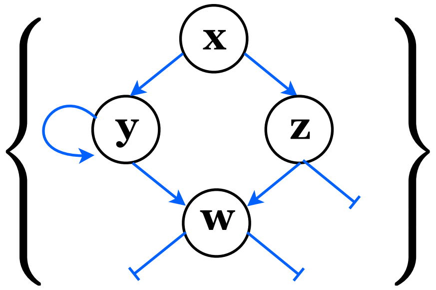


$$= [P(x\ ,\ \mathcal{R})] * \boxed{\begin{array}{l} x \mapsto 0, y, z\ * \\ G(y, x.l) * G(z, x.r) \end{array}}$$

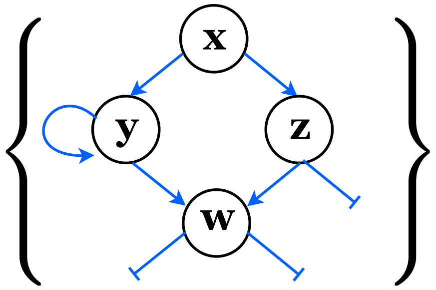
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {

    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



```
b:= spanning(x) {
```

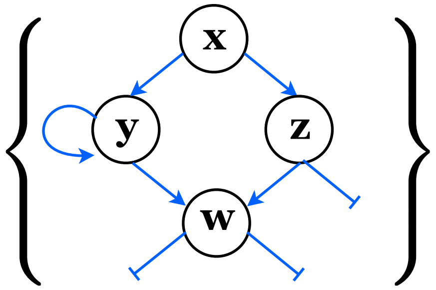


$= [P(x\ ,\ \mathcal{R})]\ *\ \boxed{\begin{array}{l} x \mapsto 0,\ y,\ z\ * \\ G(y,\ x.l) * G(z,\ x.r) \end{array}}$

```
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

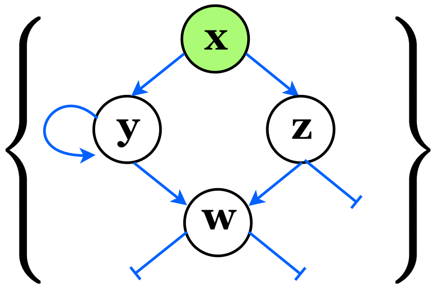
Example - Spanning Tree

```
b:= spanning(x) {
```



$$= [P(x, \mathcal{R})] * \boxed{\begin{array}{l} x \mapsto 0, y, z * \\ G(y, x.l) * G(z, x.r) \end{array}}$$

```
  b:= <CAS(x.m, 0, 1)>;
```



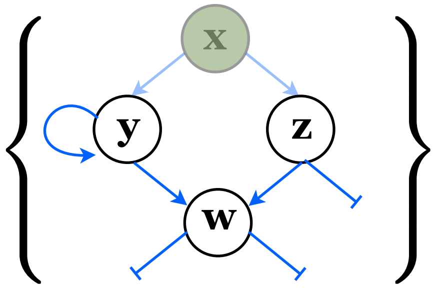
$$= [P(x, \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$$

$$x.l \mapsto y * x.r \mapsto z$$

```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

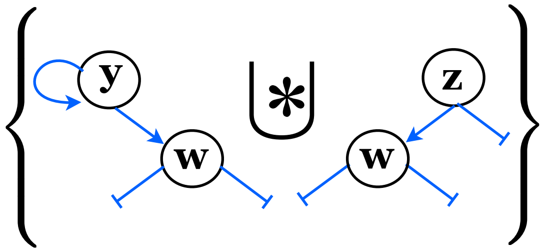
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$= [P(x , \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$

$x.l \mapsto y * x.r \mapsto z$

```
  if (b) then {
```

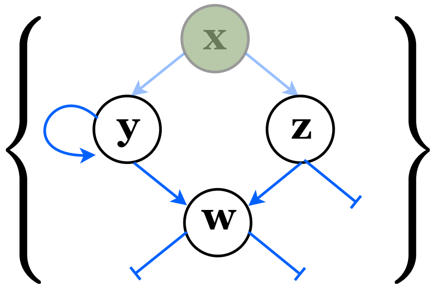


$= G(y, x.l) * G(z, x.r)$

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

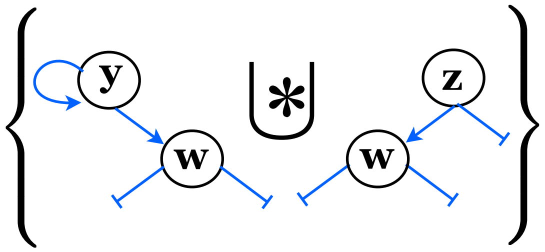
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$= [P(x\,, \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$

$x.l \mapsto y * x.r \mapsto z$

```
  if (b) then {
```

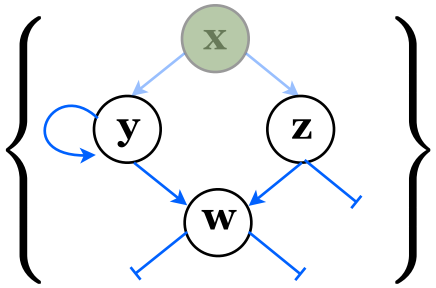


$= \begin{array}{l}[P(y, x.l)] * \\ \boxed{\begin{array}{l} y \mapsto 0, y, w * \\ G(y, y.l) * G(w, y.r) \end{array}}\end{array} * \begin{array}{l}[P(z, x.r)] * \\ \boxed{\begin{array}{l} z \mapsto 0, w, null \\ * G(w, z.l) \end{array}}\end{array}$

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

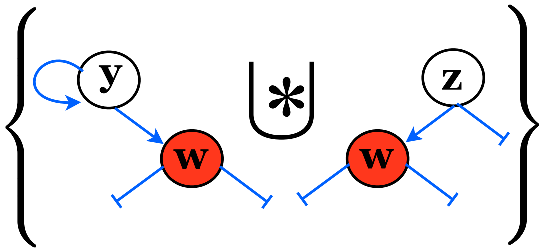
Example - Spanning Tree

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$$= [P(x\,, \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$$
$$x.l \mapsto y * x.r \mapsto z$$

```
  if (b) then {
```

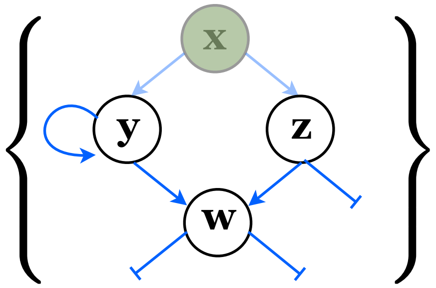


$$= [P(y, x.l)] * \boxed{\begin{array}{l} y \mapsto 0, y, w * \\ G(y, y.l) * G(w, y.r) \end{array}} * [P(z, x.r)] * \boxed{\begin{array}{l} z \mapsto 0, w, null \\ * \; G(w, z.l) \end{array}}$$

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

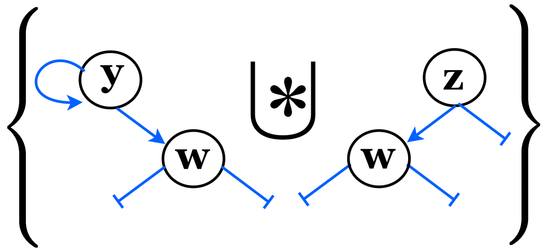
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$= [P(x\ ,\ \mathcal{R})] * \ G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$

$x.l \mapsto y * x.r \mapsto z$

```
  if (b) then {
```

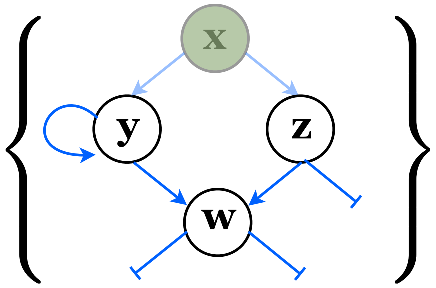


$= G(y, x.l) * G(z, x.r)$

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

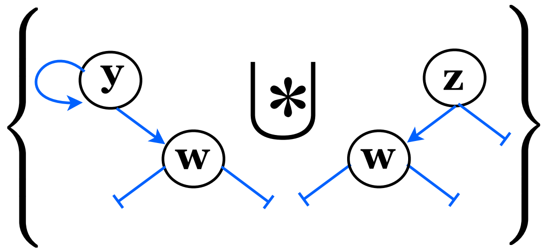
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$= [P(x , \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$

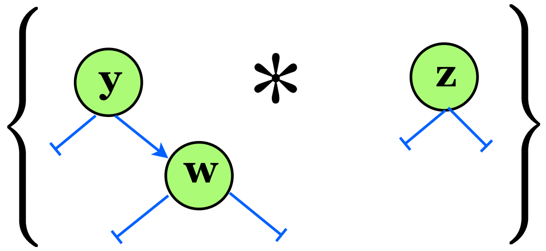
$x.l \mapsto y * x.r \mapsto z$

```
  if (b) then {
```



$= G(y, x.l) * G(z, x.r)$

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
```

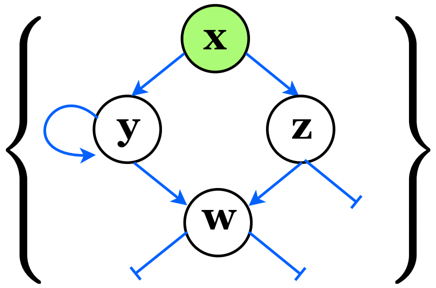


$= T(y, x.l) * T(z, x.r)$

```
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

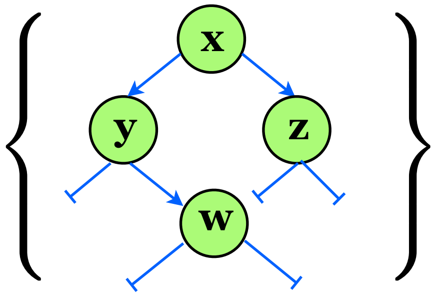
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```



$$= [P(x , \mathcal{R})] * G(y, x.l) * G(z, x.r) * \boxed{x.m \mapsto 1}$$

$$x.l \mapsto y * x.r \mapsto z$$

```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
```



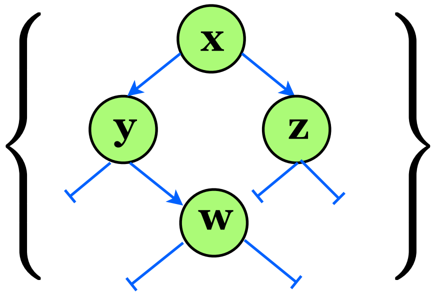
$$= [P(x , \mathcal{R})] * T(y, x.l) * T(z, x.r) \quad \boxed{x.m \mapsto 1}$$

$$x.l \mapsto y * x.r \mapsto z$$

```
  return b;
}
```

Example - Spanning Tree

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
```

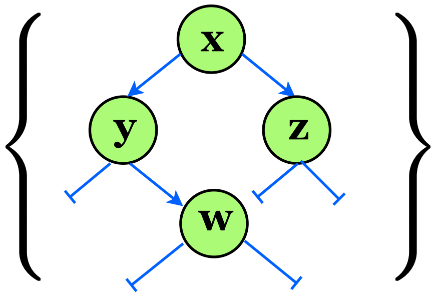


$= \mathrm{T}(x , \mathcal{R})$

```
  return b;
}
```

Example - Spanning Tree

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```



$= T(x, \mathcal{R})$

Conclusions

Conclusions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic

Conclusions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Reasoning about *concurrent* programs with *overlapping* footprints

Conclusions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Reasoning about *concurrent* programs with *overlapping* footprints
  - (dis)Entanglement allows for more *local* reasoning by *forgetting* (framing) irrelevant parts of the shared state and associated actions

Conclusions

- CoLoSL: **Co**ncurrent **Lo**cal **S**ubjective **L**ogic
  - Reasoning about concurrent programs with *overlapping* footprints
  - (dis)Entanglement allows for more *local* reasoning by *forgetting* (framing) irrelevant parts of the shared state and associated actions
  - Framing by disentanglement results in *different* yet *compatible* views giving way to *subjective* views of the shared state

Questions?

Thank you for listening