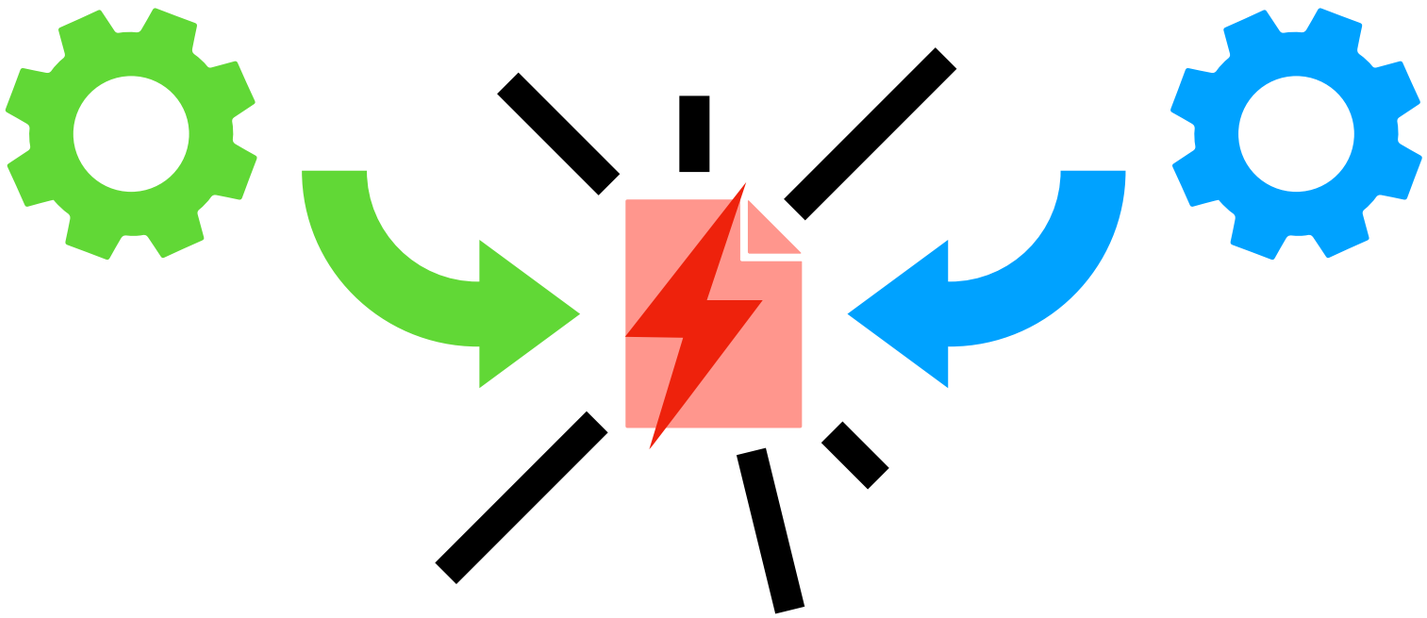
Linguaggi di Programmazione



Roberta Gori

Applicazioni del CCS

CCS: sintassi

$$
\begin{array}{lcll}
p, q & ::= & \mathbf{nil} & \text{processo inattivo} \\
 & | & x & \text{variabile di processo (per la ricorsione)} \\
 & | & \mu.p & \text{prefisso azione} \\
 & | & p\backslash\alpha & \text{canale ristretto} \\
 & | & p[\phi] & \text{rietichettatura del canale} \\
 & | & p + q & \text{scelta nondeterministica (somma)} \\
 & | & p|q & \text{composizione parallela} \\
 & | & \mathbf{rec}\ x.\ p & \text{ricorsione}
\end{array}
$$

(gli operatori sono elencati in ordine di precedenza)

Un po' di notazione

scriviamo $\sum_{i=1}^{n} p_i$ invece di $p_1 + \cdots + p_n$

scriviamo $\prod_{i=1}^{n} p_i$ invece di $p_1 | \cdots | p_n$

scriviamo $p \backslash \{a_1, ..., a_n\}$ invece di $p \backslash a_1 \cdots \backslash a_n$

scriviamo $\mu^n.p$ invece di $\underbrace{\mu.\mu....\mu.}_{n} p$

CCS op. semantica

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

CCS

Codificare un linguaggio imperativo

Prima cosa: terminazione

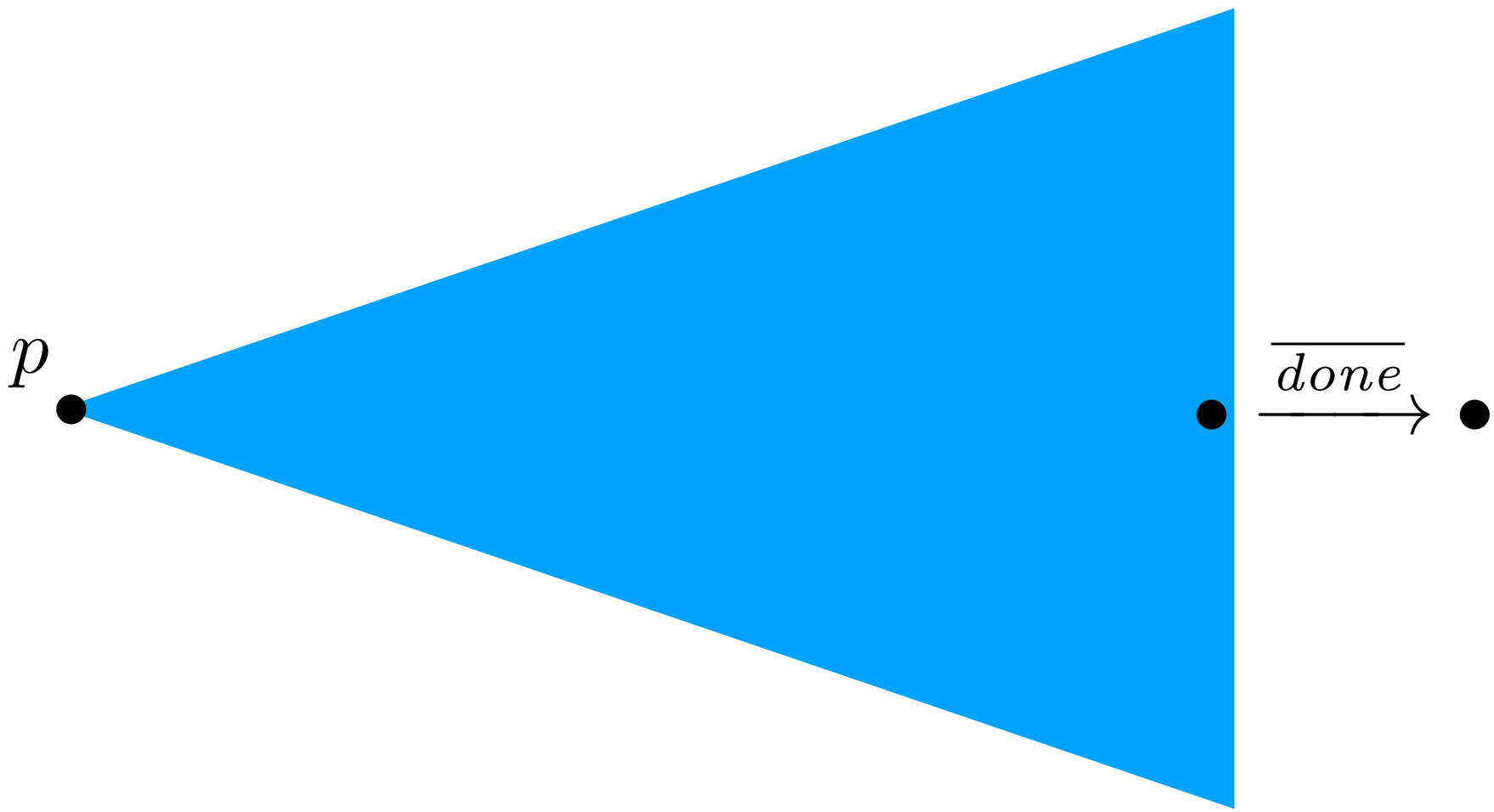
Un canale dedicato *done*:

viene inviato un messaggio quando il comando corrente termina

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

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

Terminazione



Skip

skip

non fa nulla invia *done*

$$\tau.\text{Done}$$

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

Variabili

x che varia su $V = \{v_1, ..., v_n\}$

un processo dedicato alla gestione di ogni variabile

possiamo leggere il suo valore attuale (xr_i canale)

possiamo scrivere qualsiasi valore (canale xw_i)

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

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

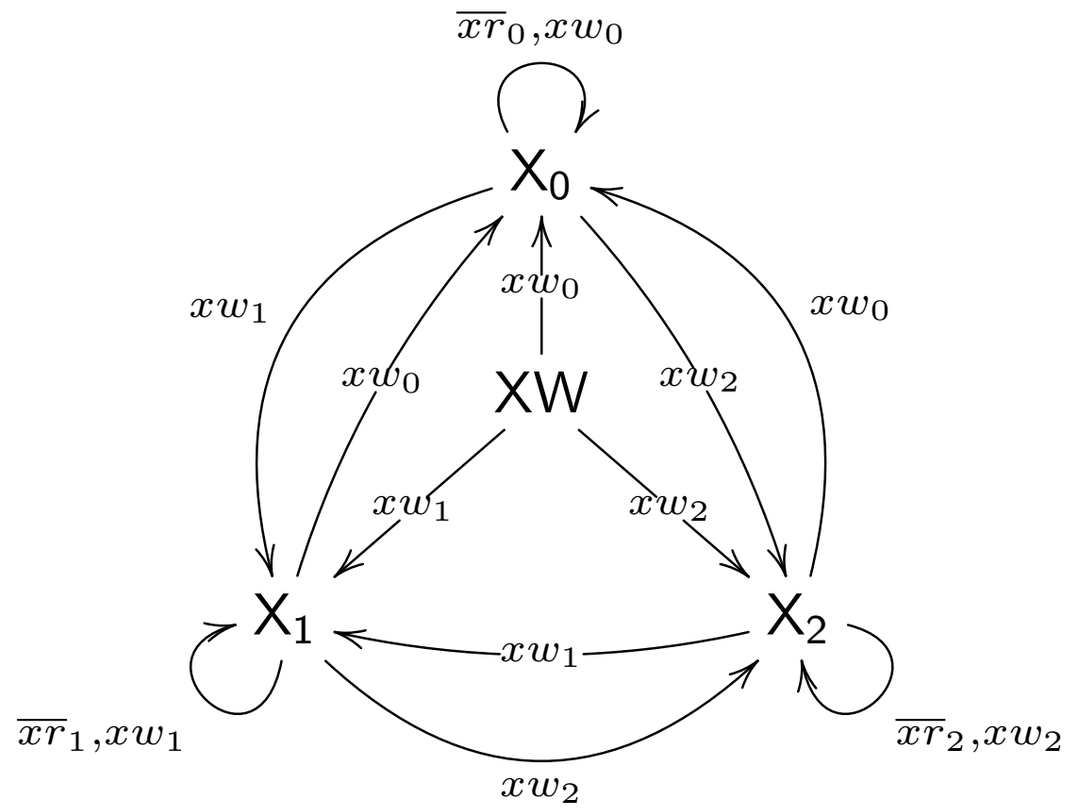
Dichiarazione di variabili

var x

rilascia una variabile non inizializzata e termina

XW|Done

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



|

Done $\xrightarrow{\overline{done}}$ **nil**

Assegnamento

$$x := v_i$$

invia un messaggio per cambiare lo stato della variabile
e poi termina

$$\overline{xw}_i.\mathsf{Done}$$

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

Composizione sequenziale

$$c_1 \; ; \; c_2$$

assumiamo p_1 modella c_1

p_2 modella c_2

$$p_1 | done.p_2$$

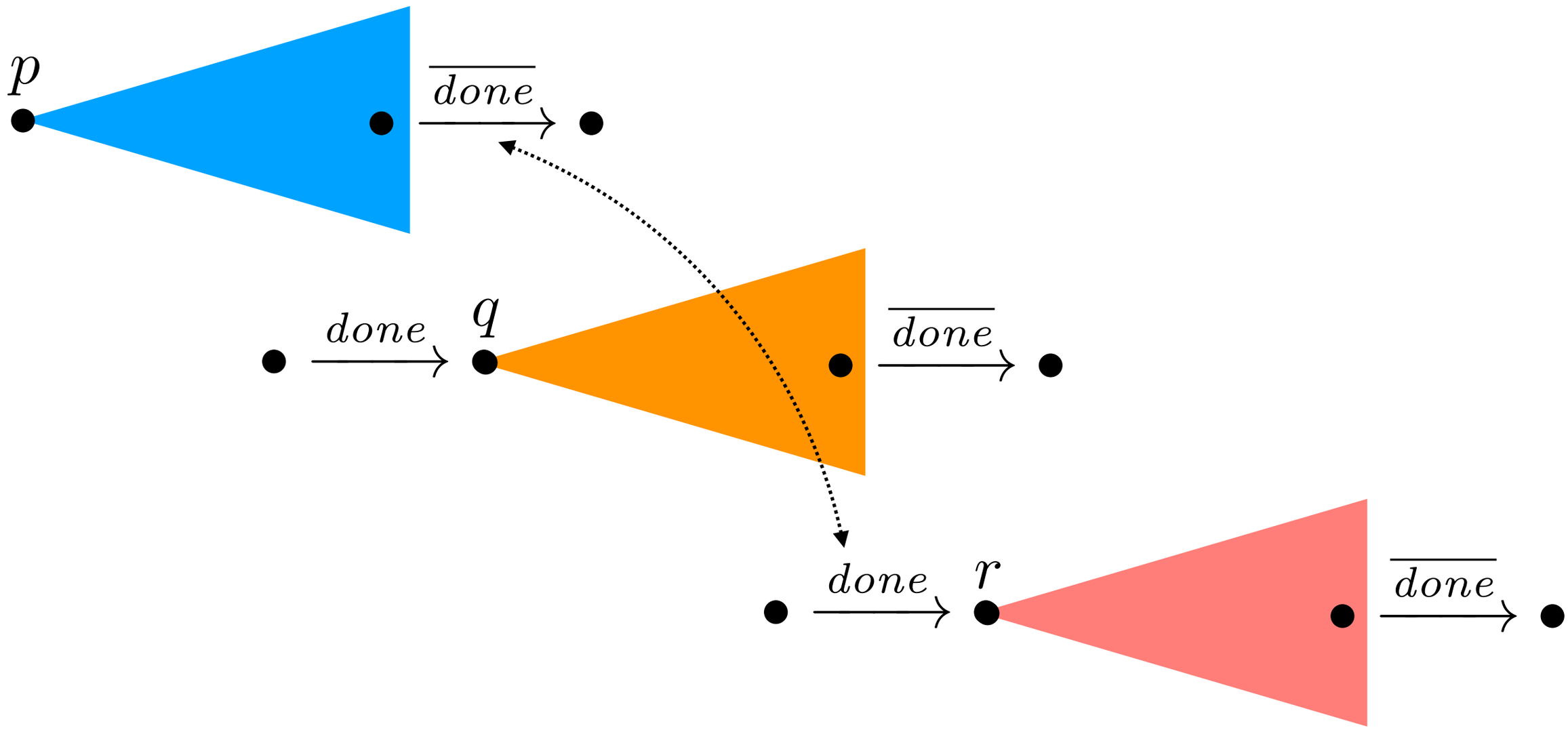
scelta infelice: non è scalabile

$$c_1; c_2; c_3$$

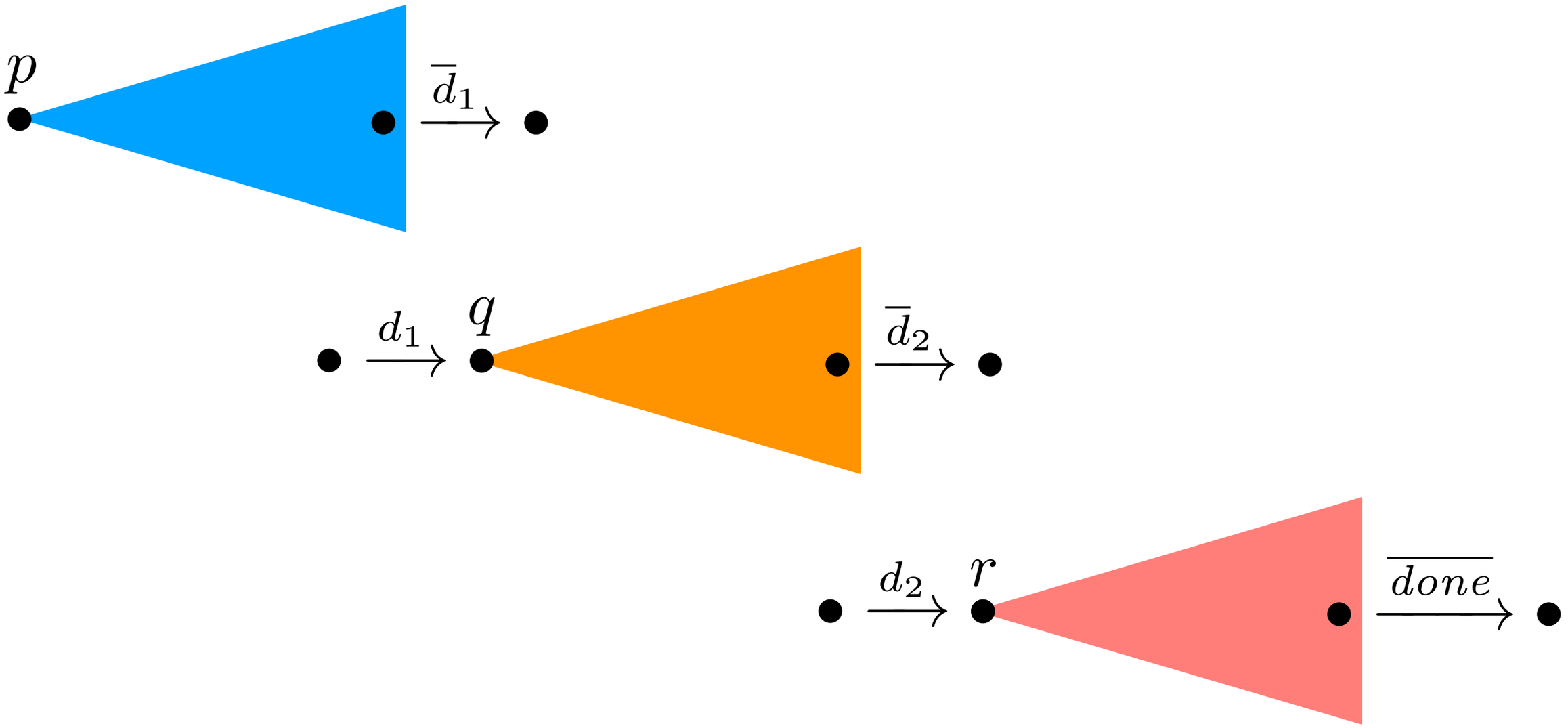
$$p_1 | done.p_2 | done.p_3$$

p_3 puo' iniziare dopo p_1

Sequenziale?



Sequential!



Composizione sequenziale

$$c_1 \; ; \; c_2$$

assumiamo p_1 modella c_1

p_2 modella c_2

$$\phi_d(done) = d$$

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

ora d e' locale a p_1 e p_2

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

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

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

Condizionale

if $x = v_i$ then c_1 else c_2

assumiamo p_1 modella c_1

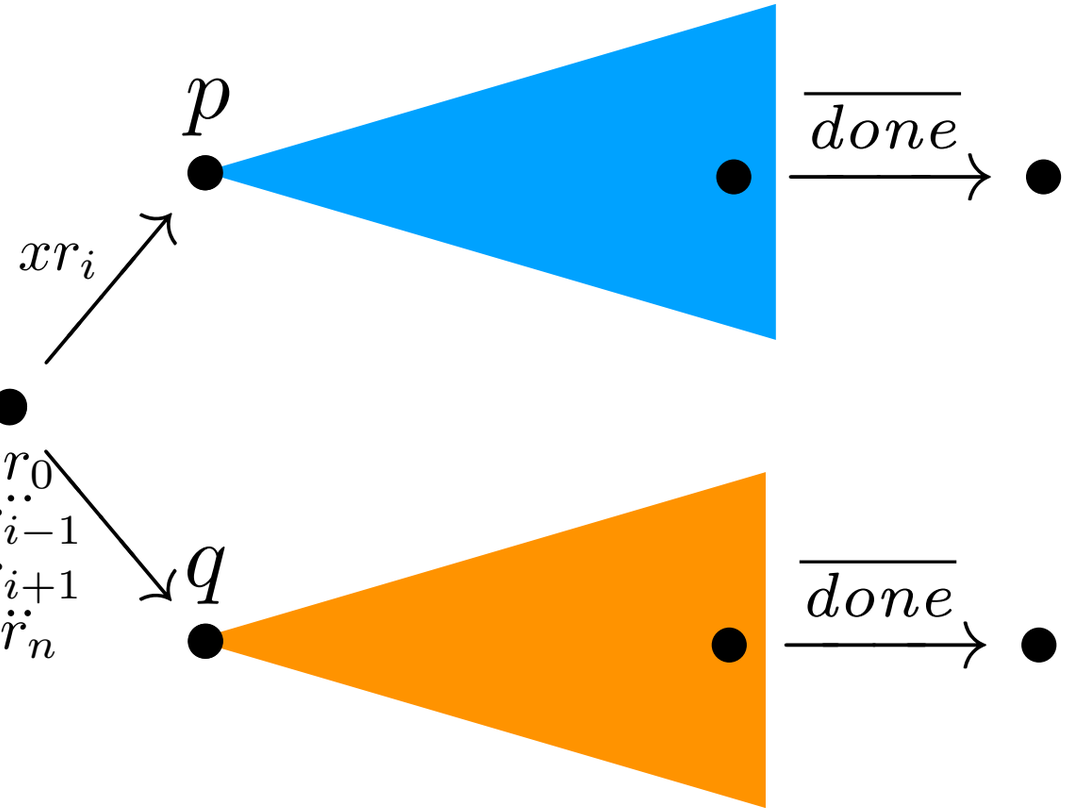
p_2 modella c_2

riceve lo stato della variabile

e poi sceglie di conseguenza

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

Condizionale



Iterazione

while $x = v_i$ do c

assumiamo p modella c

riceve lo stato della variabile
e poi sceglie di conseguenza, eventualmente ricorrendo

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

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

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

Riassumendo

tutti i canali di comunicazione con le variabili
devono essere ristretti per garantire che le richieste di lettura/scrittura siano sincronizzate

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

sono possibili diverse ottimizzazioni:

prefisso d'azione invece del collegamento per la composizione sequenziale
guardie più espressive
rimuovere le transizioni silenziose

Example: optimisation

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

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

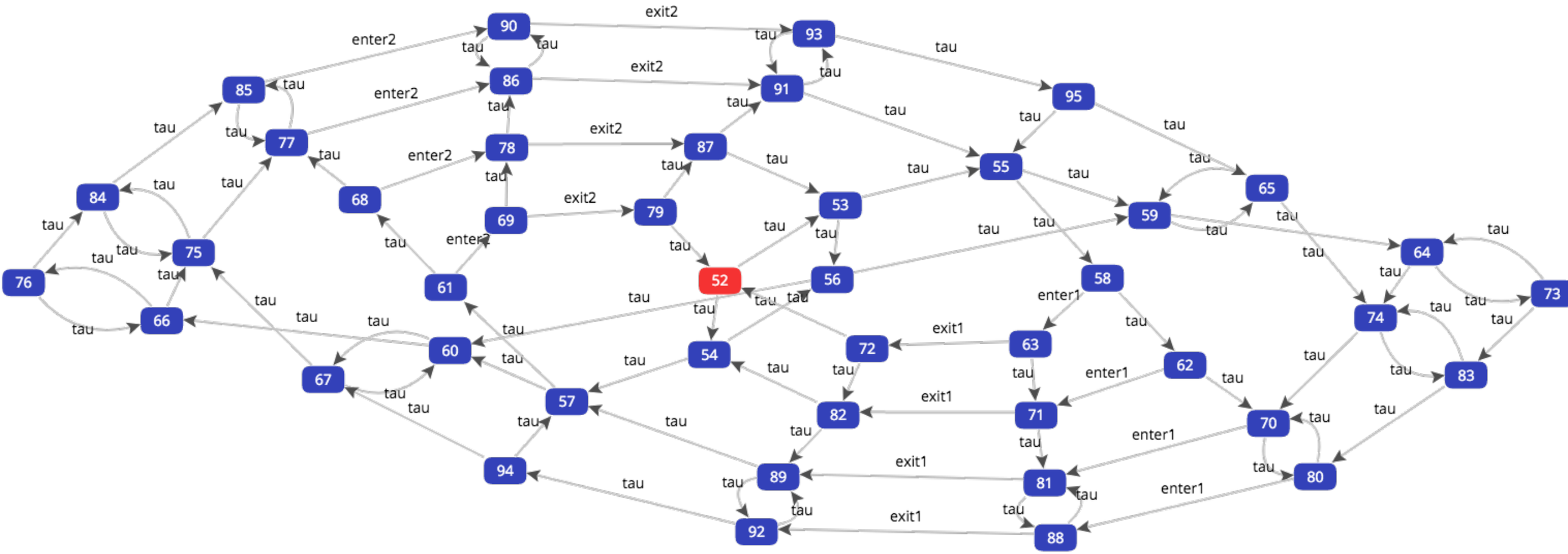
$$(\ (\overline{xw}_1.\overline{done})[\phi_d] \mid d.\overline{yw}_2.\overline{done}\ )\backslash d$$

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

CCS

Giochiamo con CAAL

Concurrency Workbench, Aalborg Edition



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

CAAL

CAAL è uno strumento basato sul web per
la modellazione, la visualizzazione e la verifica
di processi concorrenti in CCS (e la sua estensione temporale)

sviluppato per scopi educativi

Edit permette di specificare processi (definiti ricorsivamente)

Explore i loro LTS (collassando stati equivalenti)

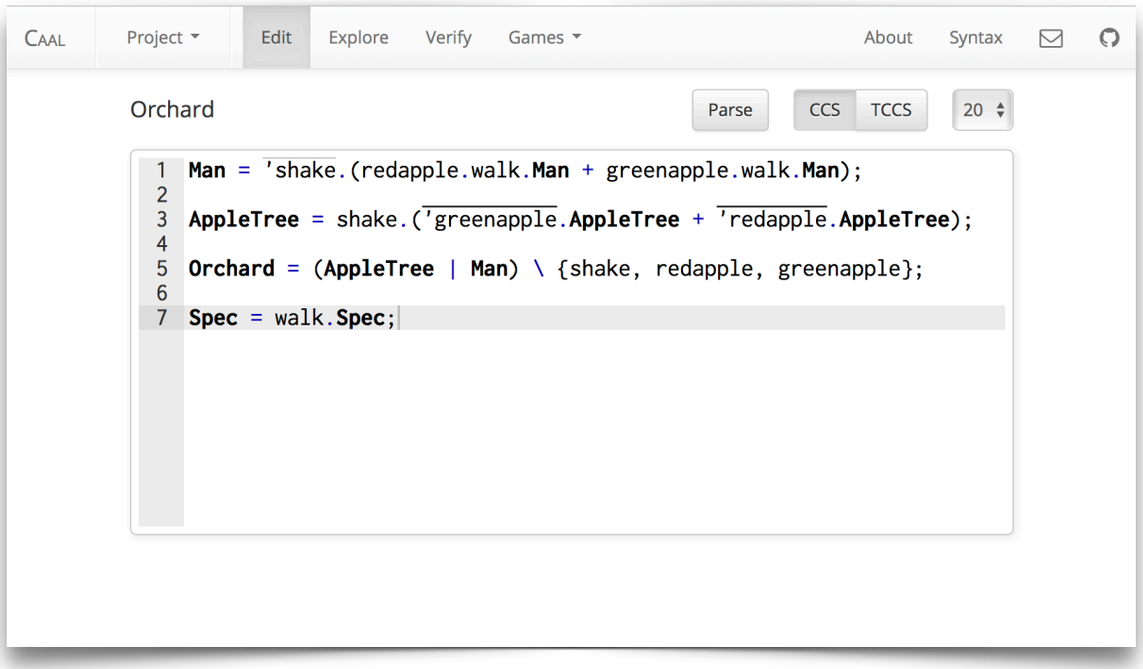
Verify controlla la soddisfazione delle formule HML (model checking)

Verify equivalenza tra processi

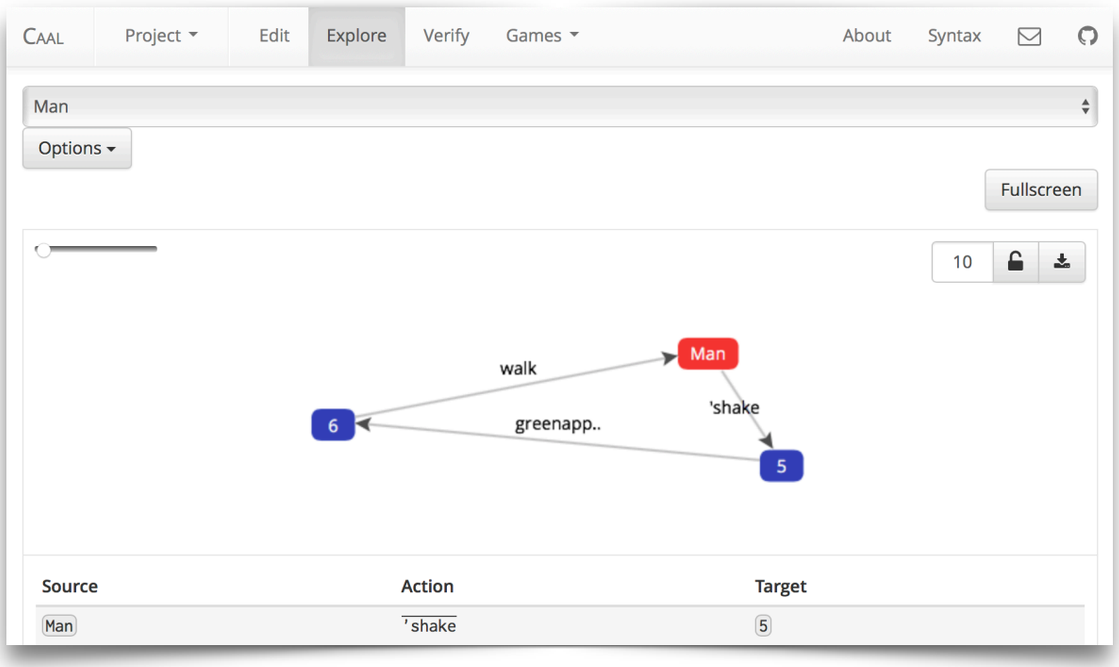
permette di giocare al **Game** della simulazione

CAAL

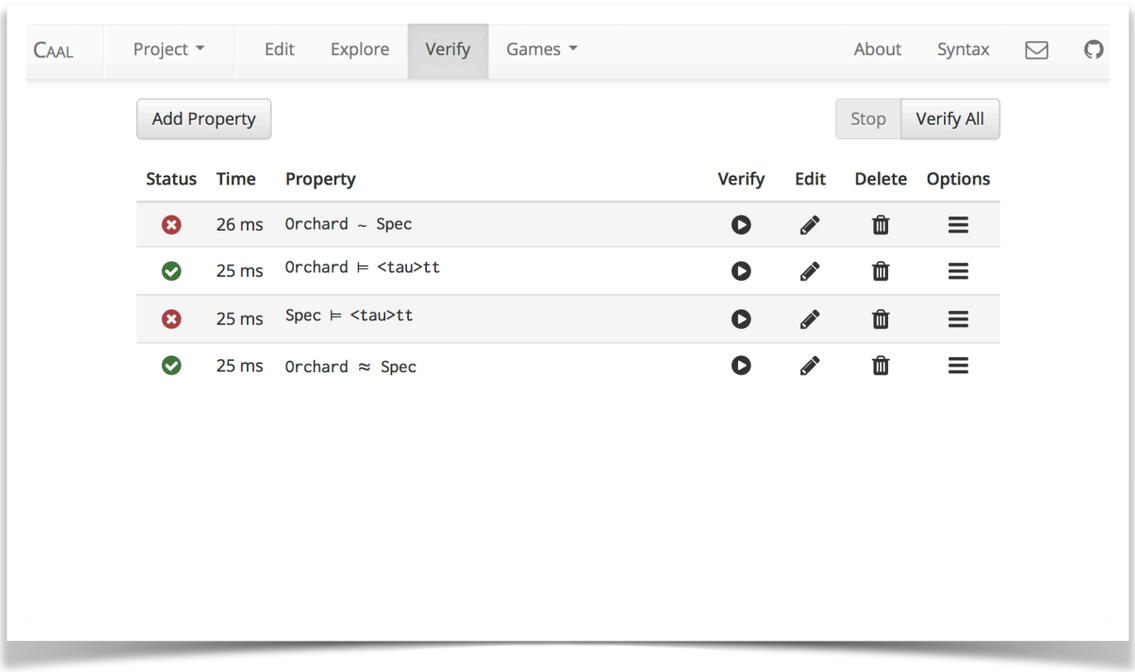
Edit processes



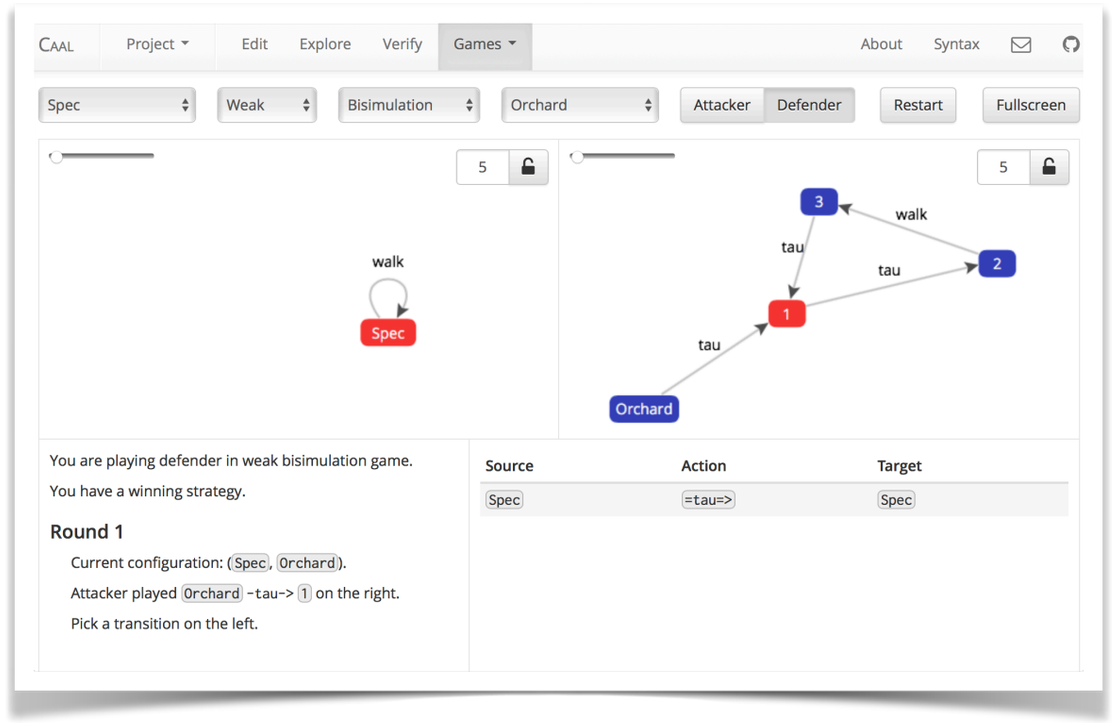
Explore LTS



Verify HML & Equivalences



play bisimulation **Game**



CAAL sintassi per CCS

$\mathbf{nil}$	0
a.P $\quad$ $\overline{\text{a}}$.P $\quad$ τ.P	a.P $\quad$ 'a.P $\quad$ tau.P
P + Q	P + Q
P | Q	P | Q
P\{a1, ..., an}	P \{a1,...,an}
$\text{P}[^{\text{b1}}/_{\text{a1}}, \ldots, ^{\text{bn}}/_{\text{an}}]$	P[b1/a1,...,bn/an]
$\text{A} \triangleq \text{P}$	A = P ;

CAAL sintassi per HML

$\mathbf{tt}$	tt
$\mathbf{ff}$	ff
$F \wedge G$	F and G
$F \vee G$	F or G
$\Diamond_{\{a1,\ldots,an\}} F$	$\langle$a1,…,an$\rangle$F
$\Box_{\{a1,\ldots,an\}} F$	[a1,…,an]F

CAAL

mutual exclusion protocols analysis

Algoritmo di mutua esclusione di Peterson

```
% Due processi P1, P2
% Due variabili booleane b1, b2 (inizialmente false)
% quando Pi vuole entrare nella sezione critica setta  bi a true
% Una variabile intera  k, con valori in  {1,2}
% (il cui valore iniziale e' arbitrario)
% il processo Pk ha priorita' sull'altro processo
%
```

```
% Il processo  P1 in pseudocodice, il process P2 e' analogo a P1
while (true) {
     ...
     b1 = true ; % vuole entrare
     k = 2 ; % da la priorita' a P2
     while (b2 && k==2) skip ; % aspetta
     ...  % entra nella sezione critica
     b1 = false %esce dalla sezione critica
}
```

P1 in CCS

$$\mathsf{B1W} \triangleq b1wf.\mathsf{B1f} + b1wt.\mathsf{B1t}$$
$$\mathsf{B1f} \triangleq \overline{b1rf}.\mathsf{B1f} + \mathsf{B1W}$$
$$\mathsf{B1t} \triangleq \overline{b1rt}.\mathsf{B1t} + \mathsf{B1W}$$

$$\mathsf{KW} \triangleq kw1.\mathsf{K1} + kw2.\mathsf{K2}$$
$$\mathsf{K1} \triangleq \overline{kr1}.\mathsf{K1} + \mathsf{KW}$$
$$\mathsf{K2} \triangleq \overline{kr2}.\mathsf{K2} + \mathsf{KW}$$

```
% Processo P1 in pseudocodice
while (true) {
    ...
    b1 = true ; % vuole entrare
    k = 2 ; % da la priorita' a P2
    while (b2 && k==2) skip ; % aspetta
    ...  % entra nella sezione critica
    b1 = false %esce dalla sezione critica
}
```

$$\mathsf{P1} \triangleq \overline{b1wt}.\overline{kw2}.\mathsf{P1a}$$
$$\mathsf{P1a} \triangleq b2rf.\mathsf{P1b} + b2rt.(kr1.\mathsf{P1b} + kr2.\mathsf{P1a})$$

Esce dal ciclo

Cicla

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

P1a senza attesa attiva

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

Le mettiamo per osservare entrata e uscita di P1

Algoritmo in CCS

$\mathsf{B2W} \triangleq b2wf.\mathsf{B2f} + b2wt.\mathsf{B2t}$

$\mathsf{B2f} \triangleq \overline{b2rf}.\mathsf{B2f} + \mathsf{B2W}$

$\mathsf{B2t} \triangleq \overline{b2rt}.\mathsf{B2t} + \mathsf{B2W}$

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

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

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

$\mathsf{P2} \triangleq \overline{b2wt}.\overline{kw1}.\mathsf{P2a}$

```
% Processo P2 in pseudocodice
while (true) {
    ...
    B2 = true ; % vuole entrare
    k = 1; % da la priorita' a P1
    while (b1 && k==1) skip ; % aspetta
    ...  % entra nella sezione critica
    B2 = false %esce dalla sezione critica
}
```

$\mathsf{P2a} \triangleq b1rf.\mathsf{P2b} + kr2.\mathsf{P2b}$

$\mathsf{P2b} \triangleq enter2.exit2.\overline{b2wf}.\mathsf{P2}$

Intero sistema

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

Algoritmo di Peterson in CCS

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

P2 $\triangleq \overline{b2wt}.\overline{kw1}$.P2a

P1a $\triangleq b2rf$.P1b $+ kr1$.P1b

P2a $\triangleq b1rf$.P2b $+ kr2$.P2b

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

P2b $\triangleq enter2.exit2.\overline{b2wf}$.P2

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

una formula per la mutua esclusione?

qualsiasi label

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

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

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

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

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

una formula definita ricorsivamente!

Algoritmo di Peterson in CCS

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

P2 $\triangleq \overline{b2wt}.\overline{kw1}$.P2a

P1a $\triangleq b2rf$.P1b $+ kr1$.P1b

P2a $\triangleq b1rf$.P2b $+ kr2$.P2b

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

P2b $\triangleq enter2.exit2.\overline{b2wf}$.P2

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

P1 ha la possibilita' di entrare?

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

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

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

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

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

una formula definita ricorsivamente!

Algoritmo di Peterson in CCS

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

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

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

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

non abbiamo deadlock?

$$H \triangleq \langle - \rangle \mathbf{tt} \qquad H \wedge [-](\cdots)$$

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

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

una formula definita ricorsivamente!

Algoritmo di Peterson in CCS

P1 $\triangleq \overline{b1wt}.req_1.\overline{kw2}$.P1a

P2 $\triangleq \overline{b2wt}\ req_2\ \overline{kw1}$.P2a

P1a $\triangleq b2rf$.P1b $+ kr1$.P1b

P2a $\triangleq b1rf$.P2b $+ kr2$.P2b

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

P2b $\triangleq enter2.exit2.\overline{b2wf}$.P2

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

P1 puo' entrare ogni volta che fa una richiesta?

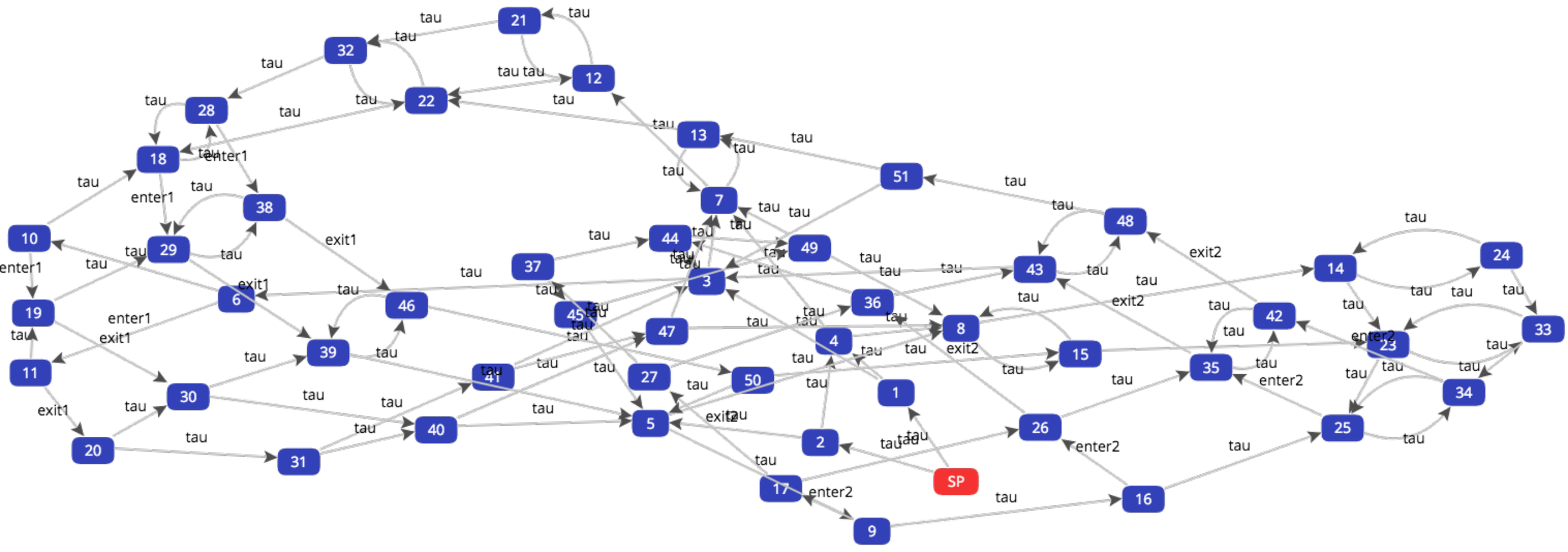
$$A \underset{\text{(min)}}{\triangleq} \langle exit_1 \rangle \mathbf{tt} \vee [-]A$$

$$REQ \underset{\text{(max)}}{\triangleq} [req_1]A \wedge [-]REQ$$

una formula definita ricorsivamente!

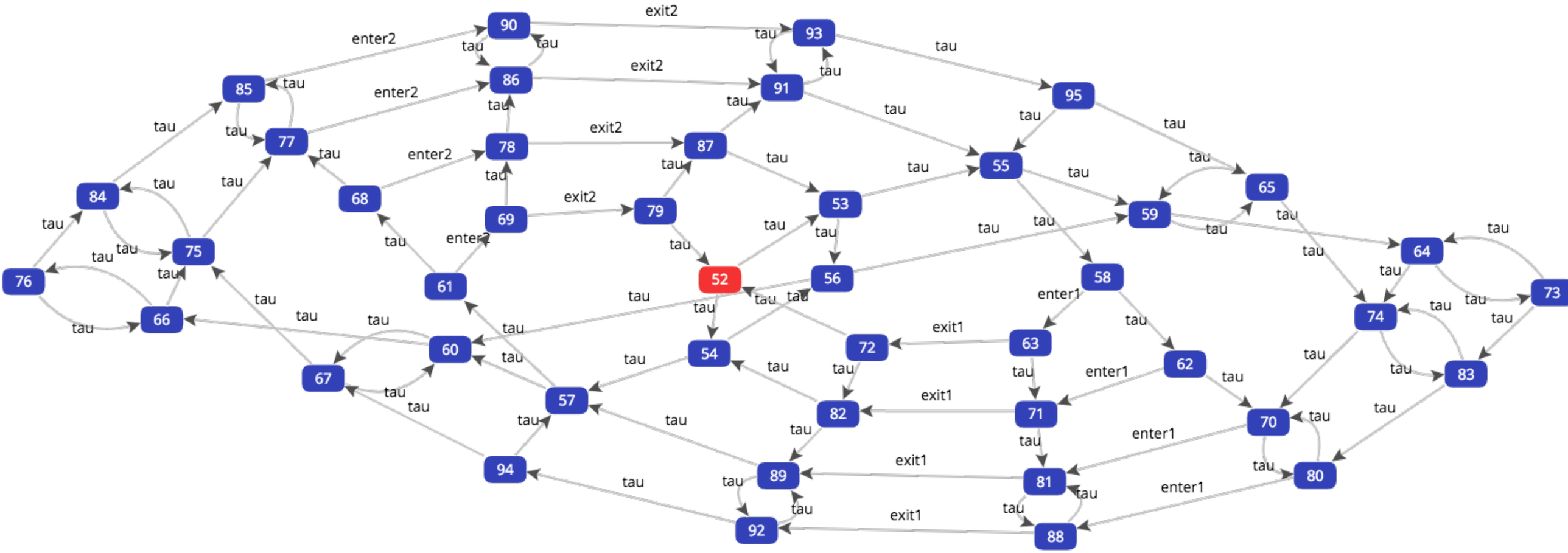
Algoritmo di Peterson in CAL

LTS



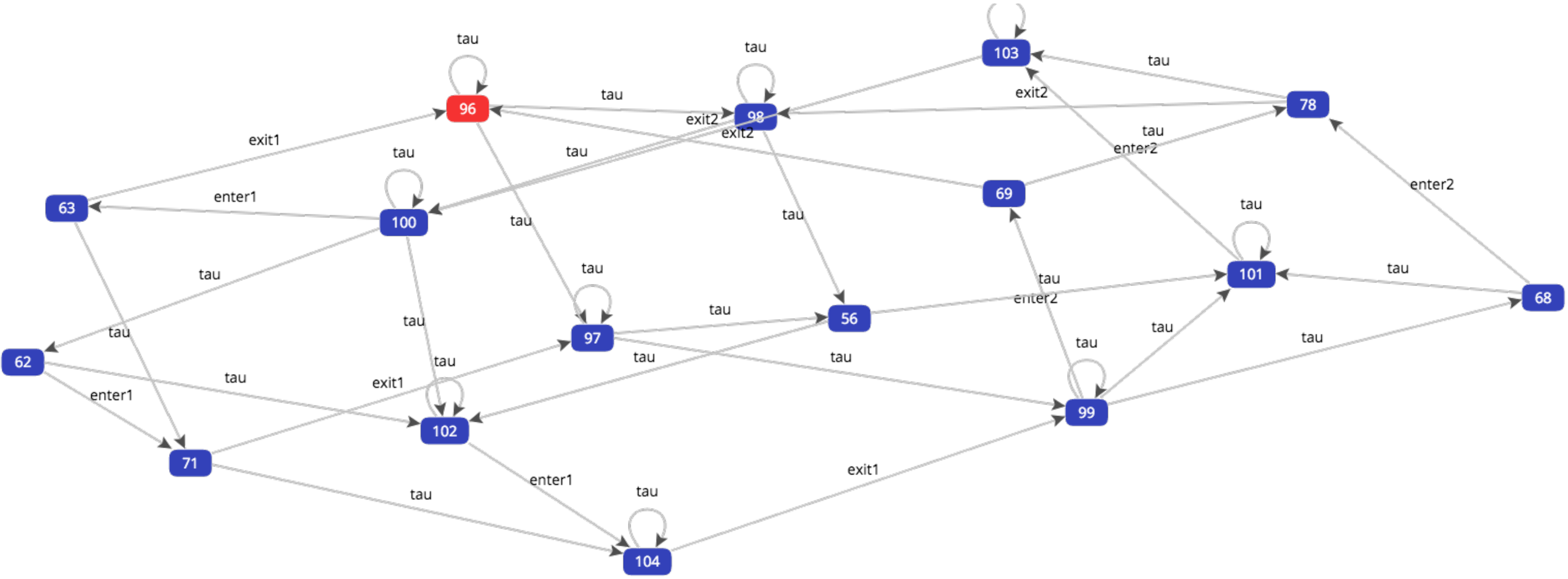
Algoritmo di Peterson in CAL

LTS up to strong bisimilarity



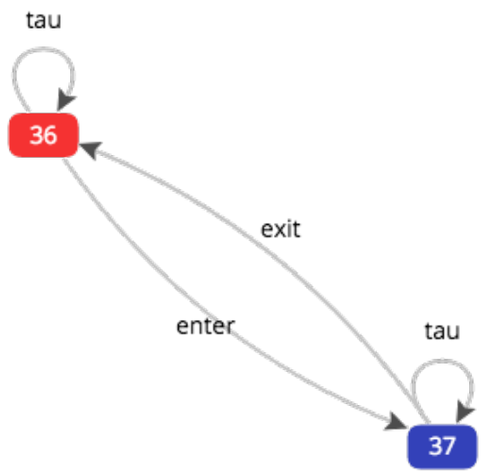
Algoritmo di Peterson in CAL

LTS up to weak bisimilarity



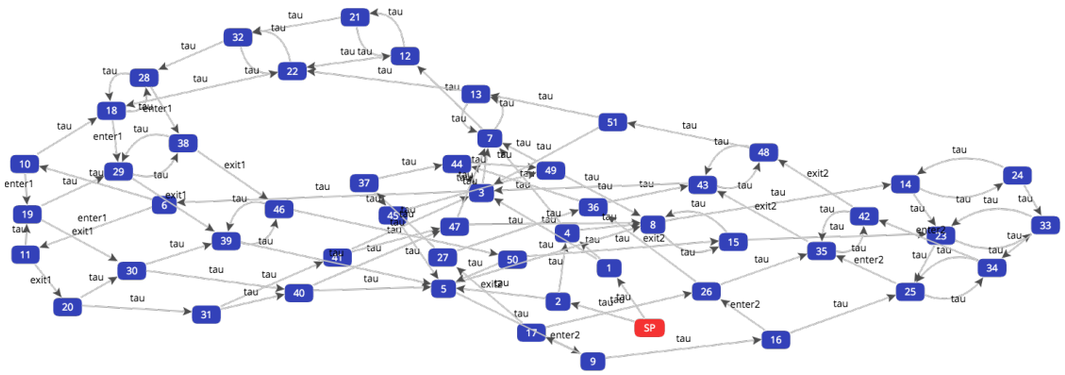
Algoritmo di Peterson in CAL

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

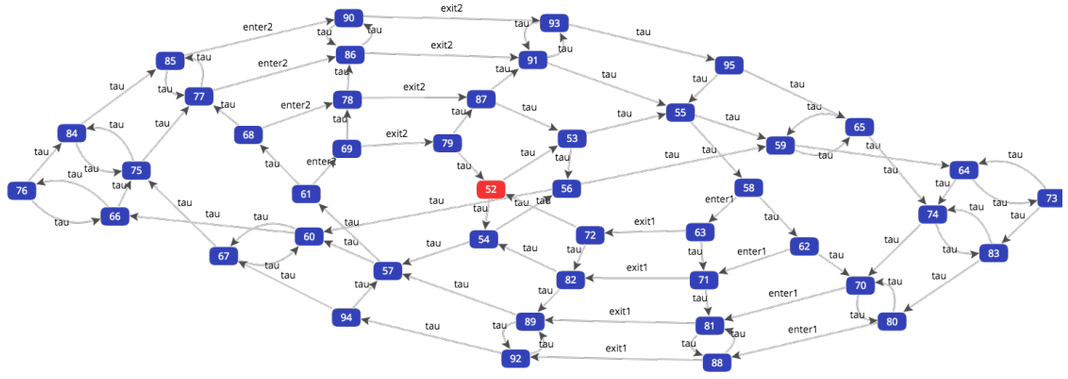


Algoritmo di Peterson in CAL

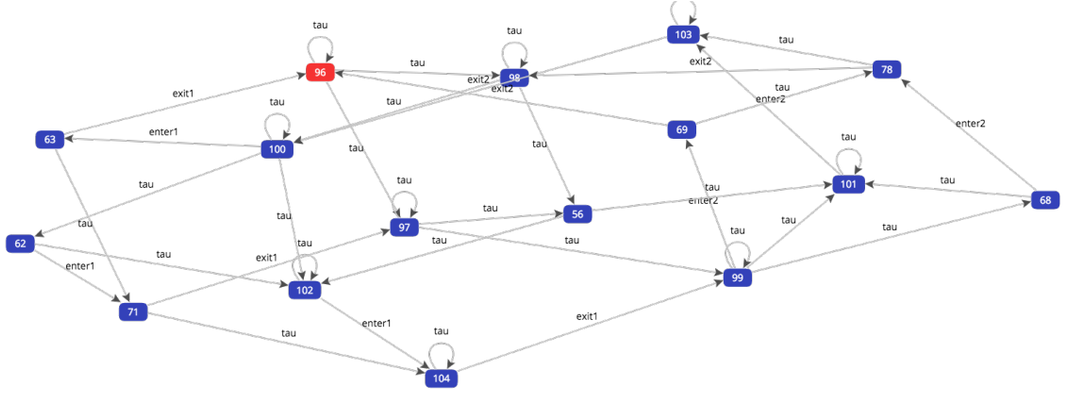
LTS



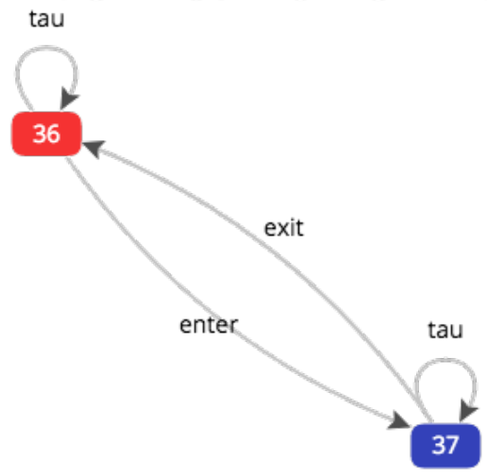
LTS up to strong bisimilarity



LTS up to weak bisimilarity



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



Algoritmo di Hyman

```
% Due processi H1, H2
% Due variabili booleane b1, b2 (inizialmente false)
% quando Hi vuole entrare nella sezione critica setta  bi a true
% Una variabile intera  k, con valori in  {1,2}
% (il cui valore iniziale e' arbitrario)
% il processo Hk ha priorita' sull'altro processo
%
% Il processo  H1 in pseudocodice, il process H2 e' analogo a H1
```

```
while (true) {
     ...
     b1 = true ;                  % H1 vuole entrare
     while (k==2){                % finche' H2 ha la priorita'
           while (b2) skip ;      % H1 aspetta
            k = 1;                % H1 setta la sua priorita'
           }                      % H1 entra nella sezione critica
….
       b1 = false                 % H1 esce dalla sezione critica
}
```

H1 in CCS

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

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

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

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

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

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

$\mathsf{B2W} \triangleq b2wf.\mathsf{B2f} + b2wt.\mathsf{B2t}$

$\mathsf{B2f} \triangleq \overline{b2rf}.\mathsf{B2f} + \mathsf{B2W}$

$\mathsf{B2t} \triangleq \overline{b2rt}.\mathsf{B2t} + \mathsf{B2W}$

```
while (true) {
      ...
      b1 = true ;                % H1 vuole entrare
      while (k==2){              % finche' H2 ha la priorita'
            while (b2) skip ;    % H1 aspetta
             k = 1;              % H1 setta la sua priorita'
            }                    % H1 entra nella sezione critica
 ….
       b1 = false                % H1 esce dalla sezione critica
}
```

$\mathsf{H1} \triangleq \overline{b1wt}.\mathsf{H1a}$

$\mathsf{H1a} \triangleq kr2.\mathsf{H1b} + kr1.\mathsf{H1d}$

$\mathsf{H1b} \triangleq b2rt.\mathsf{H1b} + b2rf.\mathsf{H1c}$

$\mathsf{H1c} \triangleq \overline{kw1}.\mathsf{H1a}$

$\mathsf{H1d} \triangleq enter1.exit1.\overline{\mathsf{b1rt}}.\mathsf{H1}$

H2 in CCS

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

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

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

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

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

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

$$\mathsf{B2W} \triangleq b2wf.\mathsf{B2f} + b2wt.\mathsf{B2t}$$

$$\mathsf{B2f} \triangleq \overline{b2rf}.\mathsf{B2f} + \mathsf{B2W}$$

$$\mathsf{B2t} \triangleq \overline{b2rt}.\mathsf{B2t} + \mathsf{B2W}$$

$$\mathsf{H1} \triangleq \overline{b1wt}.\mathsf{H1a}$$

$$\mathsf{H1a} \triangleq kr2.\mathsf{H1b} + kr1.\mathsf{H1d}$$

$$\mathsf{H1b} \triangleq b2rt.\mathsf{H1b} + b2rf.\mathsf{H1c}$$

$$\mathsf{H1c} \triangleq \overline{kw1}.\mathsf{H1a}$$

$$\mathsf{H1d} \triangleq enter1.exit1.\overline{\mathsf{b1rt}}.\mathsf{H1}$$

H2 e' analogo a H1

Intero sistema

$$\mathsf{SP} \triangleq (\mathsf{B1f}|\mathsf{B2f}|\mathsf{KW}|\mathsf{H1}|\mathsf{H2})\backslash\{kw1, kr1, kw2, kr2, ...\}$$

Algoritmo di Hyman in CAL

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