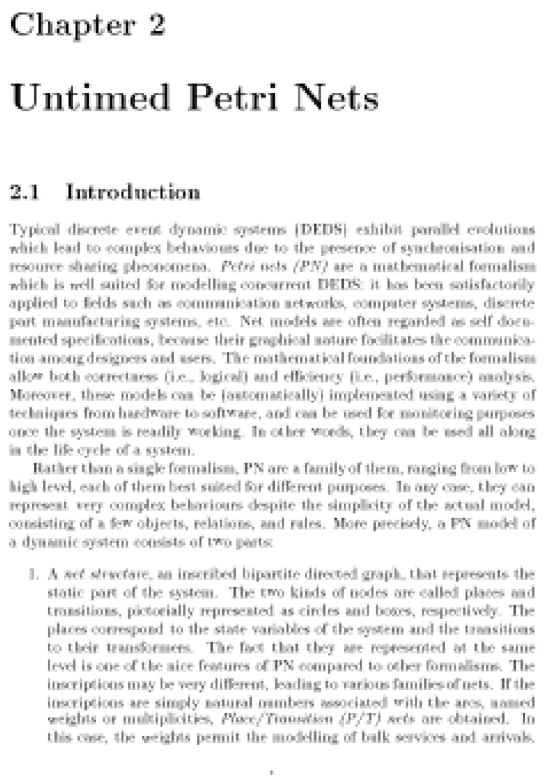
VERIFICA DEI PROCESSI CONCORRENTI
VPC 16-17

Reti di Petri colorate Well-formed nets

Prof.ssa Susanna Donatelli
Universita' di Torino
www.di.unito.it
susi@di.unito.it

Reference material books:



Chapter 2

Untimed Petri Nets

2.1 Introduction

Typical discrete event dynamic systems (DEDS) exhibit parallel evolutions which lead to complex behaviours due to the presence of synchronisation and resource sharing phenomena. *Petri nets (PN)* are a mathematical formalism which is well suited for modelling concurrent DEDS: it has been satisfactorily applied to fields such as communication networks, computer systems, discrete part manufacturing systems, etc. Net models are often regarded as self documented specifications, because their graphical nature facilitates the communication among designers and users. The mathematical foundations of the formalism allow both correctness (i.e., logical) and efficiency (i.e., performance) analysis. Moreover, these models can be (automatically) implemented using a variety of techniques from hardware to software, and can be used for monitoring purposes once the system is readily working. In other words, they can be used all along in the life cycle of a system.

Rather than a single formalism, PN are a family of them, ranging from low to high level, each of them best suited for different purposes. In any case, they can represent very complex behaviours despite the simplicity of the actual model, consisting of a few objects, relations, and rules. More precisely, a PN model of a dynamic system consists of two parts:

1. A *net structure*, an inscribed bipartite directed graph, that represents the static part of the system. The two kinds of nodes are called places and transitions, pictorially represented as circles and boxes, respectively. The places correspond to the state variables of the system and the transitions to their transformers. The fact that they are represented at the same level is one of the nice features of PN compared to other formalisms. The inscriptions may be very different, leading to various families of nets. If the inscriptions are simply natural numbers associated with the arcs, named weights or multiplicities, *Place/Transition (P/T) nets* are obtained. In this case, the weights permit the modelling of bulk services and arrivals.

Notes of the EU-sponsored Jaca MATCH school – ultima parte del capitolo 2

Acknowledgements

Transparencies adapted from the course notes and trasparencies of

- Prof.ssa Giuliana Franceschinis, Universita’ del Piemonte orientale (Italy)

Colored Petri Nets modelling

Modelling convenience: more compact representation, models are more easily parametrizable on the initial marking (and on colors)

Solution convenience: if colors and color functions are defined appropriately, it is possible to automatically exploit symmetries

Various classes:

- colored PN (CPN): colors can be any denumerable set and any function on denumerable set is allowed for defining transition enabling and firing
- Predicate-transition net (algebraic nets)
- well-formed nets (WN): CPN with syntactical restriction

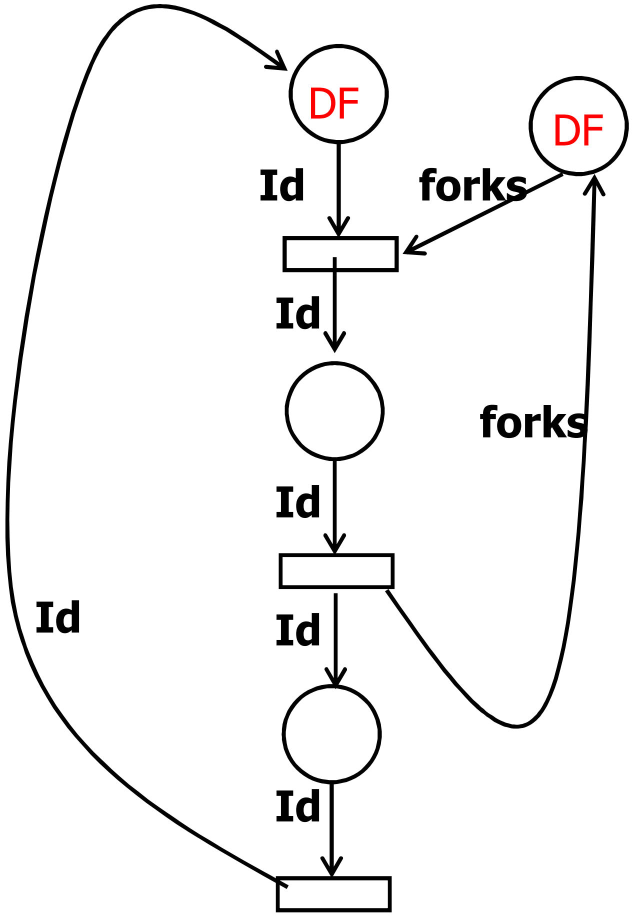
Colored Petri Nets

Basic ingredients:

- a set of colours C: *{students identified by Ids}, {yellow, red, green}, {1,2,3,…}, ..}*
- marking is a function from P to Bag(C): *place p is marked with two yellow tokens and three green ones*
- arcs have an associated function from colors of transitions to colors of places : *transition t fires for yellow if there are at least 2 green tokens and one red token in place p; the firing of t for yellow adds 2 yellow tokens to place p*
- we are assuming that the same set of colour is associated to transitions and places

Colored Petri Nets

Example: dining philosophers. Color DF={a,b,c,d}

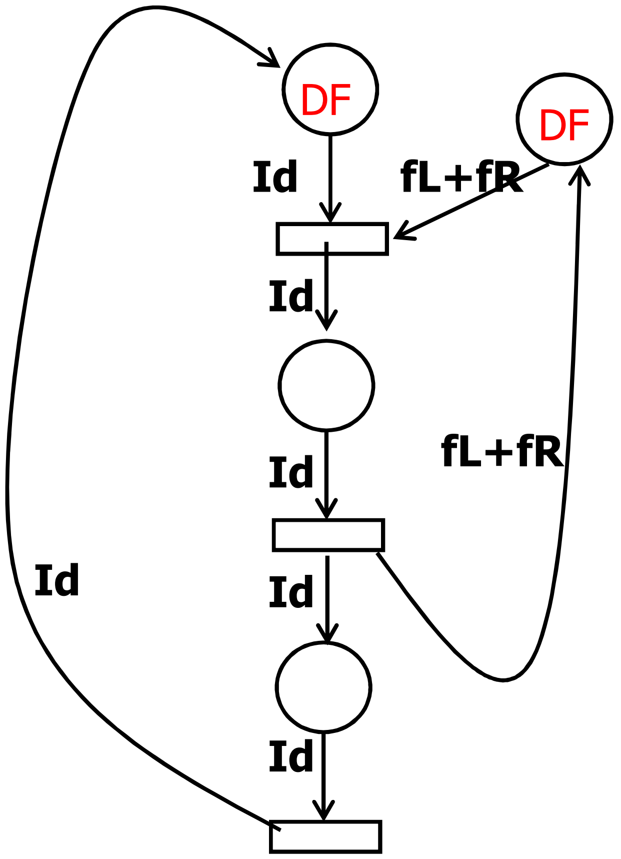


Id	a	b	c	d
a	1			
b		1		
c			1	
d				1

forks	a	b	c	d
a	1	1		
b		1	1	
c			1	1
d	1			1

Colored Petri Nets

Example: dining philosophers. Color DF={a,b,c,d}, modify for one fork at a time



fL	a	b	c	d
a	1			
b		1		
c			1	
d				1

fR	a	b	c	d
a				1
b	1			
c		1		
d			1	

Colored Petri Nets

One single set of colours? May be we do not want to mix processes and resources

Definition 2.11 (CPN, matrix oriented) *A* Coloured Petri Net *is a six-tuple:*

$$\mathcal{N} = \langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathcal{C}, cd \rangle$$

where:

1. *P and T are disjoint finite non empty sets (the places and transitions of $\mathcal{N}$),*

2. *$\mathcal{C}$ is the finite set of finite colour classes,*

3. *$cd : P \cup T \to \mathcal{C}$ is a function defining the colour domain of each place and transition,*

4. *$\mathbf{Pre}[p,t], \mathbf{Post}[p,t] : cd(t) \to \mathrm{Bag}(cd(p))$ are the* pre- *and* post- incidence matrices.

Colored Petri Nets: enabling rule

Definition 2.14 (Enabling and occurrence in CPNs)
The marking in a CPN system evolves as follows:

1. *A transition instance $\langle t, c\rangle$ is said to be* enabled *at a given marking when the multiset of coloured tokens in each input place $p \in {}^{\bullet}(t)$ contains at least as many tokens of (any) colour $c' \in cd(p)$ as the multiplicity of c' in* $\mathbf{Pre}[p,t](c)$. *Formally, $\langle t, c\rangle$ is enabled at* $\mathbf{m}$ *iff* $\mathbf{m} \geq \mathbf{Pre}[P,t](c)$.

   ~~*The number of simultaneous enablings of a transition instance $\langle t, c\rangle$ at a given marking* $\mathbf{m}$ *is called its* enabling degree, *and is denoted by* $\mathbf{e}(\mathbf{m})[\langle t, c\rangle]$. *Formally,* $\mathbf{e}(\mathbf{m})[\langle t, c\rangle] = \max\{k \in \mathbb{N}_+ \mid \mathbf{m} \geq k \cdot \mathbf{Pre}[P,t](c)\}$. *(The enabling degrees at* $\mathbf{m}$ *of all the transition instances are collected in the enabling vector,* $\mathbf{e}(\mathbf{m})$.*)*~~

Colored Petri Nets: firing rule

2. *The occurrence, or firing, of an enabled transition instance $\langle t, c\rangle$ is an atomic operation that removes from (adds to) each input (output) place the multiset of tokens obtained by applying the function associated with the corresponding arc, to the transition colour c. Formally, the occurrence of $\langle t, c\rangle$ at marking $\mathbf{m}$, denoted by $\mathbf{m}\xrightarrow{\langle t,c\rangle}\mathbf{m}'$, yields the marking $\mathbf{m}' = \mathbf{m} + \mathbf{C}[P, t](c)$*

Colored Petri Nets

Specifying arc functions can be cumbersome (for example the color of a message can be a triplet <sender, receiver, msg id>) and tables can grow quite big, unless there is some regularity, as in the philosophers model

Colours are only a modelling convenience, and can lead to very large state spaces (many markings and each marking requires also to store the colours of tokens)

Can't we use the high level description of the system to generate a state space that has already been reduced according to bisimulation (exploitation of symmetries)?

An introduction to Well-formed Nets

- Formalism syntax: example and definition

- Formalism semantics: example and definition

- Behavioral symmetries exploitation: the symbolic marking and the symbolic reachability graph

WN Syntax

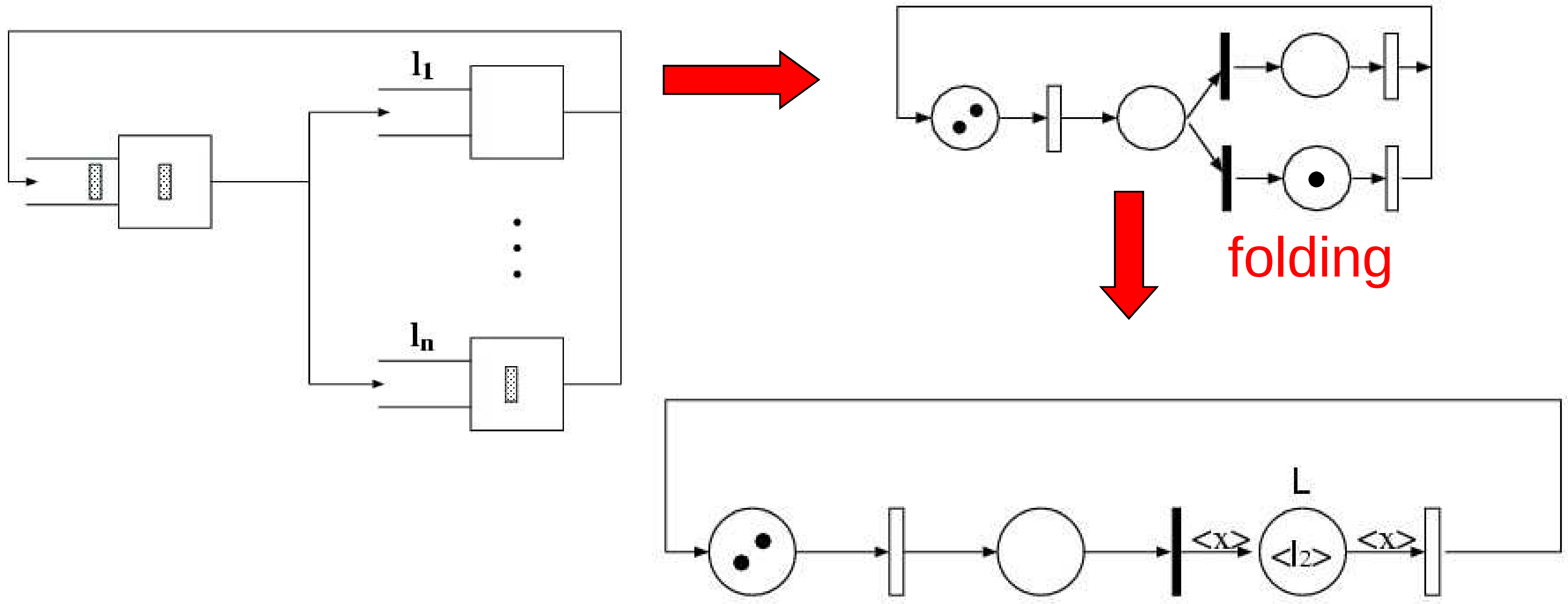
•Well-formed Nets are a *colored* version of P/T nets with priorities and inhibitor arcs

• Tokens can have different colors (i.e. can carry a data structure), so that they are no more indistinguishable. The set of possible colors that tokens can have when they are in place *P* is called *color domain* of *P*.

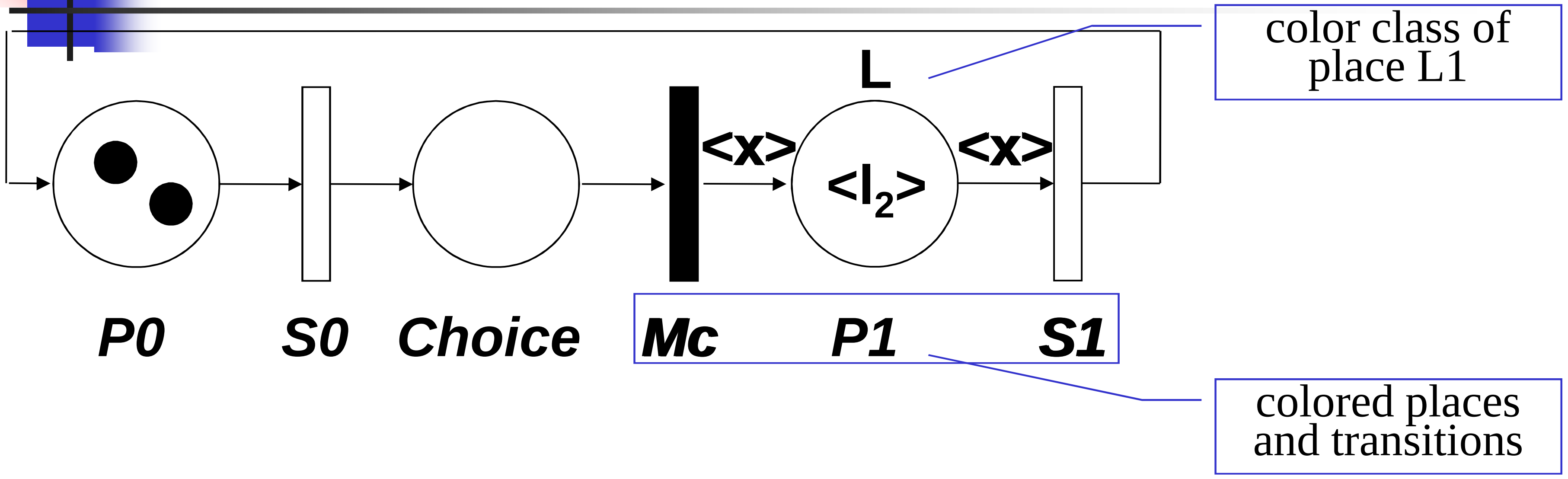
• Transitions can be parameterized through a set of typed variables: the possible values of variables are of the same nature of data (colors) associated with tokens. An assignment of values to the variables of a given transition *t* identifies an *instance* of *t*.

Advantages of colors

- It is more natural to represent data in the model
- It is possible to "fold" similar subnets into a single subnet, and use colors to identify tokens belonging to different folded subnets.



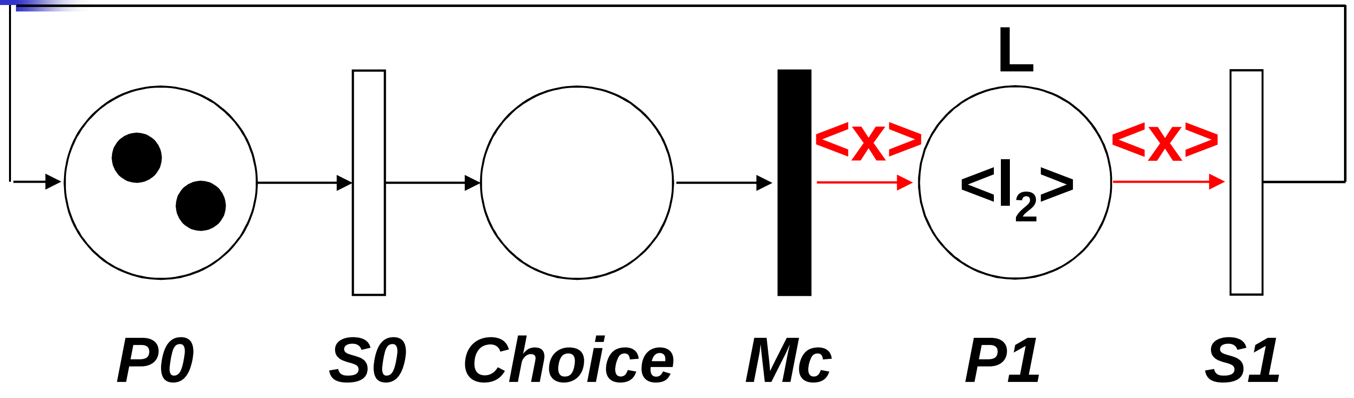
A very simple example: the syntax



- *P1* is a colored place, with color domain $\mathbf{L = \{l_1,l_2\}}$
- The marking of *P1* is $< l_2 >$ (a token of color l_2).

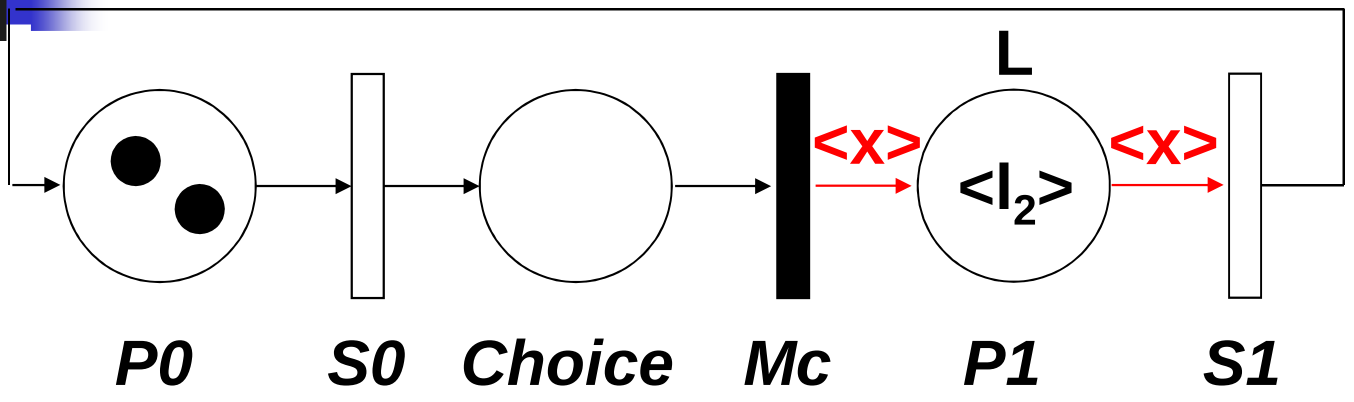
- *Mc* and *S1* are parametric transitions, with parameter **x:L**
- The instances of *Mc* are ($Mc,x=l_1$), ($Mc,x= l_2$)

A very simple example: enabling and firing



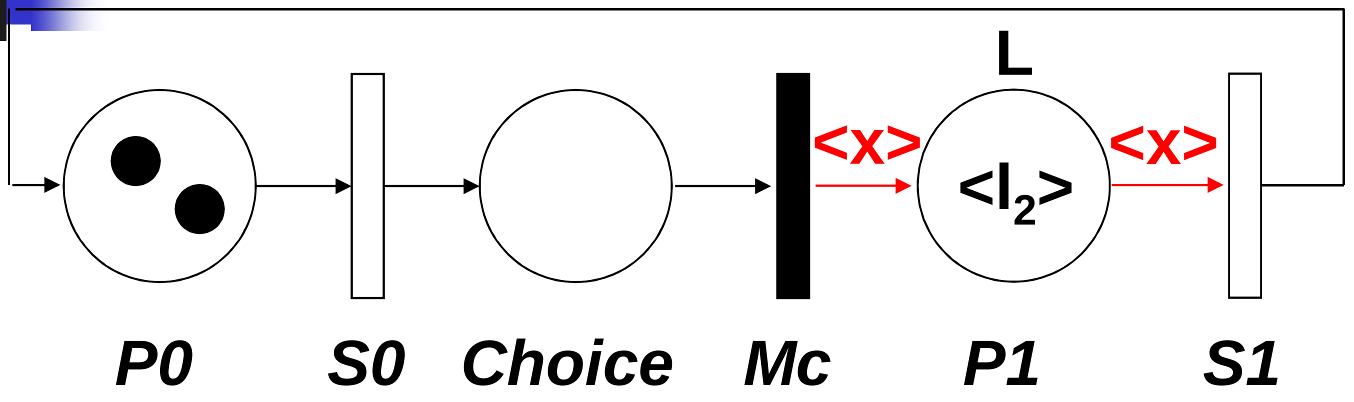
- The arc expressions **<x>** (read as “variable x” or identity) on the input and output arcs of *P1* determine the state transformation due to the firing of transition instances
- In the current marking, transition instance (*S1*,x=l2) is enabled, and its firing leads to the marking *P0(3)*

A very simple example: enabling and firing



- “Neutral” places and transitions behave as in PNs
- In the current marking “neutral” transition *S0* is enabled: after its firing a vanishing marking is reached: *P0(1)Choice(1)P1(<l2>).*
- In the new vanishing marking both instances of *Mc* are enabled (and they are in conflict !).
  - m [(*Mc*,x=l1)> *P0(1)P1(<l1>+<l2>)*
  - m [(*Mc*,x=l2)> *P0(1)P1(2<l2>)*

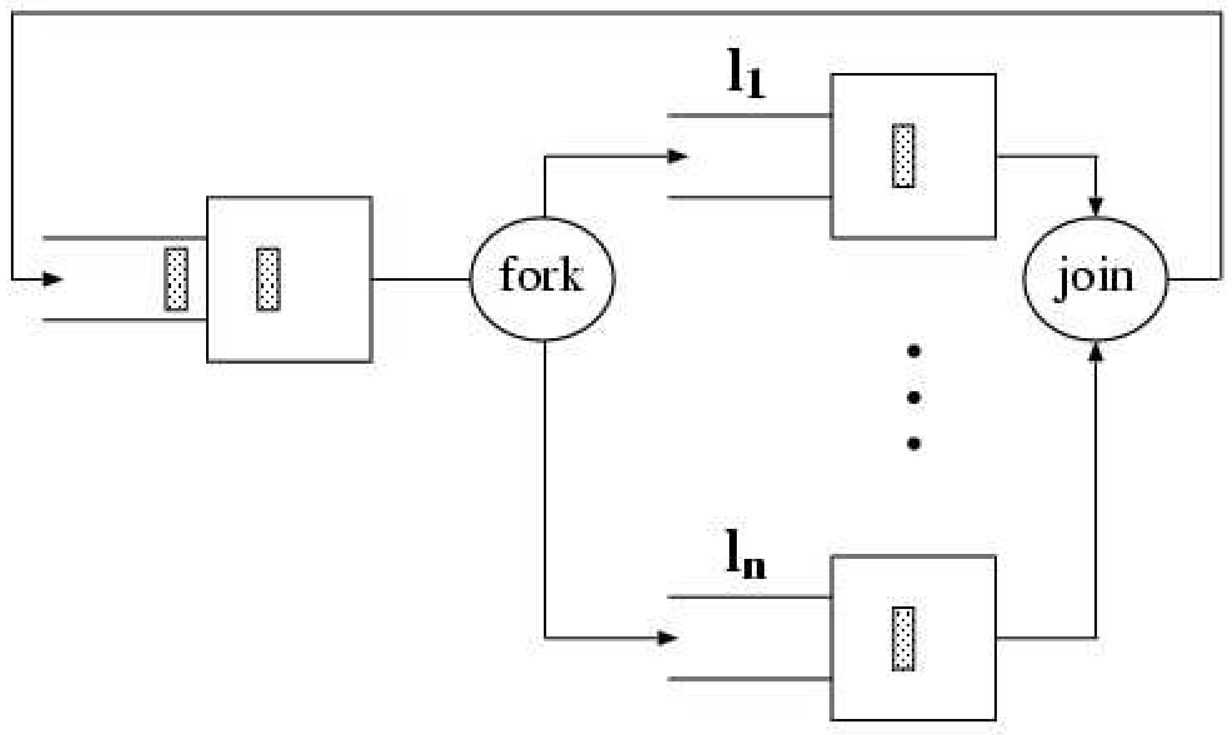
A very simple example: enabling and firing



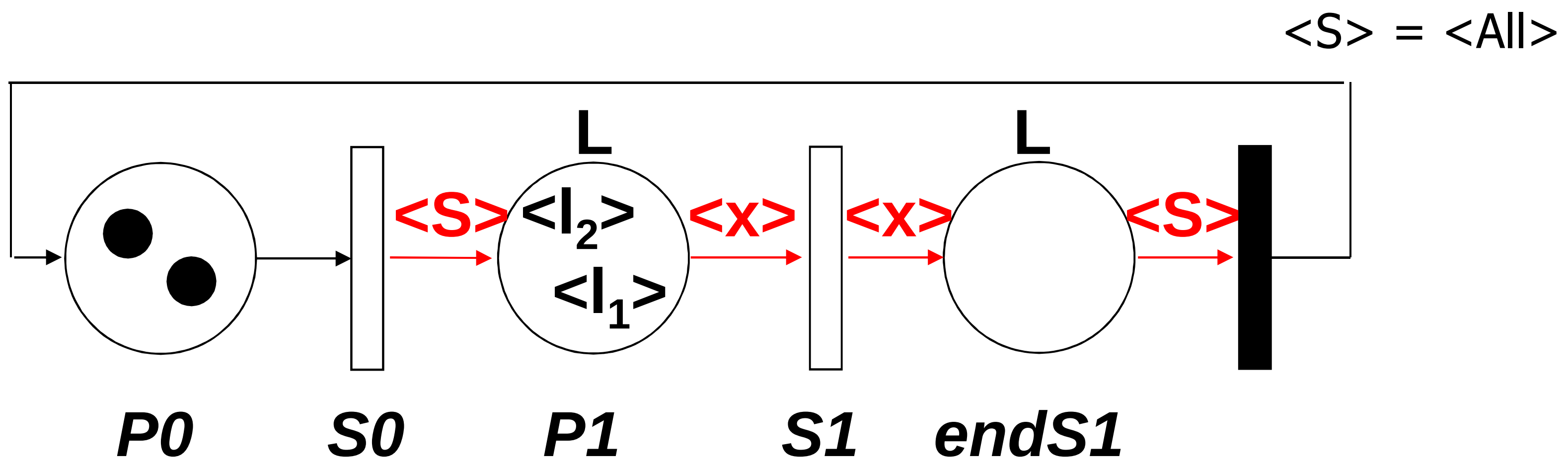
- "Neutral" places and transitions behave as in PNs
- In the current marking "neutral" transition *S0* is enabled: after its firing a vanishing marking is reached: *P0(1)Choice(1)P1(<l2>).*
- In the new vanishing marking both instances of *Mc* are enabled (and they are in conflict !).
  - m [(*Mc*,x=l1)> *P0(1)P1(<l1>+<l2>)*
  - m [(*Mc*,x=l2)> *P0(1)P1(2<l2>)*

place markings are multisets on the place color domain

Second example: Fork and Join (1)



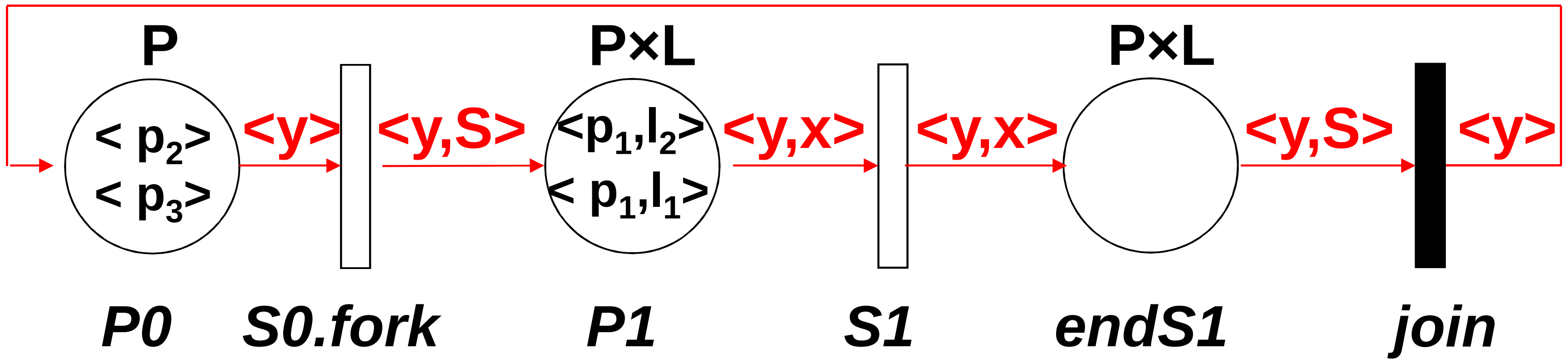
Second example: Fork and Join (2)



• <x> denotes a function of x, returning set {<x>}

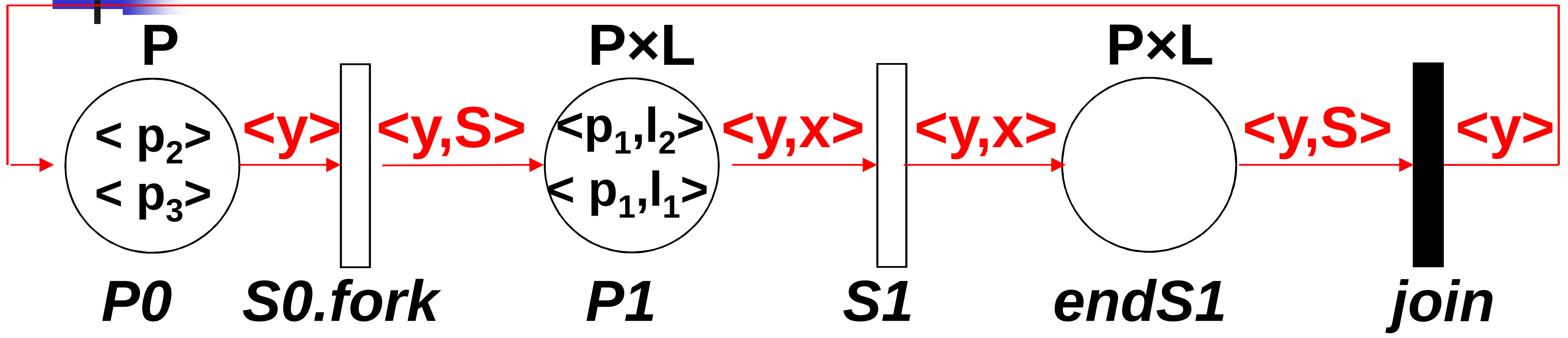
• <S> denotes a constant function always returning the set of all elements in the color domain of the corresponding place (*P1* or *endS1*). It allows to easily model fork (when used on transition output arcs) and join (when used on transition input arcs). – *It is indicated with <All> in the tool*

Second example: Fork and Join (3)



- To ensure that the final synchronization matches the threads generated by the same process, we add a new color class, and add the processes identity.
- Color domains can be Cartesian products of basic color classes (as for places *P1* and *endS1*).
- Transitions *S0.fork* and *join* have only one parameter: **y**; Transition *S1* has two parameters: **y** and **x**.

Second example: Fork and Join (3)



• Function <x,y> simply returns the set {<x,y>}; function <y,S> returns the set {<y,l1>+<y,l2>}. In general function $<f_1,f_2,\ldots,f_n>$ returns $\mathbf{f_1 \times f_2 \times \ldots \times f_n}$

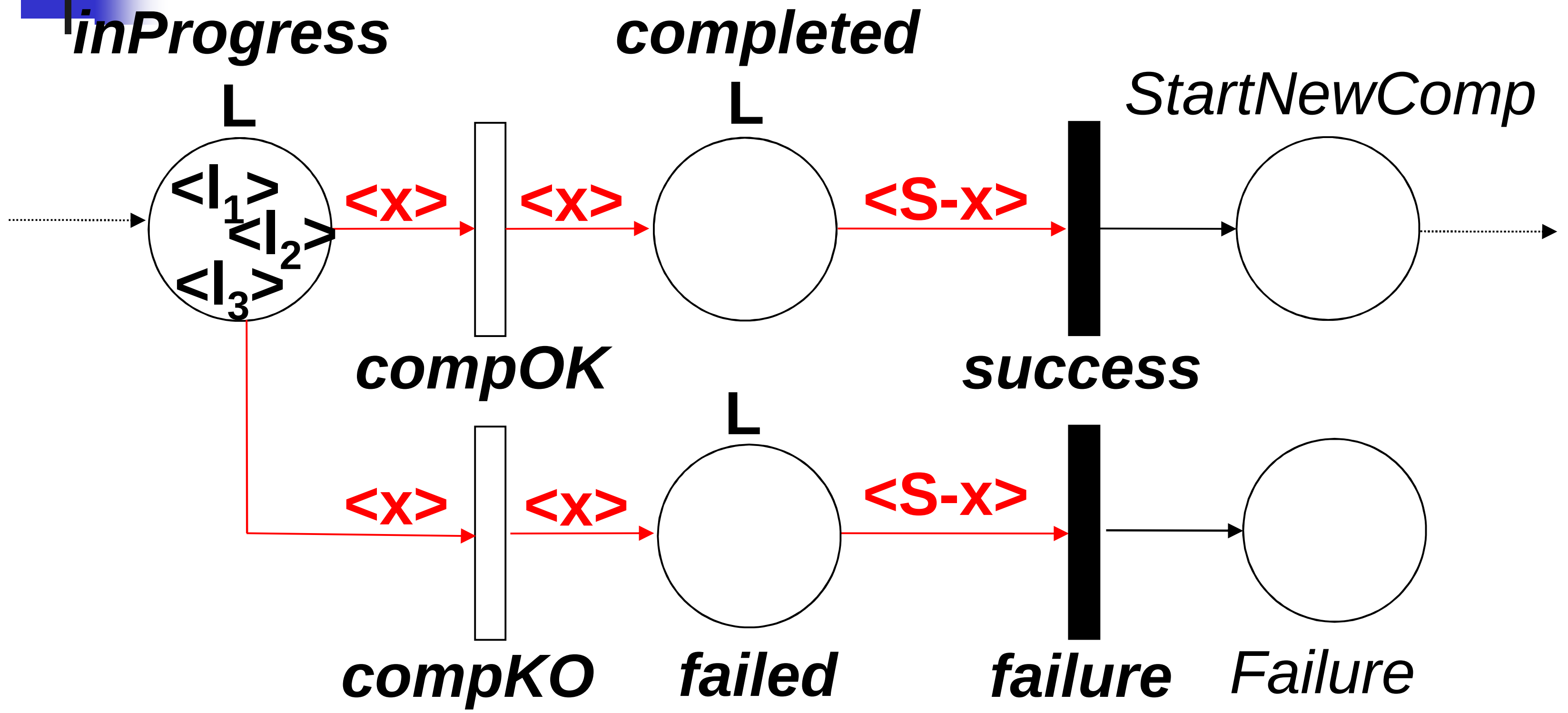
•The marking reached after firing the sequence

(*S0.fork*,y=p3), (*S1*,y=p1,x=l1), (*S1*,y=p3,x=l2)

is *P0*(<p2>)*P1*(<p1,l2>,<p3,l1>)*endS1*(<p1,l1>,<p3,l2>).

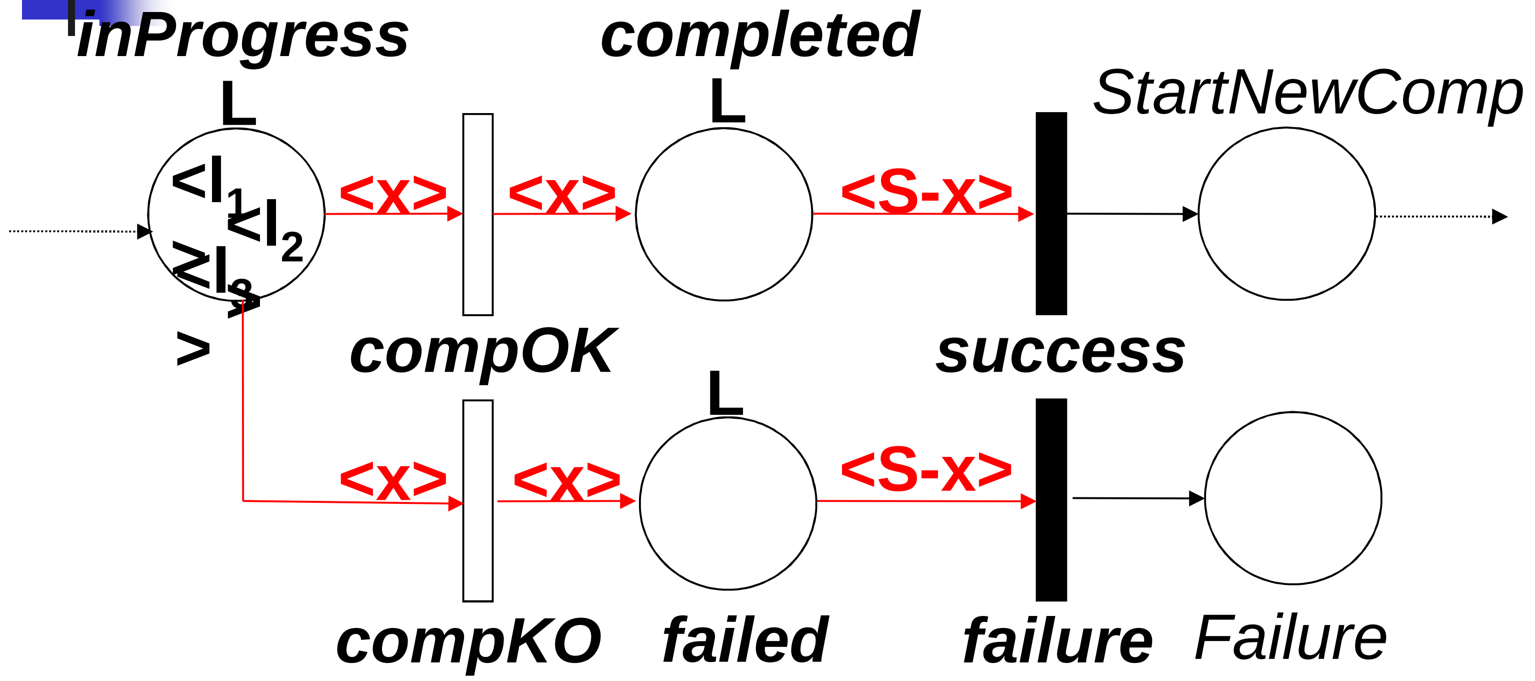
In this marking only one instance of *S0.fork* and two instances of *S1* are enabled (the two threads in *endS1* belong to different processes and hence cannot synchronize).

Third example: a 2:3 voter



- Function <S-x> returns all the elements of set **L** but one (namely, x). It implements the 2:3 voter.
- If two computations out of three complete succesfully, a new computation can start, if instead two computations out of three fail, then a failure has occurred.

Third example: a 2:3 voter



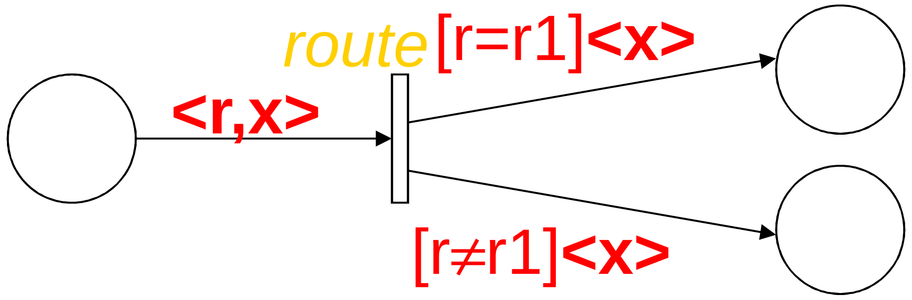
• Why do we need colours at all in this example? A weight of 2 on the arc to transitions success and failure will do! (assuming places completed and failed are uncoloured).

• But immagine 2 parallel computation, then marking of place InProgress is 2<l1>+2<l2>+2<l3> and colours are needed

• 2<l1>+2<l2>+2<l3> can also be written as 2<S>

Static subclasses

Can we distinguish behaviour according to colours?

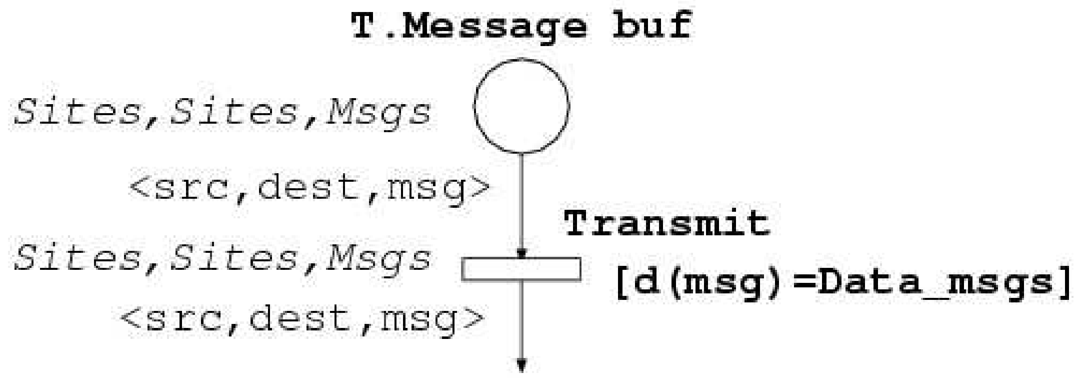


Depending on the routing the message goes to different buffers, and destination r1 requires the upper buffer, while all other destinations require the lower buffer.

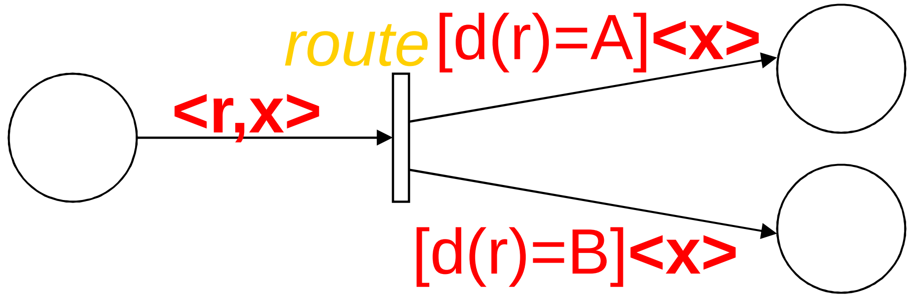
This is NOT allowed in SWN as it destroy symmetries, but a colour class may be split into colour sublasses (a partition of the colour class)

R={A,B}, A={r1}, B={r2,r3,r4,r5} and D(colour instance) = name of subclass

Transition predicates and guarded functions

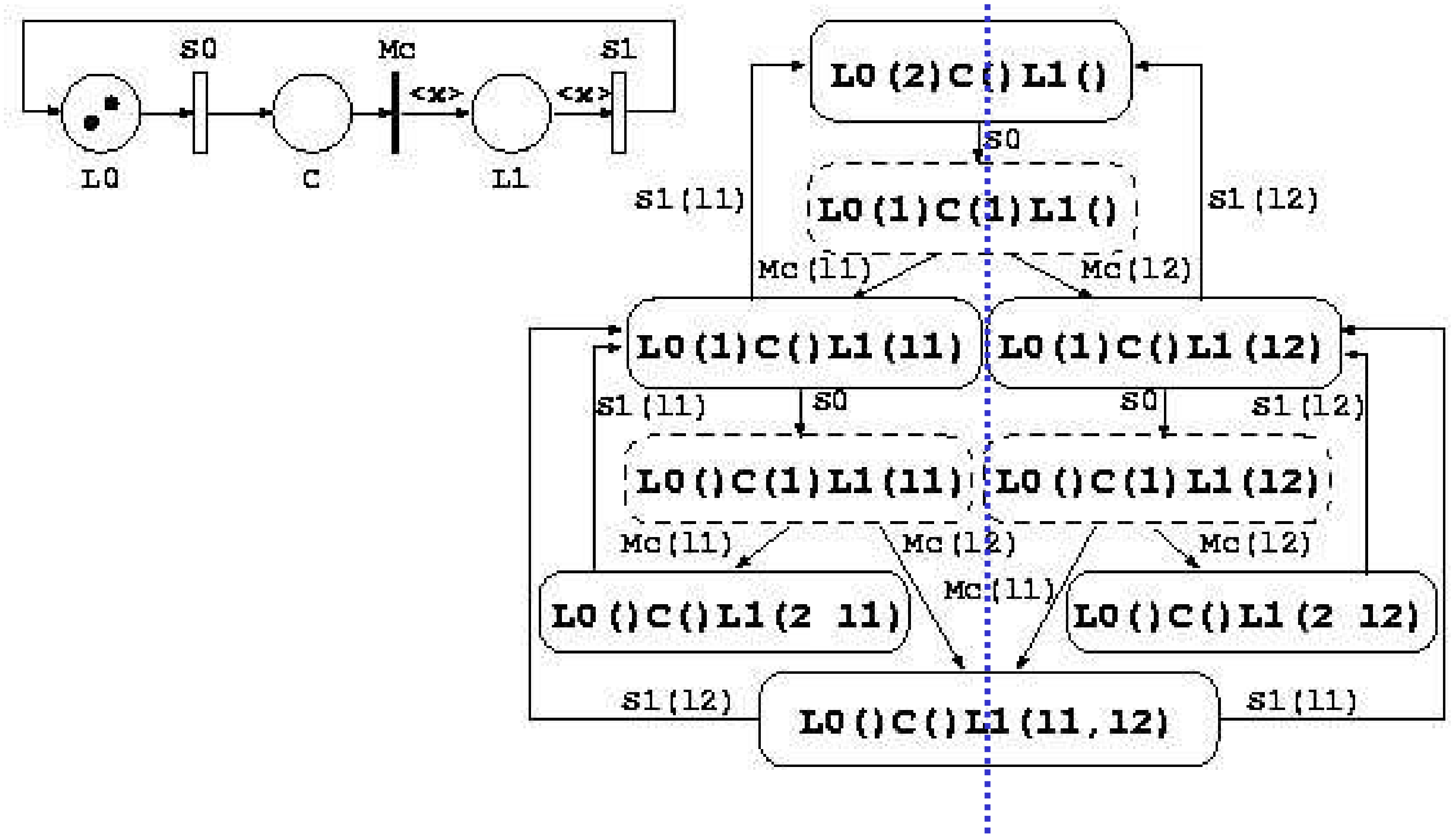


- Guards are boolean functions built on *standard predicates* on the transition variables. They restrict the set of fireable instances of a transition.
- Guards can be used within arc function, to implement transition-instance dependent arcs: an arc may be present for some instances and absent for others.



Can be avoided by adding immediate transitions.

Model semantics: RG of first example

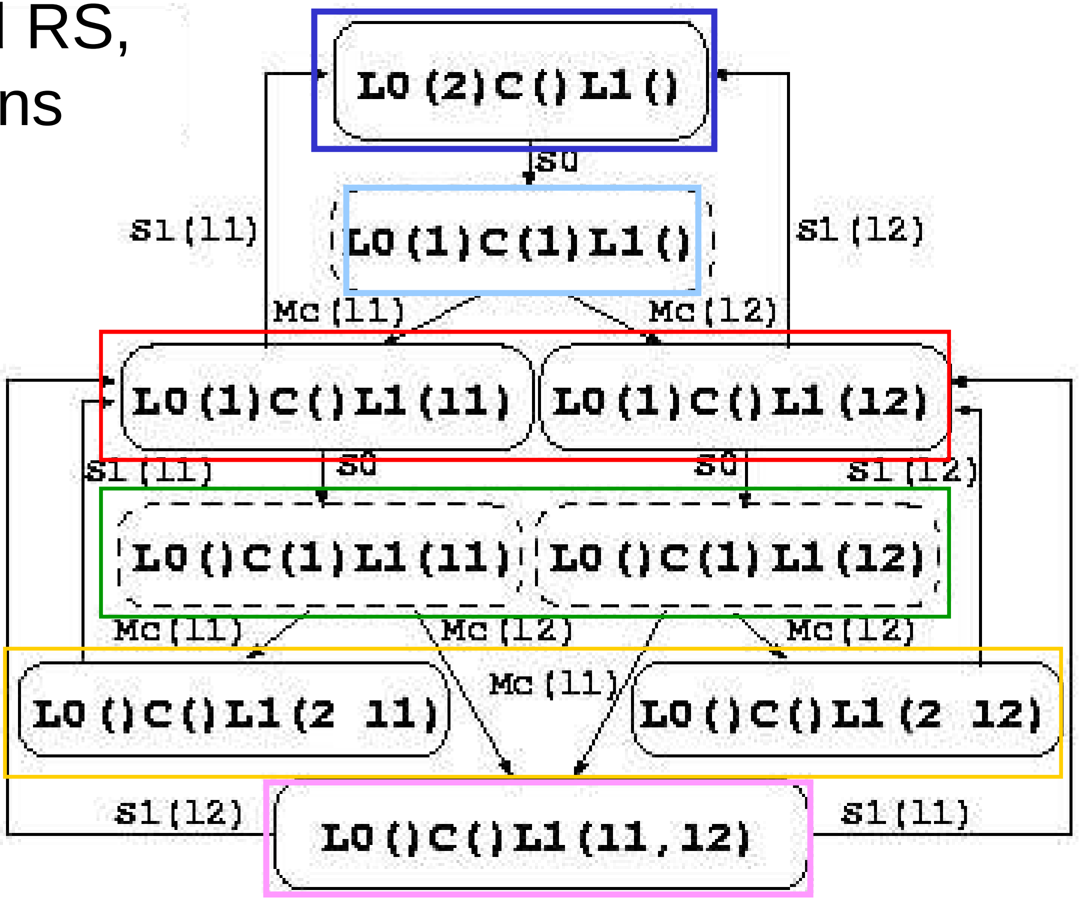


This model has a symmetrical behavior, as can be visualized on the reachability graph (RG)

Colored RG and bisimulation

What happens if we compute a bisimulation on the colored RS, using the name of transitions as the set of actions?

We get 6 states (see the colours) on which we can build an aggregated RG



Colored RG and bisimulation

But what is the point in building the RG and then aggregating it? It would be better to compute directly the aggregated state space, but how?

Problems:

• find a suitable representation for the element of the partition (equivalence classes) that allows for an easy check of

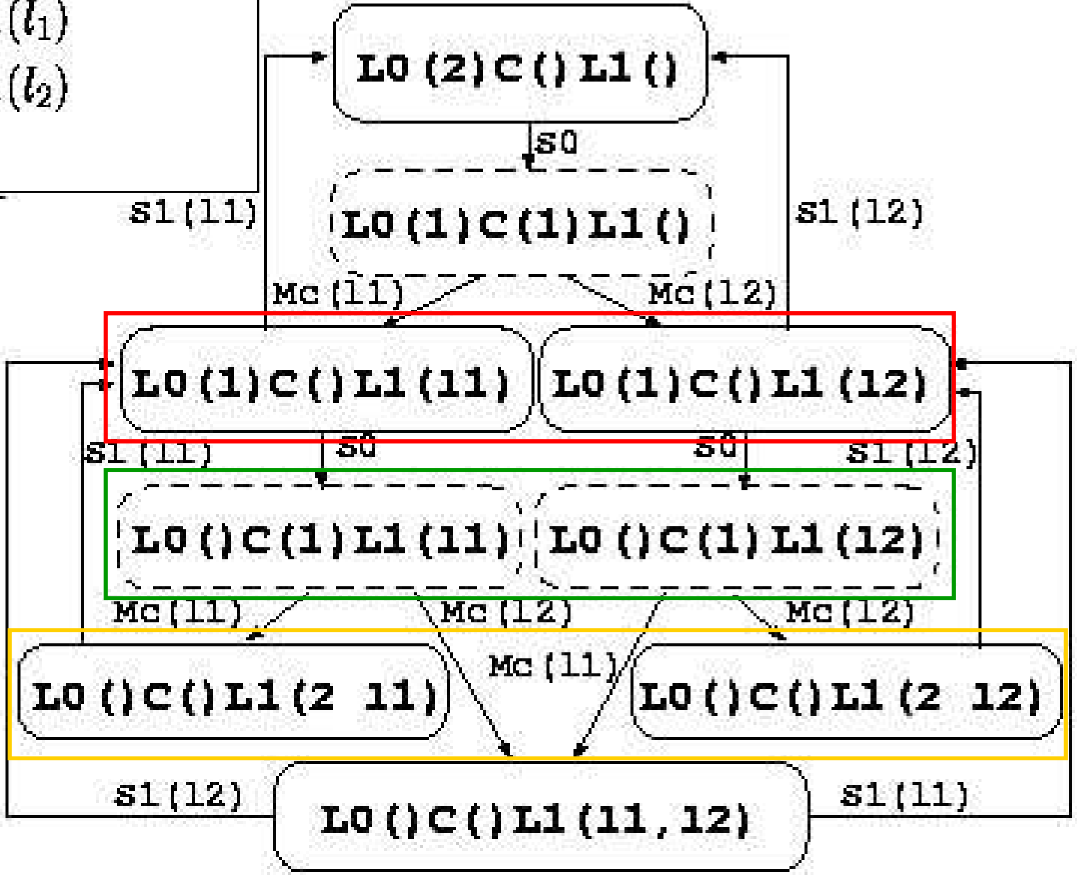
- marking is already in the set
- transition t is enabled in a marking / eq. class of markings

•Alternative: the process algebra approach: aggregate, compose, aggregate, compose, ….

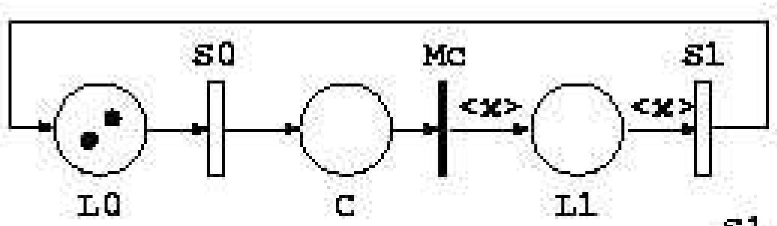
Equivalent markings and transition instances (1)

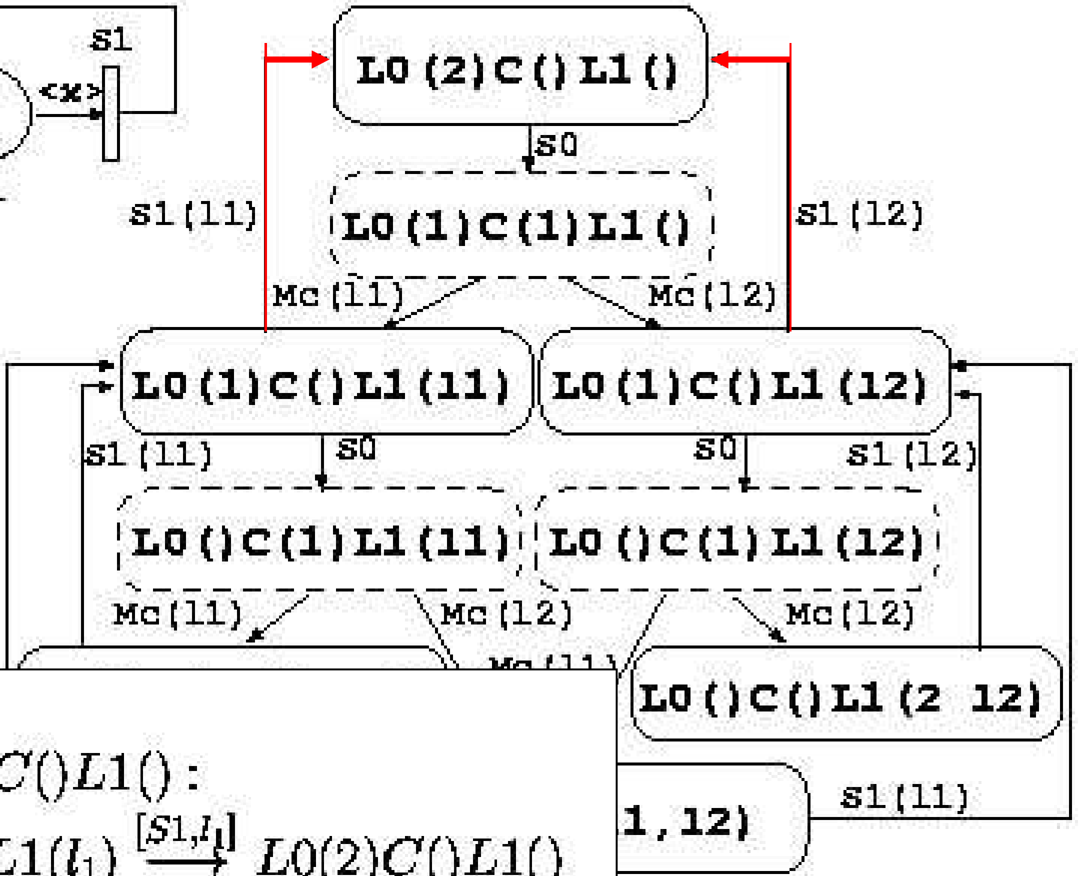
Equivalent markings

$$L0(1)C()L1(x1) : \begin{cases} L0(1)C()L1(l_1) \\ L0(1)C()L1(l_2) \end{cases}$$



Equivalent markings and transition instances (2)





Equivalent transition instances

$$L0(1)C()L1(x_1) \xrightarrow{[S1,x_1]} L0(2)C()L1():$$

$$\begin{cases} L0(1)C()L1(l_1) \xrightarrow{[S1,l_1]} L0(2)C()L1() \\ L0(1)C()L1(l_2) \xrightarrow{[S1,l_2]} L0(2)C()L1() \end{cases}$$

Equivalent markings and transition instances (3)

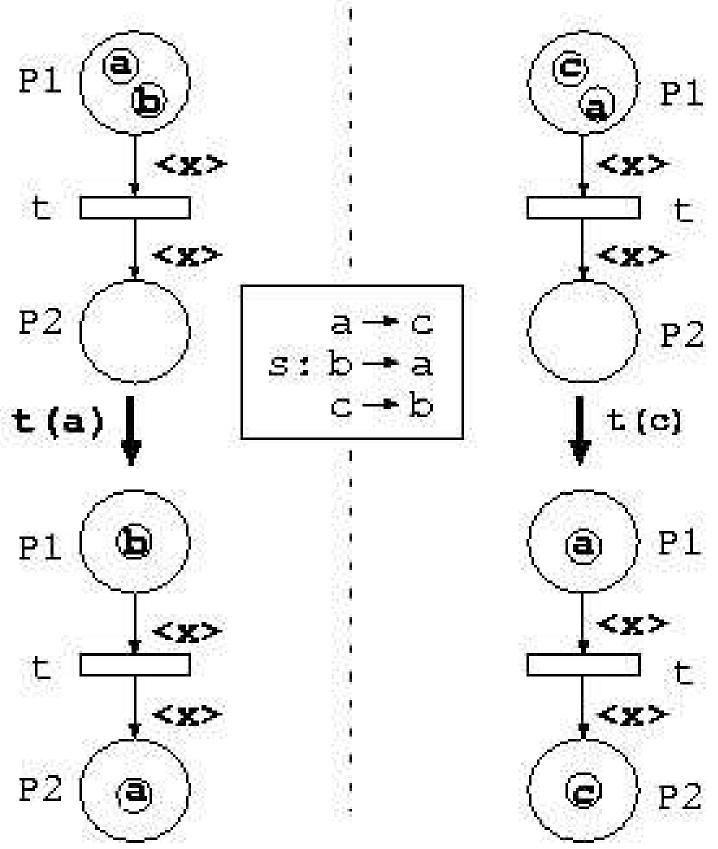
Intuitively: two markings are equivalent if they allow the same future evolution (same firable transition sequences), hence:

•equivalent markings must enable the same transition sets

• The markings reached by firing corresponding transitions from equivalent markings, must be equivalent themselves.

NOTE: this definition recalls the *bisimulation concept*

Marking and firing of equivalent transitions



Equivalence relation between markings

$$M \sim M' \Leftrightarrow \exists s \in \xi : s.M = M'$$

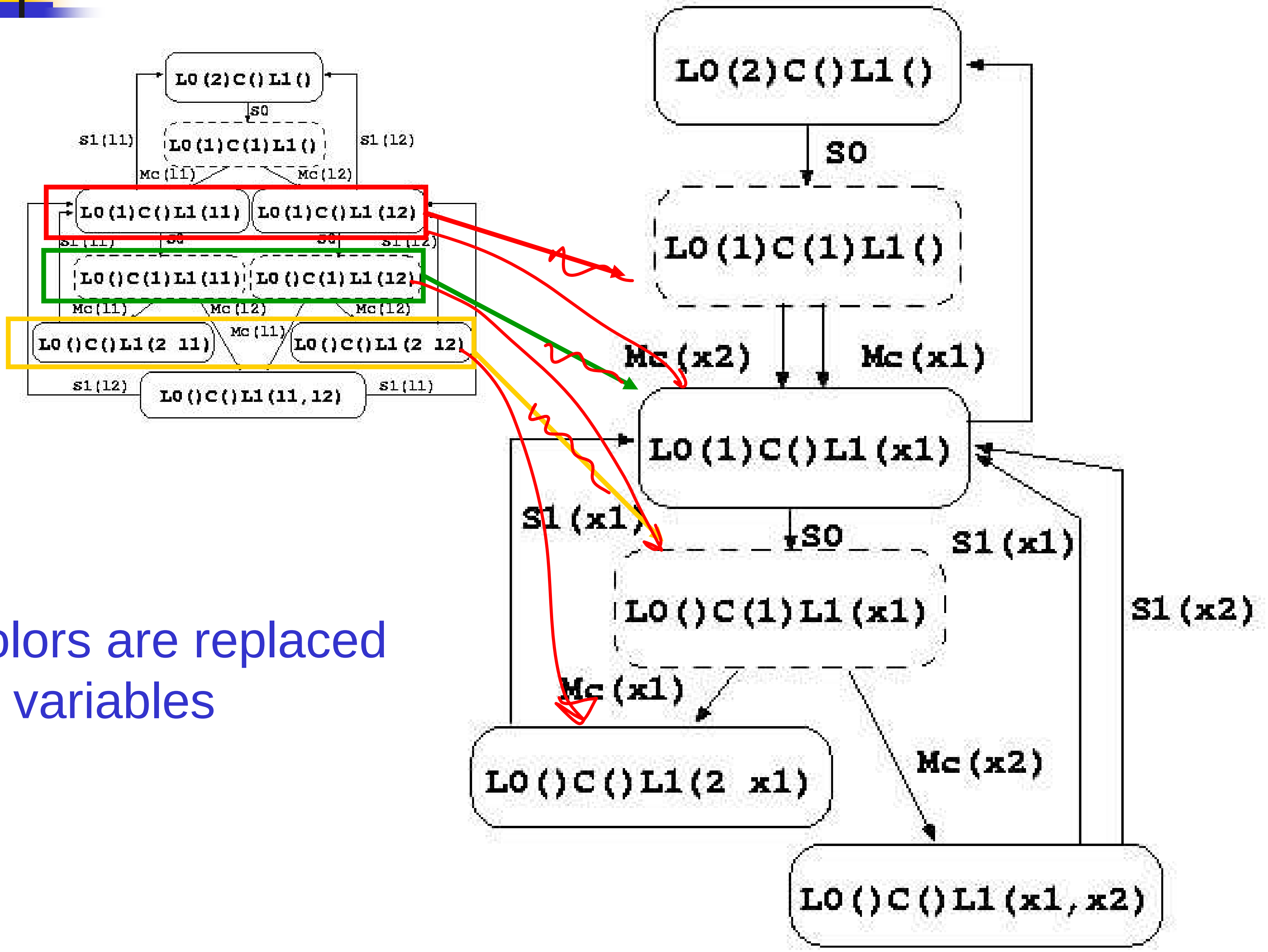
Colour permutations

“s” is a permutation that satisfies:

- it respects the static subclasses (colors of different subclasses cannot be permuted)
- for ordered color classes order should be respected (therefore only rotations are allowed)

On s we can build an equivalence relation upon which to partition the Reachability Set (RS) in *equivalence classes* (symbolic markings)

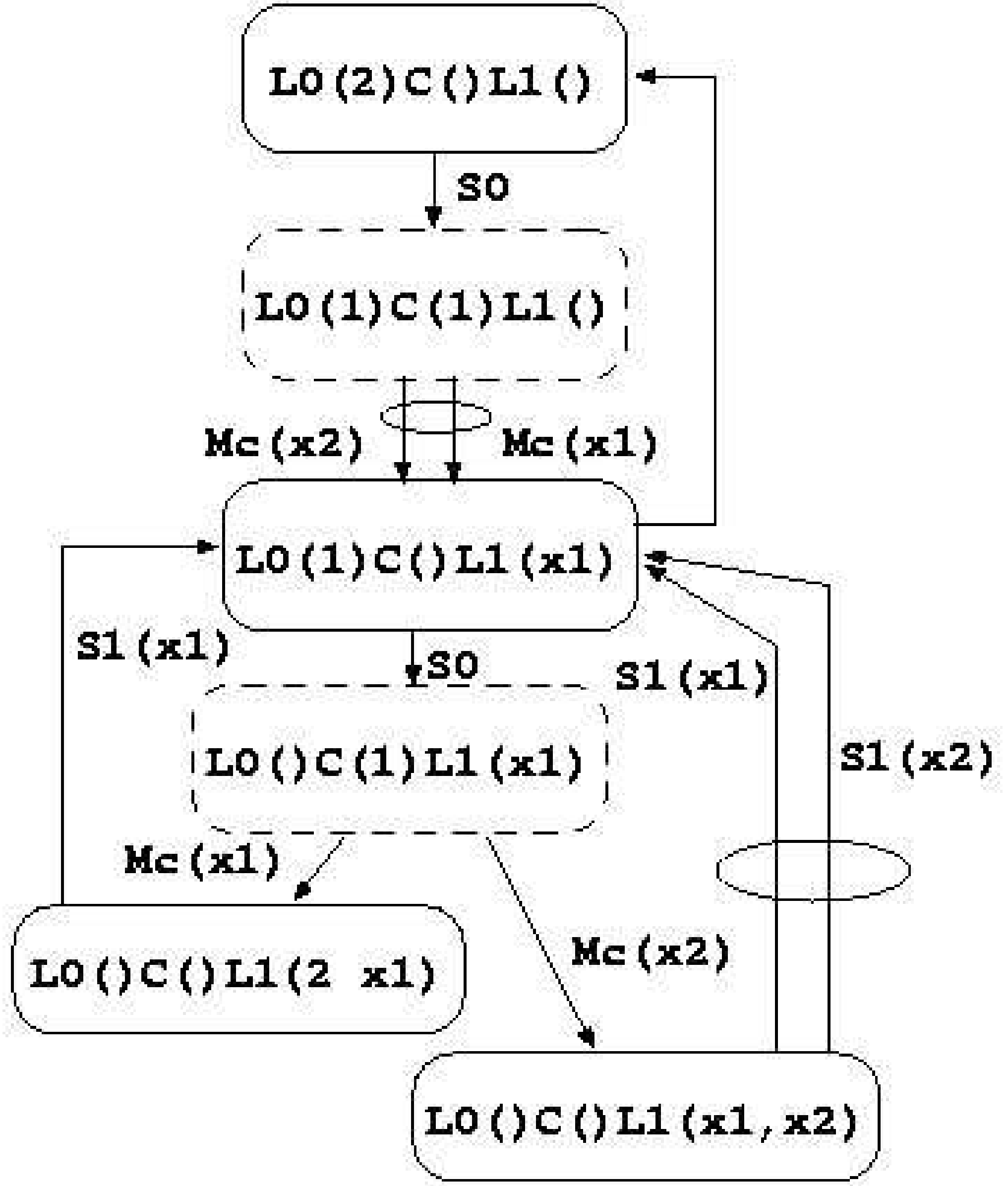
Towards a symbolic representation



Colors are replaced by variables

Towards a symbolic representation

``Similar'' transition instances are put together

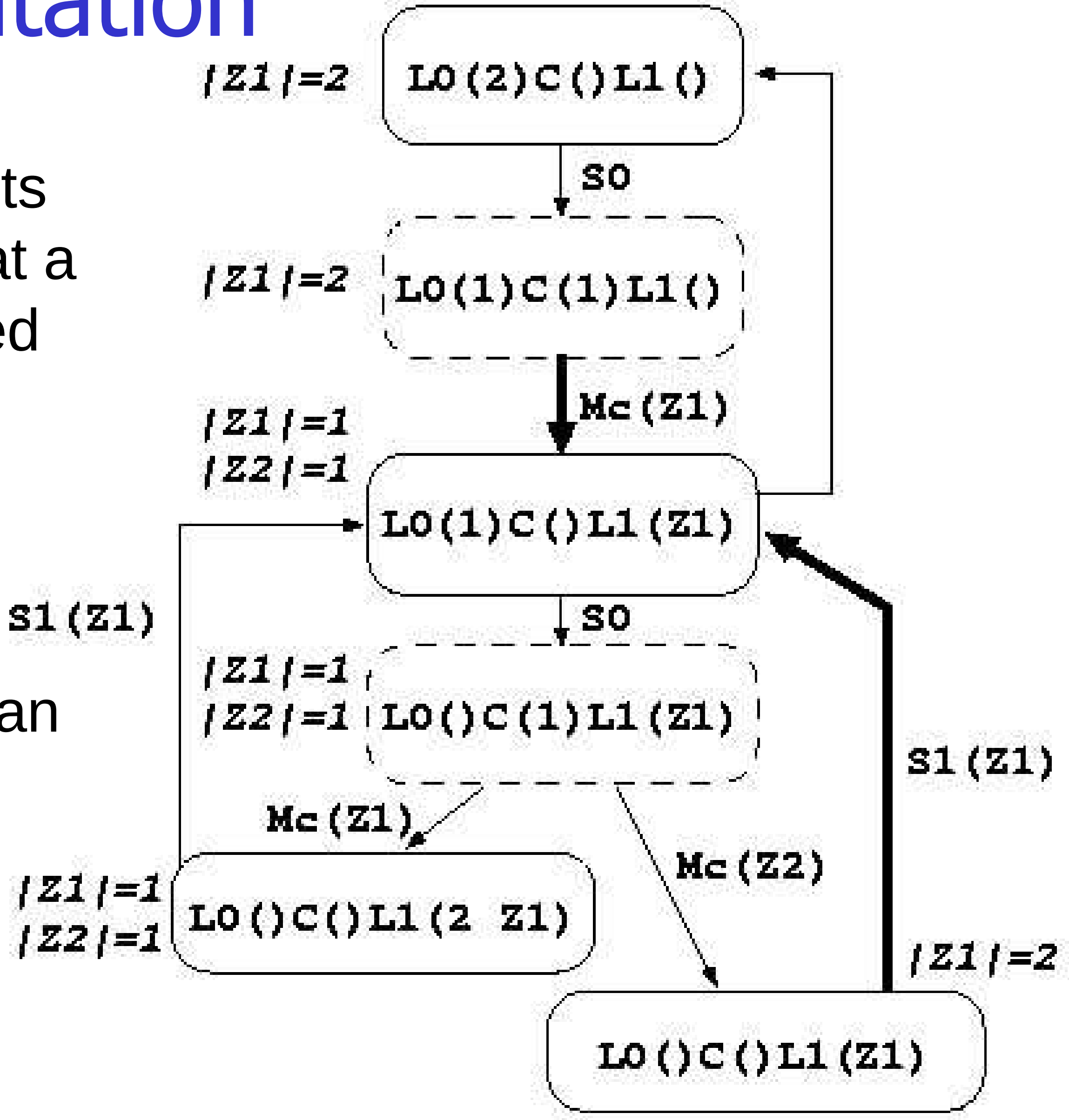


Towards a symbolic representation

"variables" now represents set of colours (instead that a single one): they are called *dynamic subclasses*

$C = Z_1 \cup Z_2$

put together colours with an equal distribution over places



Symbolic marking interpreted as state

There is a client in queue l1
and none in l2
There is a client in l2
and none in l1

⇒ One of the two queues (x1) has a client, the other is empty

Equivalent markings

$$L0(1)C()L1(x1) : \begin{cases} L0(1)C()L1(l_1) \\ L0(1)C()L1(l_2) \end{cases}$$

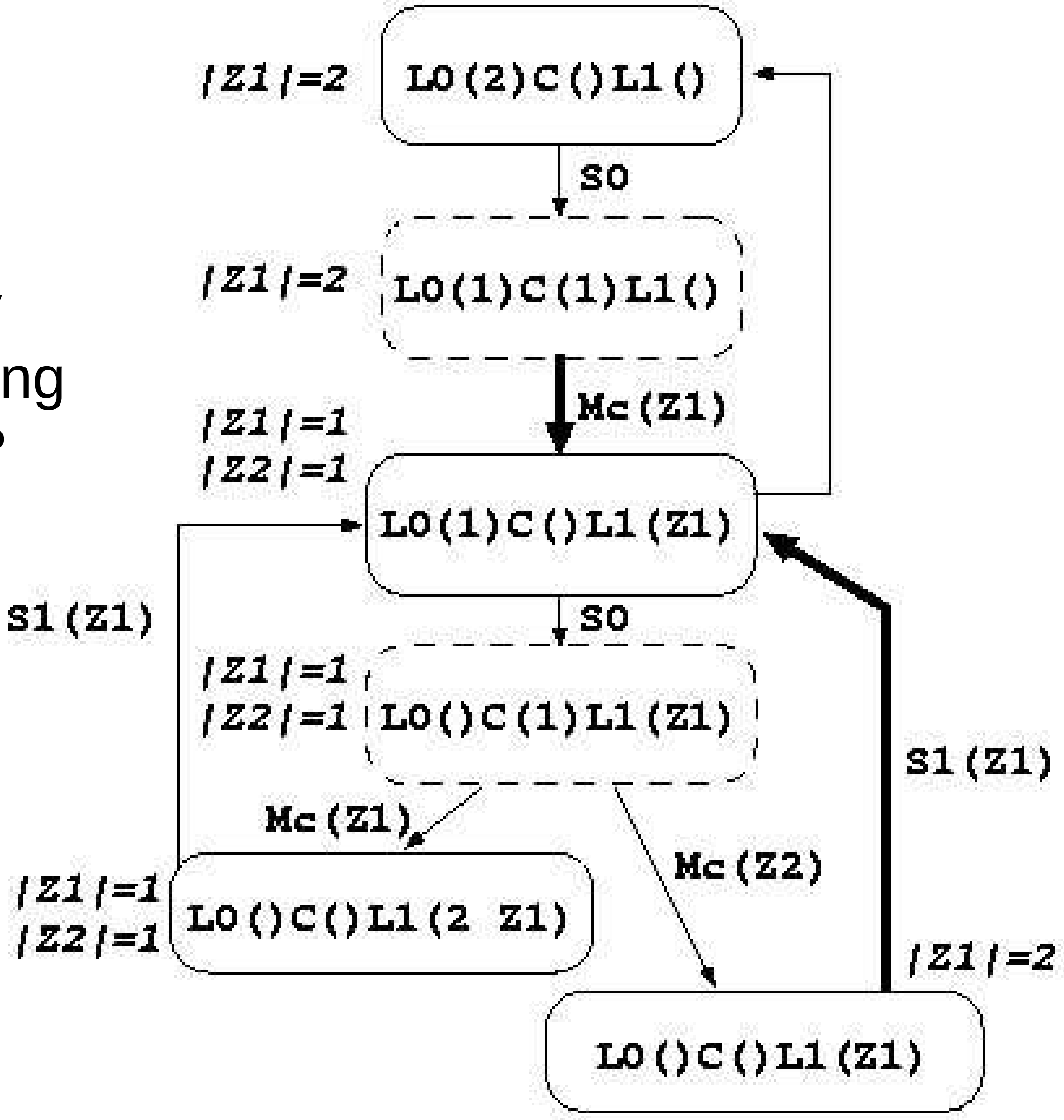
end of service at l1

⇒ End of service in the busy queue(x1)

$$L0(1)C()L1(x_1) \xrightarrow{[S1,x_1]} L0(2)C()L1() :$$
$$\begin{cases} L0(1)C()L1(l_1) \xrightarrow{[S1,l_1]} L0(2)C()L1() \\ L0(1)C()L1(l_2) \xrightarrow{[S1,l_2]} L0(2)C()L1() \end{cases}$$

Example

What are the ordinary markings corresponding to the symbolic ones?

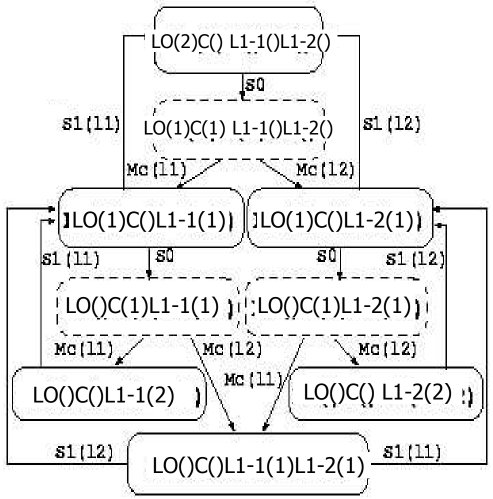


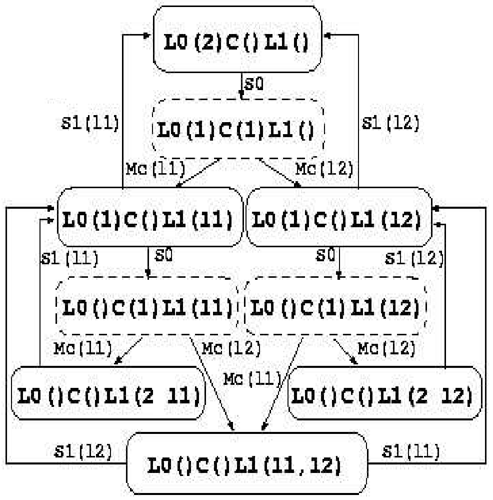
RG of unfolded, Colored RG, SRG

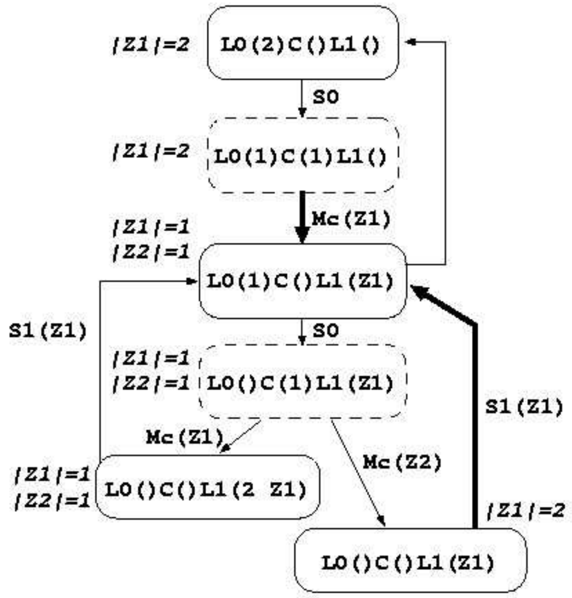
RG della rete unfolded

RG colorato della rete WN

Symbolic RG (SRG) della rete WN







Minimality and uniqueness of the representation

We need a unique symbolic reachability graph, therefore:
maximal partition and unique name for an equivalent class (symbolic marking)

Minimality and uniqueness of the representation

Se non si fornisce un criterio univoco su quanto raggruppare e che nomi dare alle sottoclassi dinamiche, la rappresentazione non e' unica, occorre:

• scegliere un raggruppamento che minimizza il numero di sottoclassi dinamiche (inducendo massima compattazione nella rappresentazione)

• numerare le sottoclassi dinamiche in modo da minimizzare (in senso lessicografico) una certa funzione della marcatura (si ottiene effettuando l'ordinamento di una rappresentazione matriciale della marcatura simbolica)

Symbolic firing rule

Goal: to build the equivalence classes and the aggregated reachability graph (Symbolic Reachability Graph) directly, and not as a partition of the ordinary reachability graph
This requires a symbolic firing rule.

-dynamic subclasses are assigned to variables (instead than single colours)
-the elements of the subclasses involved in the firing are kept separate (“splitting”)
-transition is fired and state is changed
-the obtained symbolic marking is normalized to obtain a symbolic marking according to the uniqueness criteria

Example of symbolic firing

$L0()C()L1(Z_1); \mid Z_1\mid=2$

$\Downarrow$ S1(Z1)

Splitting:

$L0()C()L1(Z_1,Z_2); \mid Z_i\mid=1$

$\Downarrow$ S1(Z1)

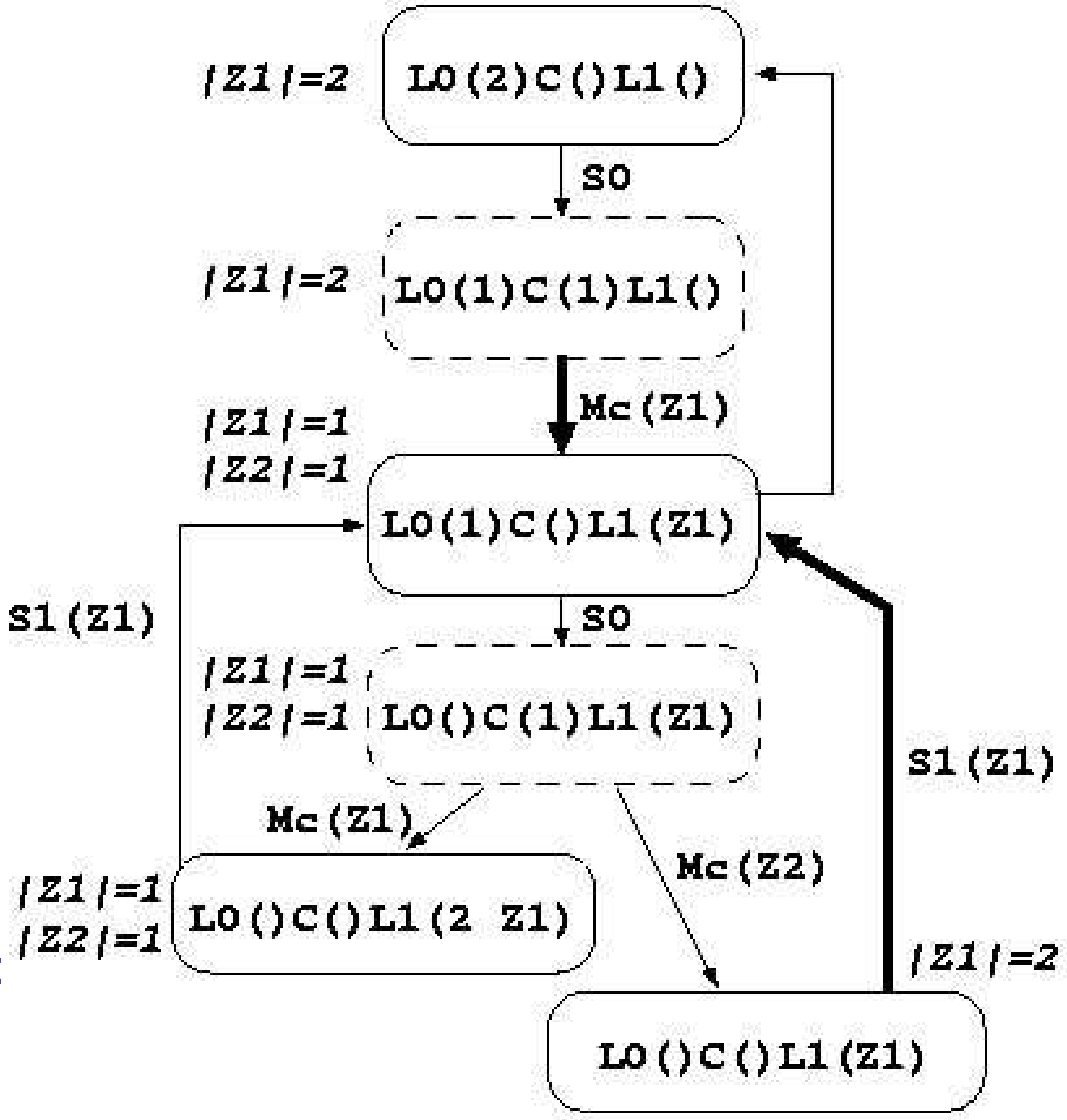
Firing:

$L0(1)C()L1(Z_2); \mid Z_i\mid=1$

Minimization is not required

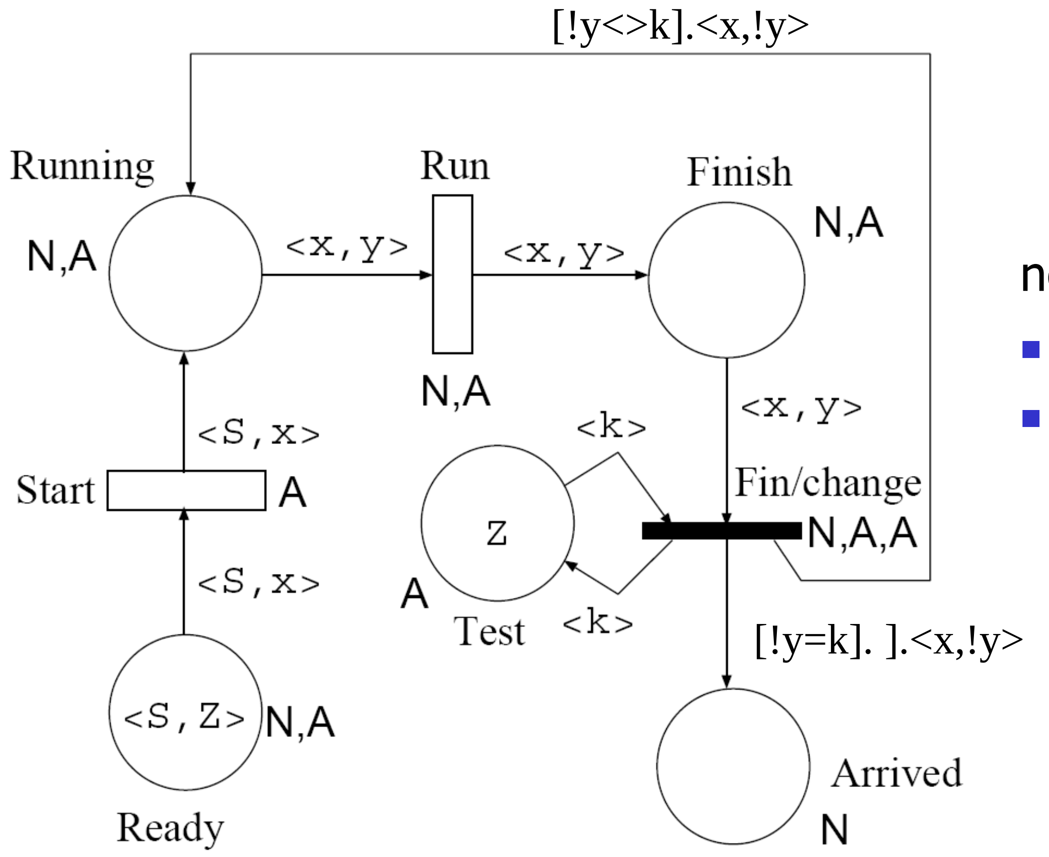
Naming for uniqueness:

$L0(1)C()L1(Z_1); \mid Z_i\mid=1$



Another example of symbolic firing

4 × 400 Olympic Relay Race



N = non-ordered set of nations

A = oredered id of 4 athlets per team

Z = dynamic subclass of A with cardinality 1

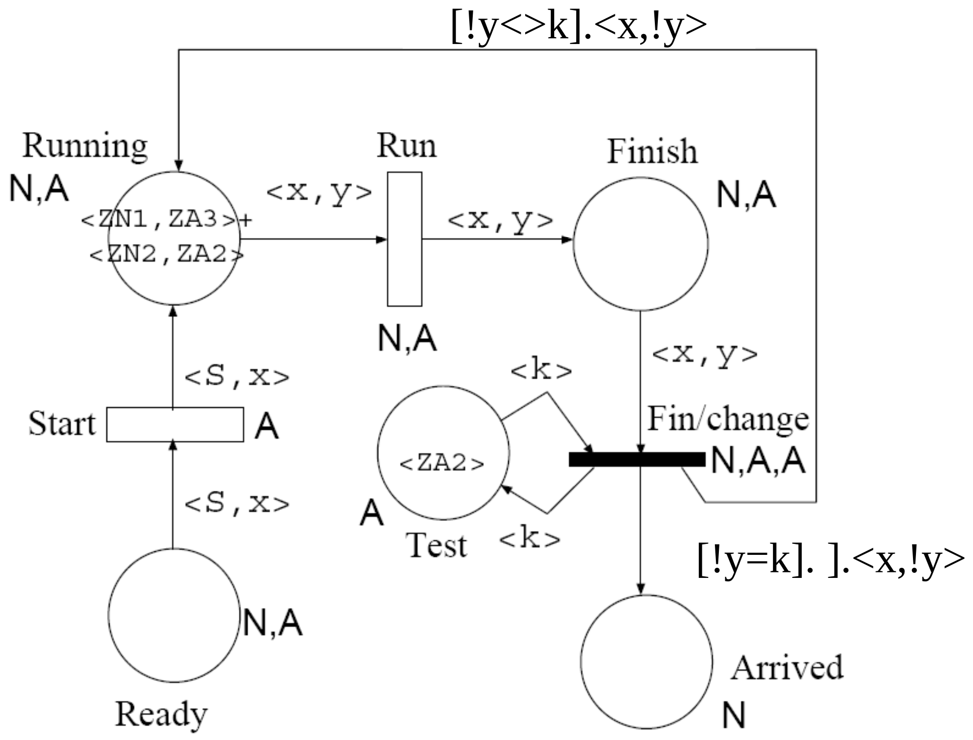
nella nuova interfaccia grafica:

- <!x> si scrive <x++>
- <S> si scrive <All>

Example of symbolic firing

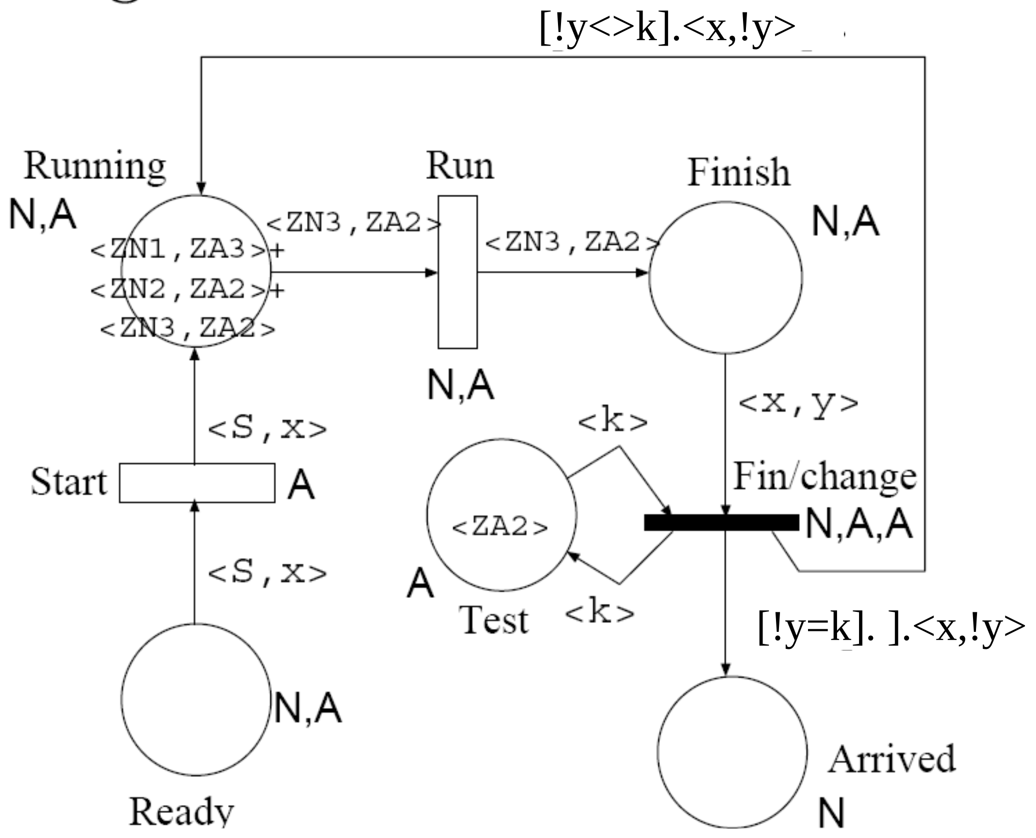
$$\mathcal{M} = Running(\langle ZN1, ZA3\rangle + \langle ZN2, ZA2\rangle)Test(\langle ZA2\rangle)$$

$$|ZN1| = 1,\ |ZN2| = 7,\ |ZA1| = 2,\ |ZA2| = 1,\ |ZA3| = 1$$



Example of symbolic firing

The splitting of marking $\mathcal{M}$:

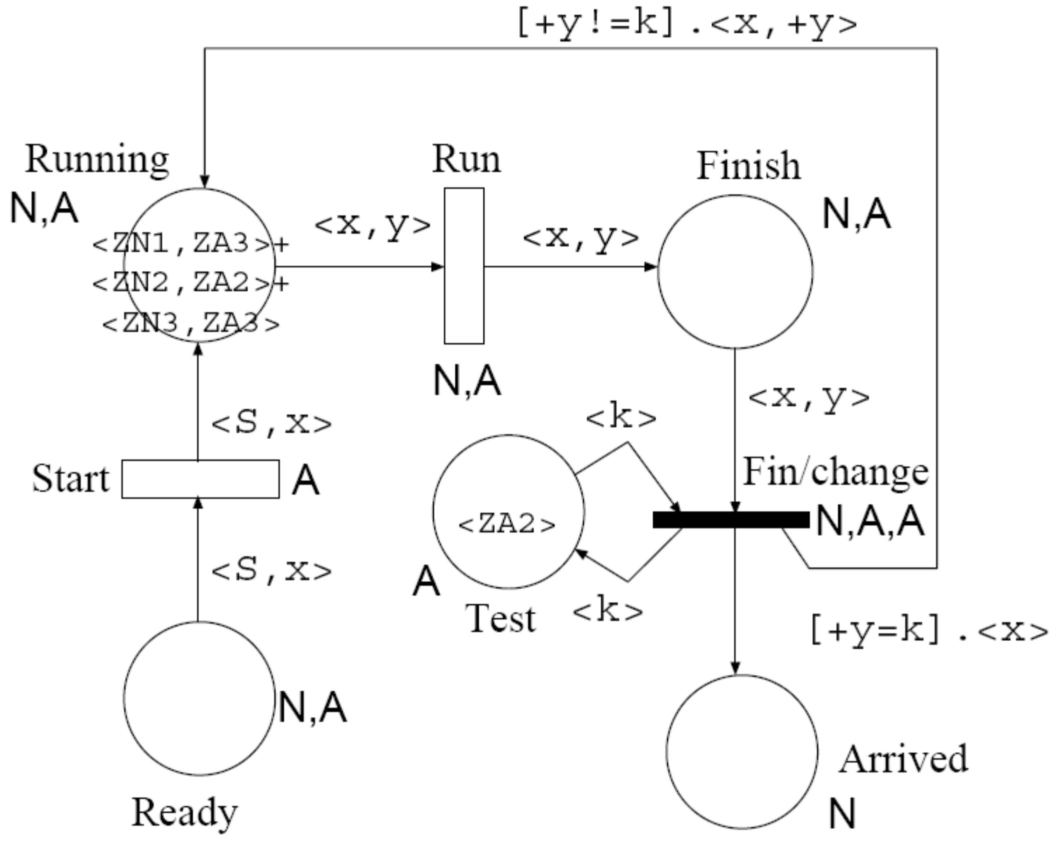


New tanglible state $\mathcal{M}'$: $\mathcal{M} \xrightarrow{Run} \cdot \xrightarrow{Fin/change} \mathcal{M}'$

$|ZN1| = 1,\ |ZN2| = 6,\ |ZN3| = 1,\ |ZA1| = 2,\ |ZA2| = 1,\ |ZA3| = 1$

Example of symbolic firing

Dynamic subclasses $ZN1$ and $ZN3$ can be merged.



After applying canonic marking algorithm:

$|ZN1| = 2,\ |ZN2| = 6,\ |ZA1| = 2,\ |ZA2| = 1,\ |ZA3| = 1$

Stochastic WN (SWN) - per gli studenti di valutazione delle prestazioni

Symbolic reachability graph + a set of rules on transition firing rates allows to derive from SRG a lumped Markov chain

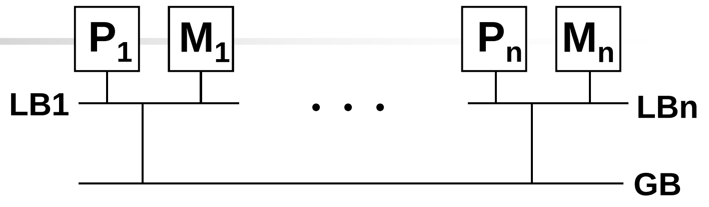
All states of an equivalence class are equiprobable, so there is no information loss w.r.t. the computation of the Markov chain generated by the ordinary coloured Reachability Graph

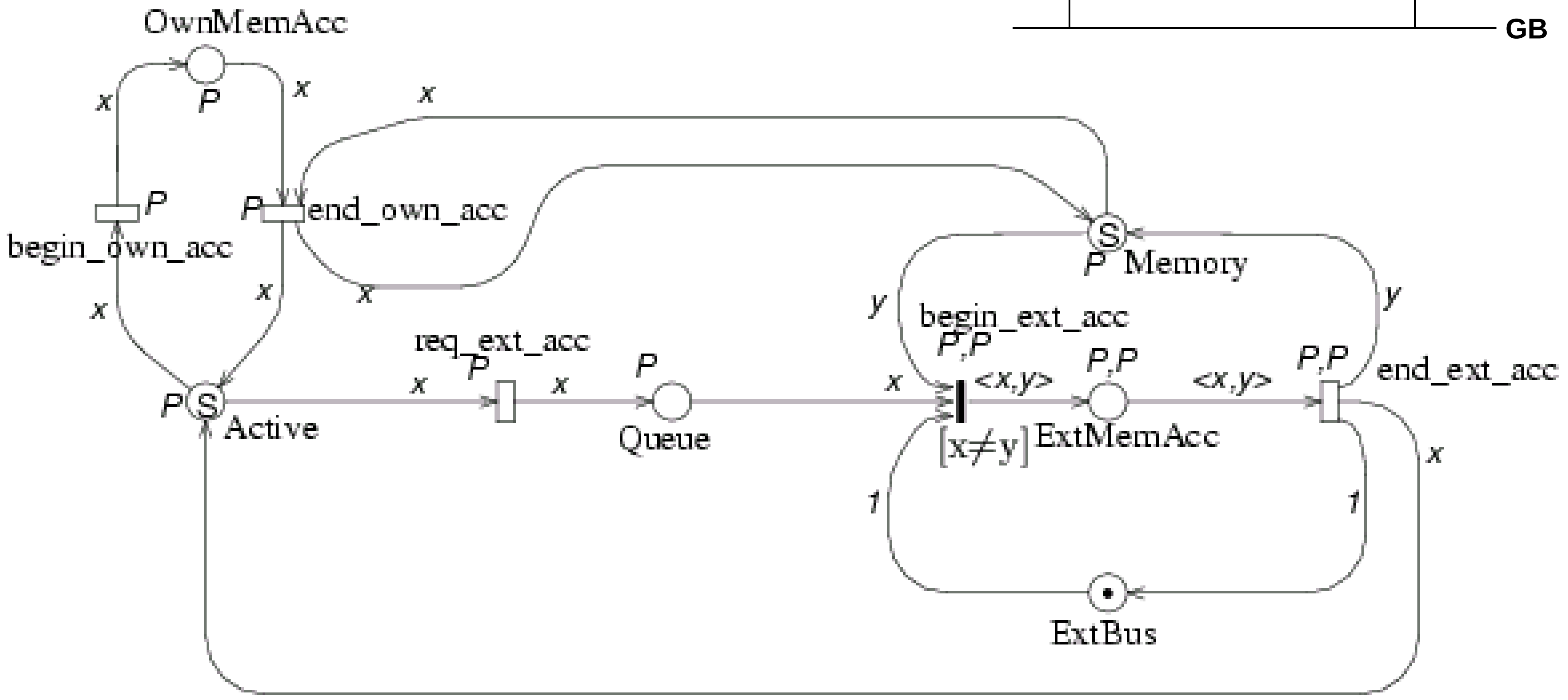
Saving memory and time

Saving depends on the level of symmetry in the (description of) the system

It can be an advantage being able to describe properties in terms of symbolic markings (for example P- and T- invariants can be described in a more abstract way): easier to reason about

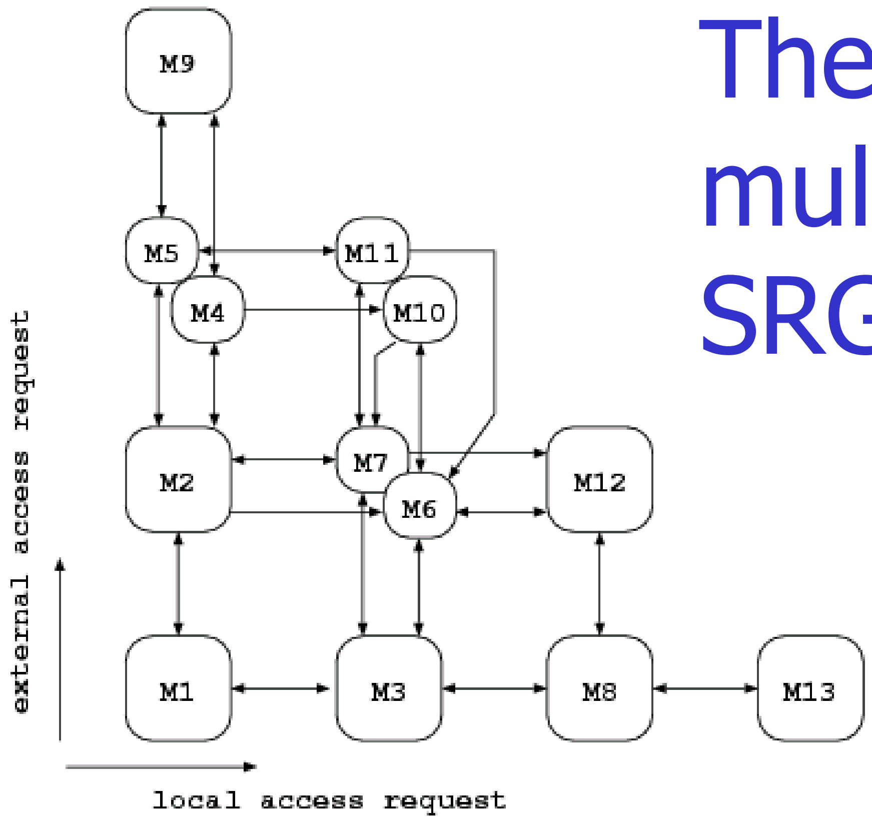
State space reduction in a multiprocessor system





States: Active, Accessing local memory (running or blocked), waiting for the GlobalBus, Accessing an external memory. An external access preempts the local bus.

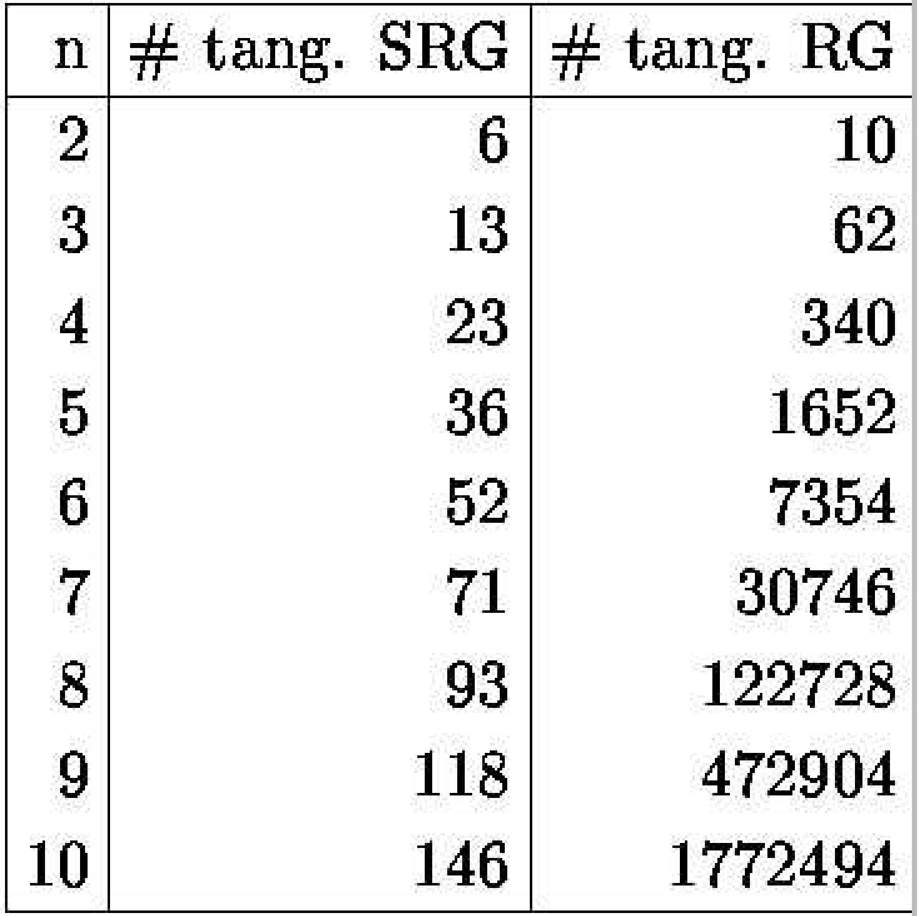
The multiprocessor SRG



Symbolic marking interpretation (forgetting processors identity)

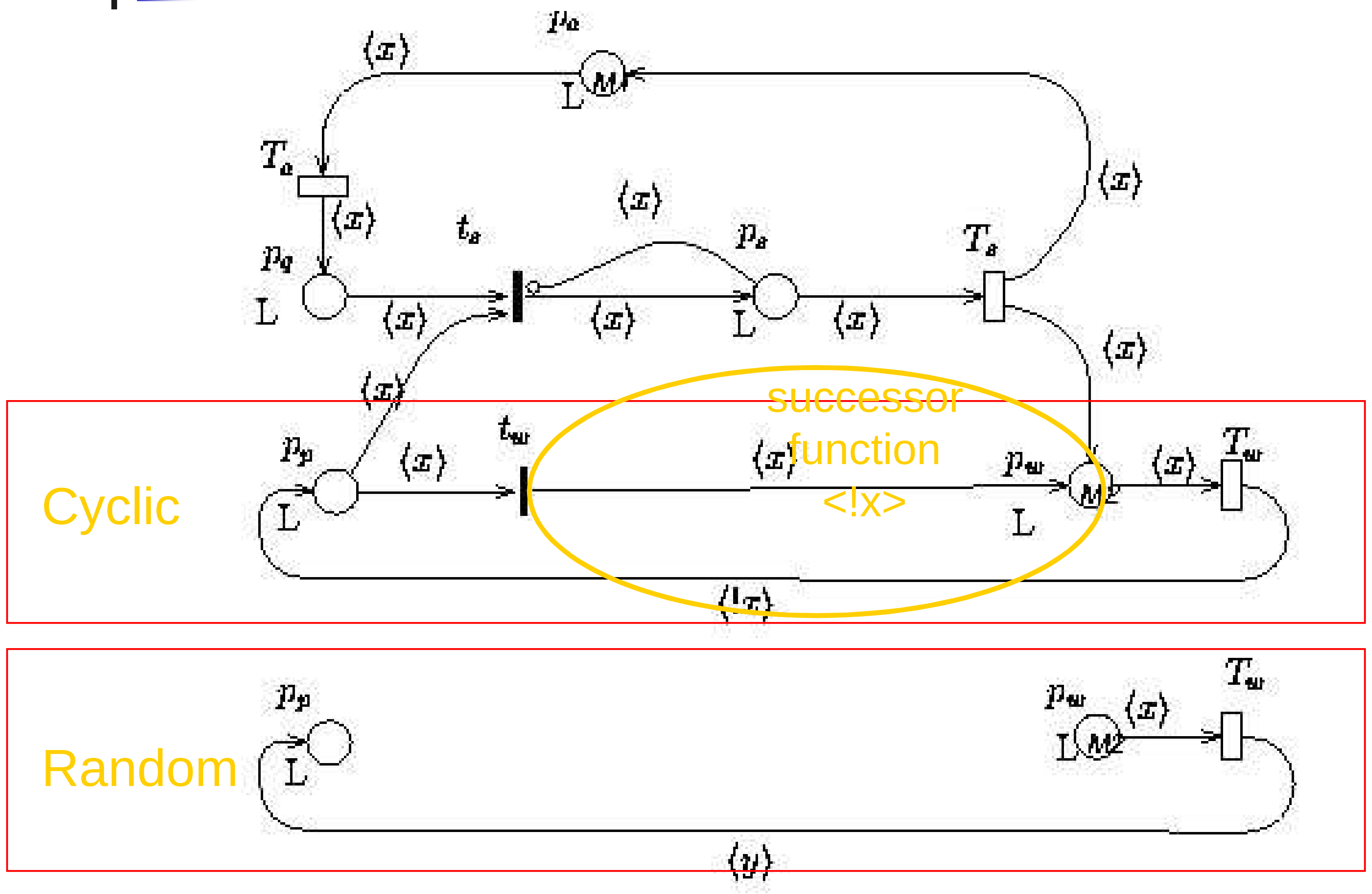
M1:	Three active processors
M2:	One external access, two active processors
M3:	One local access,two active processors
M4,M5:	One external access, one active processor, one queued
M6,M7:	One external access, one local access,one active processor
M8:	Two local accesses,one active processor
M9:	One external access, two processors queued
M10,M11:	One external access, one local access, one processor queued
M12:	One external access, two local accesses
M13:	Three local accesses

Multiprocessor: number of symbolic and ordinary states



n	# tang. SRG	# tang. RG
2	6	10
3	13	62
4	23	340
5	36	1652
6	52	7354
7	71	30746
8	93	122728
9	118	472904
10	146	1772494

State space reduction in the cyclic and random polling system

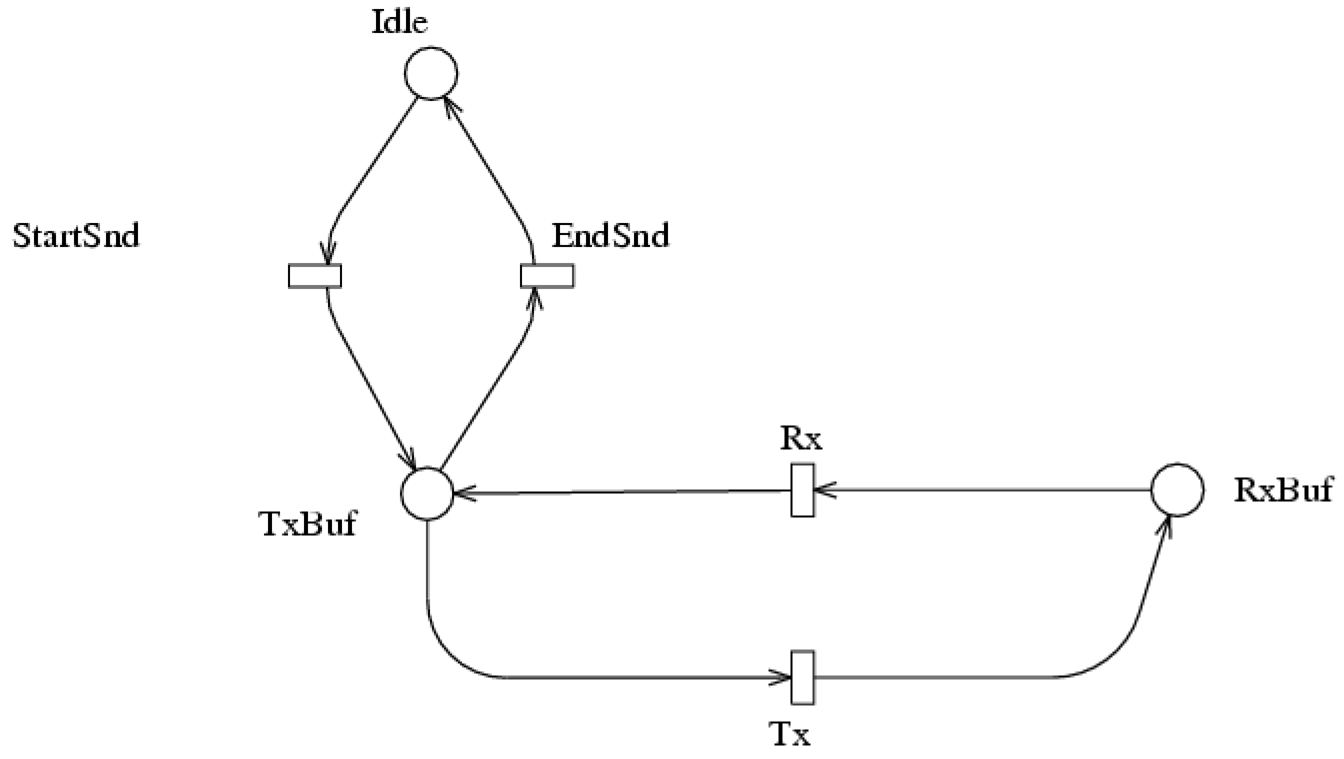


Polling system: number of symbolic and ordinary states states

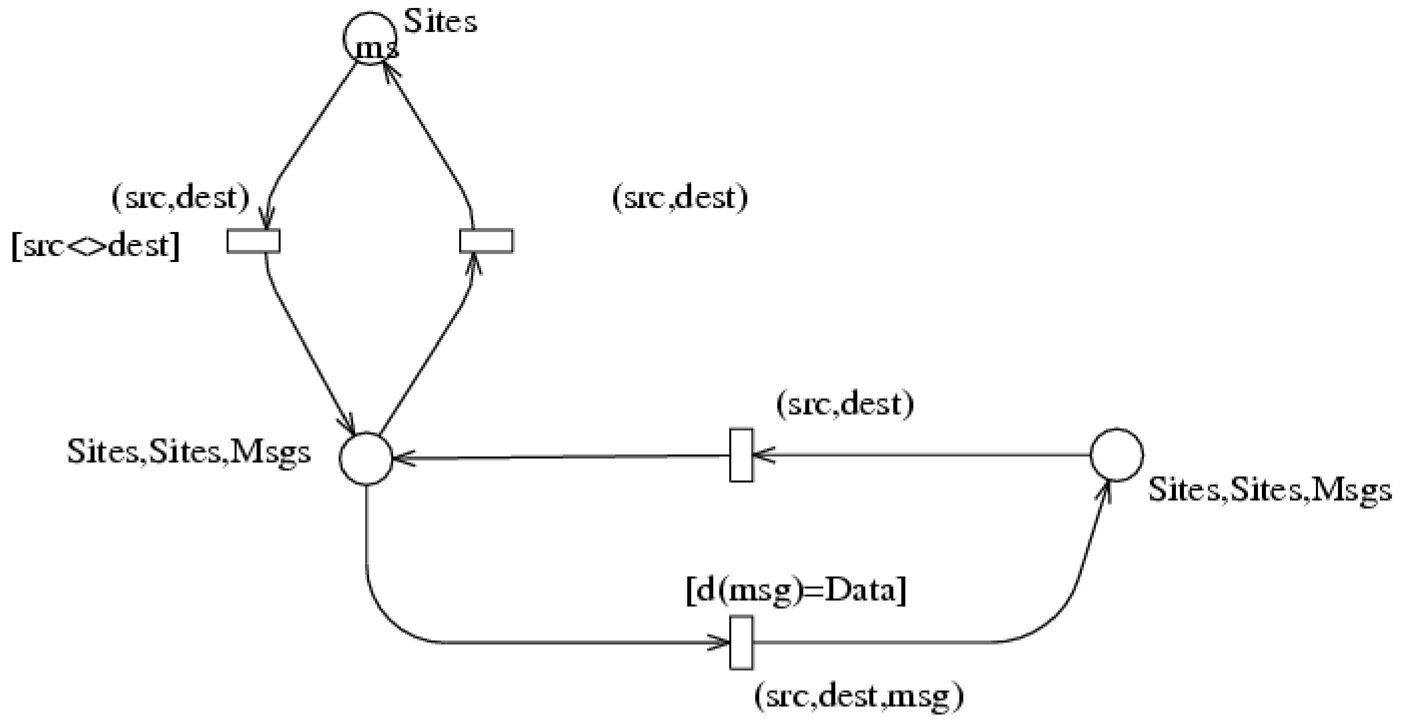
	Cyclic polling				Random polling			
	SRG		RG		SRG		RG	
$\lvert L \rvert$	# tang.	# van.	# tang.	# van.	# tang.	# van.	# tang.	# van.
4	81	96	312	384	30	30	312	384
5	192	240	960	1200	39	39	960	1200
6	462	576	2736	3456	48	48	2736	3456
7	1056	1344	7392	9408	57	56	7392	9408
8	2412	3072	19200	24576	66	66	19200	24576
9	5376	6912	48384	62208	75	75	48384	62208
10	11928	15360	119040	153600	84	84	119040	153600

- "tang." is the number of tangible states (states that enable transitions of priority zero)
- "van." is the number of vanishing states (states that enable transitions of priority one or more)

Sender-receiver

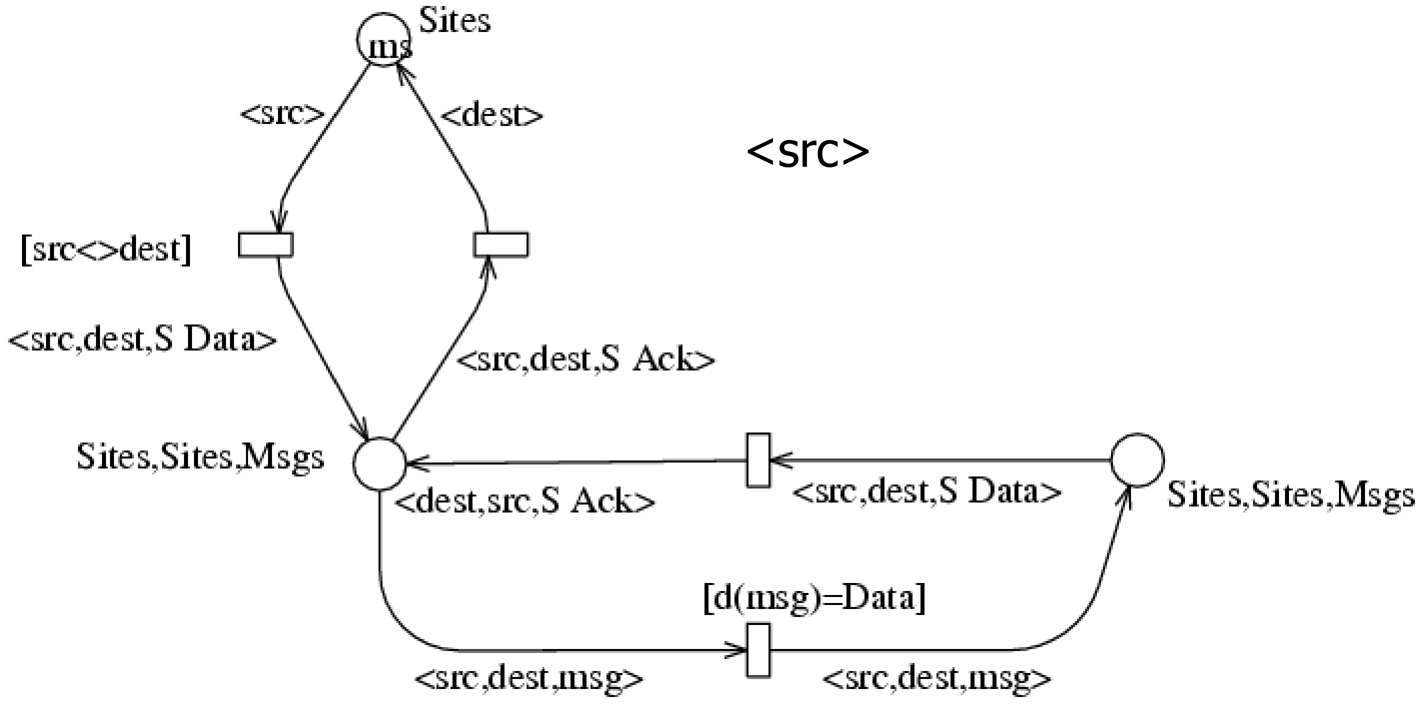


Sender-receiver



Classe dei messaggi: {m_1..m_n} is Data + {ack} is Ack

Sender-receiver

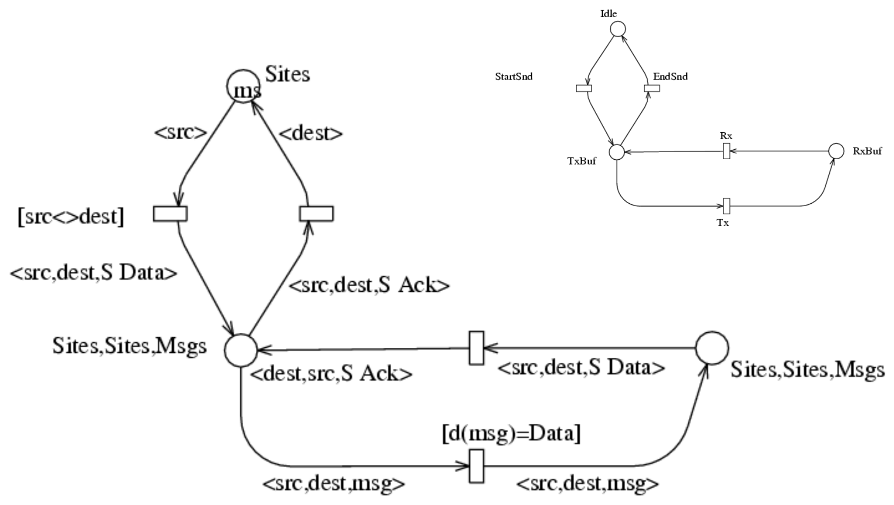


<src>

Classe dei messaggi: {m_1..m_n} is Data + {ack} is Ack

Sender-receiver

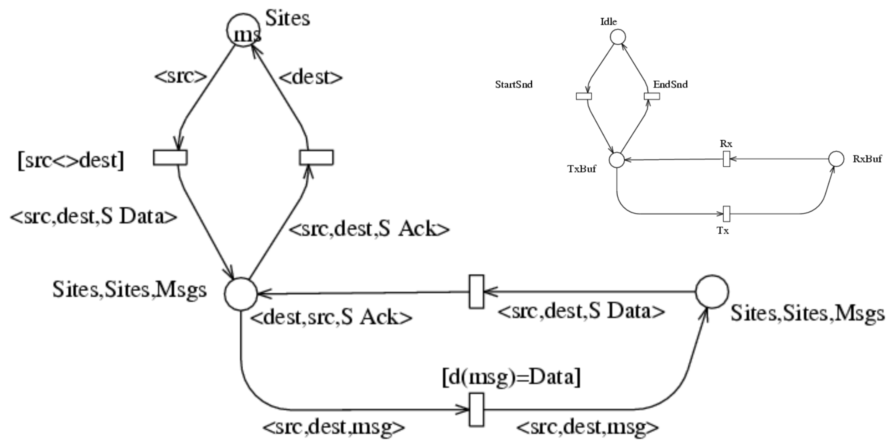
Colored reachability graph



```
Idle(1<s1>1<s2>1<s3>1<s4>)

>>>>StartSnd(src <-- s1,dest <-- s4)

Idle(1<s2>1<s3>1<s4>)TxBuf(1<s1,s4,m1>1<s1,s4,m2>)

>>>>Tx(msg <-- m1,src <-- s1 ,dest <-- s4)

Idle(1<s2>1<s3>1<s4>)TxBuf(1<s1,s4,m2>)RxBuf(1<s1,s4,m1>)

>>>>StartSnd(src <-- s3,dest <-- s1)

Idle(1<s2>1<s4>)TxBuf(1<s1,s4,m2>1<s3,s1,m1>1<s3,s1,m2>)
RxBuf(1<s1,s4,m1>)
```

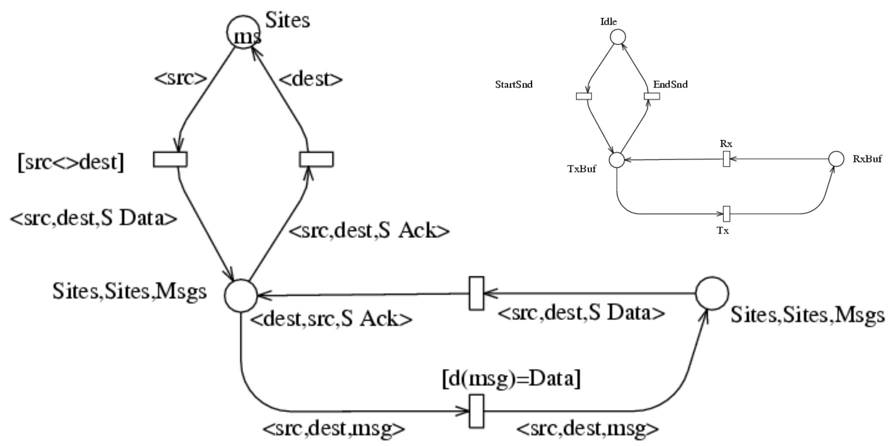
Sender-receiver

Colored reachability graph



```
Idle(1<s2>1<s4>)TxBuf(1<s1,s4,m2>1<s3,s1,m1>1<s3,s1,m2>)
RxBuf(1<s1,s4,m1>)

>>>>Tx(msg <-- m2,src <-- s1 ,dest <-- s4)

Idle(1<s2>1<s4>)TxBuf(1<s3,s1,m1>1<s3,s1,m2>)
RxBuf(1<s1,s4,m1>1<s1,s4,m2>)

>>>>Rx(src <-- s1 ,dest <-- s4)

Idle(1<s2>1<s4>)TxBuf(1<s3,s1,m1>1<s3,s1,m2>1 <s4,s1,ack>

>>>>EndSnd(src <-- s4 ,dest <-- s1

Idle(1<s1>1<s2>1<s4>)TxBuf(1<s3,s1,m1>1<s3,s1,m2>)
```

Sender-receiver



Symbolic markings

$$\text{Let } |Sites| = 4, |Data\_msgs| = 2$$

|Dat1|=1 |Dat0|=1 |Ack2|=1 |Sit0|=1 |Sit1| = 2 |Sit2|=1

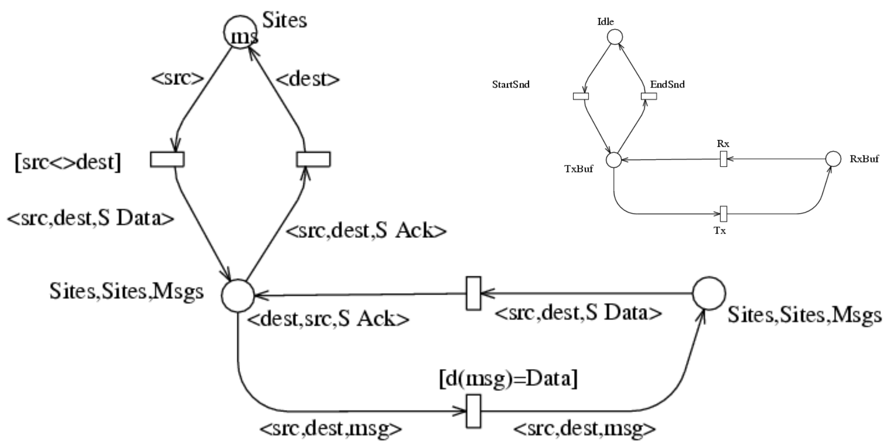
TxBuf(1<Sit0,Sit2,Dat1>)Idle(1<Sit1>1<Sit2>)
RxBuf(1<Sit0,Sit2,Dat0>)

|Dat0|=2 |Ack2|=1 |Sit0|=1 |Sit1| = 2 |Sit2|=1

Idle(1<Sit1>1<Sit2>)RxBuf(1<Sit0,Sit2,Dat0>)

Sender-receiver

Symbolic firing

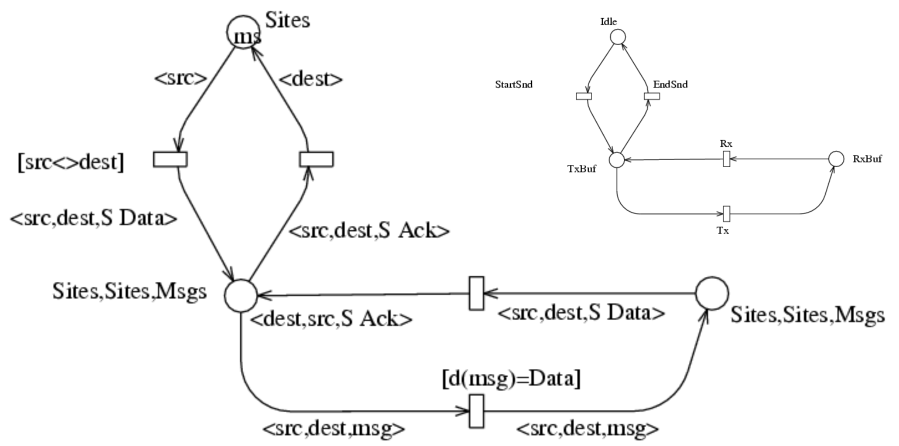


$$\text{Let } |Sites| = 4, |Data\_msgs| = 2$$

```
Idle(1<Sit0>)
|Dat0|=2 |Ack2|=1 |Sit0|=4


StartSnd(src <-- Sit0:1,dest <-- Sit0:2)


TxBuf(1<Sit0,Sit2,Dat0>)Idle(1<Sit1>1<Sit2>)
|Dat0|=2 |Ack2|=1 |Sit0|=1 |Sit1|=2 |Sit2|=1
```

Sit0 viene splittata in due: Sit0:1

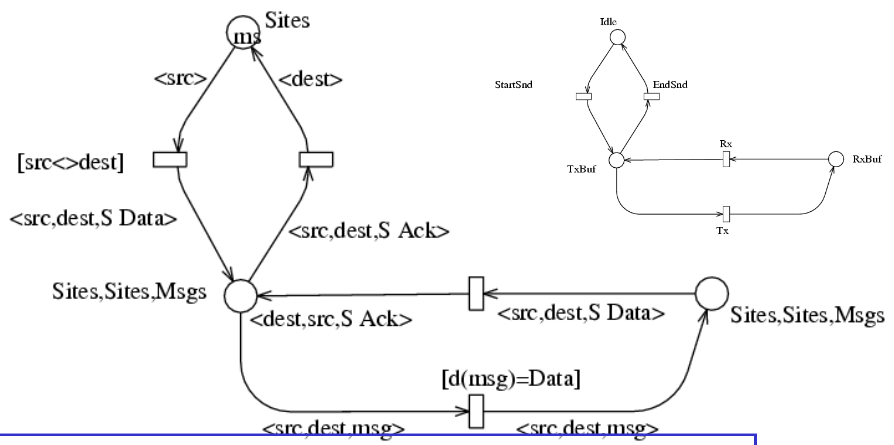
Sender-receiver



Symbolic firing

```
TxBuf(1<Sit0,Sit2,Dat0>)Idle(1<Sit1>1<Sit2>)
|Dat0|=2 |Ack2|=1 |Sit0|=1 |Sit1|=2 |Sit2|=1


Tx(msg <-- Dat0,src <-- Sit0 ,dest <-- Sit2)

Idle(1<Sit0>)
|Dat0|=2 |Ack2|=1 |Sit0|=4
```

Sender-receiver

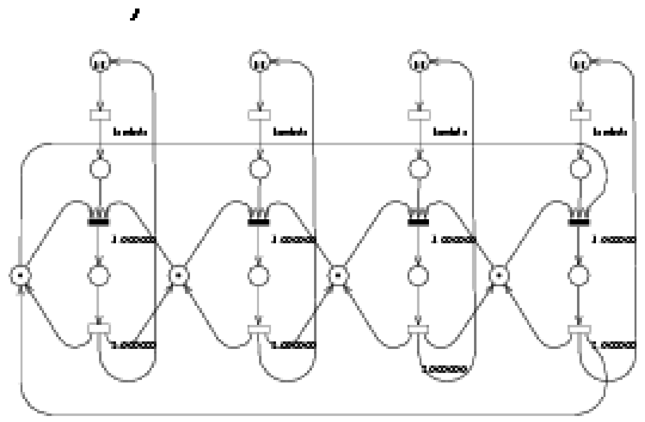


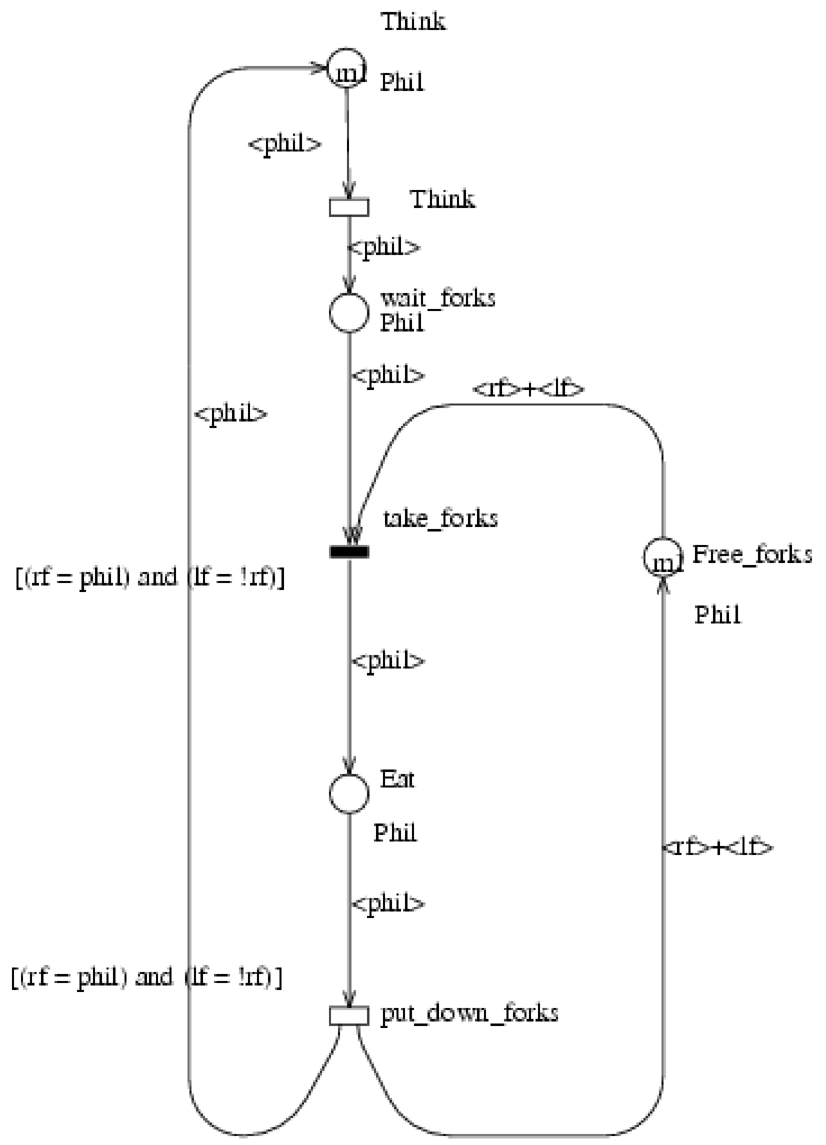
Symbolic firing

```
Idle(1<Sit0>)
|Dat0|=2  |Ack2|=1  |Sit0|=4

-- StartSnd(src <-- Sit0:1,dest <-- Sit0:2) -->

TxBuf(1<Sit0,Sit2,Dat0>)Idle(1<Sit1>1<Sit2>)
|Dat0|=2  |Ack2|=1  |Sit1|=2  |Sit0|=1  |Sit2|=1

-- Tx(msg <-- Dat0,src <-- Sit0 ,dest <-- Sit2) -->

TxBuf(1<Sit0,Sit2,Dat1>)Idle(1<Sit1>1<Sit2>)
RxBuf(1<Sit0,Sit2,Dat0>)
|Dat1|=1  |Dat0|=1  |Ack2|=1  |Sit0|=1  |Sit1|=2  |Sit2|=1
```

Philosopher

Colored and neutral





$$\text{Phil} = ph\{1-n\}$$
$$\oplus\, ph_i = ph_{((i+1)\ mod\ n)}$$