

CoLoSL

Local Reasoning At Last!

Azalea Raad

Jules Villard

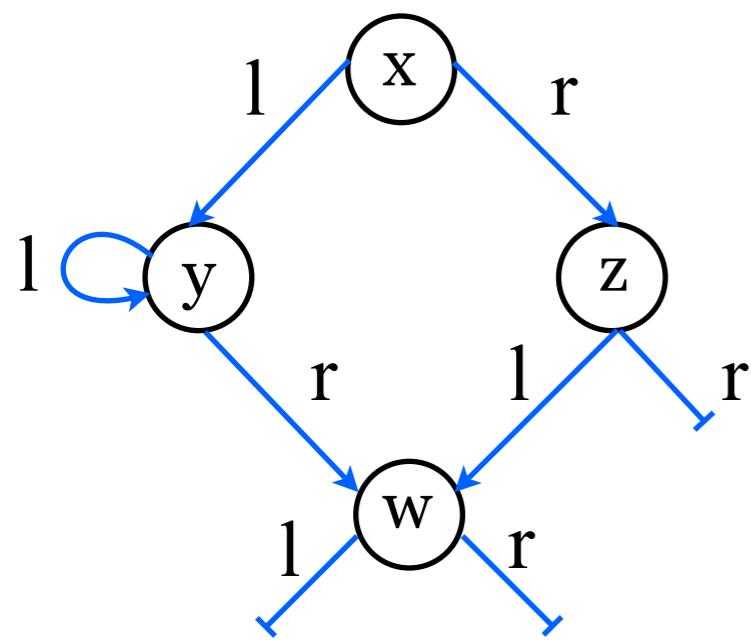
Philippa Gardner

Imperial College London

MPI-SWS Saarbrücken

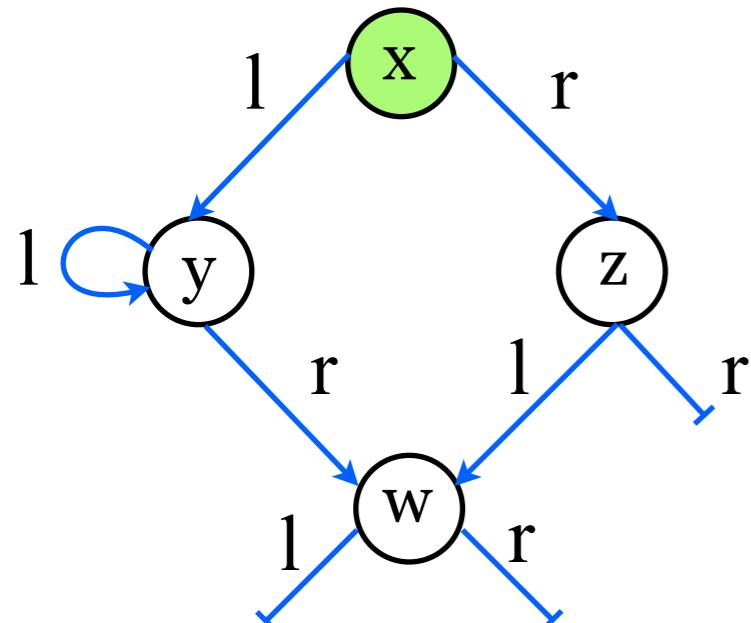
15 December 2014

Example - Spanning Tree



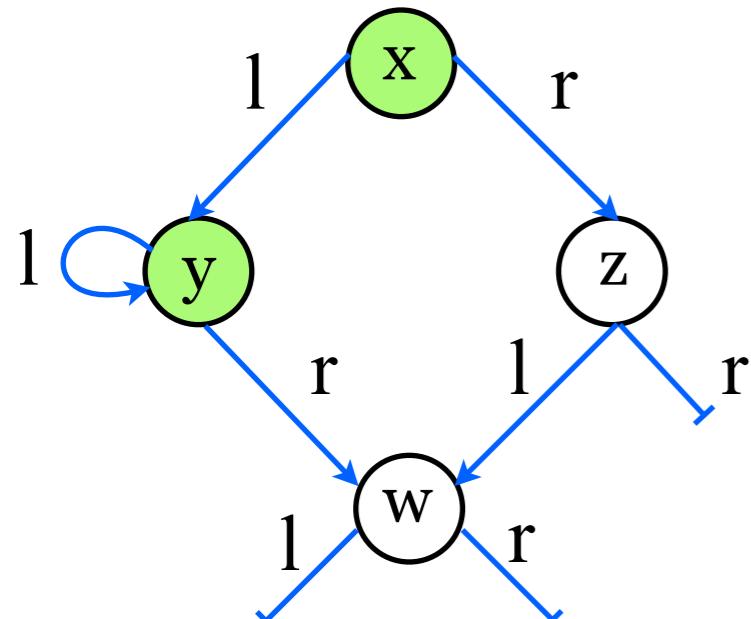
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree



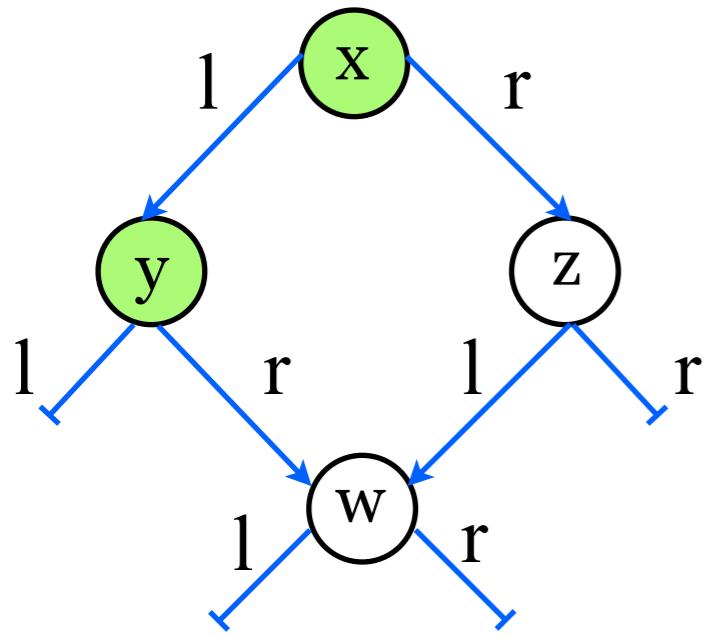
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree



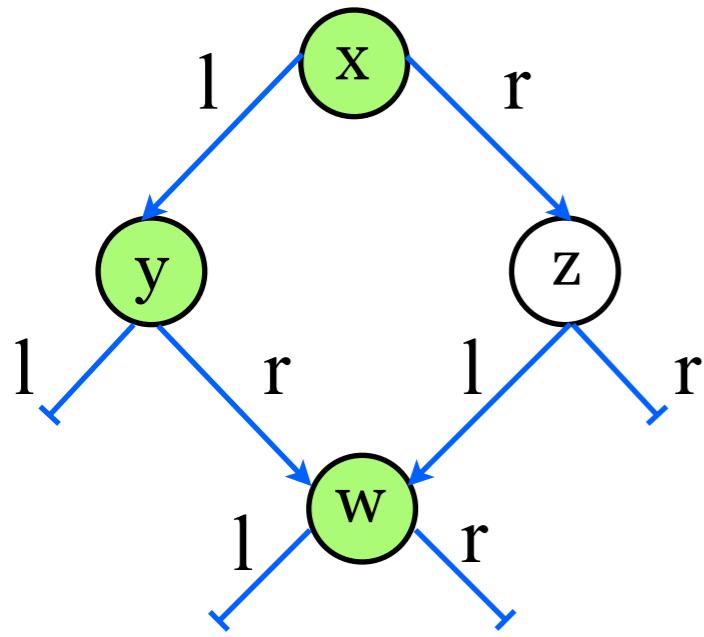
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree



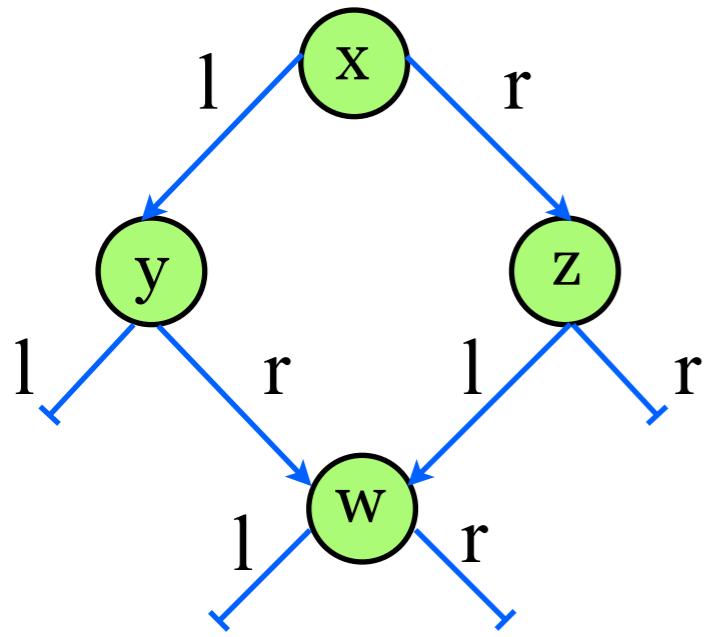
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree



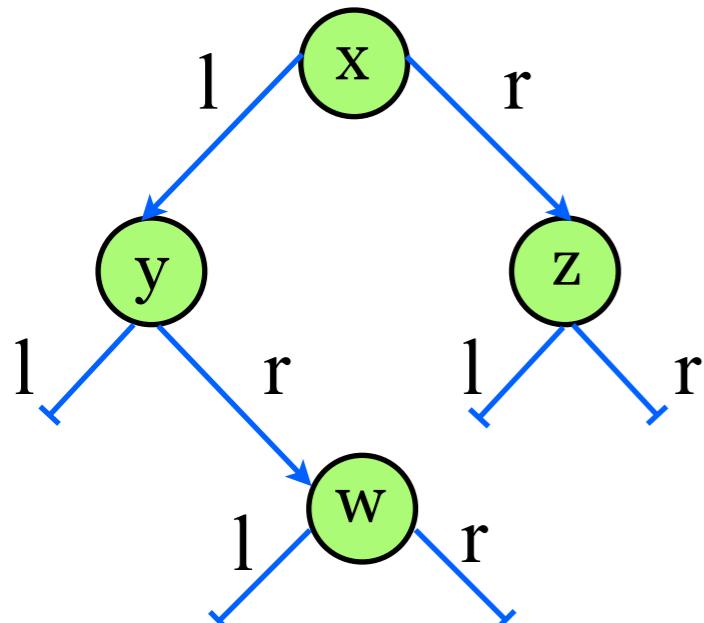
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree



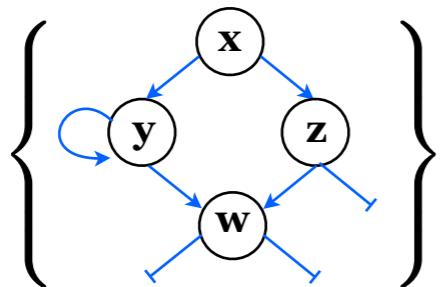
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree

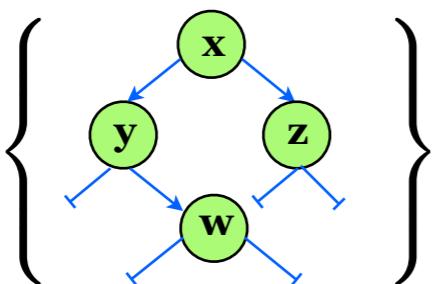


```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

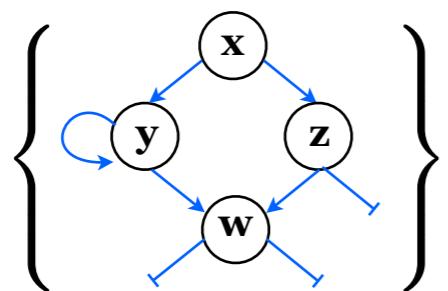
Example - Spanning Tree



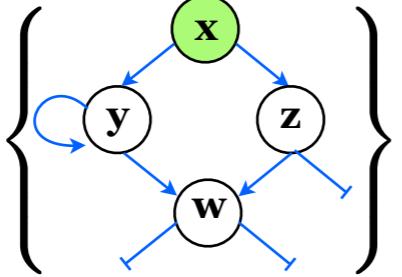
```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```



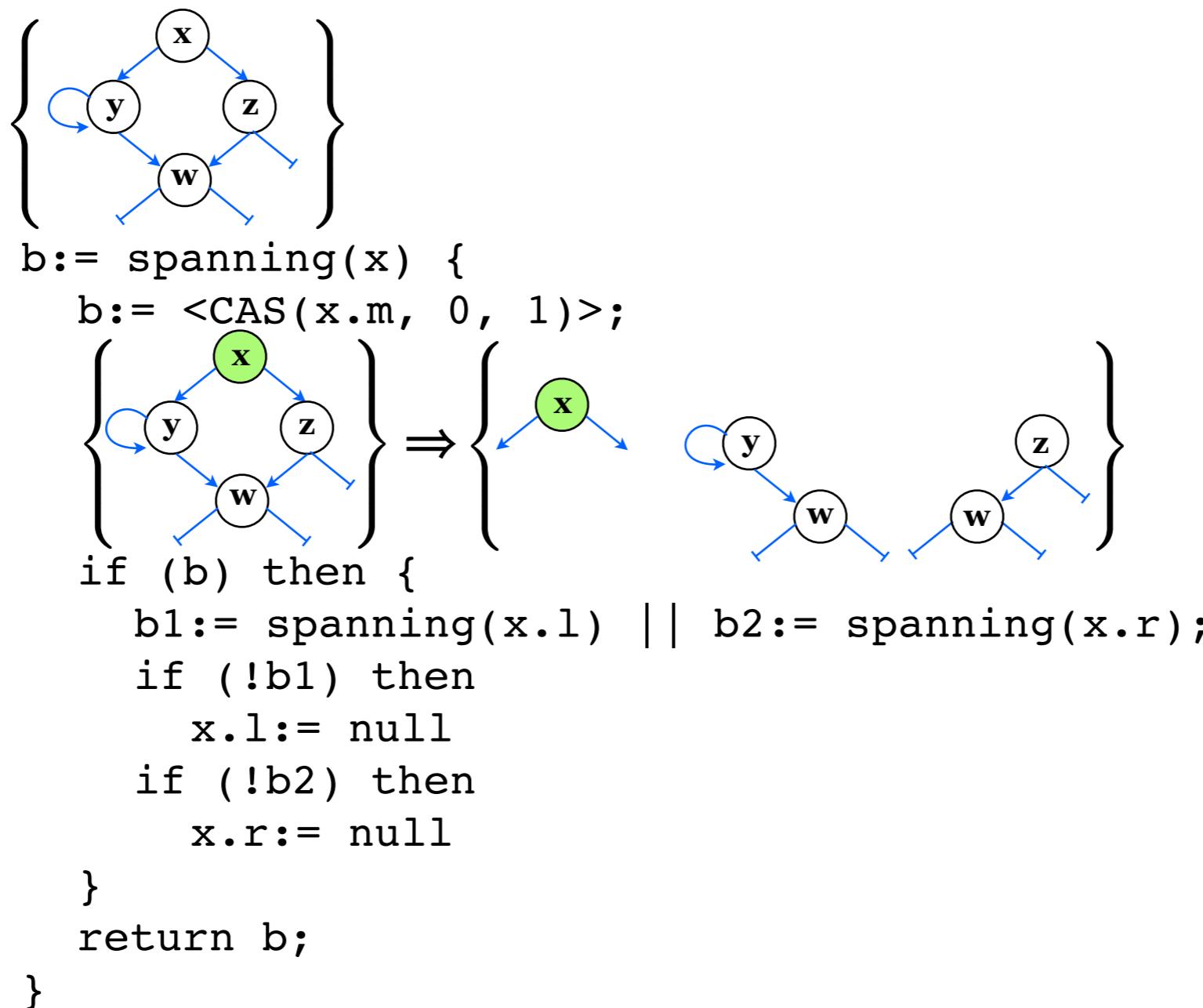
Example - Spanning Tree



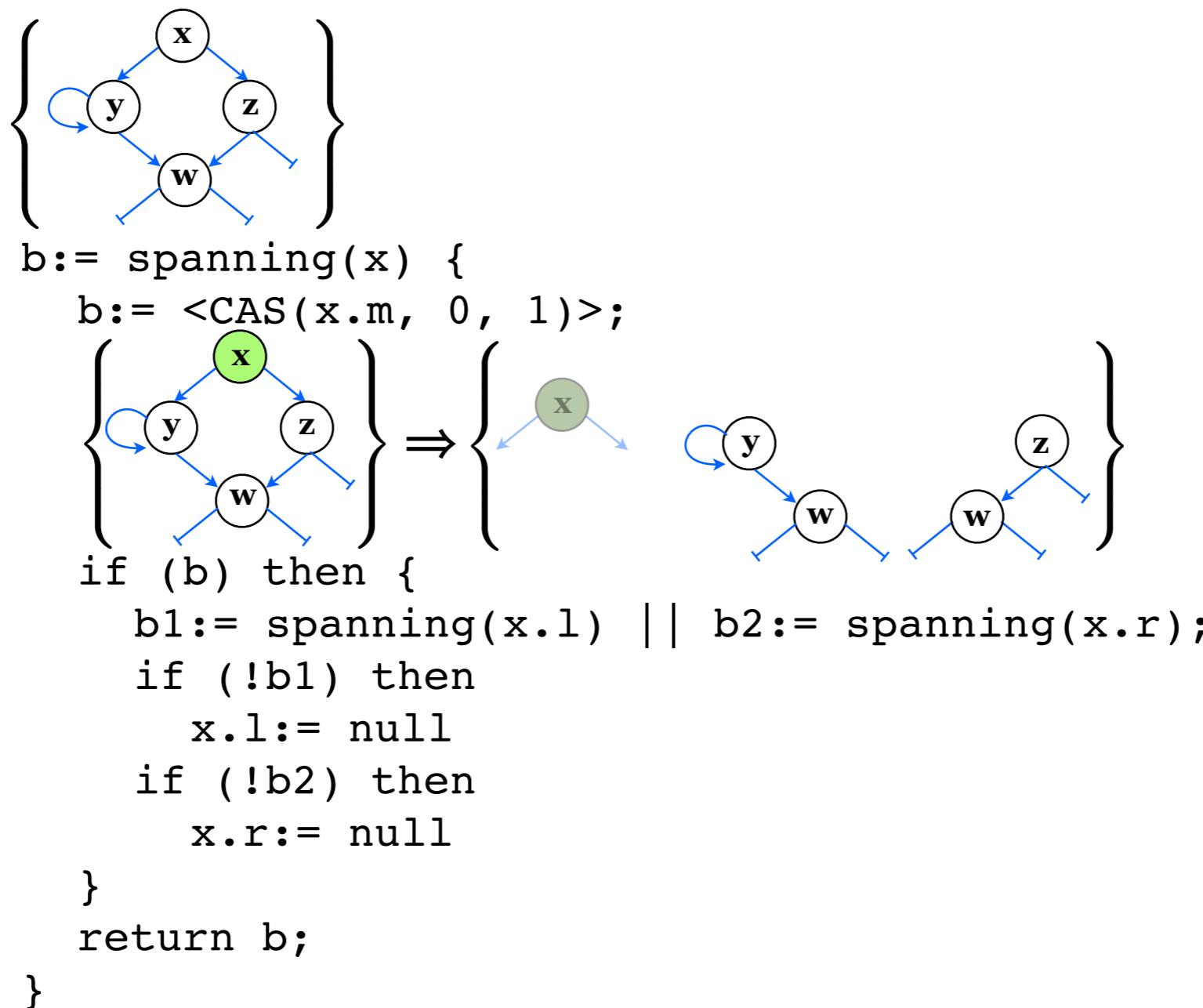
```
b := spanning(x) {  
    b := <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1 := spanning(x.l) || b2 := spanning(x.r);  
        if (!b1) then  
            x.l := null  
        if (!b2) then  
            x.r := null  
    }  
    return b;  
}
```



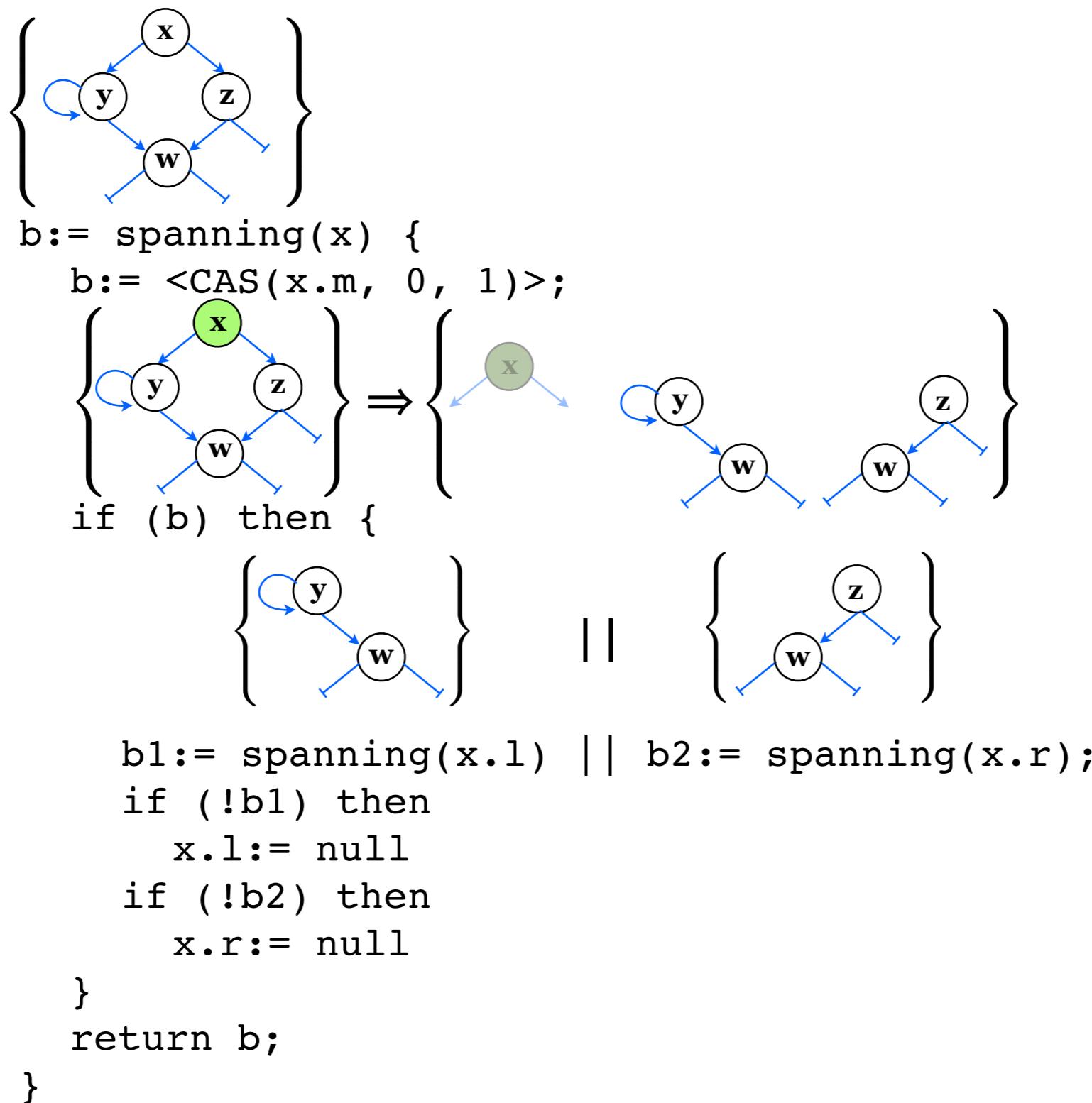
Example - Spanning Tree



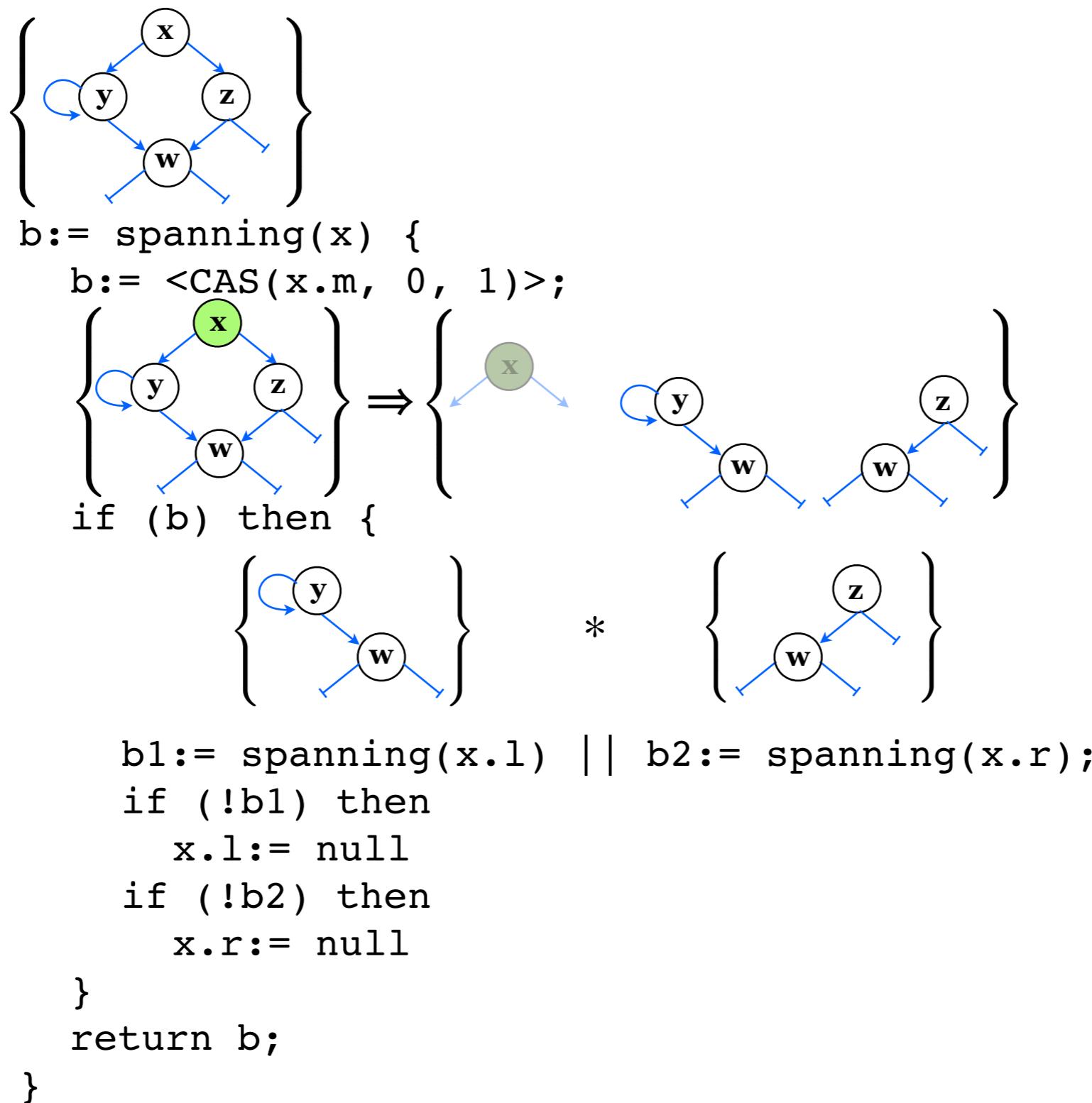
Example - Spanning Tree



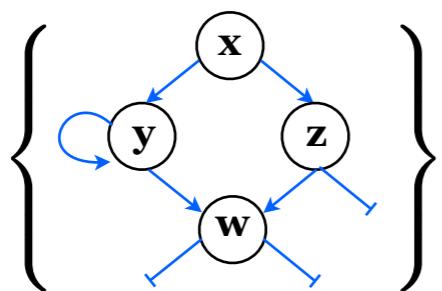
Example - Spanning Tree



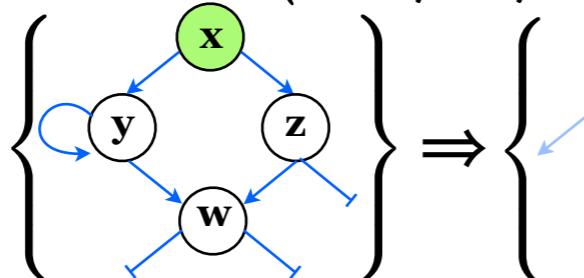
Example - Spanning Tree



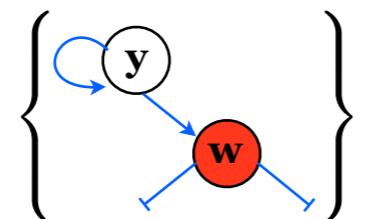
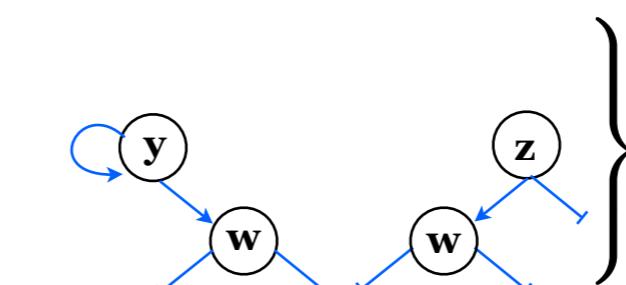
Example - Spanning Tree



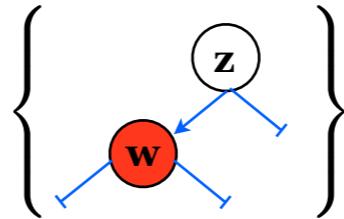
```
b := spanning(x) {  
    b := <CAS(x.m, 0, 1)>;
```



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



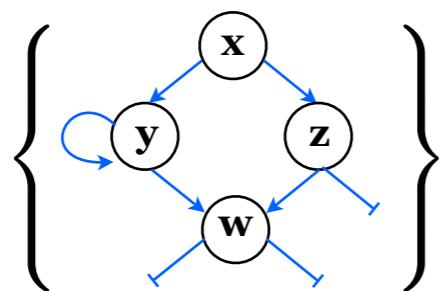
*



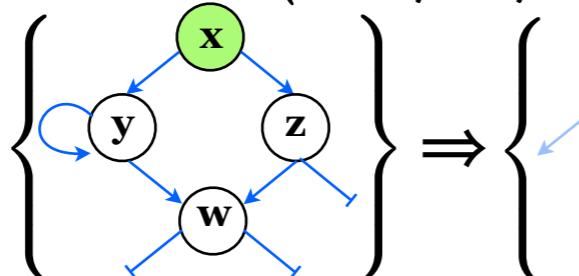
X

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

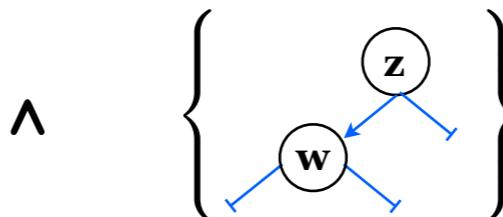
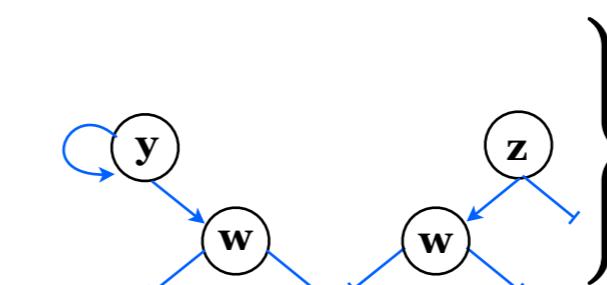
Example - Spanning Tree



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



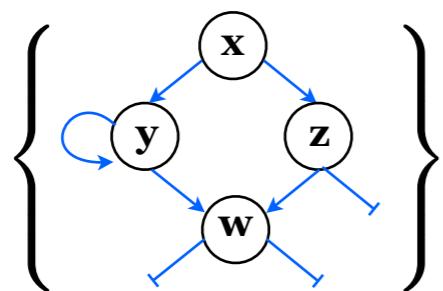
```
if (b) then {
```



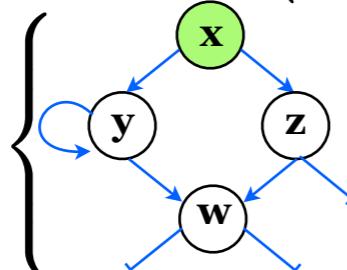
X

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

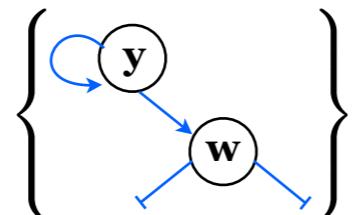
Example - Spanning Tree



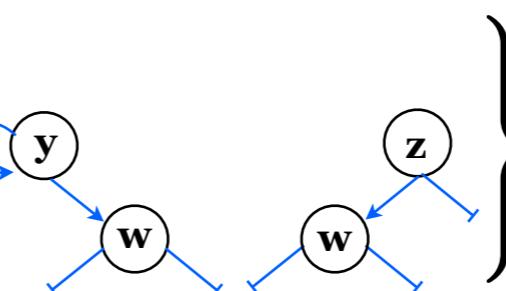
```
b := spanning(x) {  
    b := <CAS(x.m, 0, 1)>;
```



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



U



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

Global Shared Resources

$P_1 \wedge P_2$

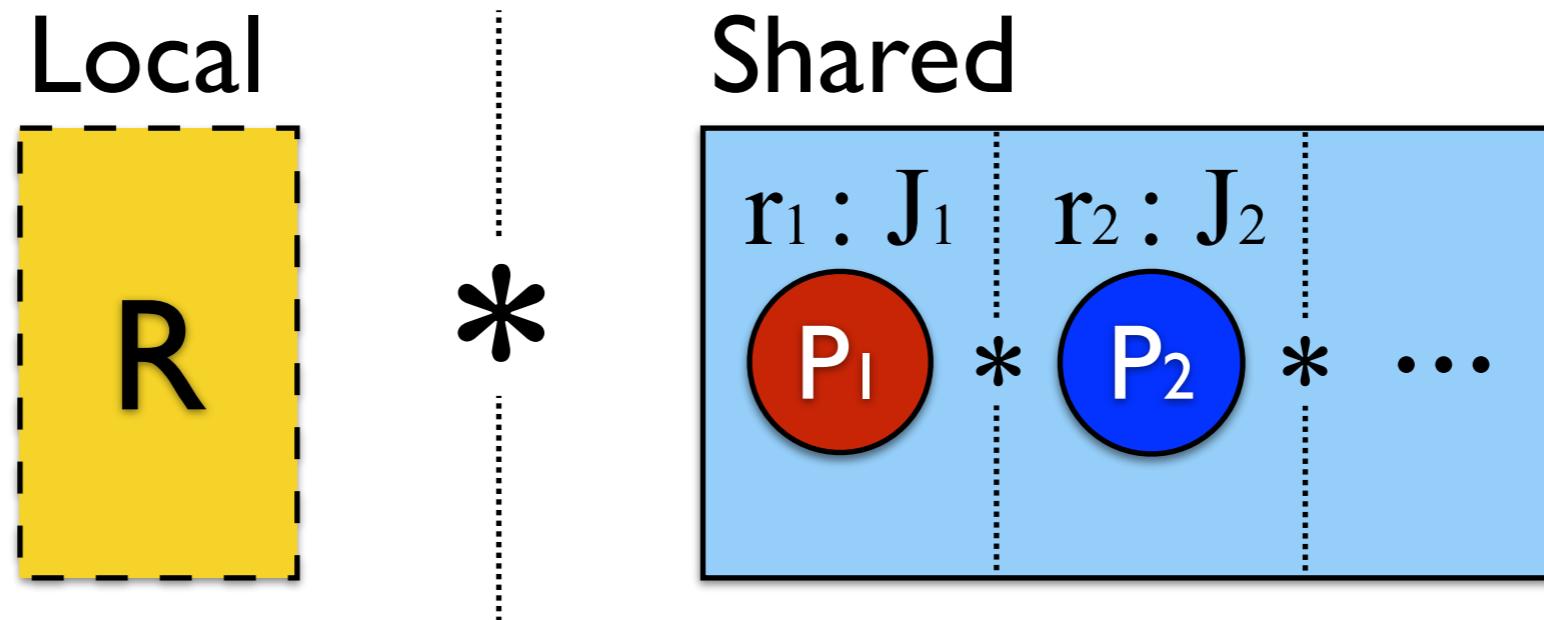
$$\frac{\left\{ \boxed{P_1} \right\} C_1 \left\{ \boxed{Q_1} \right\} \quad \left\{ \boxed{P_2} \right\} C_1 \left\{ \boxed{Q_2} \right\}}{\left\{ \boxed{P_1 \wedge P_2} \right\} C_1 \parallel C_2 \left\{ \boxed{Q_1 \wedge Q_2} \right\}}$$

Global Shared Resources

$$\frac{\left\{ \boxed{P_1} \right\} C_1 \left\{ \boxed{Q_1} \right\} \quad \left\{ \boxed{P_2} \right\} C_1 \left\{ \boxed{Q_2} \right\}}{\left\{ \boxed{P_1 \wedge P_2} \right\} C_1 \parallel C_2 \left\{ \boxed{Q_1 \wedge Q_2} \right\}}$$

- Shared resources
 - **No** framing mechanism - global resources carried around throughout reasoning
- Interference
 - **No** framing mechanism provided - interference of ALL resources carried around throughout reasoning

Disjoint Shared Resources



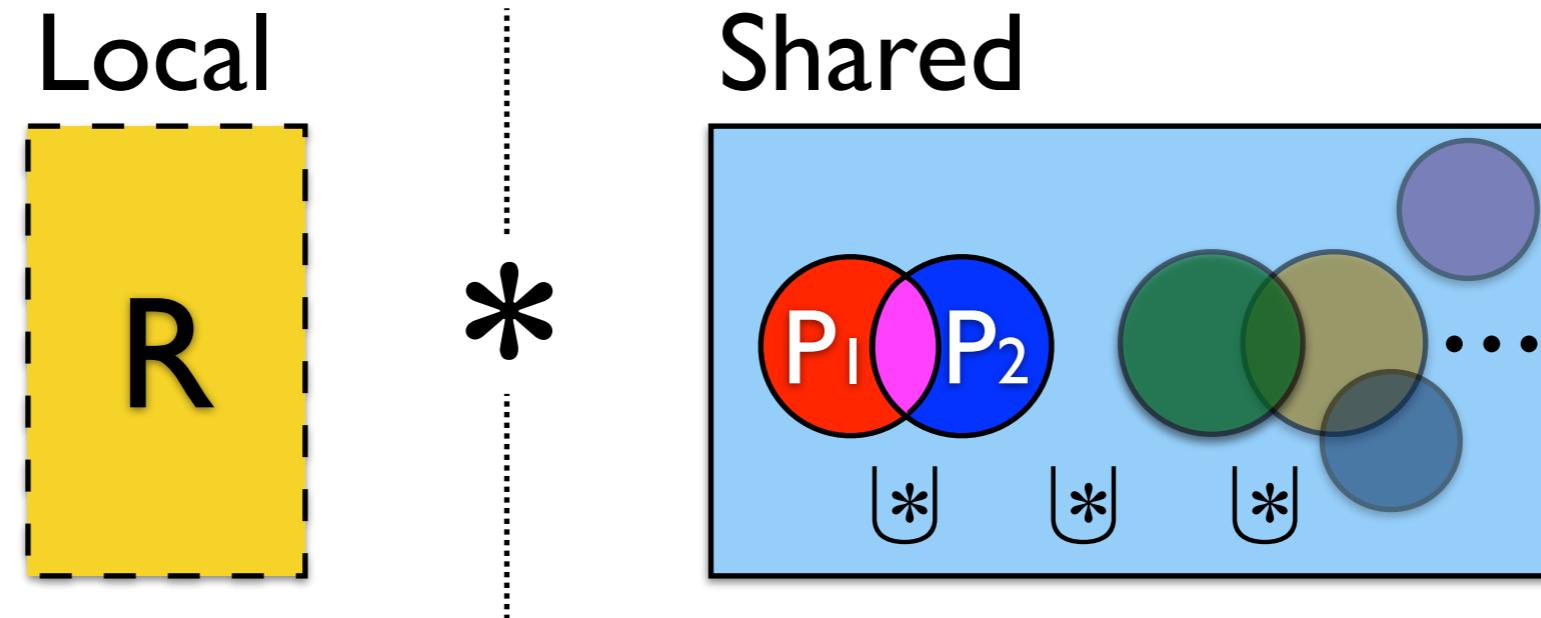
$$\frac{\left\{ \boxed{P_1} \right\} C_1 \left\{ \boxed{Q_1} \right\} \quad \left\{ \boxed{P_2} \right\} C_1 \left\{ \boxed{Q_2} \right\}}{\left\{ \boxed{P_1} * \boxed{P_2} \right\} C_1 || C_2 \left\{ \boxed{Q_1} * \boxed{Q_2} \right\}}$$

Disjoint Shared Resources

$$\frac{\begin{array}{c} \{P1\} C1 \{O1\} \quad \{P2\} C1 \{O2\} \\ \hline \end{array}}{\{P1 * P2\} C1 || C2 \{Q1 * Q2\}}$$

- Shared resources
 - Limited framing
 - **Static** (pre-determined) frames (regions/ invariants)
 - **Disjoint** frames
 - Interference
 - **No** framing mechanism provided within a region

CoLoSL: Concurrent Local Subjective Logic



$$\frac{\{P_1\} C_1 \{Q_1\} \quad \{P_2\} C_2 \{Q_2\}}{\{P_1 \uplus P_2\} C_1 \parallel C_2 \{Q_1 \uplus Q_2\}}$$

CoLoSL: Concurrent Local Subjective Logic

- Shared resources
 - **Dynamic framing**
 - **Overlapping frames**
 - Interference
 - **Dynamic framing** of irrelevant interference
 - **Subjective rewriting** of interference

Local Subjective Reasoning

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
	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
		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

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++;  
}  
}
```

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

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

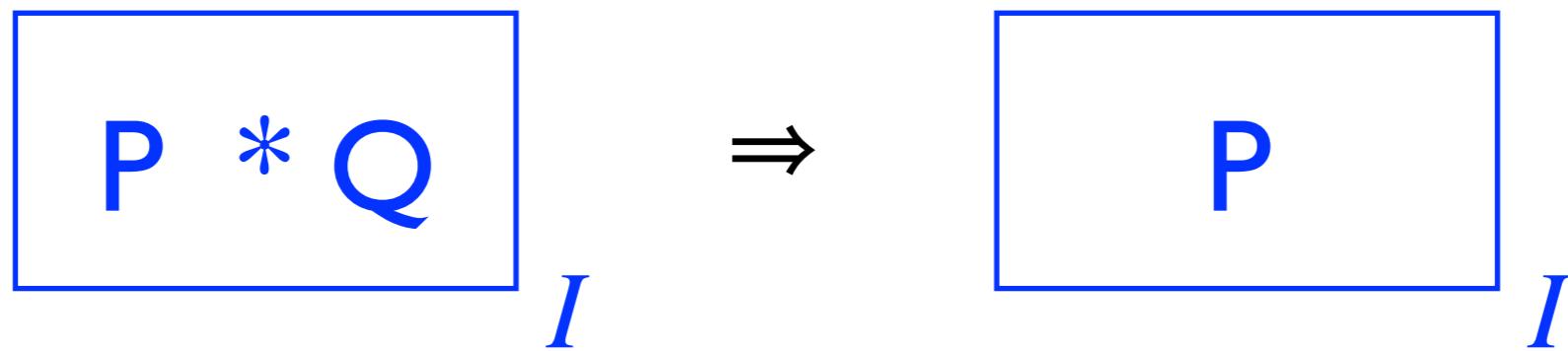
$\exists v.$

$x \mapsto v$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v+1$	*	$z \mapsto v$

I

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

Forgetting Resources



Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v+1$	*	$z \mapsto v$

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v+1$

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$	
\vee	$x \mapsto v+1$	*	$y \mapsto v$
\vee	$x \mapsto v+1$	*	$y \mapsto v+1$

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$$\begin{array}{ll} x \mapsto v & * y \mapsto v \\ \vee x \mapsto v+1 & * y \mapsto v \end{array}$$

I

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

Forgetting Interference

If

$$I \cup I' \sqsubseteq^P I$$

Then

$$\boxed{P}_{I \cup I'} \Rightarrow \boxed{P}_I$$

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$

I

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

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v.$

$$\begin{array}{ll} x \mapsto v & * y \mapsto v \\ \vee x \mapsto v+1 & * y \mapsto v \end{array}$$

NOT STABLE!!

I_2

$$I_2 = \left\{ \begin{array}{l} X : x \mapsto v * z \mapsto v \rightsquigarrow x \mapsto v+1 * z \mapsto v \\ Y : x \mapsto v+1 * y \mapsto v \rightsquigarrow x \mapsto v+1 * y \mapsto v+1 \end{array} \right.$$

Example - Dijkstra's Self-stabilising Ring

Localisation : Naïve Attempt

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

$\exists v, v'.$
 $x \mapsto v * y \mapsto v'$

I_2

$$I_2 = \left\{ \begin{array}{l} X : x \mapsto v * z \mapsto v \rightsquigarrow x \mapsto v+1 * z \mapsto v \\ Y : x \mapsto v+1 * y \mapsto v \rightsquigarrow x \mapsto v+1 * y \mapsto v+1 \end{array} \right.$$

Example - Dijkstra's Self-stabilising Ring Localisation

$T2 = \begin{array}{l} \text{while } (y < 10) \{ \\ \quad \text{if } (y < x) \text{ then} \\ \quad \quad y++; \\ \} \end{array}$

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v+1$	*	$z \mapsto v$

I

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

Example - Dijkstra's Self-stabilising Ring Localisation

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

$\exists v.$

$x \mapsto v$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v$	*	$z \mapsto v$
$\vee x \mapsto v+1$	*	$y \mapsto v+1$	*	$z \mapsto v$

I'

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

Rewriting Interference

If

$$I \approx^P I'$$

Then

$$\boxed{P}_I \Rightarrow \boxed{P}_{I'}$$

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'

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

Example - Dijkstra's Self-stabilising Ring Localisation

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

$$\boxed{\begin{array}{l} \exists v. \\ \quad x \mapsto v * y \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v \end{array}} I'$$

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

Example - Dijkstra's Self-stabilising Ring Localisation

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

$$I_2 = \left\{ \begin{array}{l} X : x \mapsto v * z \mapsto v * y \mapsto v \rightsquigarrow x \mapsto v+1 * z \mapsto v * y \mapsto v \\ Y : x \mapsto v+1 * y \mapsto v \rightsquigarrow x \mapsto v+1 * y \mapsto v+1 \end{array} \right.$$

Example - Dijkstra's Self-stabilising Ring

$X * Y * Z *$

$\exists v.$

$$\begin{array}{l} x \mapsto v * y \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v+1 * z \mapsto v \end{array}$$

I

// Localise the resources

$X * \boxed{\begin{array}{l} \exists v. \\ x \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * z \mapsto v \end{array}}$

T_1

$Y * \boxed{\begin{array}{l} \exists v. \\ x \mapsto v * y \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v \end{array}}$

$||$

T_2

$Z * \boxed{\begin{array}{l} \exists v. \\ y \mapsto v * z \mapsto v \\ \vee y \mapsto v+1 * z \mapsto v \end{array}}$

T_3

$X * \boxed{\begin{array}{l} x \mapsto 10 * z \mapsto 10 \\ \vee x \mapsto 10 * z \mapsto 9 \end{array}}$

I_1

$Y * \boxed{\begin{array}{l} x \mapsto 10 * y \mapsto 10 \\ \vee x \mapsto 11 * y \mapsto 10 \end{array}}$

I_2

$Z * \boxed{\begin{array}{l} y \mapsto 10 * z \mapsto 10 \\ \vee y \mapsto 11 * z \mapsto 10 \end{array}}$

I_3

// ????

$X * Y * Z *$

$\boxed{x \mapsto 10 * y \mapsto 10 * z \mapsto 10}$

I

Merging Resources

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

Example - Dijkstra's Self-stabilising Ring

$X * Y * Z *$

$\exists v.$

$$\begin{array}{l} x \mapsto v * y \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v+1 * z \mapsto v \end{array}$$

I

// Localise the resources

$X * \boxed{\begin{array}{l} \exists v. \\ x \mapsto v * z \mapsto v \\ \vee x \mapsto v+1 * z \mapsto v \end{array}}$

T_1

$Y * \boxed{\begin{array}{l} \exists v. \\ x \mapsto v * y \mapsto v \\ \vee x \mapsto v+1 * y \mapsto v \end{array}}$

T_2

$Z * \boxed{\begin{array}{l} \exists v. \\ y \mapsto v * z \mapsto v \\ \vee y \mapsto v+1 * z \mapsto v \end{array}}$

T_3

$X * \boxed{\begin{array}{l} x \mapsto 10 * z \mapsto 10 \\ \vee x \mapsto 10 * z \mapsto 9 \end{array}}$

I_1

$Y * \boxed{\begin{array}{l} x \mapsto 10 * y \mapsto 10 \\ \vee x \mapsto 11 * y \mapsto 10 \end{array}}$

I_2

$Z * \boxed{\begin{array}{l} y \mapsto 10 * z \mapsto 10 \\ \vee y \mapsto 11 * z \mapsto 10 \end{array}}$

I_3

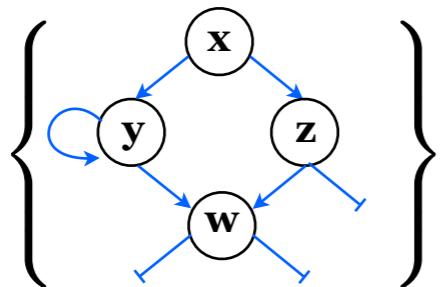
// Merge the resources

$X * Y * Z *$

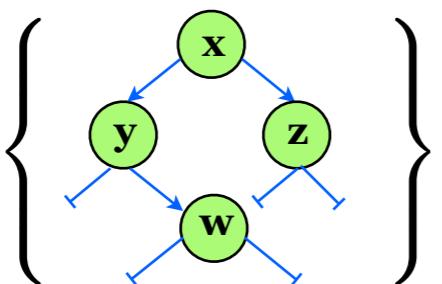
$\boxed{x \mapsto 10 * y \mapsto 10 * z \mapsto 10}$

I

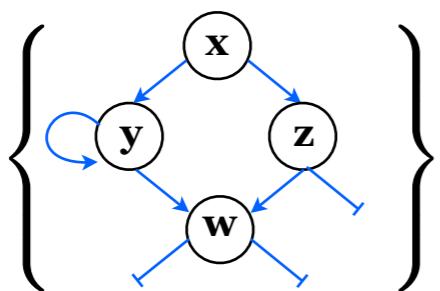
Example - Spanning Tree



```
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;  
}
```

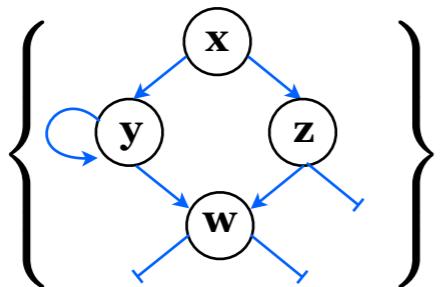


Example - Spanning Tree

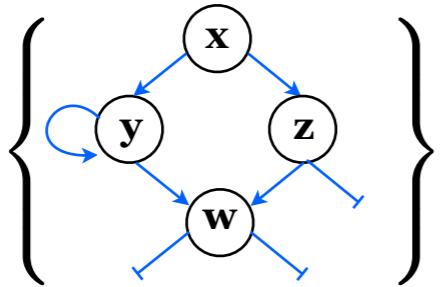


```
b:= spanning(x) {  
    b:= <CAS(x.m, 0, 1)>;  
    if (b) then {  
        b1:= spanning(x.l) || b2:= spanning(x.r);  
        if (!b1) then  
            x.l:= null  
        if (!b2) then  
            x.r:= null  
    }  
    return b;  
}
```

Example - Spanning Tree

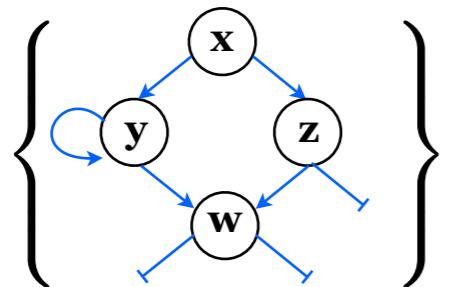


```
b := spanning(x) {
```



```
    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;
}
```

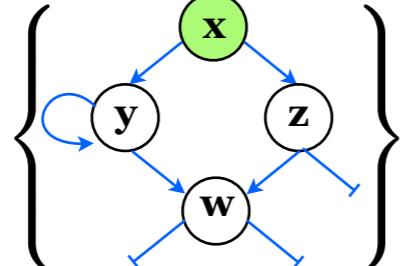
Example - Spanning Tree



```
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;  
}
```

{ }

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

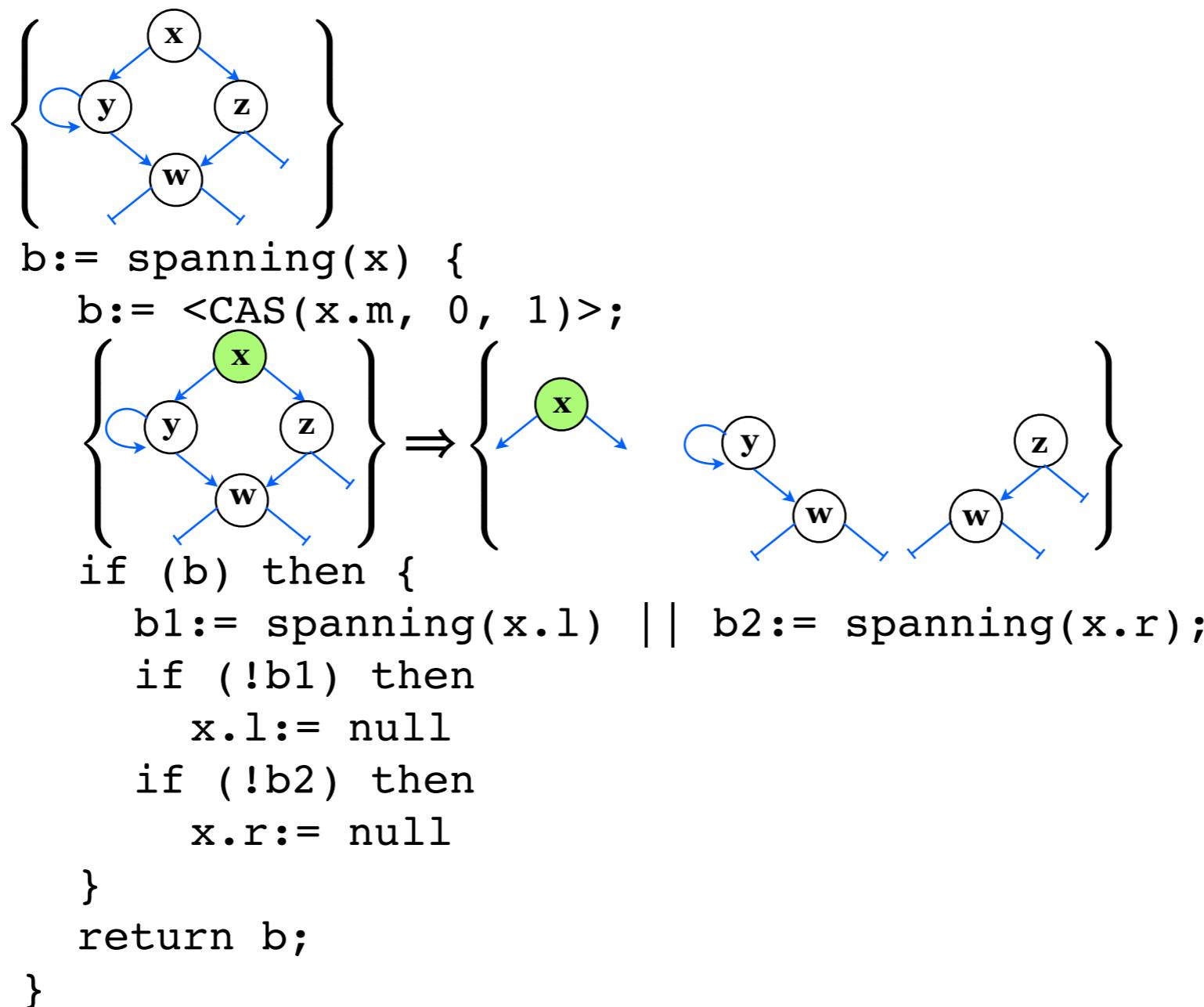
x.r := null

}

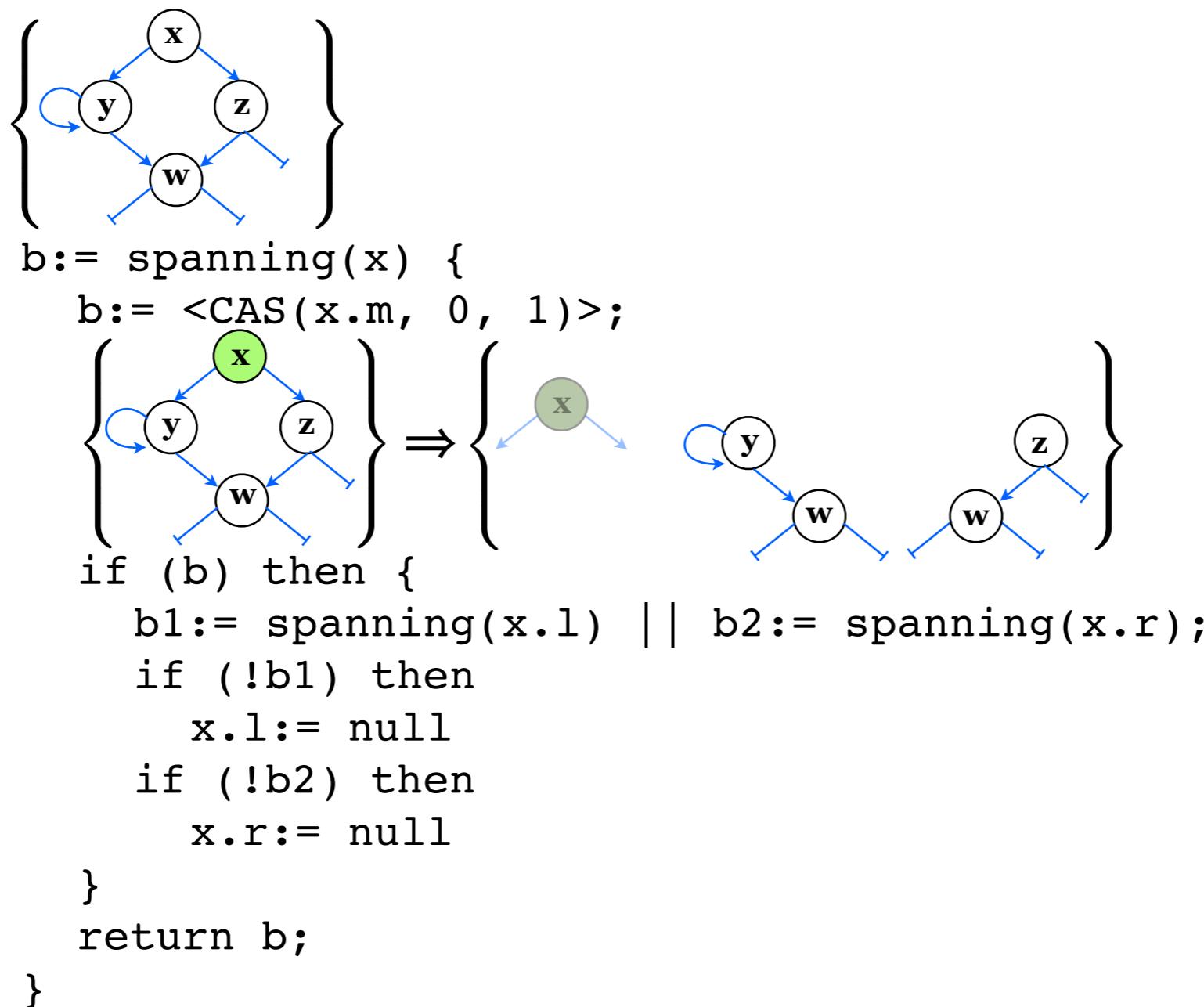
return b;

}

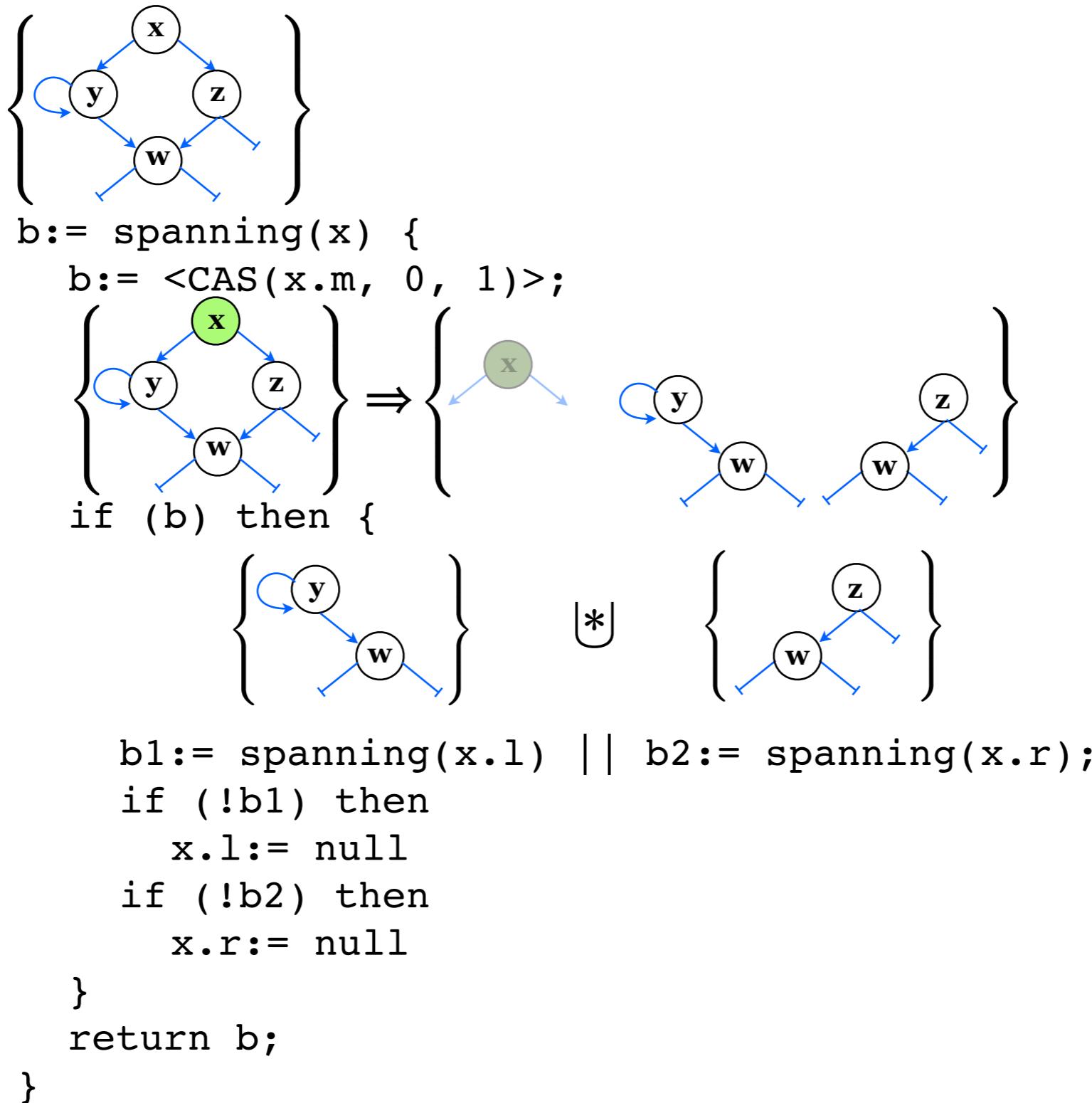
Example - Spanning Tree



Example - Spanning Tree

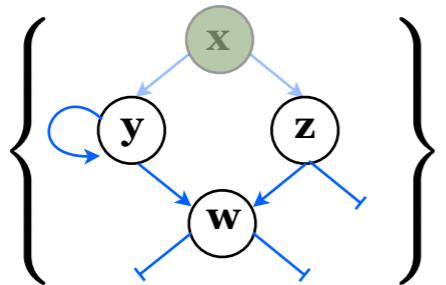


Example - Spanning Tree

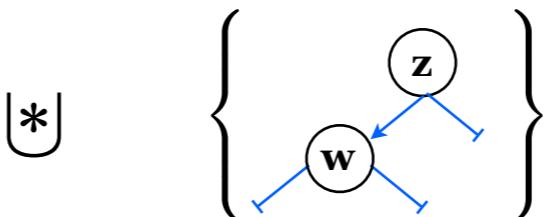
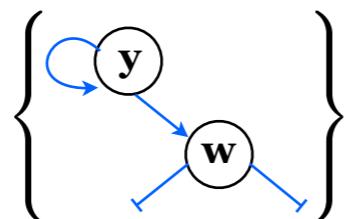


Example - Spanning Tree

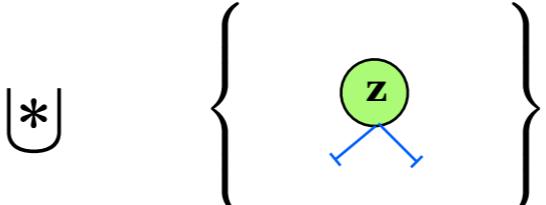
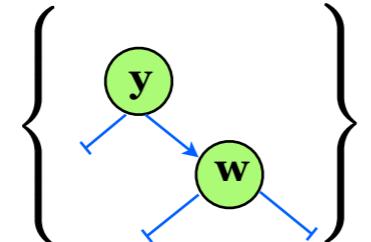
```
b := spanning(x) {  
    b := <CAS(x.m, 0, 1)>;
```



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



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



```
if (!b1) then
```

```
    x.l := null
```

```
if (!b2) then
```

```
    x.r := null
```

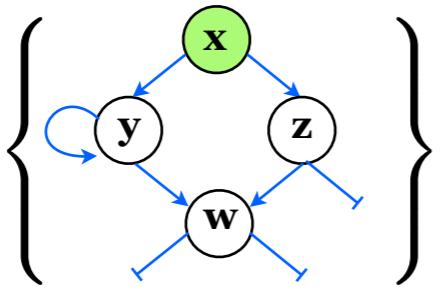
```
}
```

```
return b;
```

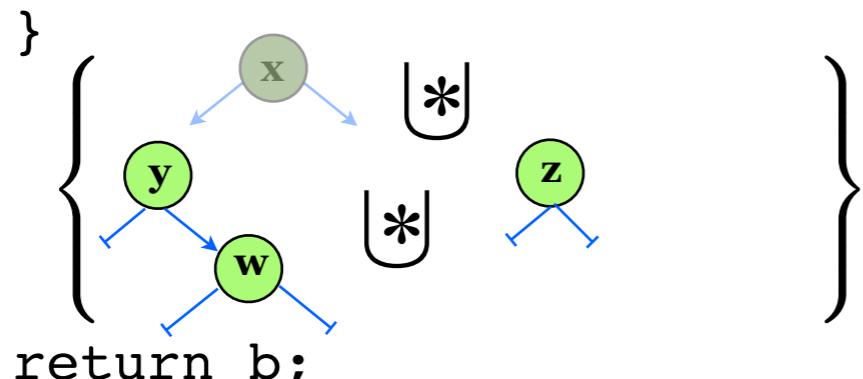
```
}
```

Example - Spanning Tree

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



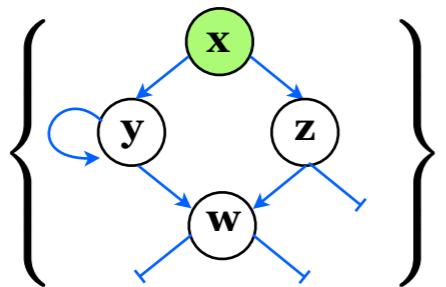
```
if (b) then {  
    b1:= spanning(x.l) || b2:= spanning(x.r);  
    if (!b1) then  
        x.l:= null  
    if (!b2) then  
        x.r:= null
```



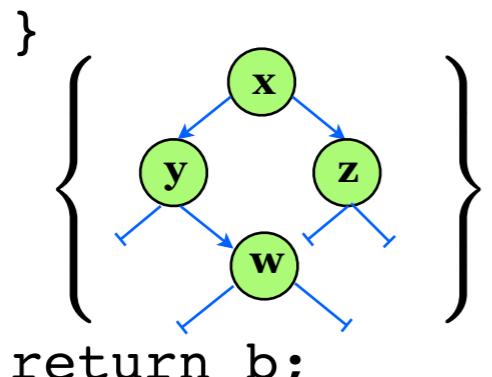
```
}
```

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
```

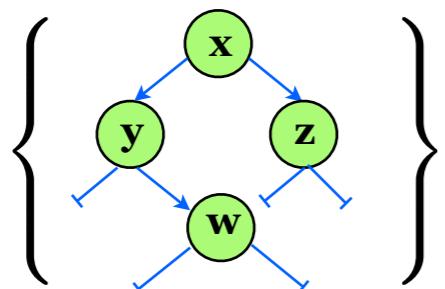


```
}
```

```
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
 - Local subjective reasoning
 - Dynamic framing on shared resources
 - Dynamic framing/rewriting of interference
 - More examples
 - Subjective thinking - can I tempt you?
 - CoLoSL in Iris?

Questions?

Thank you for listening