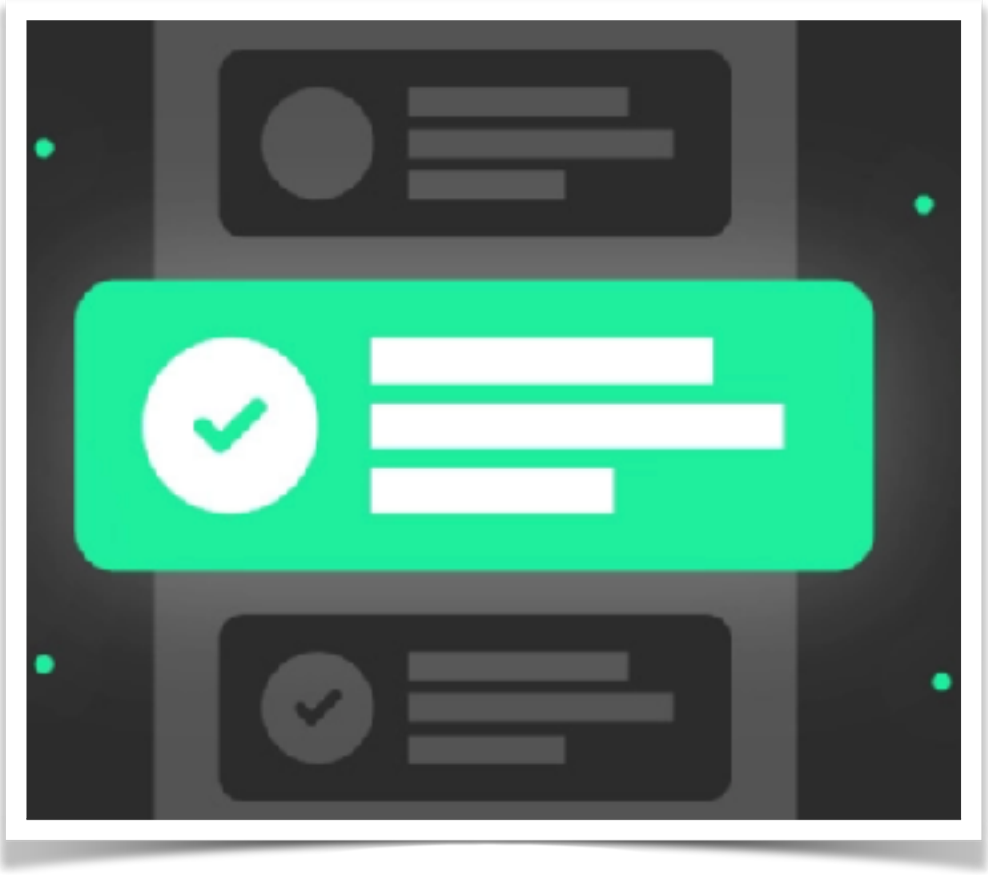


http://didawiki.di.unipi.it/doku.php/
magistraleinformatica/psc/start

PSC 2022/23 (375AA, 9CFU)

Principles for Software Composition

Roberto Bruni
http://www.di.unipi.it/~bruni/

21 - CCS at work

CCS syntax

p, q	$::=$	$\mathbf{nil}$	inactive process
	$\mid$	x	process variable (for recursion)
	$\mid$	$\mu.p$	action prefix
	$\mid$	$p\backslash\alpha$	restricted channel
	$\mid$	$p[\phi]$	channel relabelling
	$\mid$	$p + q$	nondeterministic choice (sum)
	$\mid$	$p \mid q$	parallel composition
	$\mid$	$\mathbf{rec}\ x.\ p$	recursion

(operators are listed in order of precedence)

Some notation

write $\sum_{i=1}^{n} p_i$ instead of $p_1 + \cdots + p_n$

write $\prod_{i=1}^{n} p_i$ instead of $p_1 | \cdots | p_n$

write $p \backslash \{a_1, ..., a_n\}$ instead of $p \backslash a_1 \cdots \backslash a_n$

write $\mu^n.p$ instead of $\underbrace{\mu.\mu....\mu.}_{n}\,p$

CCS op. semantics

Act) $$\frac{}{\mu.p \xrightarrow{\mu} p}$$

Res) $$\frac{p \xrightarrow{\mu} q \quad \mu \notin \{\alpha, \overline{\alpha}\}}{p\backslash\alpha \xrightarrow{\mu} q\backslash\alpha}$$

Rel) $$\frac{p \xrightarrow{\mu} q}{p[\phi] \xrightarrow{\phi(\mu)} q[\phi]}$$

SumL) $$\frac{p_1 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$$

SumR) $$\frac{p_2 \xrightarrow{\mu} q}{p_1 + p_2 \xrightarrow{\mu} q}$$

ParL) $$\frac{p_1 \xrightarrow{\mu} q_1}{p_1|p_2 \xrightarrow{\mu} q_1|p_2}$$

Com) $$\frac{p_1 \xrightarrow{\lambda} q_1 \quad p_2 \xrightarrow{\overline{\lambda}} q_2}{p_1|p_2 \xrightarrow{\tau} q_1|q_2}$$

ParR) $$\frac{p_2 \xrightarrow{\mu} q_2}{p_1|p_2 \xrightarrow{\mu} p_1|q_2}$$

Rec) $$\frac{p[{}^{\mathbf{rec}\ x.\ p}/_{x}] \xrightarrow{\mu} q}{\mathbf{rec}\ x.\ p \xrightarrow{\mu} q}$$

CCS
Encoding imperative languages

Preliminaries: termination

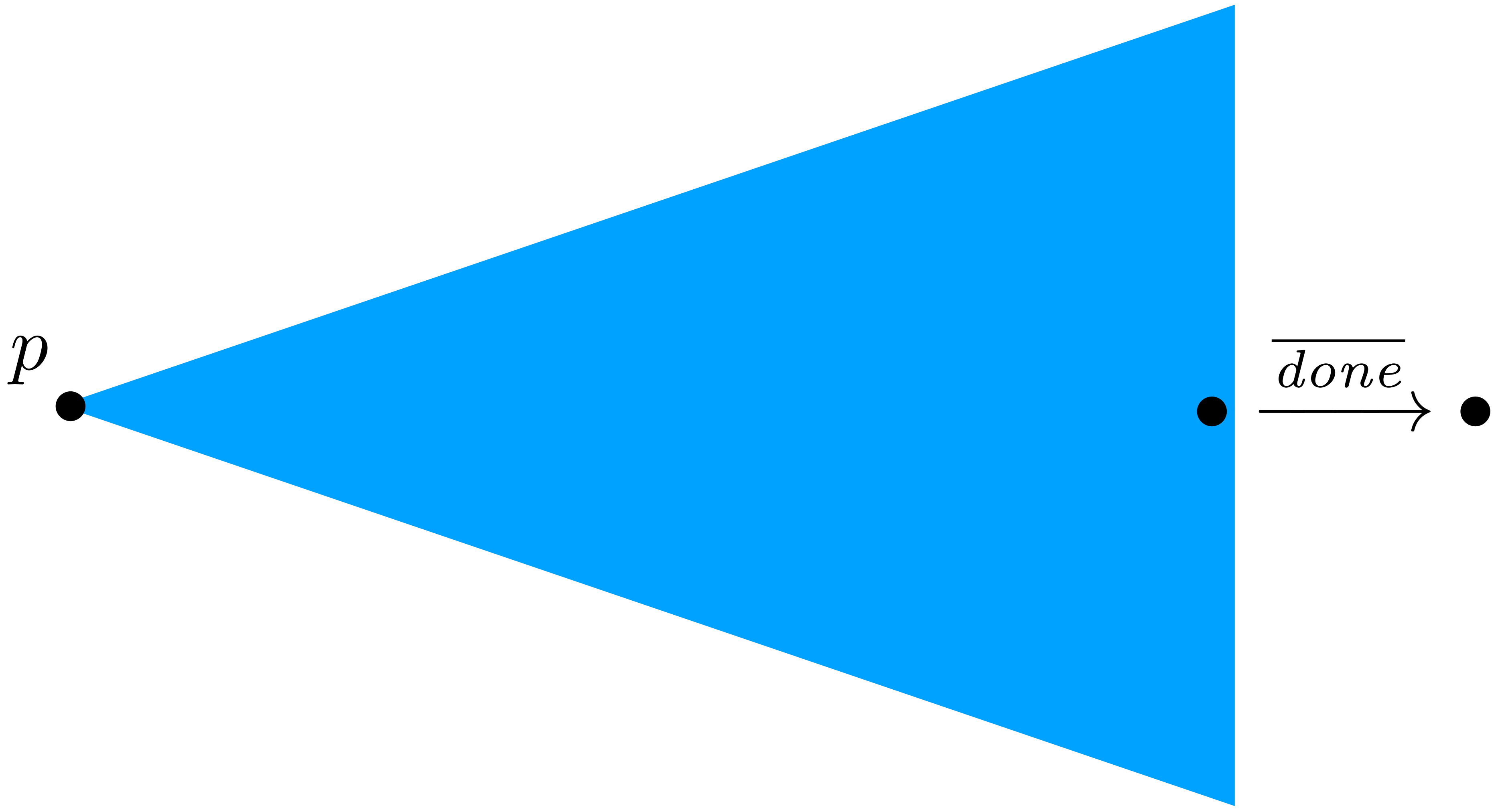
A dedicated channel *done*:

a message is sent when the current command terminates

$$\mathsf{Done} \triangleq \overline{done}$$

$$\mathsf{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Termination



Skip

skip

does nothing and sends *done*

$\tau.\mathsf{Done}$

$$\bullet \xrightarrow{\tau} \mathsf{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Variables

x ranging over $V = \{v_1, ..., v_n\}$

a dedicated process for managing each variable

we can read its current value (channel xr_i)

we can write any value (channel xw_i)

$$\begin{aligned}\mathsf{XW} &\triangleq \sum_{i=1}^{n} xw_i.\mathsf{X}_i \\ &= xw_1.\mathsf{X}_1 + \cdots + xw_n.\mathsf{X}_n\end{aligned}$$

$$\mathsf{X}_i \triangleq \overline{xr}_i.\mathsf{X}_i + \mathsf{XW}$$

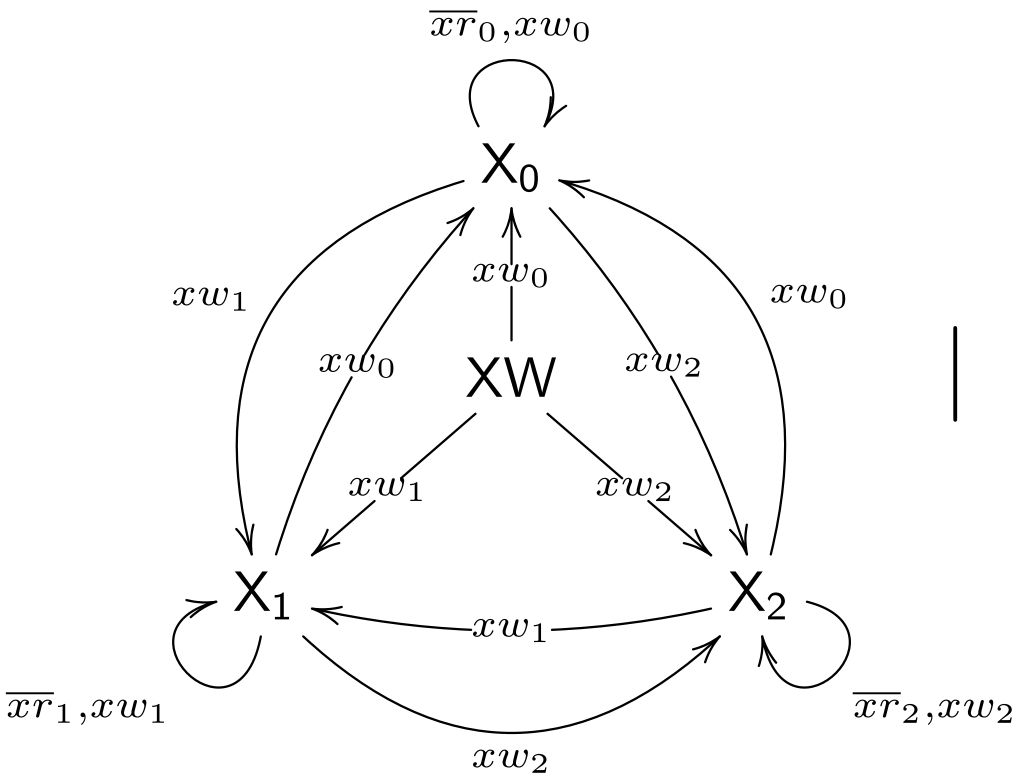
Variable declaration

var x

releases an uninitialised variable and terminates

$\mathsf{XW} | \mathsf{Done}$

$$V = \{v_0, v_1, v_2\}$$



$$\Big| \quad \mathsf{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Assignment

$$x := v_i$$

sends a message to change the state of the variable and then terminates

$$\overline{xw_i}.\mathsf{Done}$$

$$\bullet \xrightarrow{\overline{xw_i}} \mathsf{Done} \xrightarrow{\overline{done}} \mathbf{nil}$$

Sequential composition

$$c_1\ ;\ c_2$$

suppose p_1 models c_1

p_2 models c_2

$$p_1 | done.p_2$$

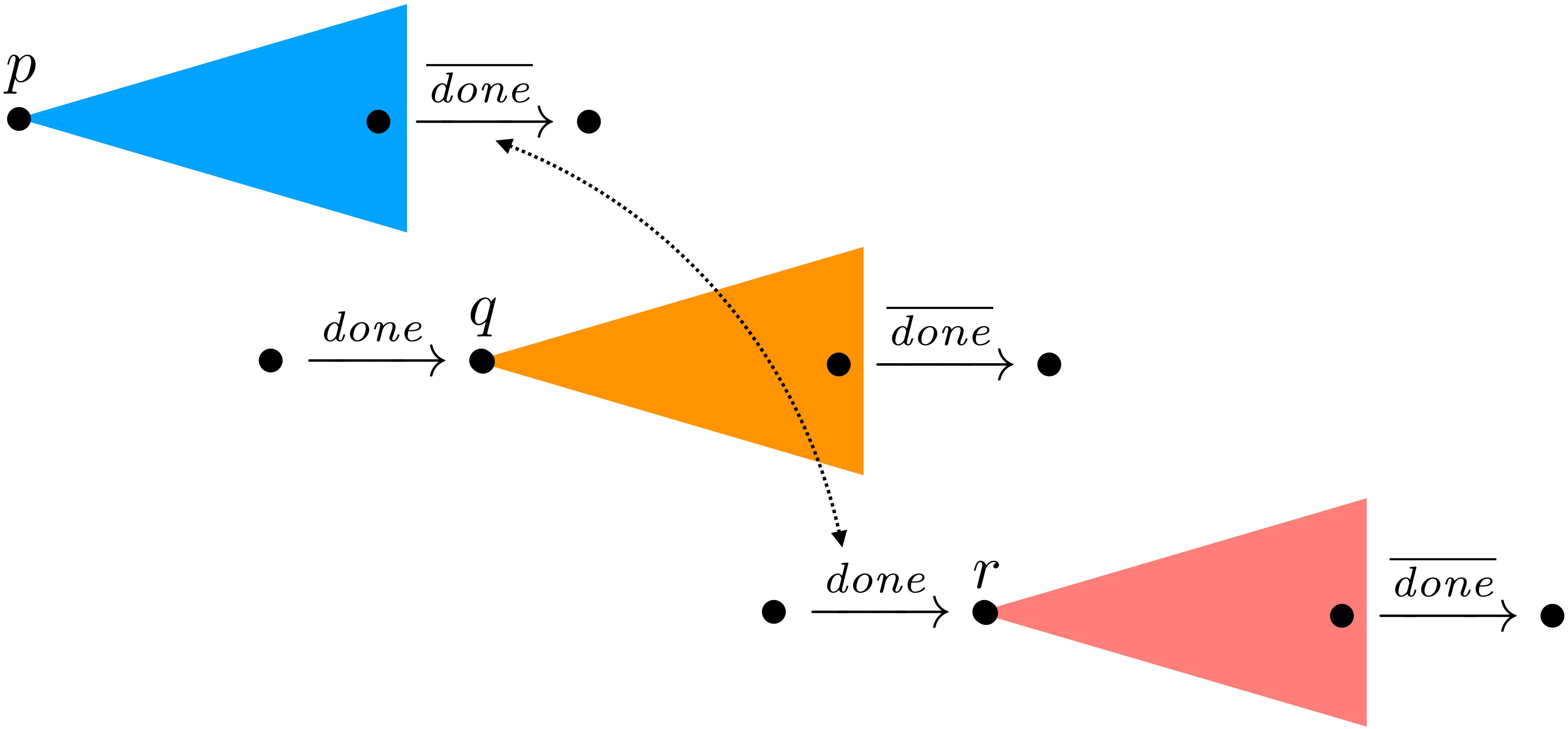
unfortunate choice: does not scale

$$c_1 ; c_2 ; c_3$$

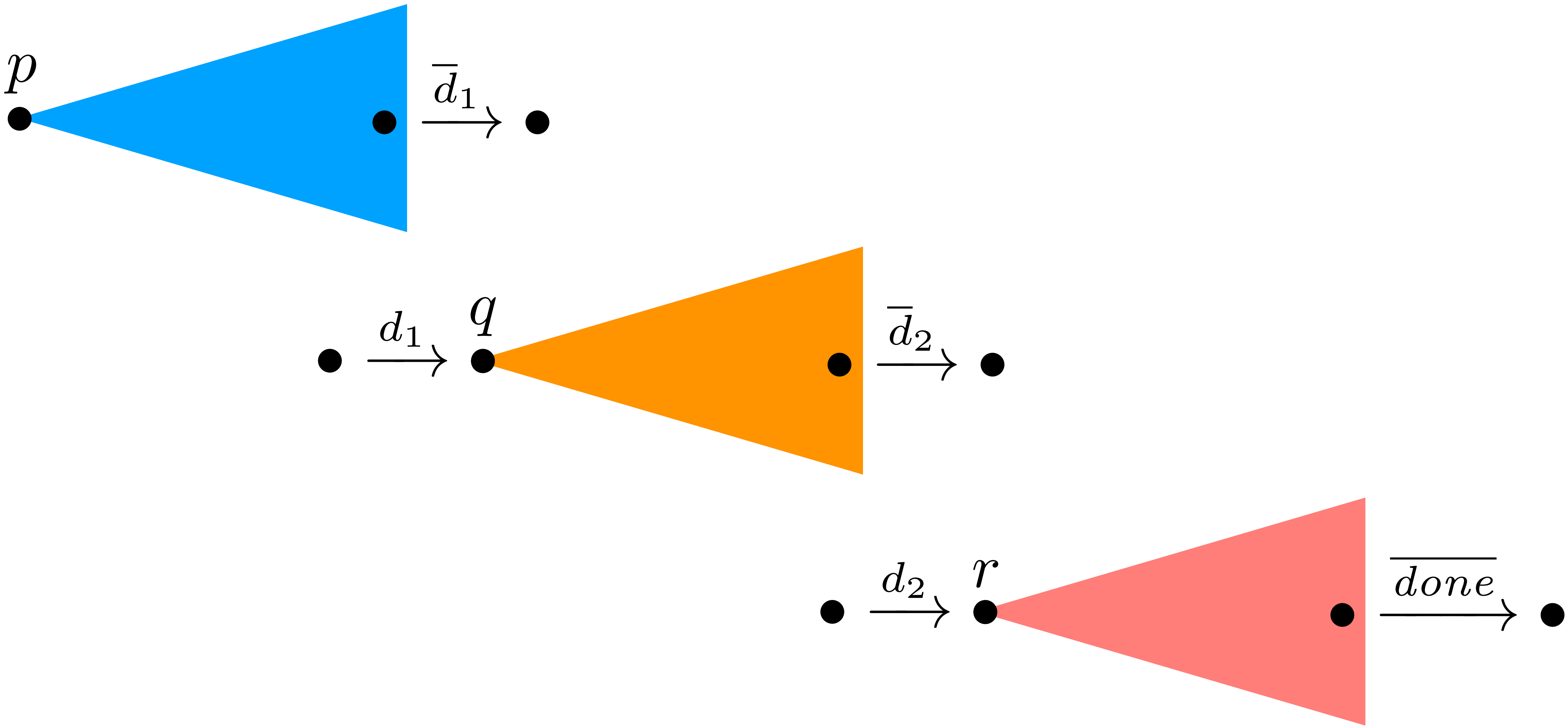
$$p_1 | done.p_2 | done.p_3$$

p_3 can start after p_1

Sequential?



Sequential!


p
$\xrightarrow{\overline{d_1}}$
$\xrightarrow{d_1}$
q
$\xrightarrow{\overline{d_2}}$
$\xrightarrow{d_2}$
r
$\xrightarrow{\overline{done}}$

Sequential compositon

$$c_1 \,;\, c_2$$

suppose p_1 models c_1

p_2 models c_2

$$\phi_d(done) = d$$

$$p_1 \frown p_2 \triangleq (p_1[\phi_d] | d.p_2)\backslash d$$

now d is local to p_1 and p_2

$$(\ ((p_1[\phi_{d_1}] \mid d_1.p_2)\backslash d_1)[\phi_{d_2}] \mid d_2.p_3)\backslash d_2$$

$$(\ ((p_1[\phi_d] \mid d.p_2)\backslash d)[\phi_d] \mid d.p_3)\backslash d$$

$$(p_1 \frown p_2) \frown p_3$$

Conditional

if $x = v_i$ then c_1 else c_2

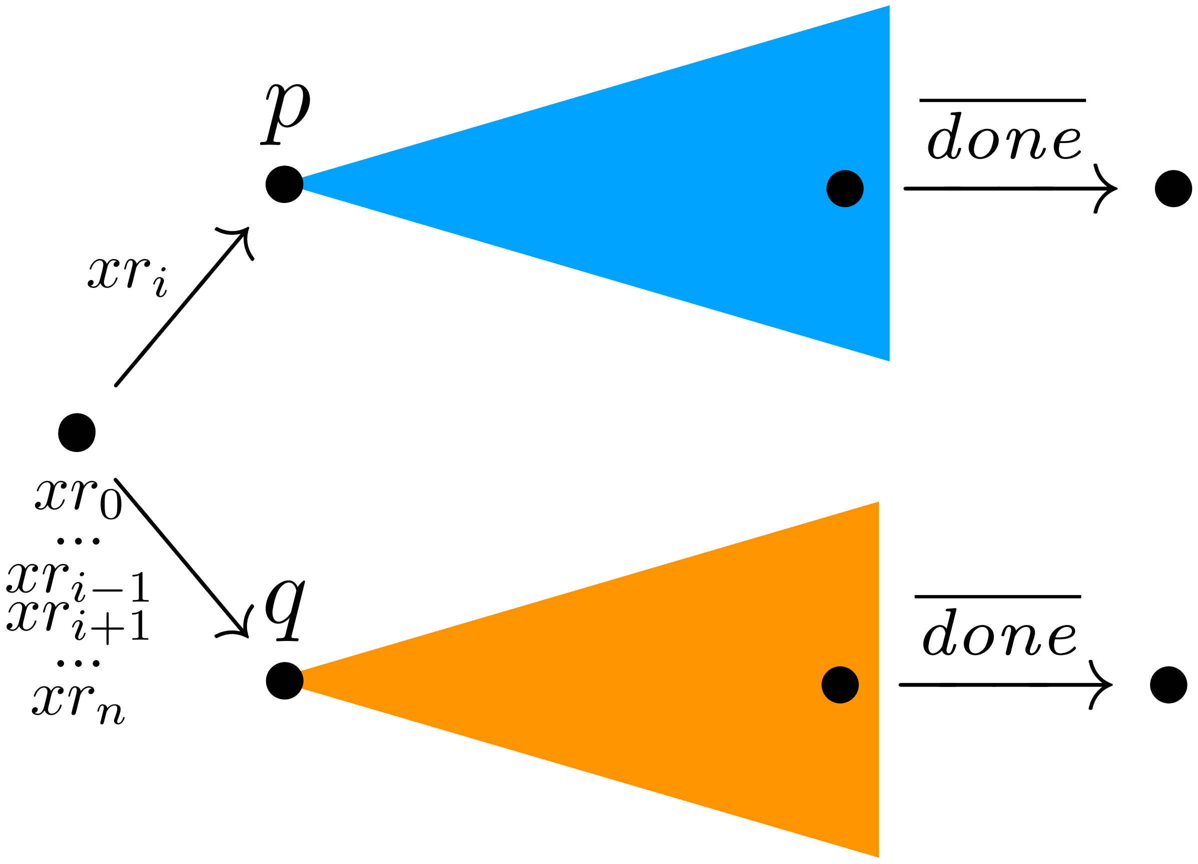
suppose p_1 models c_1

p_2 models c_2

receives the state of the variable
and then chooses accordingly

$$xr_i.p_1 + \sum_{j \neq i} xr_j.p_2$$

Conditional



Iteration

$$\text{while } x = v_i \text{ do } c$$

suppose p models c

receives the state of the variable
and then chooses accordingly, possibly recurring

$$\mathbf{rec}\ y.\ xr_i.(p[\phi_d]|d.y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p[\phi_d]|d.Y)\backslash d + \sum_{j \neq i} xr_j.\mathsf{Done}$$

$$Y \triangleq xr_i.(p \frown Y) + \sum_{j \neq i} xr_j.\mathsf{Done}$$

Concurrency

$$c_1 \mid c_2$$

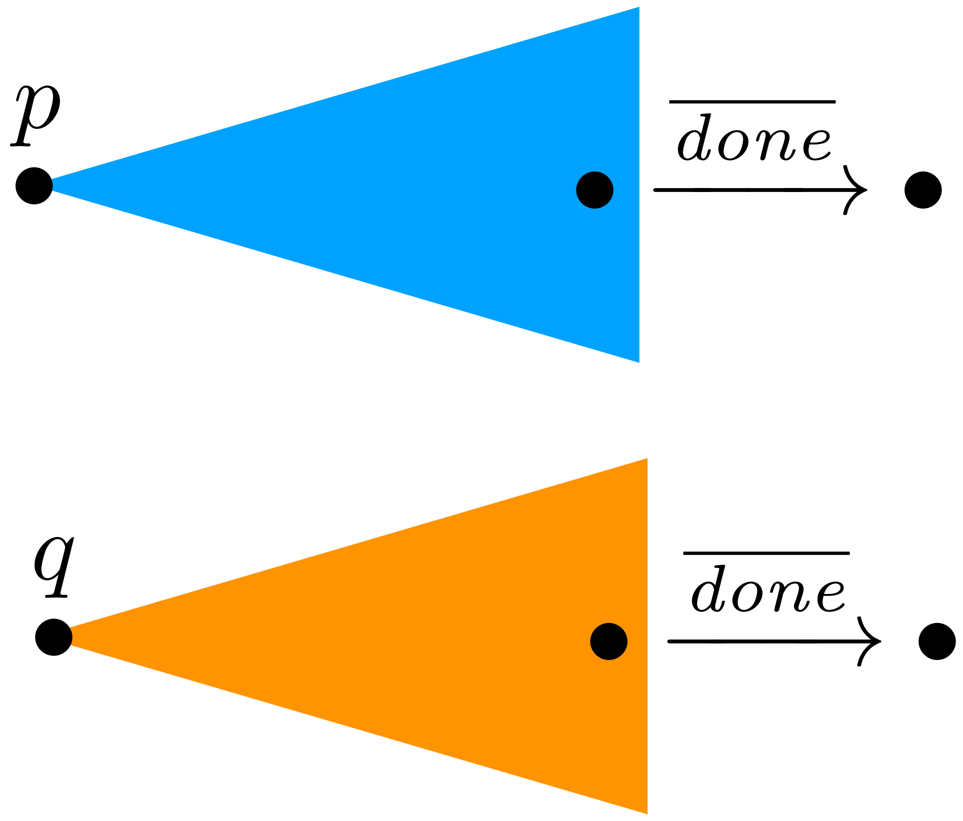
suppose p_1 models c_1

p_2 models c_2

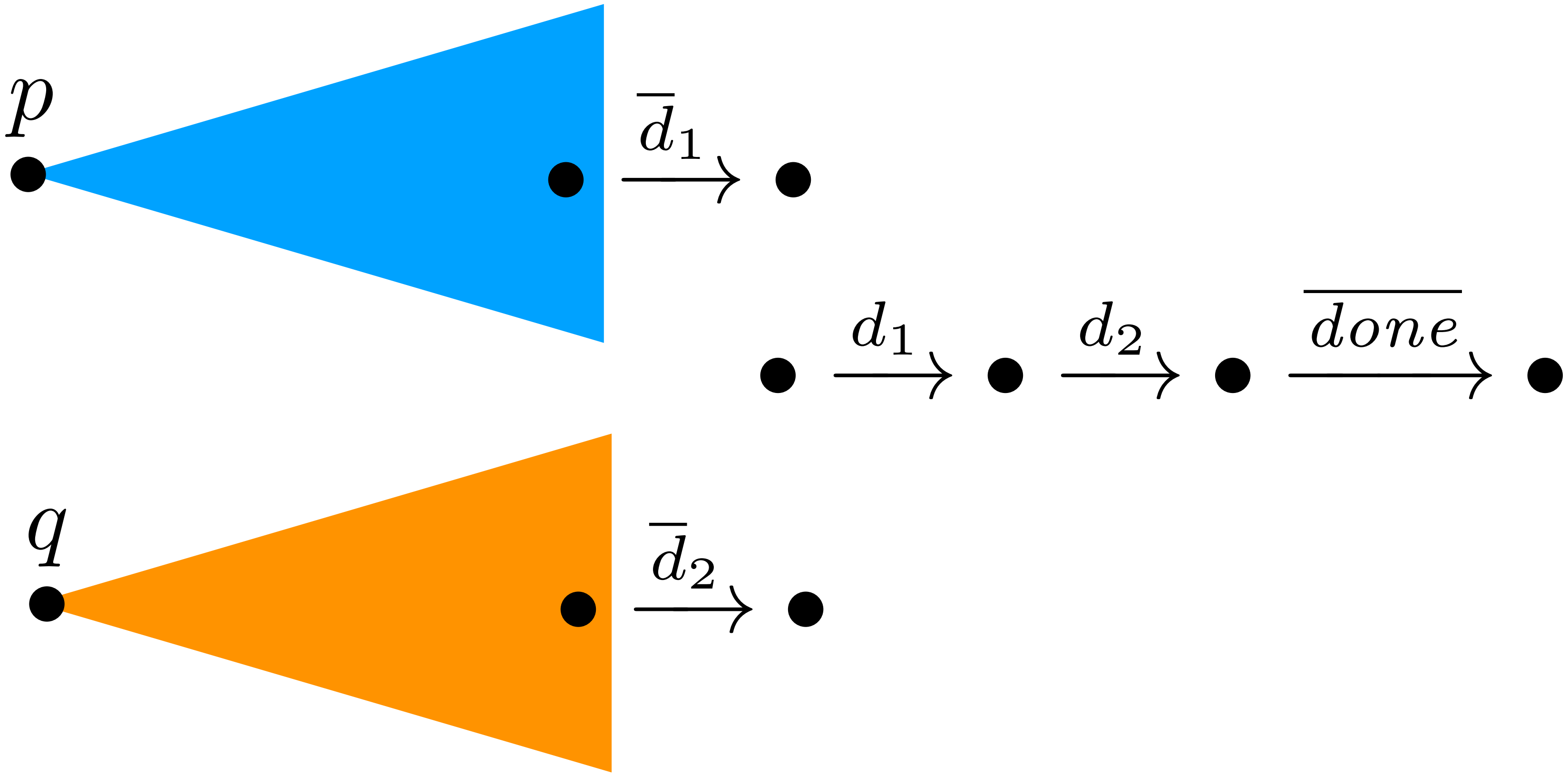
$$p_1 | p_2$$

unfortunate choice: *done* is possibly issued twice

Concurrent termination



Joint termination



Concurrency

$$c_1 \mid c_2$$

suppose p_1 models c_1

p_2 models c_2

$$(p_1[\phi_{d_1}] \mid p_2[\phi_{d_2}] \mid d_1.d_2.\mathsf{Done})\backslash d_1 \backslash d_2$$

the order is not important: both must be done to terminate

$$(p_1[\phi_d] \mid p_2[\phi_d] \mid d.d.\mathsf{Done})\backslash d$$

$$(p_1[\phi_d] \mid p_2[\phi_d] \mid d^2.\mathsf{Done})\backslash d$$

Finally

all channels for communicating with variables
must be restricted at the top
to guarantee read/write requests are synchronised

$$p\backslash\{xw_1, xr_1, \cdots\}$$

several optimisations are possible:
action prefix instead of linking for sequential composition
allows more expressive guards
remove silent transitions
introduce constants for loops
implement expressions
…

Example: optimisation

$$x := 1; y := 2$$

$$\overline{xw}_1.\overline{done} \quad \frown \quad \overline{yw}_2.\overline{done}$$

$$(\ (\overline{xw}_1.\overline{done})[\phi_d] \mid d.\overline{yw}_2.\overline{done}\ )\backslash d$$

$$\overline{xw}_1.\overline{yw}_2.\overline{done}$$

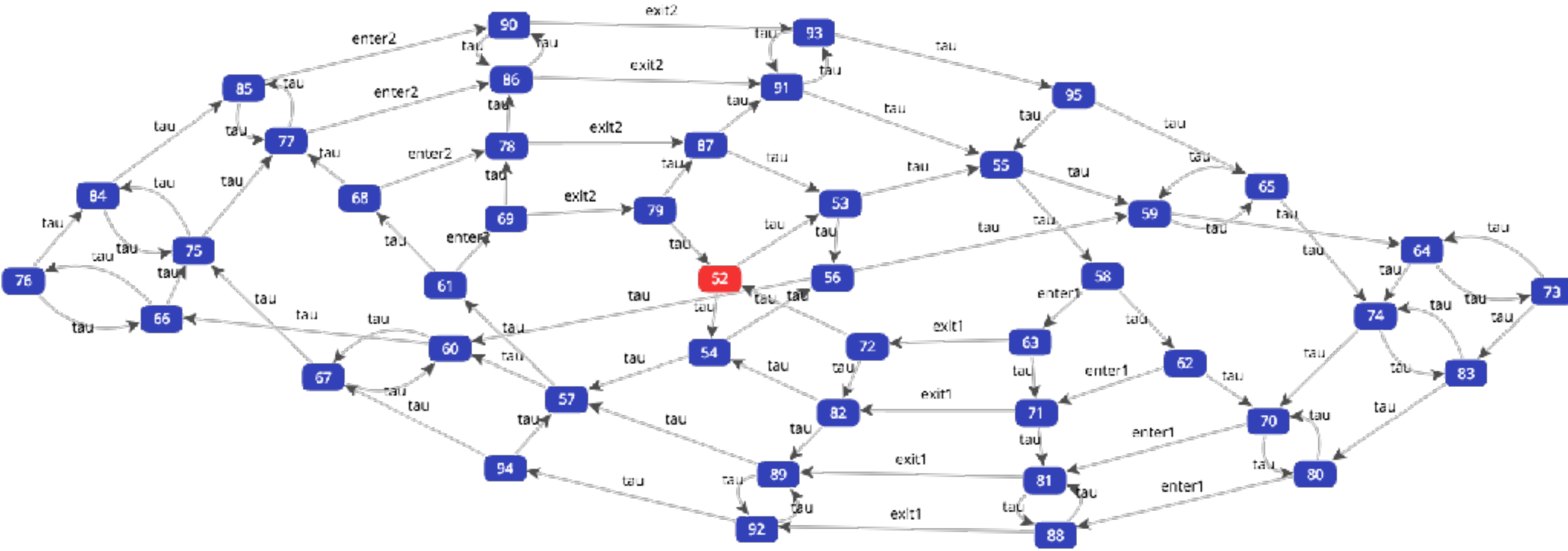
Example: optimisation

$$\textbf{if } b(x) \textbf{ then } c_1 \textbf{ else } c_2$$

$$\sum_{b(v_i)} xr_i.p_1 + \sum_{\neg b(v_j)} xr_j.p_2$$

CCS
Playing with CAAL

Concurrency Workbench, Aalborg Edition



http://caal.cs.aau.dk//

CAAL

CAAL is a web-based tool for
modeling, visualization and verification
of concurrent processes in CCS (and its timed extension)

developed for educational purposes

Edit (recursively defined) processes

Explore their LTS (collapse states up to equivalences)

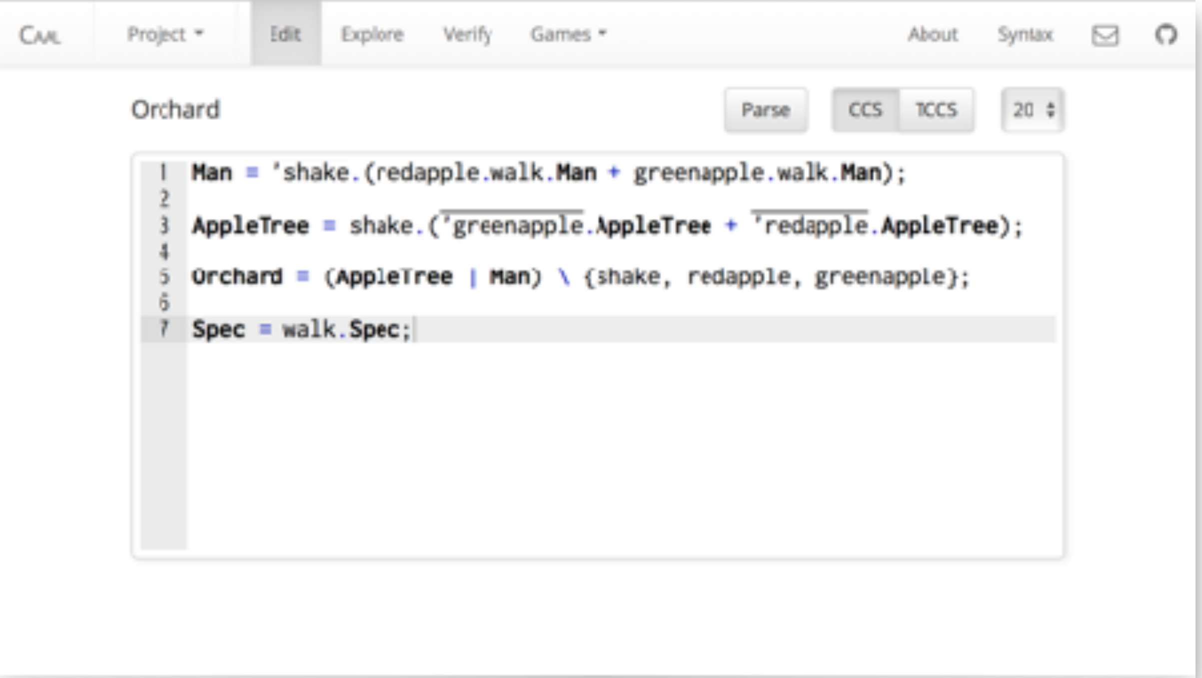
Verify HML formulas satisfaction (model checking)

Verify equivalences between processes
(generate distinguishing formulas)

play bisimulation **Game**

CAAL

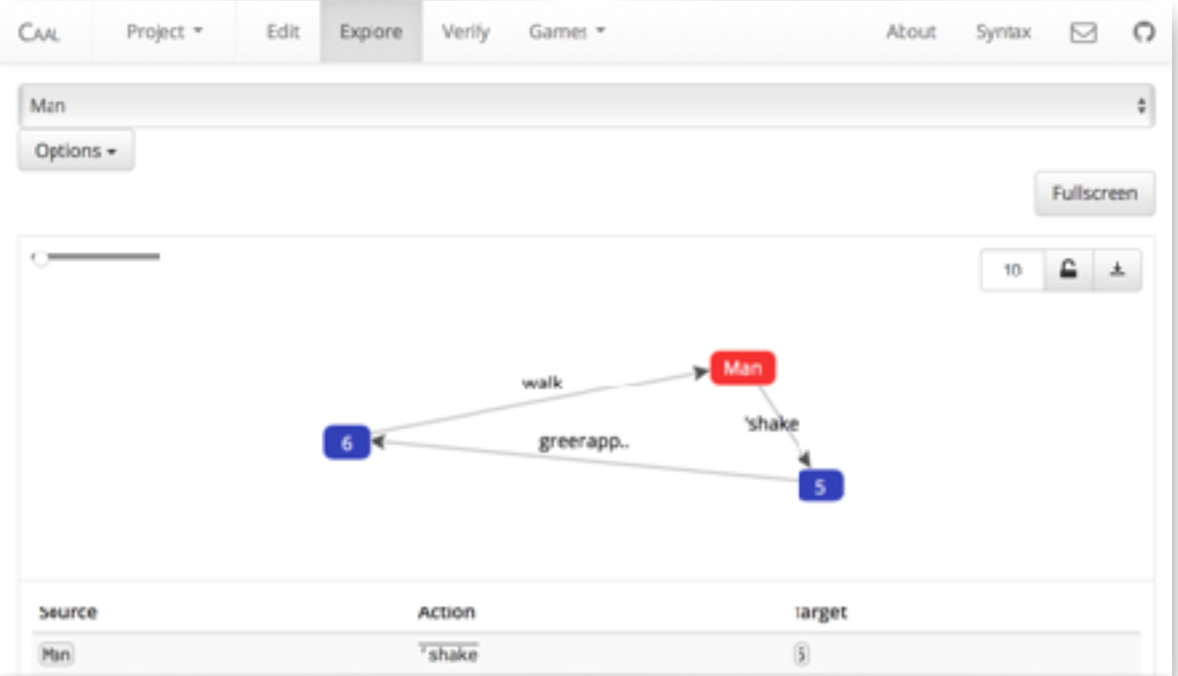
Edit processes



Orchard

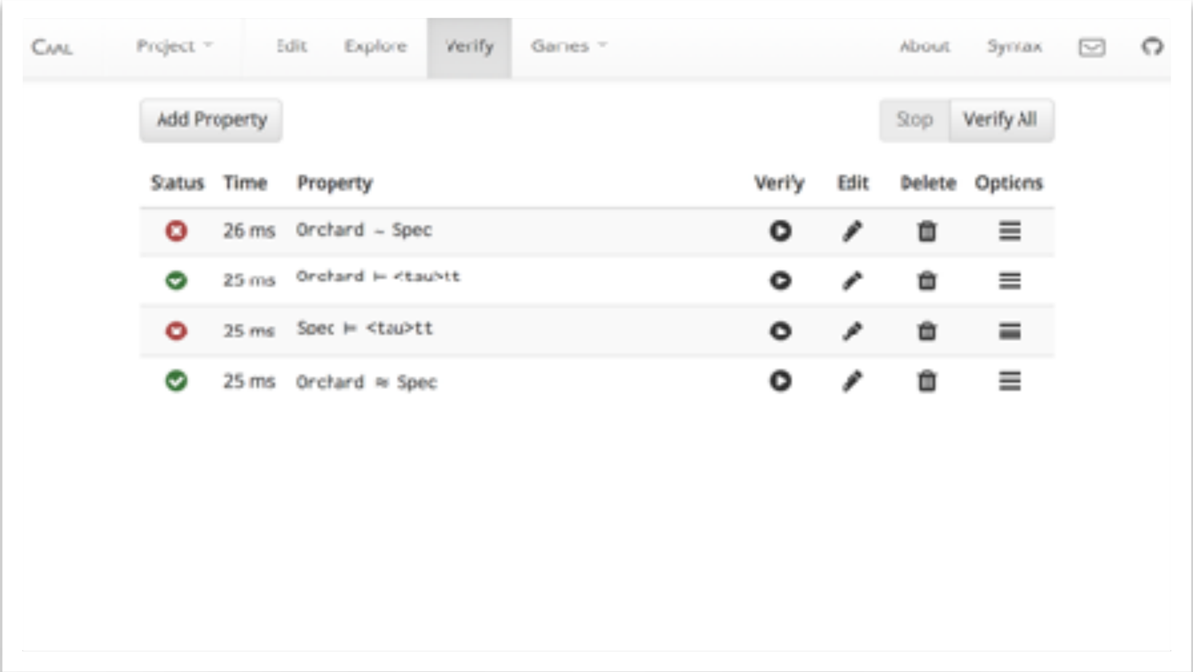
```
Man = 'shake.(redapple.walk.Man + greenapple.walk.Man);

AppleTree = shake.('greenapple.AppleTree + 'redapple.AppleTree);

Orchard = (AppleTree | Man) \ {shake, redapple, greenapple};

Spec = walk.Spec;
```

Explore LTS



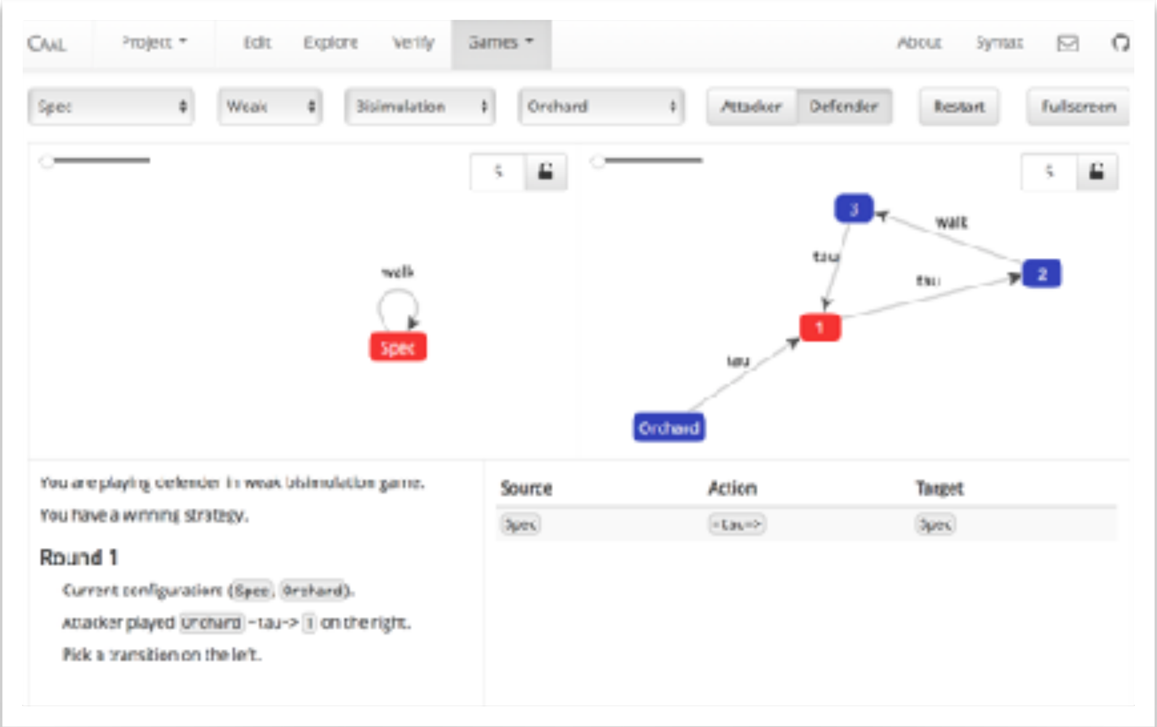
Source	Action	Target
Man	'shake	5

Verify HML & Equivalences



Status	Time	Property	Verify	Edit	Delete	Options
✗	26 ms	Orchard ∼ Spec	▶	✎	🗑	≡
✓	25 ms	Orchard ⊨ <tau>tt	▶	✎	🗑	≡
✗	25 ms	Spec ⊨ <tau>tt	▶	✎	🗑	≡
✓	25 ms	Orchard ≈ Spec	▶	✎	🗑	≡

play bisimulation **Game**



You are playing defender in weak bisimulation game.

You have a winning strategy.

Round 1

Current configuration: (Spec, Orchard).

Attacker played Orchard -tau-> 1 on the right.

Pick a transition on the left.

Source	Action	Target
Spec	=tau=>	Spec

CAAL syntax for CCS

<table>
<tr><th></th><th></th></tr>
<tr><td>$\mathbf{nil}$</td><td>0</td></tr>
<tr><td>$a.P \quad \overline{a}.P \quad \tau.P$</td><td>a.P ‘a.P tau.P</td></tr>
<tr><td>$P + Q$</td><td>P + Q</td></tr>
<tr><td>$P \mid Q$</td><td>P | Q</td></tr>
<tr><td>$P \backslash \{a1, ..., an\}$</td><td>P \ {a1,…,an}</td></tr>
<tr><td>$P[^{b1}/_{a1}, ...,^{bn}/_{an}]$</td><td>P[b1/a1,…,bn/an]</td></tr>
<tr><td>$A \triangleq P$</td><td>A = P ;</td></tr>
</table>

CAAL syntax for HML

$\mathbf{tt}$	tt
$\mathbf{ff}$	ff
$F \wedge G$	F and G
$F \vee G$	F or G
$\Diamond_{\{a1,\ldots,an\}} F$	⟨a1,...,an⟩F
$\Box_{\{a1,\ldots,an\}} F$	[a1,...,an]F

CAAL
mutual exclusion protocols analysis

Peterson's mex algorithm

```
% Two processes P1, P2
% Two boolean variables b1, b2 (both initially false)
% when Pi wants to enter the critical section, then it sets bi to true
% An integer variable k, taking values in {1,2}
% (initial value is arbitrary)
% the process Pk has priority over the other process
%
% Process P1 in pseudocode
while (true) {
    ...                             % non critical section
    b1 = true ;                     % P1 wants to enter the critical section
    k = 2 ;                         % P1 gives priority to the other process
    while (b2 && k==2) skip ;       % P1 waits its turn
    ...                             % P1 enters the critical section
    b1 = false                      % P1 leaves the critical section
}

% Process P2 is analogous to P1
```

Peterson's mex in CCS

$$\mathrm{B1W} \triangleq \mathit{b1wf}.\mathrm{B1f} + \mathit{b1wt}.\mathrm{B1t}$$

$$\mathrm{B1f} \triangleq \overline{\mathit{b1rf}}.\mathrm{B1f} + \mathrm{B1W}$$

$$\mathrm{B1t} \triangleq \overline{\mathit{b1rt}}.\mathrm{B1t} + \mathrm{B1W}$$

$$\mathrm{KW} \triangleq \mathit{kw1}.\mathrm{K1} + \mathit{kw2}.\mathrm{K2}$$

$$\mathrm{K1} \triangleq \overline{\mathit{kr1}}.\mathrm{K1} + \mathrm{KW}$$

$$\mathrm{K2} \triangleq \overline{\mathit{kr2}}.\mathrm{K2} + \mathrm{KW}$$

```
% Process P1 in pseudocode
while (true) {
    ...
    b1 = true ; % wants to enter
    k = 2 ; % gives priority to P2
    while (b2 && k==2) skip ; % waits
    ...   % enters critical section
    b1 = false % leaves
}

% Process P2 is analogous to P1
```

$$\mathrm{P1} \triangleq \overline{\mathit{b1wt}}.\overline{\mathit{kw2}}.\mathrm{P1a}$$

$$\mathrm{P1a} \triangleq \mathit{b2rf}.\mathrm{P1b} + \mathit{b2rt}.(\mathit{kr1}.\mathrm{P1b} + \mathit{kr2}.\mathrm{P1a})$$

leave cycle

loop

$$\mathrm{P1b} \triangleq \mathit{enter1}.\mathit{exit1}.\overline{\mathit{b1wf}}.\mathrm{P1}$$

just something to observe

$$\mathrm{SP} \triangleq (\mathrm{B1f}|\mathrm{B2f}|\mathrm{KW}|\mathrm{P1}|\mathrm{P2})\backslash\{\mathit{kw1}, \mathit{kr1}, \mathit{kw2}, \mathit{kr2}, \ldots\}$$

Peterson's mex in CCS

$$\mathsf{B1W} \triangleq \mathit{b1wf}.\mathsf{B1f} + \mathit{b1wt}.\mathsf{B1t}$$

$$\mathsf{B1f} \triangleq \overline{\mathit{b1rf}}.\mathsf{B1f} + \mathsf{B1W}$$

$$\mathsf{B1t} \triangleq \overline{\mathit{b1rt}}.\mathsf{B1t} + \mathsf{B1W}$$

$$\mathsf{KW} \triangleq \mathit{kw1}.\mathsf{K1} + \mathit{kw2}.\mathsf{K2}$$

$$\mathsf{K1} \triangleq \overline{\mathit{kr1}}.\mathsf{K1} + \mathsf{KW}$$

$$\mathsf{K2} \triangleq \overline{\mathit{kr2}}.\mathsf{K2} + \mathsf{KW}$$

```
% Process P1 in pseudocode
while (true) {
    ...
    b1 = true ; % wants to enter
    k = 2 ; % gives priority to P2
    while (b2 && k==2) skip ; % waits
    ...  % enters critical section
    b1 = false % leaves
}

% Process P2 is analogous to P1
```

$$\mathsf{P1} \triangleq \overline{\mathit{b1wt}}.\overline{\mathit{kw2}}.\mathsf{P1a}$$

no busy wait

$$\mathsf{P1a} \triangleq \mathit{b2rf}.\mathsf{P1b} + \mathit{kr1}.\mathsf{P1b}$$

$$\mathsf{P1b} \triangleq \mathit{enter1}.\mathit{exit1}.\overline{\mathit{b1wf}}.\mathsf{P1}$$

$$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{P1}|\mathsf{P2})\backslash\{\mathit{kw1}, \mathit{kr1}, \mathit{kw2}, \mathit{kr2}, ...\}$$

Peterson's mex in CCS

$$\mathsf{P1} \triangleq \overline{b1wt}.\overline{kw2}.\mathsf{P1a}$$

$$\mathsf{P1a} \triangleq b2rf.\mathsf{P1b} + kr1.\mathsf{P1b}$$

$$\mathsf{P1b} \triangleq enter1.exit1.\overline{b1wf}.\mathsf{P1}$$

$$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{P1}|\mathsf{P2})\backslash\{kw1, kr1, kw2, kr2, ...\}$$

a formula for mutual exclusion?

any label

$$F \triangleq [exit_1]\mathbf{ff} \vee [exit_2]\mathbf{ff}$$

$$F \wedge [-](\cdots)$$

$$F \wedge [-](F \wedge [-](\cdots))$$

$$MEX \underset{\text{(max)}}{\triangleq} F \wedge [-]MEX$$

$$F \wedge [-](F \wedge [-](F \wedge [-](\cdots)))$$

a recursively defined formula!

Peterson's mex in CCS

$\text{P1} \triangleq \overline{b1wt}.\overline{kw2}.\text{P1a}$

$\text{P1a} \triangleq b2rf.\text{P1b} + kr1.\text{P1b}$

$\text{P1b} \triangleq enter1.exit1.\overline{b1wf}.\text{P1}$

$\text{SP} \triangleq (\text{B1f}|\text{B2f}|\text{KW}|\text{P1}|\text{P2})\backslash\{kw1, kr1, kw2, kr2, ...\}$

has P1 the possibility to enter?

$G \triangleq \langle enter_1 \rangle \mathbf{tt}$

$G \vee \langle - \rangle (\cdots)$

$G \vee \langle - \rangle (G \vee \langle - \rangle (\cdots))$

$EN \underset{\text{(min)}}{\triangleq} G \vee \langle - \rangle EN$

$G \vee \langle - \rangle (G \vee \langle - \rangle (G \vee \langle - \rangle (\cdots)))$

a recursively defined formula!

Peterson's mex in CCS

$\mathsf{P1} \triangleq \overline{\mathit{b1wt}}.\overline{\mathit{kw2}}.\mathsf{P1a}$

$\mathsf{P1a} \triangleq \mathit{b2rf}.\mathsf{P1b} + \mathit{kr1}.\mathsf{P1b}$

$\mathsf{P1b} \triangleq \mathit{enter1}.\mathit{exit1}.\overline{\mathit{b1wf}}.\mathsf{P1}$

$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{P1}|\mathsf{P2})\backslash\{\mathit{kw1}, \mathit{kr1}, \mathit{kw2}, \mathit{kr2}, ...\}$

has P1 the possibility to enter from any reachable state ?

$$EN \underset{\text{(min)}}{\triangleq} G \vee \langle - \rangle EN \qquad EN \wedge [-](\cdots)$$

$$EN \wedge [-](EN \wedge [-](\cdots))$$

$$AEN \underset{\text{(max)}}{\triangleq} EN \wedge [-]AEN \qquad EN \wedge [-](EN \wedge [-](EN \wedge [-](\cdots)))$$

a recursively defined formula!

Peterson's mex in CCS

$$\mathsf{P1} \triangleq \overline{b1wt}.\overline{kw2}.\mathsf{P1a}$$

$$\mathsf{P1a} \triangleq b2rf.\mathsf{P1b} + kr1.\mathsf{P1b}$$

$$\mathsf{P1b} \triangleq enter1.exit1.\overline{b1wf}.\mathsf{P1}$$

$$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{P1}|\mathsf{P2})\backslash\{kw1, kr1, kw2, kr2, ...\}$$

deadlock freedom?

$$H \triangleq \langle - \rangle \mathbf{tt}$$

$$H \wedge [-](\cdots)$$

$$H \wedge [-](H \wedge [-](\cdots))$$

$$DF \underset{\text{(max)}}{\triangleq} H \wedge [-]DF$$

$$H \wedge [-](H \wedge [-](H \wedge [-](\cdots)))$$

a recursively defined formula!

Peterson's mex in CCS

$$\mathsf{P1} \triangleq \overline{\mathit{b1wt}}\boxed{.\mathit{req}_1.}\overline{\mathit{kw2}}.\mathsf{P1a}$$

$$\mathsf{P1a} \triangleq \mathit{b2rf}.\mathsf{P1b} + \mathit{kr1}.\mathsf{P1b}$$

$$\mathsf{P1b} \triangleq \mathit{enter1}.\mathit{exit1}.\overline{\mathit{b1wf}}.\mathsf{P1}$$

$$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{P1}|\mathsf{P2})\backslash\{\mathit{kw1}, \mathit{kr1}, \mathit{kw2}, \mathit{kr2}, ...\}$$

does P1 access every time it issues a request?

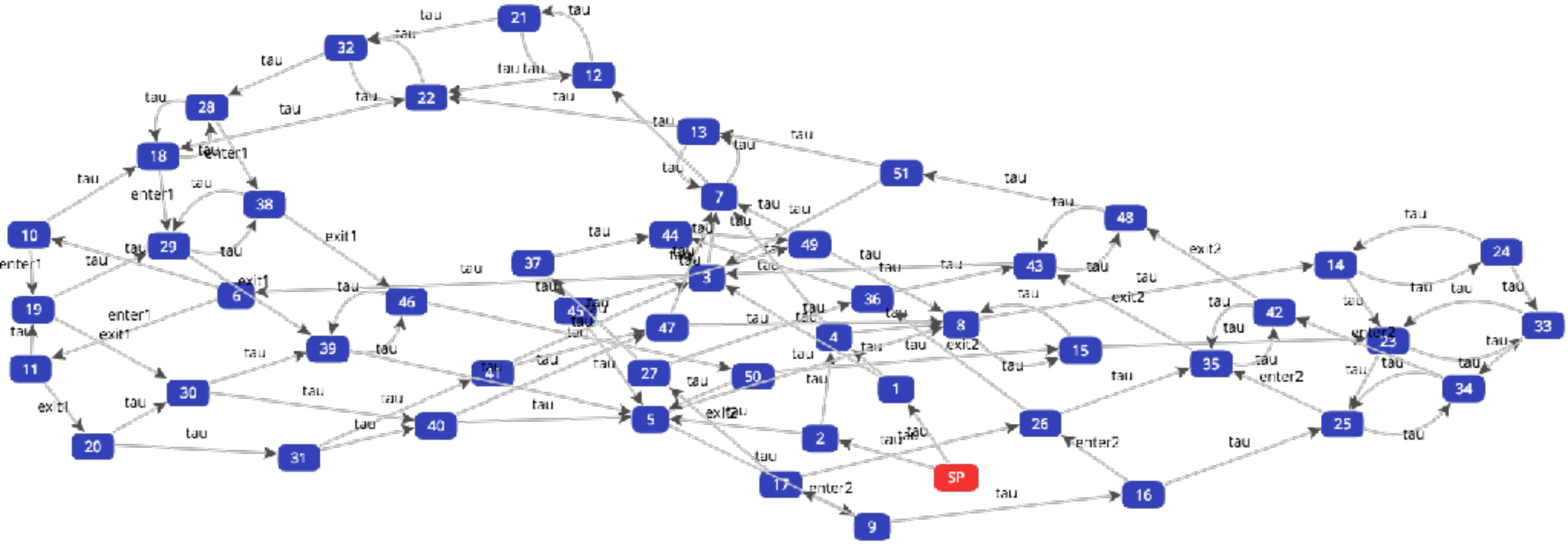
$$A \underset{\text{(min)}}{\triangleq} \langle \mathit{exit}_1 \rangle \mathbf{tt} \vee [-]A$$

$$REQ \underset{\text{(max)}}{\triangleq} [\mathit{req}_1]A \wedge [-]REQ$$

a recursively defined formula!

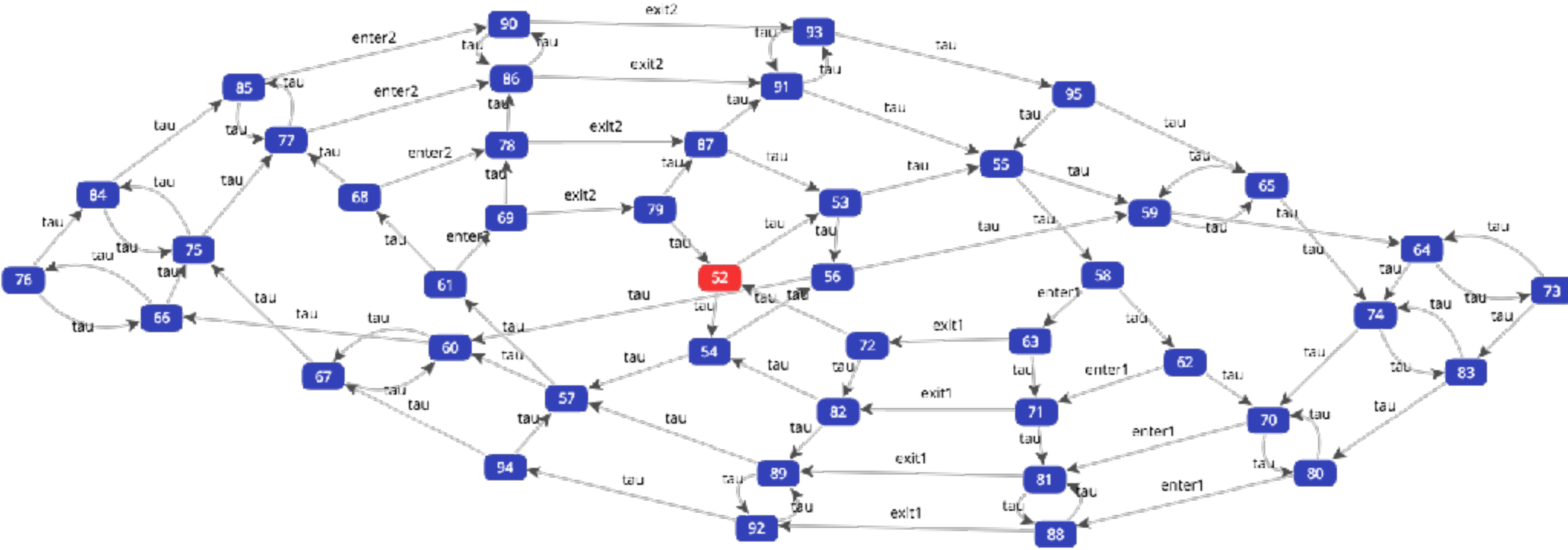
Peterson's mex in CAAL

LTS



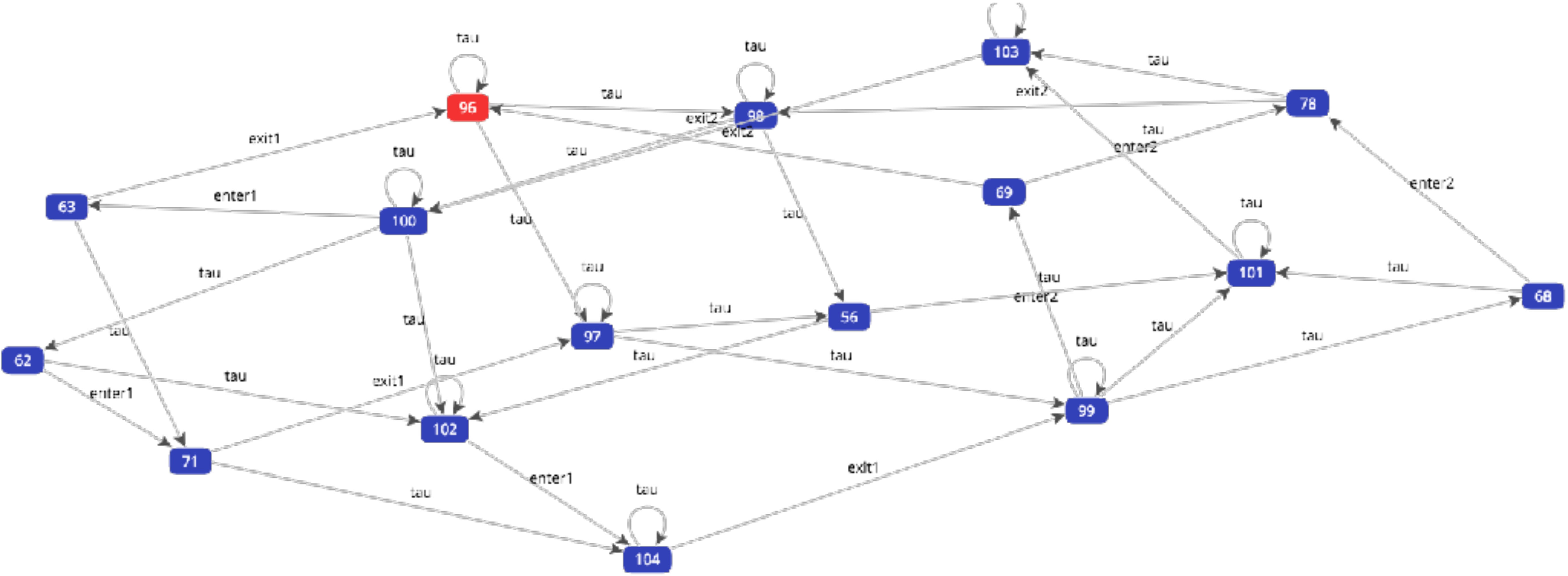
Peterson's mex in CAAL

LTS up to strong bisimilarity



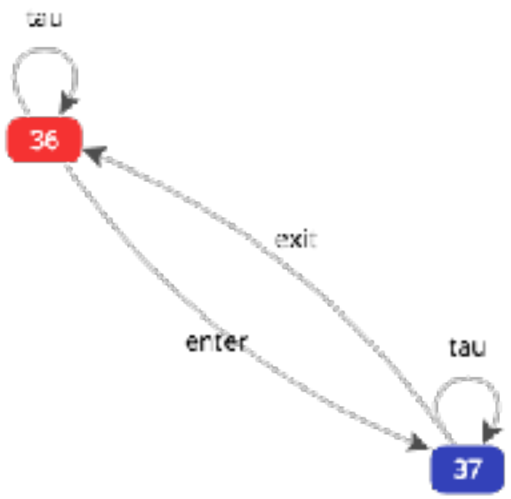
Peterson's mex in CAAL

LTS up to weak bisimilarity



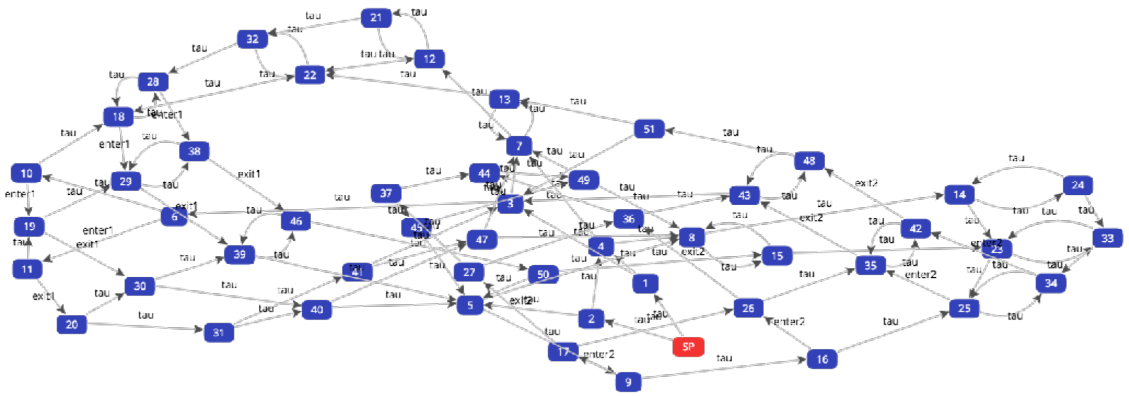
Peterson's mex in CAAL

LTS up to weak bisimilarity, after renaming *enter1/2* to *enter* and *exit1/2* to *exit*

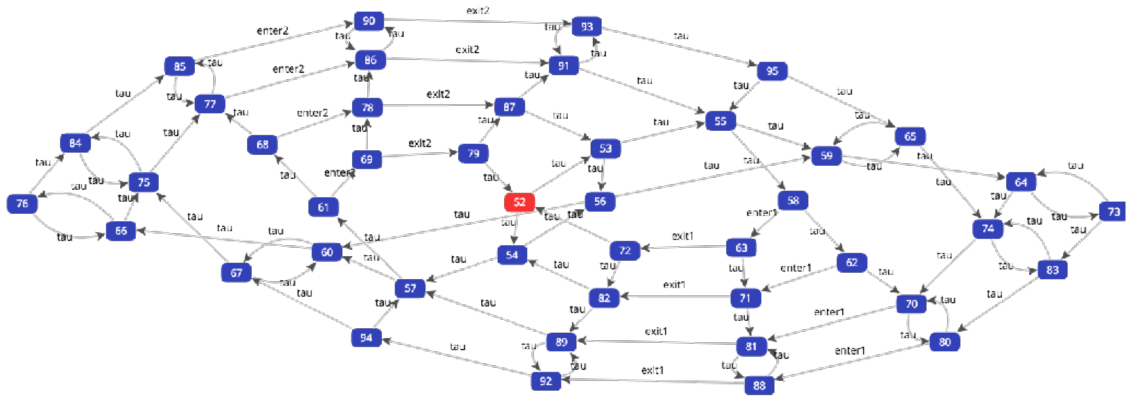


Peterson's mex in CAAL

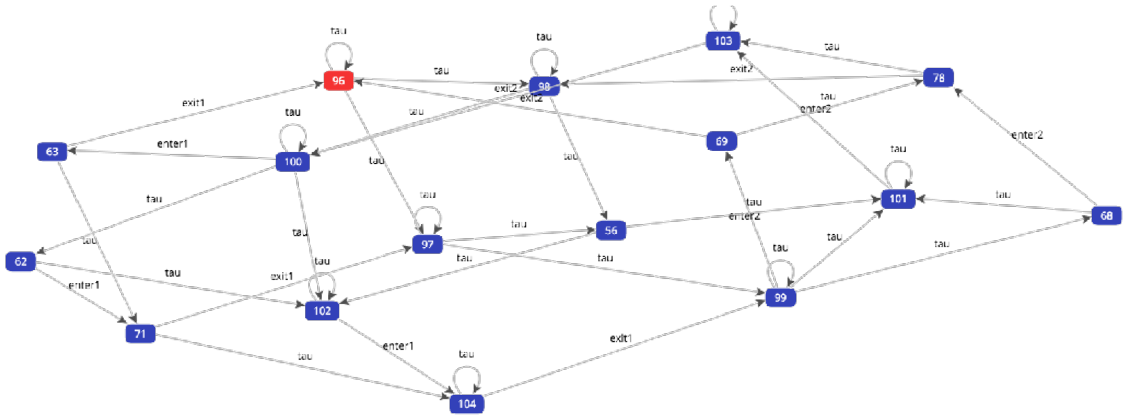
LTS



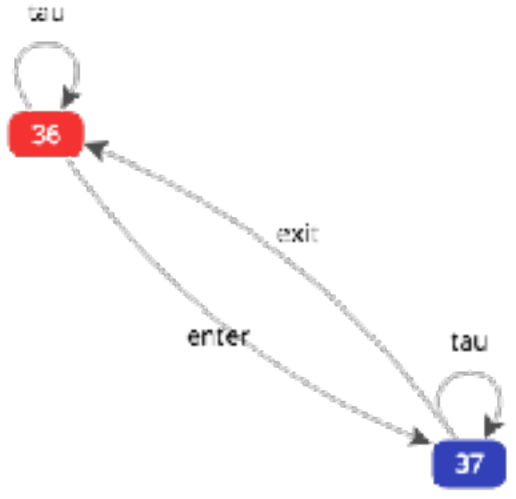
LTS up to strong bisimilarity



LTS up to weak bisimilarity



LTS up to weak bisimilarity, *enter* and *exit*

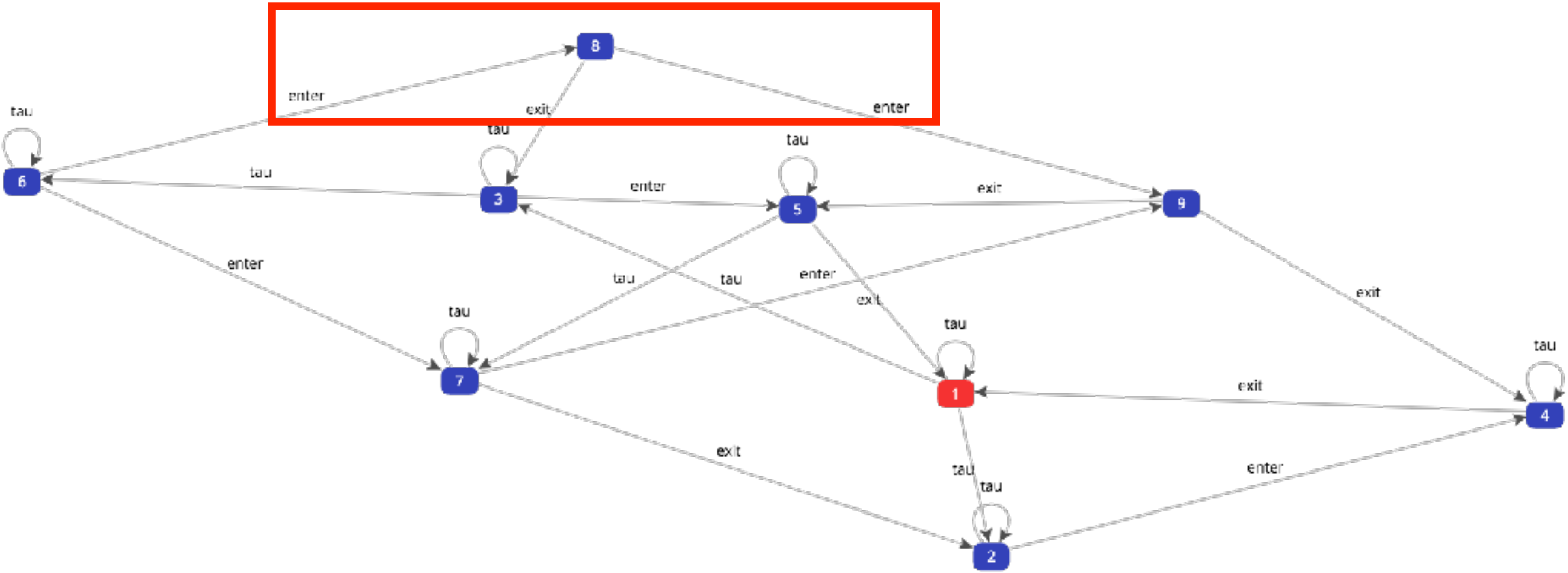


Hyman's mex algorithm

```
% Two processes H1, H2
% Two boolean variables b1, b2 (both initially false)
% when Hi wants to enter the critical section, then it sets bi to true
% An integer variable k, taking values in {1,2}
% (initial value is arbitrary)
% the process Hk has priority over the other process
%
% Process H1 in pseudocode
while (true) {
    ...                       % non critical section
    b1 = true ;               % H1 wants to enter the critical section
    while (k==2) {            % while H2 has priority
        while (b2) skip ;     % H1 waits
        k = 1;                % H1 sets priority to itself
    }
    ...                       % H1 enters the critical section
    b1 = false                % H1 leaves the critical section
}

% Process H2 is analogous to H1
```

Peterson's mex in CAAL

LTS up to weak bisimilarity, after renaming *enter1/2* to *enter* and *exit1/2* to *exit*



50 prisoners puzzle

50 prisoners kept in separate cells got a chance to be released:
from time to time one of them will be carried in a special room and then back to cell.
(in no particular order, possibly multiple times consecutively, but fairly to avoid infinite wait)

The room is completely empty except for a switch that can turn the light on or off
(the light is not visible from outside).

At any time, if any of them claims rightfully that all the prisoners have already entered the room at least once, then all prisoners will be released
(but if it proves wrong, then the chance ends and they will never be released).

The prisoners have the possibility to discuss in advance some protocol to follow
(not all prisoner must behave in the same way).

Can you find a winning strategy for the prisoners?
Can you formalise it in CCS (for 2 to 4 prisoners)?

- Easy case: it is known that the light in the room is initially off
- Hard case: the initial state of the light in the room is not known.