CoLoSL
Compositional Reasoning At Last!

Azalea Raad Jules Villard Philippa Gardner

Imperial College London

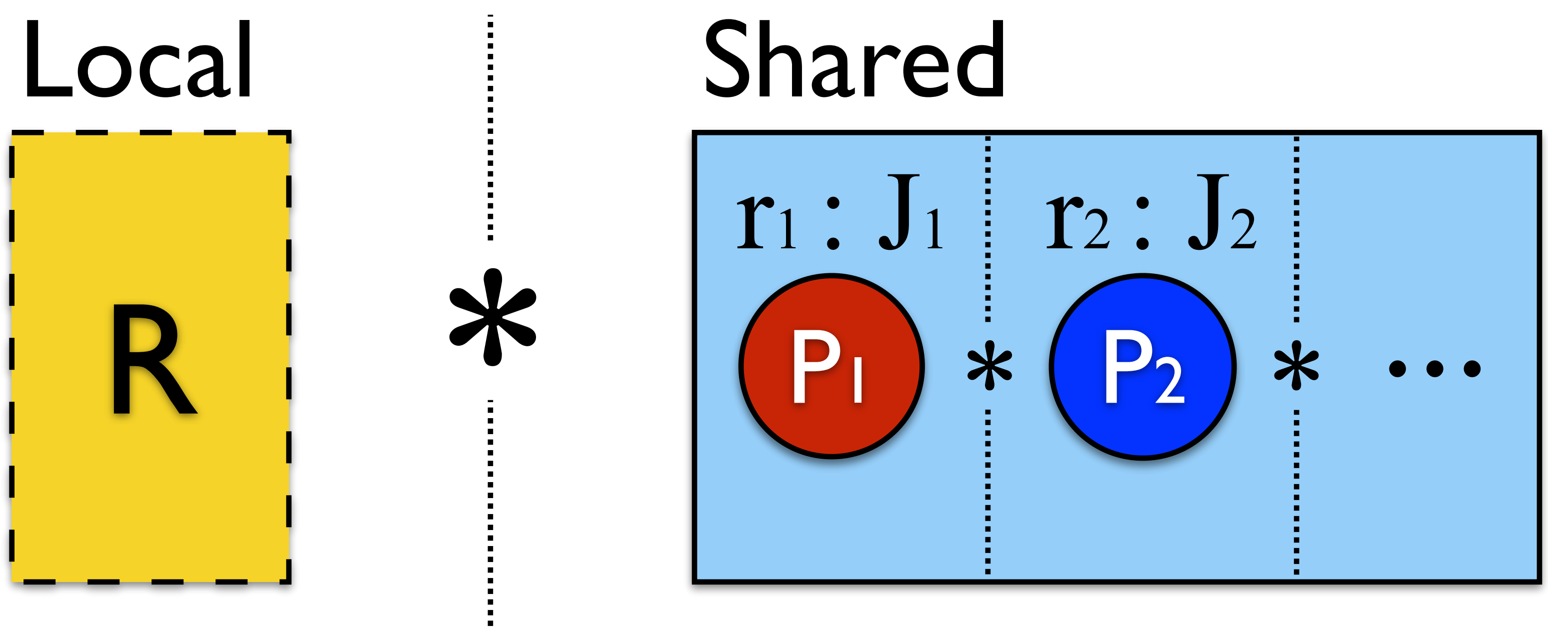
INVEST Workshop
29 November 2014

Owicki-Gries / Rely-Guarantee

$P_1 \wedge P_2$

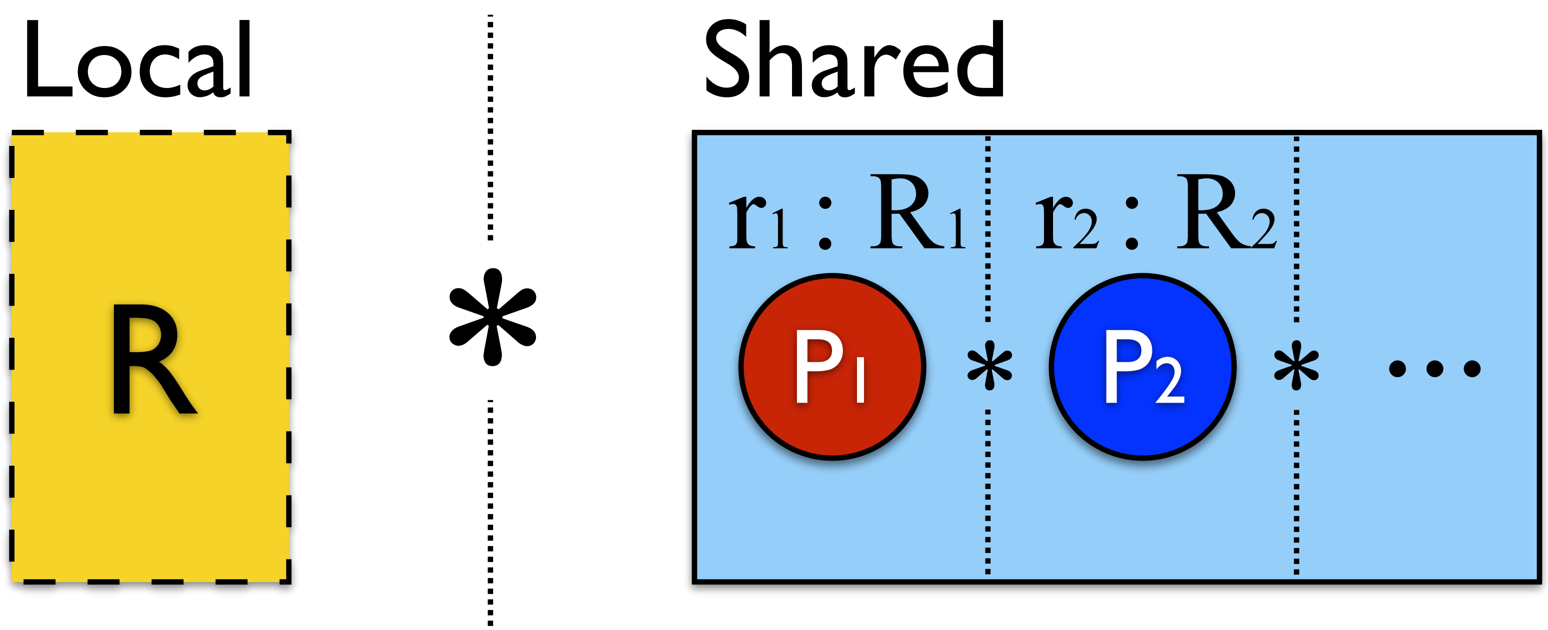
$$\frac{\{P1\}\ C1\ \{Q1\} \qquad \{P2\}\ C2\ \{Q2\}}{\{P1 \wedge P2\}\ C1 \parallel C2\ \{Q1 \wedge Q2\}}$$

Concurrent Separation Logic



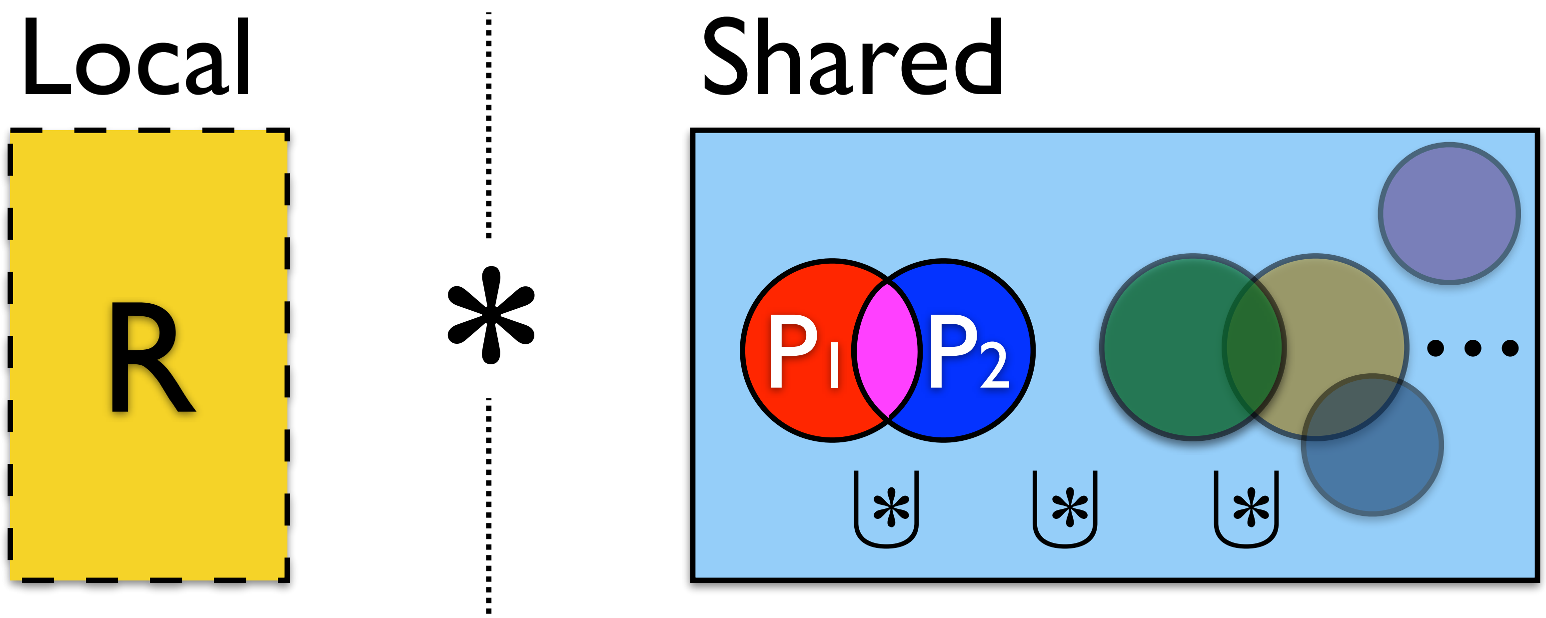
$$\frac{r_1{:}J_1, r_2{:}J_2, \ldots \vdash \{P1\}\ C1\ \{Q1\} \qquad r_1{:}J_1, r_2{:}J_2, \ldots \vdash \{P2\}\ C2\ \{Q2\}}{r_1{:}J_1, r_2{:}J_2, \ldots \vdash \{P1 * P2\}\ C1 \parallel C2\ \{Q1 * Q2\}}$$

From CAP to TaDA



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

CoLoSL: Concurrent Local Subjective Logic



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

Example - Dijkstra’s Self-stabilising Ring

```
while (x < 10) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 10) {
  if (z < y) then
    z++;
}
```

x	y	z
0	0	0

Example - Dijkstra's Self-stabilising Ring

```
while (x < 10) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 10) {
  if (z < y) then
    z++;
}
```

x	y	z
1	0	0

Example - Dijkstra's Self-stabilising Ring

```
while (x < 10) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 10) {
  if (z < y) then
    z++;
}
```

x	y	z
1	1	0

Example - Dijkstra's Self-stabilising Ring

```
while (x < 10) {          while (y < 10) {           while (z < 10) {
  if (x==z) then      ||    if (y < x) then      ||    if (z < y) then
    x++;                      y++;                       z++;
}                         }                          }
```

x	y	z
1	1	1

Example - Dijkstra's Self-stabilising Ring

```
while (x < 10) {
  if (x==z) then
    x++;
}
```

||

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

||

```
while (z < 10) {
  if (z < y) then
    z++;
}
```

x	y	z
10	10	10

Example - Dijkstra's Self-stabilising Ring

```
while (x < 10) {          while (y < 10) {          while (z < 10) {
   if (x==z) then     ||     if (y < x) then     ||     if (z < y) then
      x++;                      y++;                      z++;
}                         }                         }
```

$$\exists v.\;\; (\mathtt{x} \mapsto v * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \vee (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \vee (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v)$$

I

$$I = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \\ \mathsf{Y} : \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 \\ \mathsf{Z} : \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring
Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v.\; (\mathtt{x} \mapsto v * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \lor (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \lor (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v)$$

I

$$I = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \\ \mathsf{Y} : \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 \\ \mathsf{Z} : \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v{+}1 \end{cases}$$

Forgetting Resources

$$\boxed{P \ast Q}_{I} \Rightarrow \boxed{P}_{I}$$

Example - Dijkstra's Self-stabilising Ring
Localisation : Naïve Attempt

T2 =
```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists\, v.\ \ (\mathtt{x} \mapsto v * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \lor (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v * \mathtt{z} \mapsto v) \lor (\mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v)$$

I

$$I = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{z} \mapsto v \\ \mathsf{Y} : \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v{+}1 * \mathtt{y} \mapsto v{+}1 \\ \mathsf{Z} : \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{y} \mapsto v{+}1 * \mathtt{z} \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring
Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v. \quad x \mapsto v * y \mapsto v \;\lor\; x \mapsto v{+}1 * y \mapsto v \;\lor\; x \mapsto v{+}1 * y \mapsto v{+}1$$

I

$$I = \begin{cases} X : x \mapsto v * z \mapsto v \rightsquigarrow x \mapsto v{+}1 * z \mapsto v \\ Y : x \mapsto v{+}1 * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * y \mapsto v{+}1 \\ Z : y \mapsto v{+}1 * z \mapsto v \rightsquigarrow y \mapsto v{+}1 * z \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring
Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v. \quad (x \mapsto v * y \mapsto v) \lor (x \mapsto v+1 * y \mapsto v) \lor (x \mapsto v+1 * y \mapsto v+1)$$

I

$$I = \begin{cases} X : x \mapsto v * z \mapsto v \rightsquigarrow x \mapsto v+1 * z \mapsto v \\ Y : x \mapsto v+1 * y \mapsto v \rightsquigarrow x \mapsto v+1 * y \mapsto v+1 \\ Z : y \mapsto v+1 * z \mapsto v \rightsquigarrow y \mapsto v+1 * z \mapsto v+1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v. \quad \mathtt{x} \mapsto v \;*\; \mathtt{y} \mapsto v \;\vee\; \mathtt{x} \mapsto v{+}1 \;*\; \mathtt{y} \mapsto v$$

I

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

Forgetting Interference

If $\quad I \cup I' \sqsubseteq^{\mathsf{P}} I$

Then $\quad \boxed{\mathsf{P}}_{I \cup I'} \Rightarrow \boxed{\mathsf{P}}_{I}$

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists\, v.\quad \mathtt{x} \mapsto v \;*\; \mathtt{y} \mapsto v \;\;\vee\;\; \mathtt{x} \mapsto v{+}1 \;*\; \mathtt{y} \mapsto v$$

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

T2 =
```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v.\; (x \mapsto v * y \mapsto v) \vee (x \mapsto v{+}1 * y \mapsto v)$$

I_2

NOT STABLE!!

$$I_2 = \begin{cases} X : x \mapsto v * z \mapsto v \rightsquigarrow x \mapsto v{+}1 * z \mapsto v \\ Y : x \mapsto v{+}1 * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * y \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists\, v, v'.\quad \mathtt{x} \mapsto v \;*\; \mathtt{y} \mapsto v'$$

I_2

$$I_2 = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v \;*\; \mathtt{z} \mapsto v \;\rightsquigarrow\; \mathtt{x} \mapsto v{+}1 \;*\; \mathtt{z} \mapsto v \\ \mathsf{Y} : \mathtt{x} \mapsto v{+}1 \;*\; \mathtt{y} \mapsto v \;\rightsquigarrow\; \mathtt{x} \mapsto v{+}1 \;*\; \mathtt{y} \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring

Localisation

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v.\; \begin{array}{lll} & x \mapsto v & * \; y \mapsto v & * \; z \mapsto v \\ \vee & x \mapsto v{+}1 & * \; y \mapsto v & * \; z \mapsto v \\ \vee & x \mapsto v{+}1 & * \; y \mapsto v{+}1 & * \; z \mapsto v \end{array}$$

I

$$I = \left\{ \begin{array}{ll} X : x \mapsto v \;*\; z \mapsto v & \rightsquigarrow \; x \mapsto v{+}1 \;*\; z \mapsto v \\ Y : x \mapsto v{+}1 \;*\; y \mapsto v & \rightsquigarrow \; x \mapsto v{+}1 \;*\; y \mapsto v{+}1 \\ Z : y \mapsto v{+}1 \;*\; z \mapsto v & \rightsquigarrow \; y \mapsto v{+}1 \;*\; z \mapsto v{+}1 \end{array} \right.$$

Example - Dijkstra's Self-stabilising Ring

Localisation

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v. \quad x \mapsto v * y \mapsto v * z \mapsto v \;\vee\; x \mapsto v{+}1 * y \mapsto v * z \mapsto v \;\vee\; x \mapsto v{+}1 * y \mapsto v{+}1 * z \mapsto v$$

I'

$$I' = \begin{cases} X : x \mapsto v * z \mapsto v * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * z \mapsto v * y \mapsto v \\ Y : x \mapsto v{+}1 * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * y \mapsto v{+}1 \\ Z : y \mapsto v{+}1 * z \mapsto v \rightsquigarrow y \mapsto v{+}1 * z \mapsto v{+}1 \end{cases}$$

Forgetting Interference

If $I \approx^{\mathsf{P}} I'$

Then $\boxed{\mathsf{P}}_{I} \Rightarrow \boxed{\mathsf{P}}_{I'}$

Example - Dijkstra's Self-stabilising Ring
Localisation

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v.\ \ (x \mapsto v * y \mapsto v * z \mapsto v) \vee (x \mapsto v{+}1 * y \mapsto v * z \mapsto v) \vee (x \mapsto v{+}1 * y \mapsto v{+}1 * z \mapsto v)$$

I'

$$I' = \begin{cases} X : x \mapsto v * z \mapsto v * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * z \mapsto v * y \mapsto v \\ Y : x \mapsto v{+}1 * y \mapsto v \rightsquigarrow x \mapsto v{+}1 * y \mapsto v{+}1 \\ Z : y \mapsto v{+}1 * z \mapsto v \rightsquigarrow y \mapsto v{+}1 * z \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring
Localisation

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists v.\quad \mathtt{x} \mapsto v \ * \ \mathtt{y} \mapsto v \ \lor \ \mathtt{x} \mapsto v+1 \ * \ \mathtt{y} \mapsto v$$

I'

$$I' = \begin{cases} \mathsf{X} : \mathtt{x} \mapsto v * \mathtt{z} \mapsto v * \mathtt{y} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v+1 * \mathtt{z} \mapsto v * \mathtt{y} \mapsto v \\ \mathsf{Y} : \mathtt{x} \mapsto v+1 * \mathtt{y} \mapsto v \rightsquigarrow \mathtt{x} \mapsto v+1 * \mathtt{y} \mapsto v+1 \\ \mathsf{Z} : \mathtt{y} \mapsto v+1 * \mathtt{z} \mapsto v \rightsquigarrow \mathtt{y} \mapsto v+1 * \mathtt{z} \mapsto v+1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring

Localisation

T2 =

```
while (y < 10) {
  if (y < x) then
    y++;
}
```

$$\exists\, v.\quad x \mapsto v \;*\; y \mapsto v \;\vee\; x \mapsto v{+}1 \;*\; y \mapsto v$$

I_2

$$I_2 = \begin{cases} \mathsf{X} : x \mapsto v \;*\; z \mapsto v \;*\; y \mapsto v \;\rightsquigarrow\; x \mapsto v{+}1 \;*\; z \mapsto v \;*\; y \mapsto v \\ \mathsf{Y} : x \mapsto v{+}1 \;*\; y \mapsto v \;\rightsquigarrow\; x \mapsto v{+}1 \;*\; y \mapsto v{+}1 \end{cases}$$

Example - Dijkstra's Self-stabilising Ring

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

// Localise the resources

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

T1 || T2 || T3

$$X * \boxed{\begin{array}{l} \quad x \mapsto 10 * z \mapsto 10 \\ \vee \; x \mapsto 10 * z \mapsto 9 \end{array}}_{I_1} \qquad Y * \boxed{\begin{array}{l} \quad x \mapsto 10 * y \mapsto 10 \\ \vee \; x \mapsto 11 * y \mapsto 10 \end{array}}_{I_2} \qquad Z * \boxed{\begin{array}{l} \quad y \mapsto 10 * z \mapsto 10 \\ \vee \; y \mapsto 11 * z \mapsto 10 \end{array}}_{I_3}$$

// ????

$$X * Y * Z * \boxed{x \mapsto 10 * y \mapsto 10 * z \mapsto 10}_{I}$$

Merging Resources

$$\boxed{P}_{I_1} * \boxed{Q}_{I_2} \Rightarrow \boxed{P \uplus\!\!\!* \; Q}_{I_1 \cup I_2}$$

Example - Dijkstra's Self-stabilising Ring

$$X * Y * Z * \boxed{\exists v.\; (x \mapsto v * y \mapsto v * z \mapsto v) \lor (x \mapsto v{+}1 * y \mapsto v * z \mapsto v) \lor (x \mapsto v{+}1 * y \mapsto v{+}1 * z \mapsto v)}_{I}$$

// Localise the resources

$$X * \boxed{\exists v.\; (x \mapsto v * z \mapsto v) \lor (x \mapsto v{+}1 * z \mapsto v)}_{I_1} \quad \| \quad Y * \boxed{\exists v.\; (x \mapsto v * y \mapsto v) \lor (x \mapsto v{+}1 * y \mapsto v)}_{I_2} \quad \| \quad Z * \boxed{\exists v.\; (y \mapsto v * z \mapsto v) \lor (y \mapsto v{+}1 * z \mapsto v)}_{I_3}$$

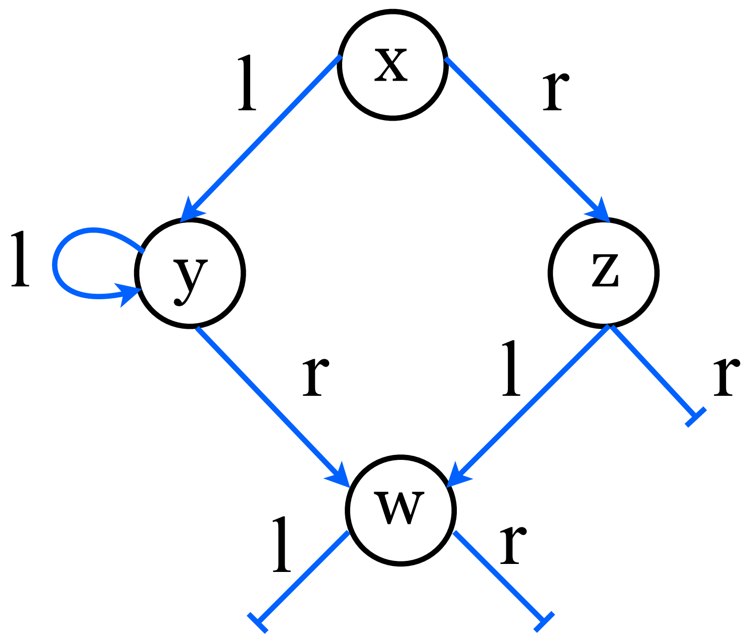
T1 || **T2** || **T3**

$$X * \boxed{(x \mapsto 10 * z \mapsto 10) \lor (x \mapsto 10 * z \mapsto 9)}_{I_1} \quad Y * \boxed{(x \mapsto 10 * y \mapsto 10) \lor (x \mapsto 11 * y \mapsto 10)}_{I_2} \quad Z * \boxed{(y \mapsto 10 * z \mapsto 10) \lor (y \mapsto 11 * z \mapsto 10)}_{I_3}$$

// ?? Merge the resources

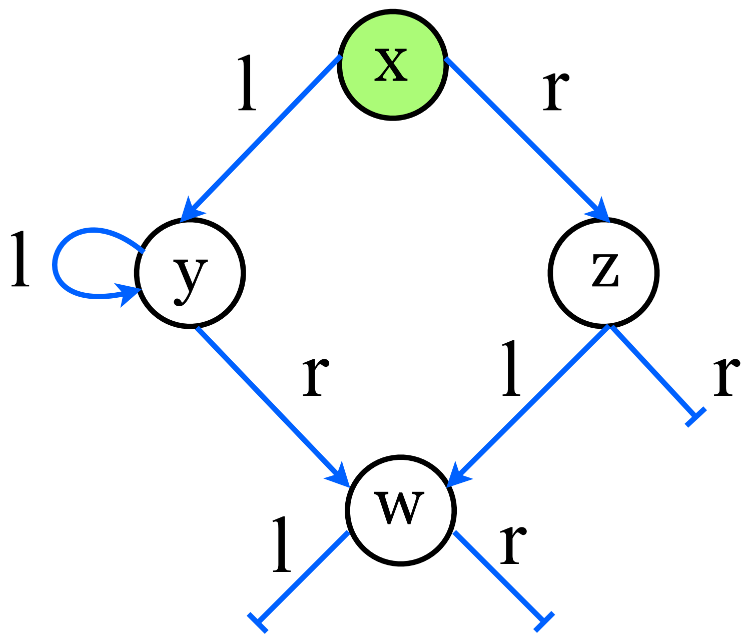
$$X * Y * Z * \boxed{x \mapsto 10 * y \mapsto 10 * z \mapsto 10}_{I}$$

Example - Spanning Tree



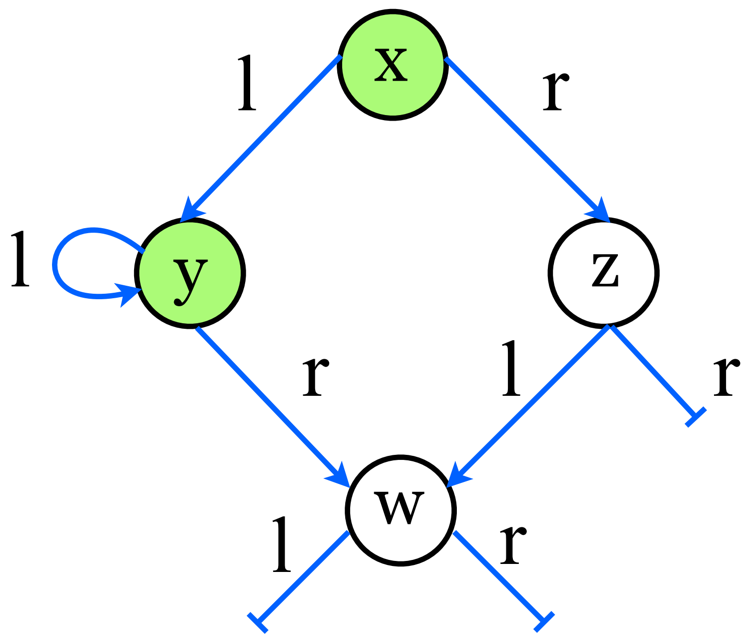

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



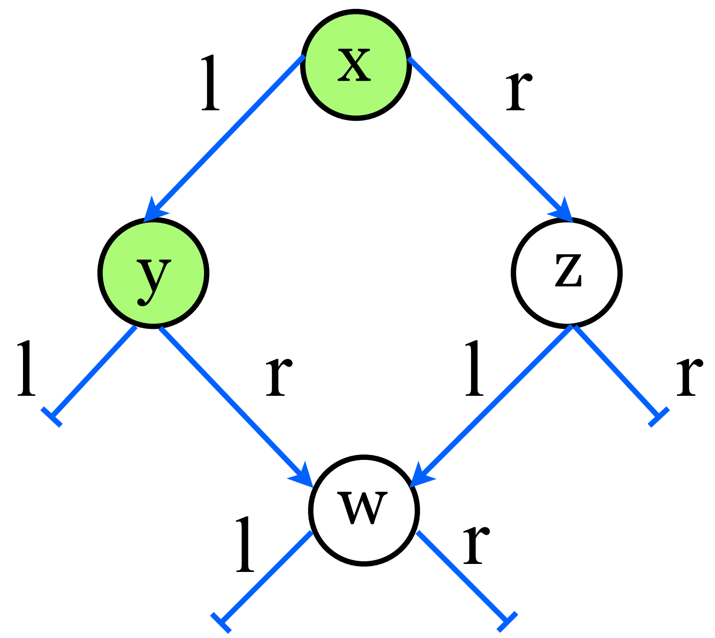

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



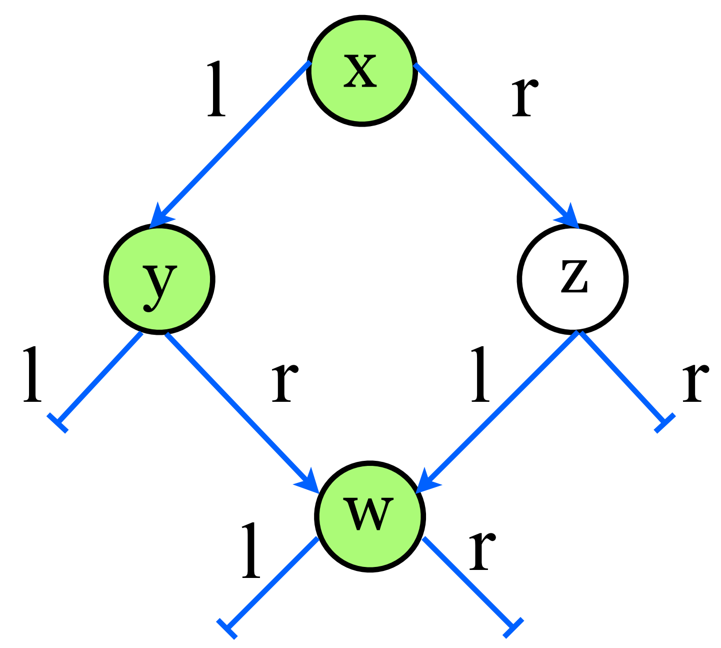

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



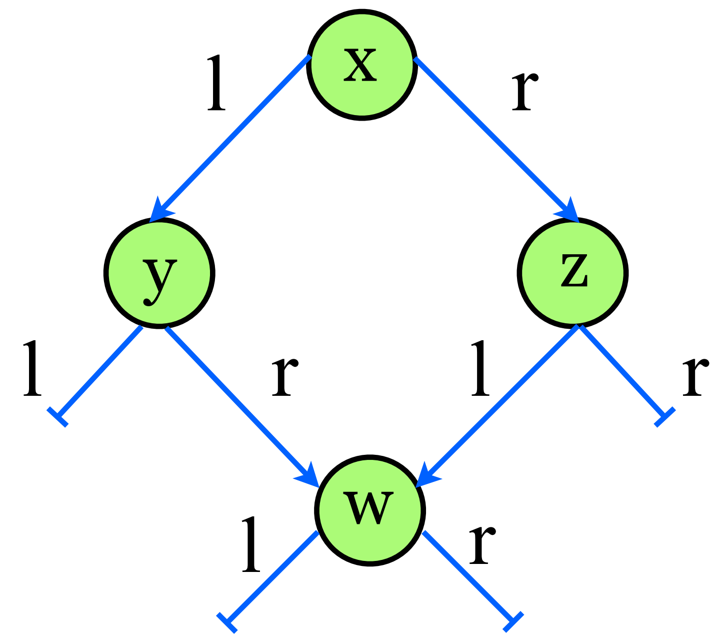

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



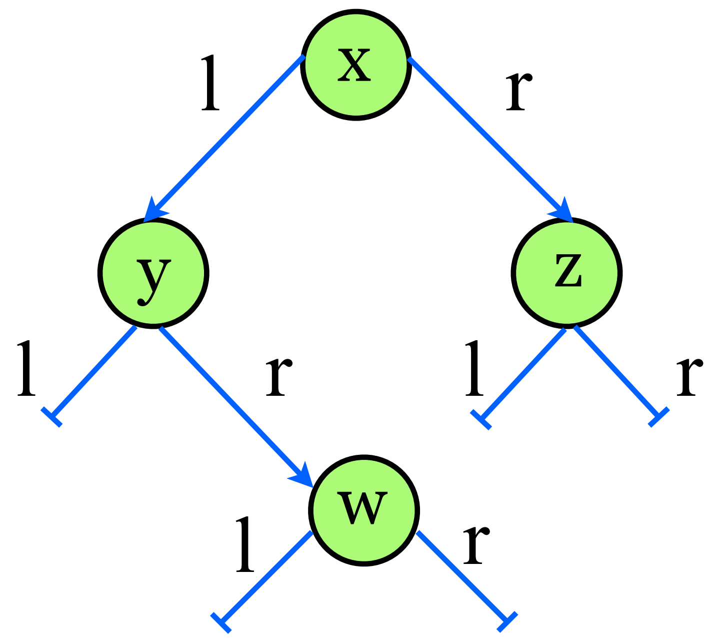

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree



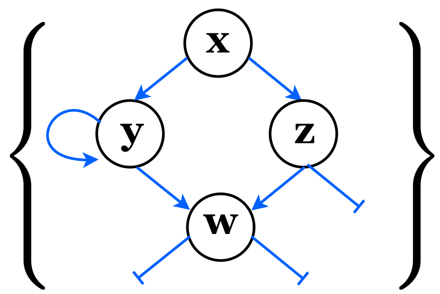

```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

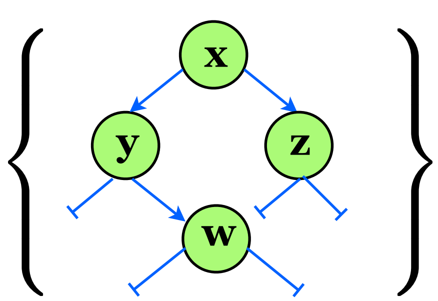



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

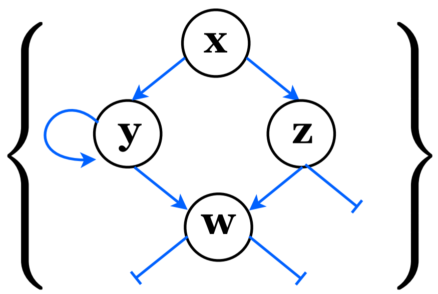
Example - Spanning Tree




```
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;
}
```

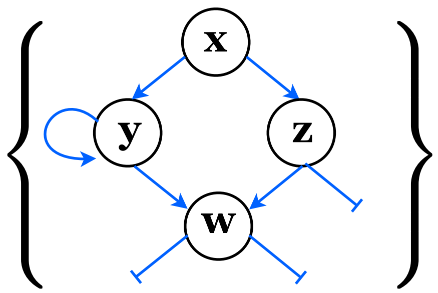


Example - Spanning Tree

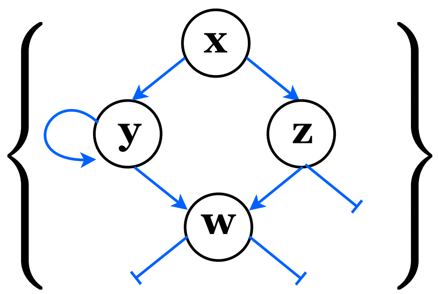



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

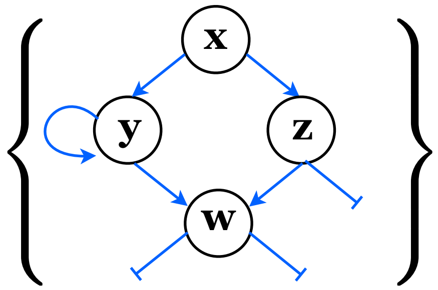



```
b:= spanning(x) {
```

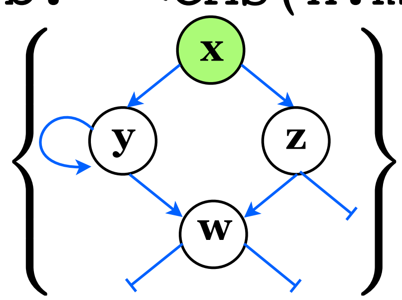



```
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

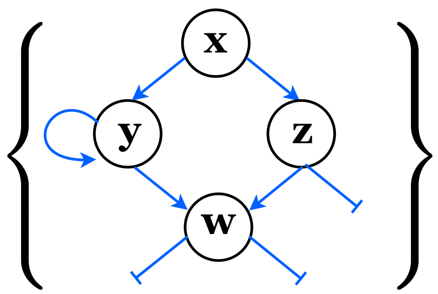



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

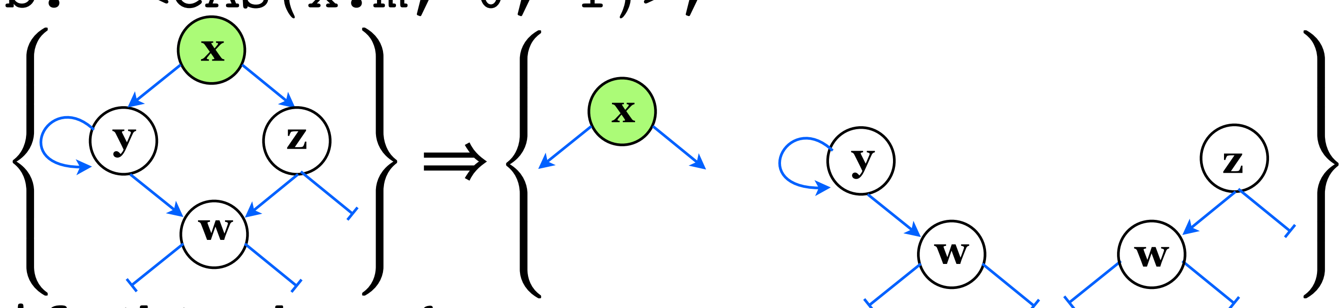



```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

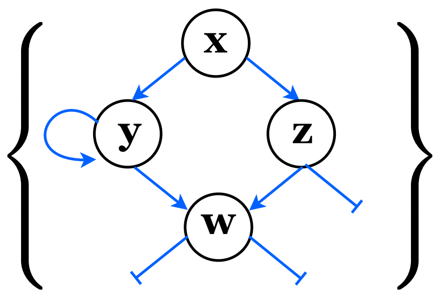



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

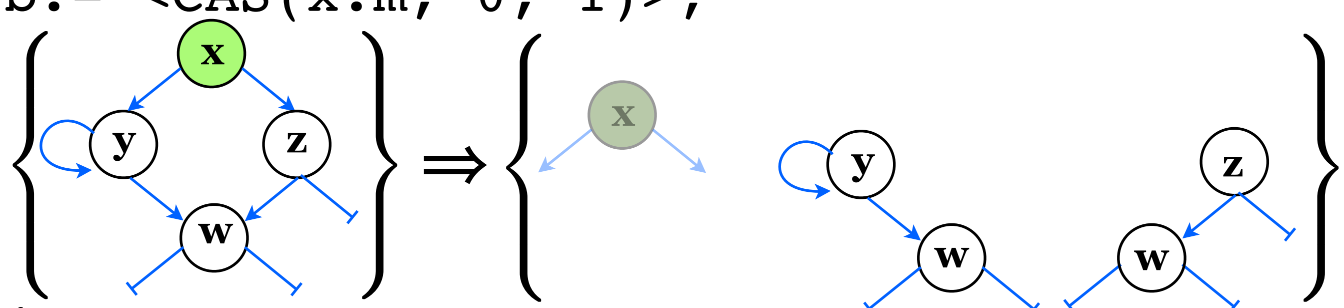



```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

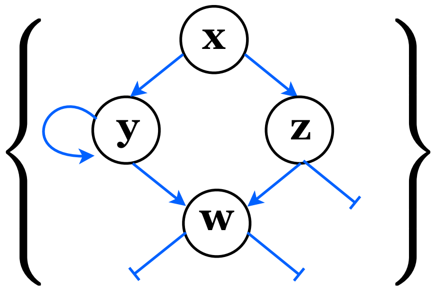



```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```

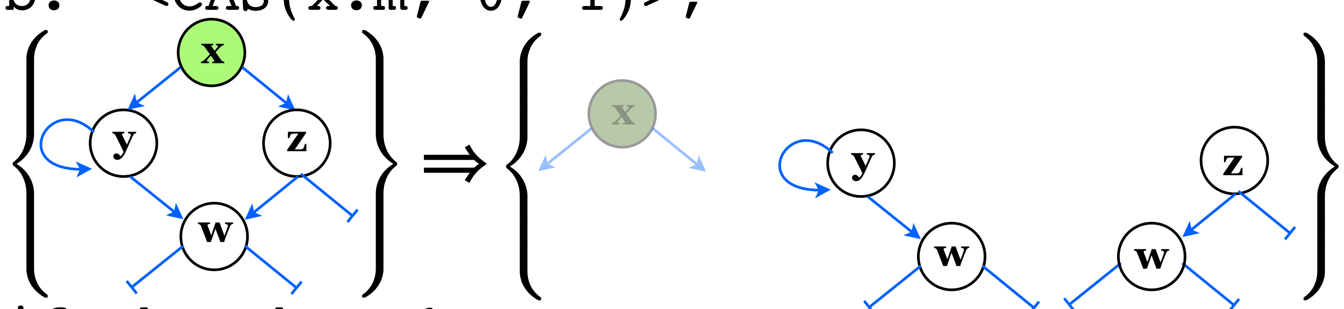



```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

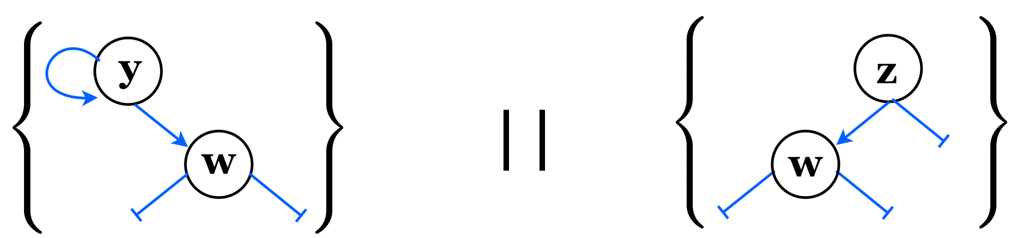
Example - Spanning Tree




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

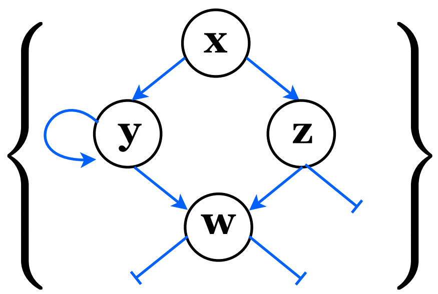



```
  if (b) then {
```

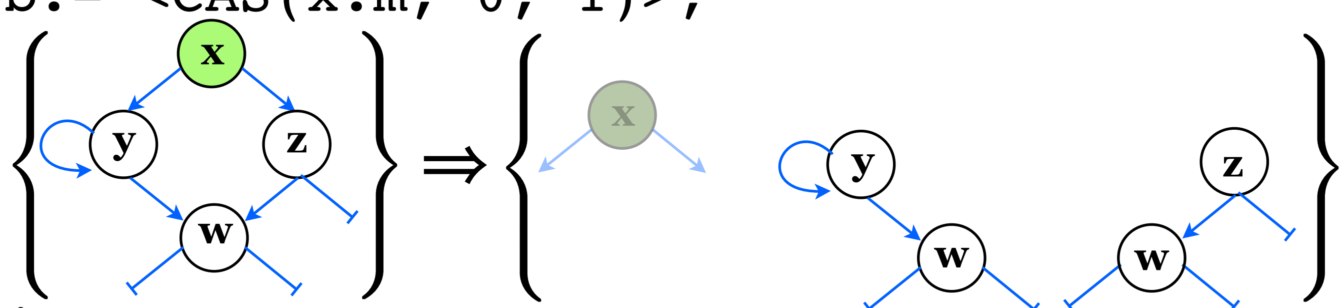



```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

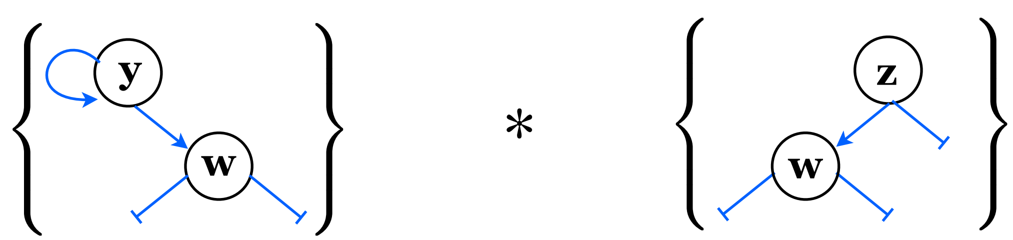
Example - Spanning Tree




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

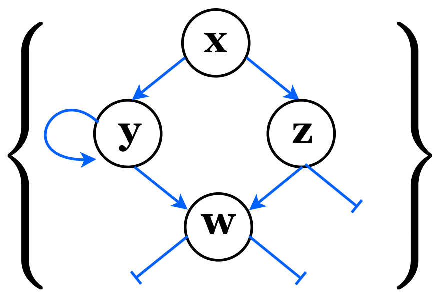



```
  if (b) then {
```

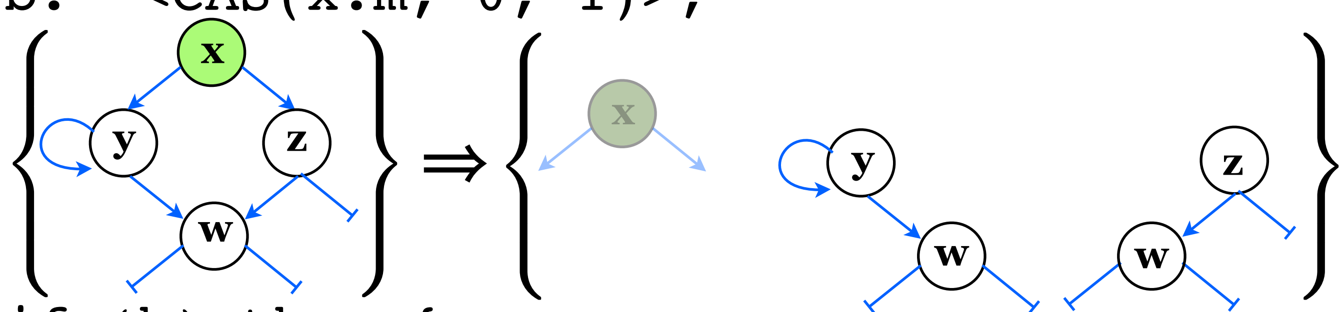



```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

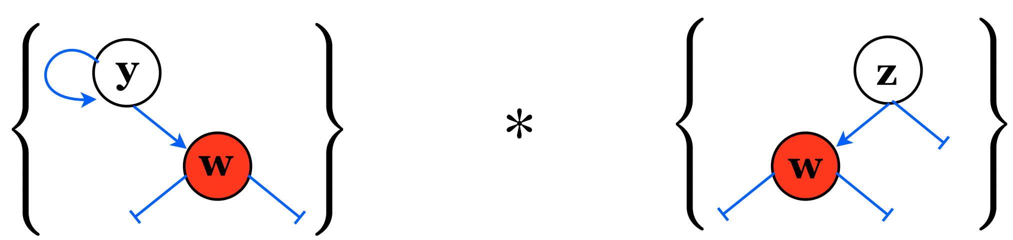
Example - Spanning Tree




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

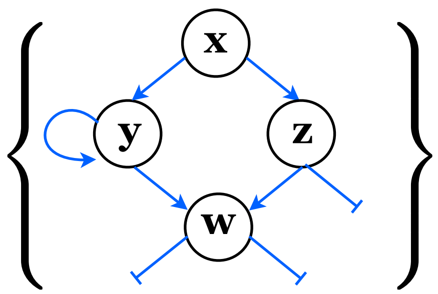



```
  if (b) then {
```

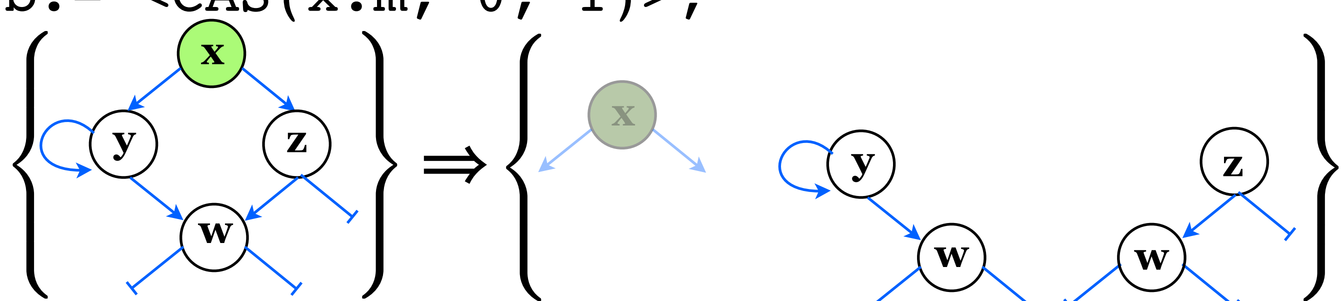


```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

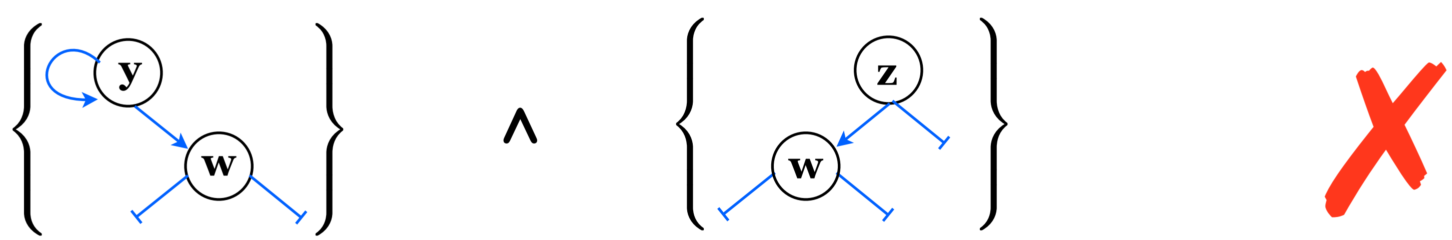
Example - Spanning Tree




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

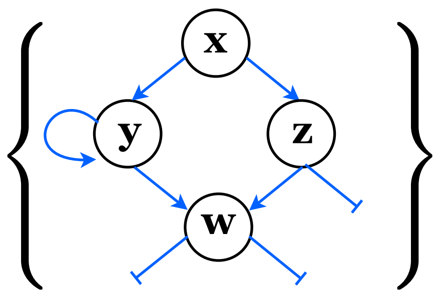



```
  if (b) then {
```

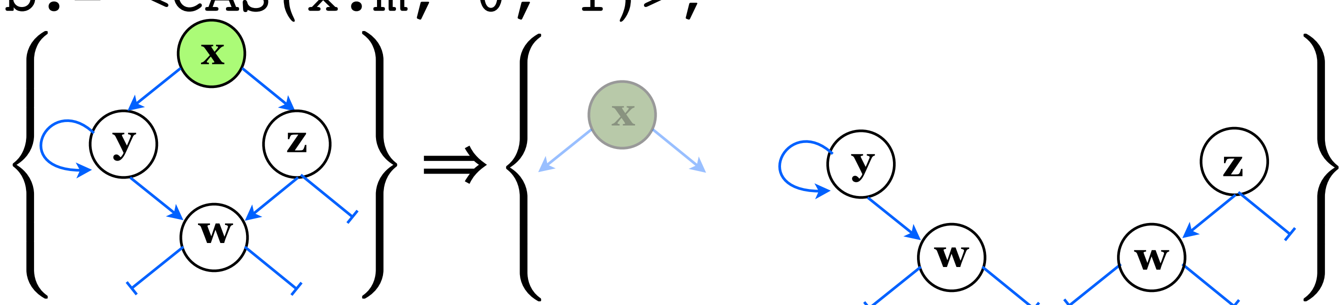



```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

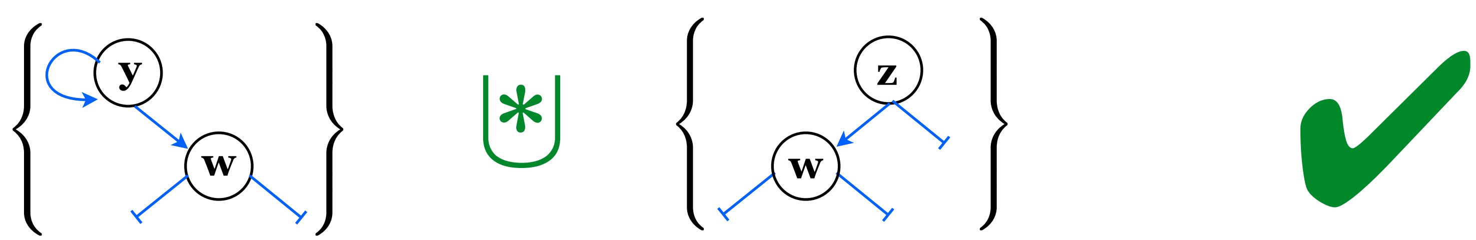
Example - Spanning Tree




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



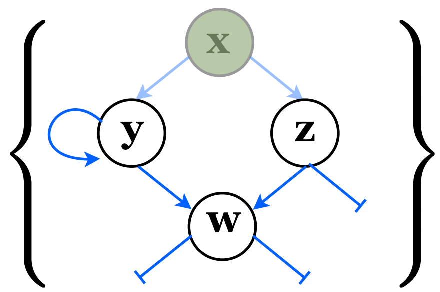

```
  if (b) then {
```



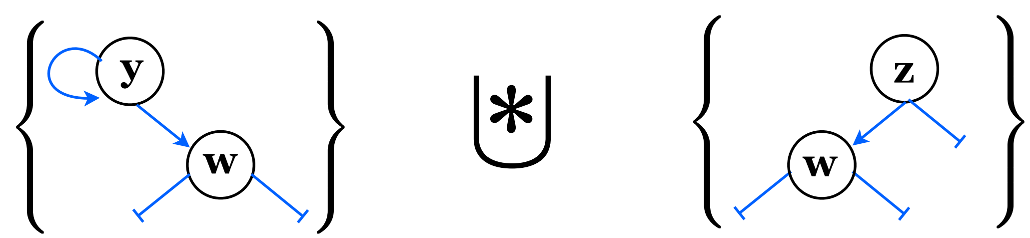

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

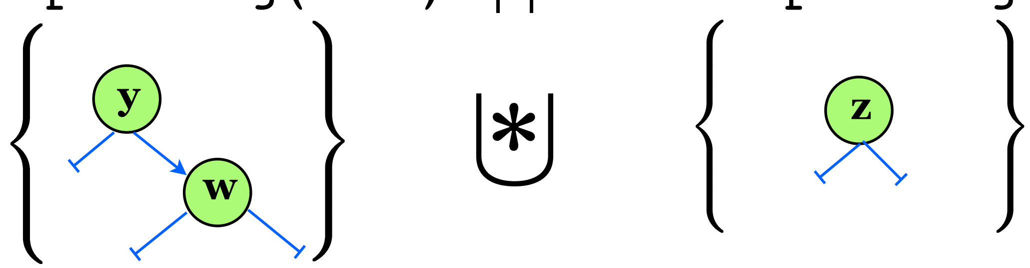
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
```




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



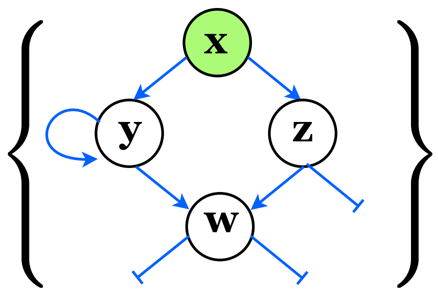

```
    b1:= spanning(x.l) || b2:= spanning(x.r);
```



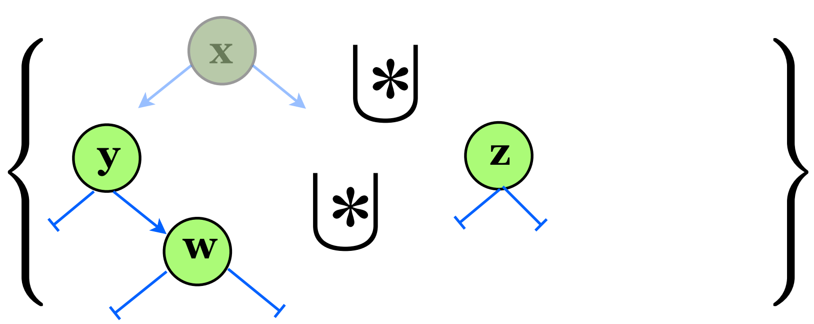

```
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
  return b;
}
```

Example - Spanning Tree

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



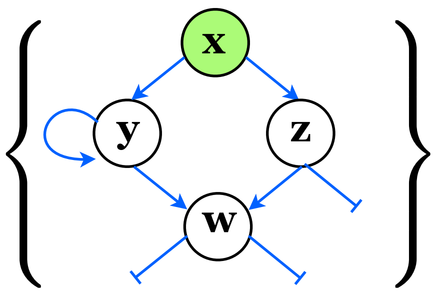

```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
```



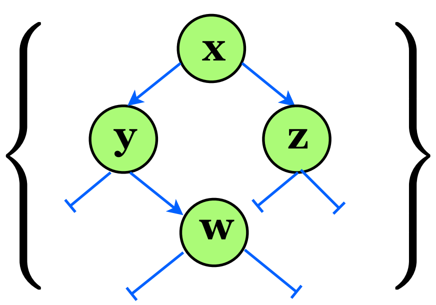

```
  return b;
}
```

Example - Spanning Tree

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



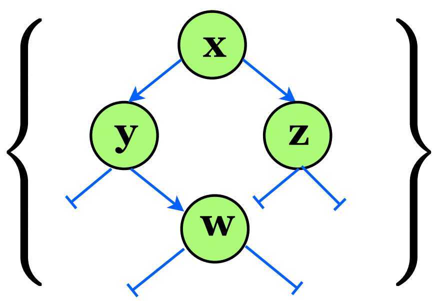

```
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
```




```
  return b;
}
```

Example - Spanning Tree

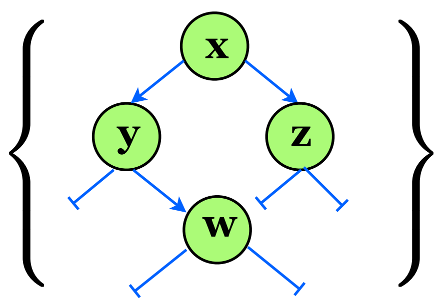
```
b:= spanning(x) {
  b:= <CAS(x.m, 0, 1)>;
  if (b) then {
    b1:= spanning(x.l) || b2:= spanning(x.r);
    if (!b1) then
      x.l:= null
    if (!b2) then
      x.r:= null
  }
```




```
  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;
}
```



Conclusions

- From RG/OG to CAP/TaDA
  - Huge steps towards compositionality/locality
  - Not good enough
- CoLoSL
  - Even more compositional/local
  - Examples - Dijkstra's algorithm, Spanning tree, Set
  - How to get the rest of the field to join subjective thinking?
  - More Examples

Questions?

Thank you for listening