Petri Nets Tutorial, from Symmetric Nets to Symmetric Nets with Bags (session 3)

Souheib Baarir, Fabrice Kordon, Laure Petrucci

Souheib.Baarir@lrde.epita.fr — LRDE, Epita
Fabrice.Kordon@lip6.fr — LIP6, Université Pierre & Marie Curie
Laure.Petrucci@lipn.univ-paris13.fr — LIPN, Université Paris 13

June 23th, 2015

Outline

- Symmetries in Symmetric Nets
  - Towards the use of symmetries
  - Symbolic Marking
  - Symbolic Firing
  - Symbolic Reachability Graph (SRG)
- Symmetric nets with Bags (SNB)
  - Syntactic extensions
  - Semantics (Firing rule)
  - “unfolding” into SN (when finite)
- Conclusion

Symmetries in Symmetric Nets

Introduction

At this stage, you know:

- Symmetric Nets with their syntax and semantics
- how to build a Reachability Graph
- how it can be used for system analysis
- how to use CosyVerif platform to practice these concepts and formalisms.

Introduction

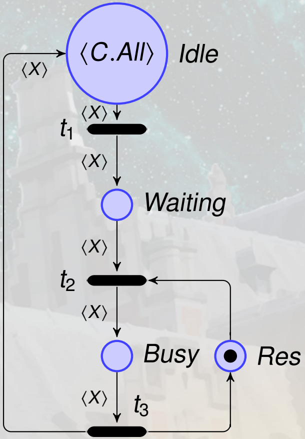
At this stage, you know:

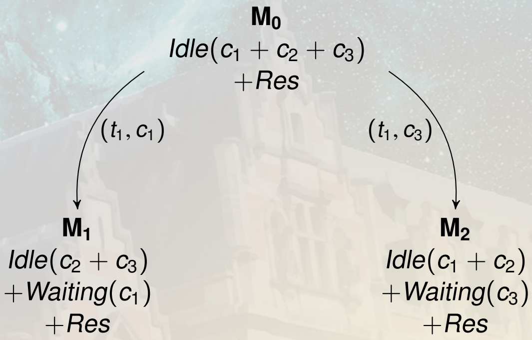
- Symmetric Nets with their syntax and semantics
- how to build a Reachability Graph
- how it can be used for system analysis
- how to use CosyVerif platform to practice these concepts and formalisms.

Let's now have an idea about the use of symmetries to reduce the size of the constructed structures.

Towards the use of symmetries (1/2)

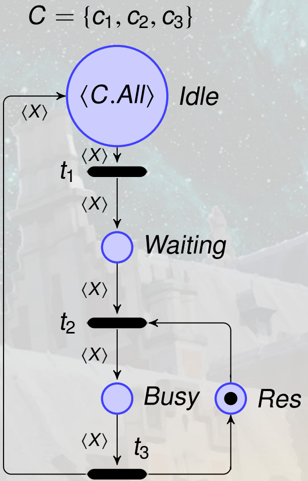
$C = \{c_1, c_2, c_3\}$

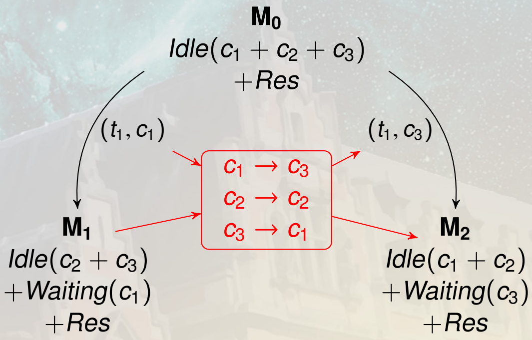




- In the initial Marking, t_1 is enabled for each colour instance marking of *Idle*.

Towards the use of symmetries (1/2)





- In the initial Marking, t_1 is enabled for each colour instance marking of *Idle*.
- If we apply a permutation on the transition colour, the obtained markings are identical up to this permutation.

Towards the use of symmetries (2/2)

- We can represent this set of firings using variables:

$$\begin{array}{c} Idle(x+y+z)+Res \\ \Big\downarrow{\scriptstyle (t_1,z)} \\ Idle(x+y)+Waiting(z)+Res \end{array} \qquad \begin{array}{l} x,y,z \in C \\ x \neq y \neq z \end{array}$$

- Then, we obtain the actual firings by testing all possible instantiations for x, y and z.

Permutations on Bags

- Let A be a set, s a permutation on A, and a a bag of A.

$$s.a = s(\sum_{x \in A} a(x).x) = \sum_{x \in A} a(x).s(x)$$

- In particular : $s.a(s.x) = a(x)$ (notation : $s.c = s(c)$)

- Example:
  - Let $a = c_1 + 2.c_2$ be a bag of $A = \{c_1, c_2, c_3\}$, and
  - s, with $s.c_1 = c_3, \quad s.c_2 = c_1, \quad s.c_3 = c_2$, be a permutation on A,
  - then, $s.a = s(c_1 + 2.c_2) = s.c_1 + 2.(s.c_2) = c_3 + 2.c_1$

Conclusion

At this stage, you:

- know Symmetric Nets with their syntax and semantics,
- have an intuitive idea about the notion of symmetries in SNs.

Conclusion

At this stage, you:

- know Symmetric Nets with their syntax and semantics,
- have an intuitive idea about the notion of symmetries in SNs.

Let's now study, formally, these symmetries and their usage for the construction of a reduced reachability graph (next sequence).

Symmetries to reduce the Reachability Graph

Now, you:

- know Symmetric Nets with their syntax and semantics,
- have an intuitive idea about the notion of symmetries in SNs.

Introduction

Now, you:

- know Symmetric Nets with their syntax and semantics,
- have an intuitive idea about the notion of symmetries in SNs.

Let's now study, formally, these symmetries and their usage for the construction of a reduced reachability graph.

Symmetries and SNs

- Consider a net $N = \langle P, T, C, W^-, W^+, M_0 \rangle$.[1]

- Consider the set $S = \{\langle s_1, \ldots, s_n \rangle | s_i \in S_i\}$, where,
  1. With each unordered class C_i, we associate the (total) permutation group S_i.
  2. With each ordered class C_i, we associate the (total) rotation group S_i.

  We call S the set of symmetries of a N.

- Useful properties: let C_i be a colour class and $f_i : C(t) \rightarrow Bag(C_i)$ (a colour function) and s_i the associated symmetry.
  1. $f_i = X_i^j \Rightarrow s_i \circ f_i = f_i \circ s_i, \forall s_i \in S_i$.
  2. $f_i = C_i.All \Rightarrow s_i \circ f_i = f_i \circ s_i = C_i.All, \forall s_i \in S_i$.
  3. $f_i = X_i^j{++} \Rightarrow r_i \circ f_i = f_i \circ r_i, \forall r_i \in S_i$. (when C_i is ordered).

[1] At this step, we consider that transition guards do not refrence colors explicitly!

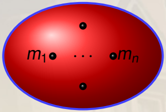
Markings Equivalence and Markings Classes

- Markings equivalence ($\equiv_S$):

$$M \equiv_S M' \Leftrightarrow \exists s \in S, M' = s.M$$

- For each marking M, we define its marking class (orbit) with respect to S, $[M]_S$:

$$[M]_S = \{M' \mid \exists s \in S, M' = s.M\}$$

$[M]_S$ 

Enabling Equivalence

(t, c_t) **is enabled in a marking** M

$$\Updownarrow$$

$(t, s.c_t)$ **is enabled in the marking** $s.M$

- (t, c_t) is enabled in a marking M

  $\Leftrightarrow\ M(p) \geq W^{-}(p, t)(c_t)$

  $\Leftrightarrow\ \forall c \in C(p), M(p)(c) \geq W^{-}(p, t)(c_t)(c)$

  $\Leftrightarrow\ \forall c \in C(p), s.M(p)(s.c) \geq s.W^{-}(p, t)(c_t)(s.c)$

  Since, $s.W^{-}(p, t)(c_t) = W^{-}(p, t)(s.c_t)$, then

  $\Leftrightarrow\ \forall c \in C(p), s.M(p)(s.c) \geq W^{-}(p, t)(s.c_t)(s.c)$

  $\Leftrightarrow\ \forall c \in C(p), s.M(p)(c) \geq s.W^{-}(p, t)(c_t)(c)$

  $\Leftrightarrow\ (t, s.c_t)$ is enabled in a marking $s.M$

Firing Equivalence

$$M \xrightarrow{(t,c_t)} M' \Leftrightarrow s.M \xrightarrow{(t,s.c_t)} s.M'$$

- $M \xrightarrow{(t,c_t)} M'$
  $\Leftrightarrow \; M'(p) = M(p) - W^{-}(p,t)(c_t) + W^{+}(p,t)(c_t)$
  $\Leftrightarrow \; s.M'(p) = s.M(p) - s.W^{-}(p,t)(c_t) + s.W^{+}(p,t)(c_t)$
  Since, $s.W^{-}(p,t)(c_t) = W^{-}(p,t)(s.c_t)$, and
  $s.W^{+}(p,t)(c_t) = W^{+}(p,t)(s.c_t)$, then
  $\Leftrightarrow \; s.M'(p) = s.M(p) - W^{-}(p,t)(s.c_t) + W^{+}(p,t)(s.c_t)$
  $\Leftrightarrow \; s.M \xrightarrow{(t,s.c_t)} s.M'$

Conclusion

At this stage, you know:

- Symmetric Nets with their syntax and semantics,
- the formal definition definition of symmetries in SNs,
- the formal definition of markings and firings equivalences.

Conclusion

At this stage, you know:

- Symmetric Nets with their syntax and semantics,
- the formal definition definition of symmetries in SNs,
- the formal definition of markings and firings equivalences.

How to use this notions to derive automatically a quotient reachability graph (next sequence).

Dynamic subclasses and Symbolic markings

Introduction

The definition of an adequate representation for marking classes, first consists in constructing a quotient graph that represents the ordinary reachability graph.

Introduction

The definition of an adequate representation for marking classes, first consists in constructing a quotient graph that represents the ordinary reachability graph.

This is achieved through the notions of:

- **Dynamic subclasses.**
- **Symbolic markings.**

Dynamic subclasses for unordered classes

- We group in a set (dynamic subclass) the objects of C_i that have the same marking.
- Example:
  - $M = Idle(c_1 + c_2) + Waiting(c_3) + Res$

    $$\Rightarrow Idle(x + y) + Waiting(z) + Res$$
    $$M(x) = M(y) \rightarrow Z^1, |Z^1| = 2$$
    $$M(z) \neq M(x) \text{ et } M(z) \neq M(y) \rightarrow Z^2, |Z^2| = 1$$

Dynamic subclasses for unordered classes

- We group in a set (dynamic subclass) the objects of C_i that have the same marking.
- Example:
  - $M = Idle(c_1 + c_2) + Waiting(c_3) + Res$

$$\Rightarrow Idle(x + y) + Waiting(z) + Res$$
$$M(x) = M(y) \rightarrow Z^1, |Z^1| = 2$$
$$M(z) \neq M(x) \text{ et } M(z) \neq M(y) \rightarrow Z^2, |Z^2| = 1$$

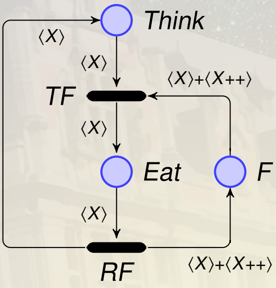
$$\Rightarrow \widehat{M} = Idle(Z^1) + Waiting(Z^2) + Res$$
$$|Z^1| = 2, |Z^2| = 1$$
(Symbolic Marking)

Dynamic subclasses for ordered classes

- A dynamic subclass represents objects that have the same marking and
  - are consecutive in the class enumeration order, and
  - the successor of the last element represented by Z^i is represented by Z^{i+1}.

- Example:
  - $Think(c_2 + c_4 + c_5) + Eat(c_1 + c_3) + F(c_5)$
    $\Rightarrow$ A dynamic subclass by object.
  - $Think(Z^2 + Z^4 + Z^5) + Eat(Z^1 + Z^3) + F(Z^5)$, $|Z^i| = 1$
  - $Think(c_1 + c_3 + c_5) + Eat(c_2 + c_4) + F(c_1)$
    $Think(c_1 + c_2 + c_4) + Eat(c_3 + c_5) + F(c_2)$
    $Think(c_2 + c_3 + c_5) + Eat(c_1 + c_4) + F(c_3)$
    $Think(c_1 + c_3 + c_4) + Eat(c_2 + c_5) + F(c_4)$
    $Think(c_2 + c_4 + c_5) + Eat(c_1 + c_3) + F(c_5)$

$C = \{c_1, c_2, c_3, c_4, c_5\}$



Conclusion

So far, we know:

- how to represent, in symbolic and unique way, the marking classes.

Conclusion

So far, we know:

- how to represent, in symbolic and unique way, the marking classes.

To construct directly a quotient graph that represents the ordinary reachability graph, we need a way to perform a firing rule, but applied directly to the symbolic markings (next sequence).

Symbolic Firing Rule

Introduction

We know:

- how to represent, in symbolic and unique way, the marking classes.

Introduction

We know:

- how to represent, in symbolic and unique way, the marking classes.

The definition of a symbolic firing rule that applies directly on symbolic representations, constitutes the second and final stage to obtain a quotient graph.

Symbolic Firing rule

- Before firing, we decompose the dynamic subclasses to isolate the objects that are used to instantiate the colour functions.
- Example:

$$\begin{array}{c} Idle(Z) + Res \\ |Z| = 3 \end{array} \longrightarrow \begin{array}{c} Idle(Z^1 + Z^{1,0}) + Res \\ |Z^1| = 2, |Z^{1,0}| = 1 \end{array}$$

  $Z^{1,0}$ contains the chosen object to instantiate X, Z^1 those that are not participating in the firing.

- We then apply the classical firing rule.
- After the firing, we must group the resulting subclasses...

Example

$$Think(Z^1 + Z^3) + F(Z^1) + Eat(Z^2)$$
$$|Z^1| = 3, |Z^2| = |Z^3| = 1$$

Example

$$Think(Z^1 + Z^3) + F(Z^1) + Eat(Z^2)$$
$$|Z^1| = 3, |Z^2| = |Z^3| = 1$$
$$Think(Z^{1,0} + Z^{1,1} + Z^{1,2} + Z^3) +$$
$$F(Z^{1,0} + Z^{1,1} + Z^{1,2}) + Eat(Z^2)$$
$$|Z^i| = 1$$

Example

$$Think(Z^1 + Z^3) + F(Z^1) + Eat(Z^2) \xrightarrow{(RF, Z^2)} Think(Z) + F(Z)$$

$$|Z^1| = 3, |Z^2| = |Z^3| = 1 \qquad |Z| = 5$$

$$Think(Z^{1,0} + Z^{1,1} + Z^{1,2} + Z^3) +$$
$$F(Z^{1,0} + Z^{1,1} + Z^{1,2}) + Eat(Z^2)$$
$$|Z^i| = 1$$

Example

$$\begin{array}{c} Think(Z^1+Z^3)+F(Z^1)+Eat(Z^2) \\ |Z^1|=3, |Z^2|=|Z^3|=1 \\ Think(Z^{1,0}+Z^{1,1}+Z^{1,2}+Z^3)+ \\ F(Z^{1,0}+Z^{1,1}+Z^{1,2})+Eat(Z^2) \\ |Z^i|=1 \end{array} \xrightarrow{(RF, Z^2)} \begin{array}{c} Think(Z)+F(Z) \\ |Z|=5 \end{array}$$

$$\Big\downarrow (TF, Z^{1,0})$$

$$\begin{array}{c} Think(Z^{1,1}+Z^{1,2}+Z^3)+ \\ F(Z^{1,2})+Eat(Z^{1,0}+Z^2) \\ Think(Z^2+Z^3+Z^5)+F(Z^3)+ \\ Eat(Z^1+Z^4) \\ |Z^i|=1 \end{array}$$

Example

$$Think(Z^1+Z^3)+F(Z^1)+Eat(Z^2)$$
$$|Z^1|=3, |Z^2|=|Z^3|=1$$
$$Think(Z^{1,0}+Z^{1,1}+Z^{1,2}+Z^3)+$$
$$F(Z^{1,0}+Z^{1,1}+Z^{1,2})+Eat(Z^2)$$
$$|Z^i|=1$$

$$\xrightarrow{(RF, Z^2)}$$

$$Think(Z)+F(Z)$$
$$|Z|=5$$

$$\downarrow (TF, Z^{1,0})$$

$$Think(Z^{1,1}+Z^{1,2}+Z^3)+$$
$$F(Z^{1,2})+Eat(Z^{1,0}+Z^2)$$
$$Think(Z^2+Z^3+Z^5)+F(Z^3)+$$
$$Eat(Z^1+Z^4)$$
$$|Z^i|=1$$

$$\searrow (TF, Z^{1,1})$$

$$Think(Z^{1,0}+Z^{1,2}+Z^3)+$$
$$F(Z^{1,0})+Eat(Z^{1,1}+Z^2)$$
$$Think(Z^1+Z^3+Z^5)+F(Z^1)+$$
$$Eat(Z^2+Z^4)$$
$$|Z^i|=1$$

Conclusion

At this stage, we know:

- how to represent, in symbolic and unique way, the marking classes,
- how to fire from a symbolic marking, a symbolic instance, to obtain the symbolic successor.

Conclusion

At this stage, we know:

- how to represent, in symbolic and unique way, the marking classes,
- how to fire from a symbolic marking, a symbolic instance, to obtain the symbolic successor.

We are ready to derive an algorithm to construct the symbolic reachability graph (next sequence).

Symbolic Reachability Graph

Introduction

Now, we know:

- how to represent, in symbolic and unique way, the marking classes,
- how to fire from a symbolic marking, a symbolic instance, to obtain the symbolic successor.

Introduction

Now, we know:

- how to represent, in symbolic and unique way, the marking classes,
- how to fire from a symbolic marking, a symbolic instance, to obtain the symbolic successor.

We are ready to derive an algorithm to construct the symbolic reachability graph.

SRG construction Algorithm

SRG_Construction$(N = \langle P, T, C, W^-, W^+, M_0 \rangle)$

$SRG.Q = \{\hat{M}_0\}$; $SRG.\delta = \emptyset$;

$SRG.q_0 = \hat{M}_0$; $sStates = \{\hat{M}_0\}$;

While (sStates <> $\emptyset$) {

  $\hat{s}$ = pick a state in sStates ;

  sStates = sStates \ $\{\hat{s}\}$;

  for each $t \in T, \hat{c} \in \hat{C}(t)$ {

    if $(\hat{s}[(t, \hat{c})\rangle)$ {

      $\hat{s}[(t, \hat{c})\rangle \hat{ns}$;

      if $(\hat{ns} \notin SRG.Q)$ {

        $SRG.Q = SRG.Q \cup \{\hat{ns}\}$;

        $sStates = sStates \cup \{\hat{ns}\}$;

      }

      $SRG.\delta = SRG.\delta \cup \{(\hat{s}, \hat{ns})\}$;

      $SRG.\lambda(\hat{s}, \hat{ns}) = (t, \hat{c})$;

    }

  }

}

return SRG;

Example: SRG of the critical section access model

$$
\begin{tikzcd}
\substack{\mathit{Idle}(Z) + \mathit{Res} \\ |Z| = 3} \arrow[d, "{(t_1, Z)}"'] & \\
\substack{\mathit{Idle}(Z^1) + \mathit{Waiting}(Z^2) + \mathit{Res} \\ |Z^1| = 2, |Z^2| = 1} \arrow[r, "{(t_2, Z^2)}"] \arrow[d, "{(t_1, Z^1)}"'] & \substack{\mathit{Idle}(Z^1) + \mathit{Busy}(Z^2) \\ |Z^1| = 2, |Z^2| = 1} \arrow[ul, "{(t_3, Z^2)}"'] \\
\substack{\mathit{Idle}(Z^1) + \mathit{Waiting}(Z^2) + \mathit{Res} \\ |Z^1| = 1, |Z^2| = 2} \arrow[r, "{(t_2, Z^2)}"] \arrow[d, "{(t_1, Z^1)}"'] & \substack{\mathit{Idle}(Z^1) + \mathit{Waiting}(Z^2) + \mathit{Busy}(Z^3) \\ |Z^1| = |Z^2| = |Z^3| = 1} \arrow[ul, "{(t_3, Z^2)}"'] \\
\substack{\mathit{Waiting}(Z^2) + \mathit{Res} \\ |Z^2| = 3} \arrow[r, "{(t_2, Z^2)}"] & \substack{\mathit{Waiting}(Z^1) + \mathit{Busy}(Z^2) \\ |Z^1| = 2, |Z^2| = 1} \arrow[ul, "{(t_3, Z^1)}"']
\end{tikzcd}
$$

7 classes instead of
20 markings

Example: SRG of the dining philosophers problem

$$
\begin{array}{c}
Think(Z) + F(Z) \\
|Z| = 5
\end{array}
$$

$$\downarrow (TF, Z) \qquad \uparrow (PF, Z^2)$$

$$
\begin{array}{c}
Think(Z^1 + Z^3) + F(Z^1) + Eat(Z^2) \\
|Z^1| = 3, |Z^2| = |Z^3| = 1
\end{array}
$$

$$\downarrow (TF, Z^{1,0}), \downarrow (TF, Z^{1,1}) \qquad \uparrow (PF, Z^1), \uparrow (PF, Z^4)$$

$$
\begin{array}{c}
Think(Z^2 + Z^3 + Z^5) + F(Z^3) + Eat(Z^1 + Z^4) \\
|Z^i| = 1
\end{array}
$$

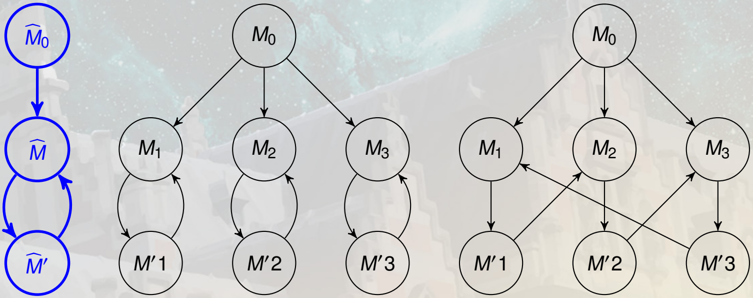
3 symbolic markings instead of 11 markings

What does the Symbolic Reachability Graph preserve?

- Each marking represented by a class (a symbolic marking) is reachable.
- Each reachable marking is represented by a class.
- Each firing sequence of the RG is represented in the SRG.
- To each sequence of the symbolic graph corresponds a sequence of the RG.

Then, what is missing?

- We cannot distinguish between the following situations:



Conclusion

- So far, the approach presented imposes that all objects of the same class behave identically.
  - A class groups a set of objects that have the same nature.
  - The obtained reduction, SRG vs. RG, is maximal.

- How to deal with the case where objects have the same nature, *but with potentially different behaviours*?
  - Example: a class that represents a set of processors divided in two subsets: fast and slow.

Conclusion

- So far, the approach presented imposes that all objects of the same class behave identically.
  - A class groups a set of objects that have the same nature.
  - The obtained reduction, SRG vs. RG, is maximal.
- How to deal with the case where objects have the same nature, *but with potentially different behaviours*?
  - Example: a class that represents a set of processors divided in two subsets: fast and slow.
- Use of static subclasses...
  - Each class is partitioned into cells, called static subclasses, where the objects of the same cell behave identically.
  - Symmetries of nets easily extend as follows... (next sequence)

Static subclasses

Introduction

- So far, the approach presented imposes that all objects of the same class behave identically.
  - A class groups a set of objects that have the same nature.
  - The obtained reduction, SRG vs. RG, is maximal.
- How to deal with the case where objects have the same nature, *but with potentially different behaviours*?
  - Example: a class that represents a set of processors divided in two subsets: fast and slow.
- Use of static subclasses...
  - Each class is partitioned into cells, called static subclasses, where the objects of the same cell behave identically.
  - Symmetries of nets easily extend as follows...

Symmetries, static subclasses and SNs

- Consider a net $N = \langle P, T, C, W^-, W^+, M_0 \rangle$, where,
  - Each class C_i is partitioned into n_i cells.

$$C_i = \bigcup_{j=1}^{n_i} D_{i,j}, \text{ such that } \begin{cases} \forall\ 0 < j \leq n_i, |D_{i,j}| > 0, \\ \forall\ 0 < j' \leq n_i,\ j \neq j' \Rightarrow D_{i,j} \cap D_{i,j'} = \emptyset. \end{cases}$$

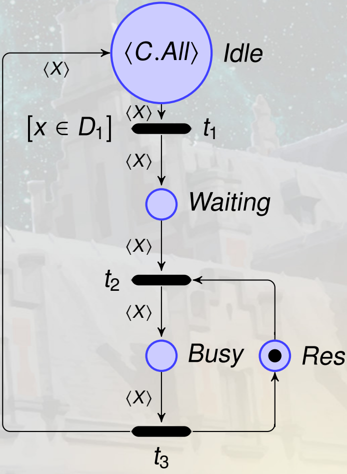
  - $D_{i,j}$ is called a static subclass.

- The symmetries of N are defined by the set $S = \{\langle s_1, \ldots, s_n \rangle \mid s_i \in S_i\}$, where:
  1. With each unordered class C_i, we associate a permutation subgroup S_i,
  2. With each ordered class C_i, we associate a rotation subgroup S_i,
  3. $\forall D_{i,j}, \forall s_i \in S_i : s_i(D_{i,j}) = D_{i,j}$.

- Additional syntax constraints:
  - Broadcast functions are defined w.r.t. subclasses (e.g. $D_{i,j}.All$)
  - Transition Guards are defined w.r.t. subclasses (e.g. $[x \in D_{i,j}]$)

Example of SN with static subclasses

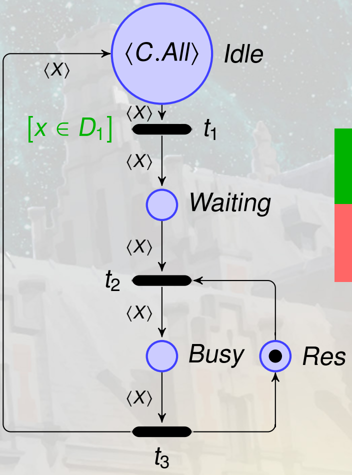
$$C = D_1 \cup D_2 \text{ where}$$
$$D_1 = \{c_1, c_2\},\ D_2 = \{c_3, c_4\}$$



- Colour class C is partitioned into two static subclasses: D_1 and D_2.
- Transition t_1 can be enabled (and fired) only by elements of D_1.

Impact of static subclasses on the SRG (1/2)

$C = D_1 \cup D_2$ where
$D_1 = \{c_1, c_2\}$, $D_2 = \{c_3, c_4\}$



$$Idle(Z) + Res$$
$$|Z| = 4$$
$$\Big\downarrow (t_1, Z)$$
$$Idle(Z^1) + Waiting(Z^2) + Res$$
$$|Z^1| = 3, |Z^2| = 1$$

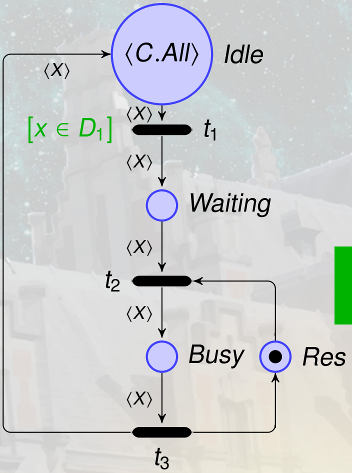
$$Idle(c_1 + c_3 + c_4) + Waiting(c^2) + Res$$
$$Idle(c_2 + c_3 + c_4) + Waiting(c^1) + Res$$

$$Idle(c_1 + c_2 + c_3) + Waiting(c^4) + Res$$
$$Idle(c_1 + c_2 + c_4) + Waiting(c^3) + Res$$

- The symbolic marking defined assumes that all colours of a class are symmetric. So, the instantiation is trivial!
- This is **no more correct** when static subclasses are introduced.

Impact of static subclasses on the SRG (2/2)

$C = D_1 \cup D_2$ where
$D_1 = \{c_1, c_2\}$, $D_2 = \{c_3, c_4\}$



$$Idle(Z^1 + Z^2) + Res$$
$$|Z^1| = 2, |Z^2| = 2$$
$$Z^1 \subseteq D_1,\ Z^2 \subseteq D_2$$

$$\Big\downarrow ((t_1, Z^1)$$

$$Idle(Z^1 + Z^3) + Waiting(Z^2) + Res$$
$$|Z^1| = |Z^2| = 1, |Z^3| = 2$$
$$Z^1, Z^2 \subseteq D_1,\ Z^3 \subseteq D_2$$

$$Idle(c_1 + c_3 + c_4) + Waiting(c^2) + Res$$
$$Idle(c_2 + c_3 + c_4) + Waiting(c^1) + Res$$

- A dynamic subclass must refer to the static subclass to which it belongs (i.e. to which the elements it represents belong).

Conclusion

- Static subclasses are needed to model complex algorithms in a compact way.
- A symbolic marking must refer, in its definition, to these static subclasses, otherwise the underlying represented markings will be spurious!
- The efficiency of the constructed SRG (the reduction factor) depends on these static subclasses:
  - When each class of the net contains only one static subclass, the reduction is maximal.
  - When the classes of the net are partitioned into static subclasses with only one element, there is no reduction.

Conclusion

- Static subclasses are needed to model complex algorithms in a compact way.
- A symbolic marking must refer, in its definition, to these static subclasses, otherwise the underlying represented markings will be spurious!
- The efficiency of the constructed SRG (the reduction factor) depends on these static subclasses:
  - When each class of the net contains only one static subclass, the reduction is maximal.
  - When the classes of the net are partitioned into static subclasses with only one element, there is no reduction.

How to deal with this last case (next sequence).

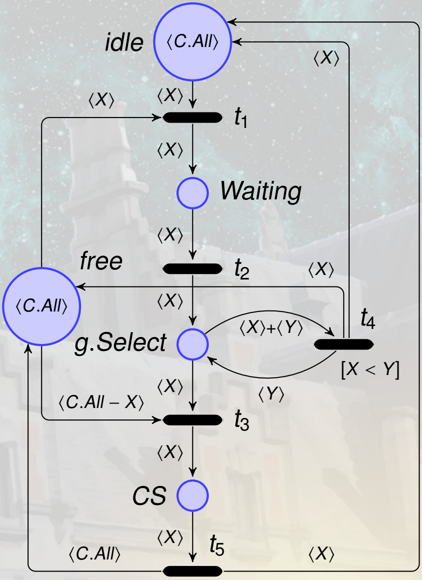
SN and Local Symmetries

Introduction

- Static subclasses are needed to model complex algorithms in a compact way.
- A symbolic marking must refer, in its definition, to the these static subclasses, otherwise the underlying represented markings will be spurious!
- The efficiency of the constructed SRG (the reduction factor) depends on these static subclasses:
  - When each class of the net contains only one static subclass, the reduction is maximal.
  - When the classes of the net are partitioned into static subclasses with only one element, there is no reduction.

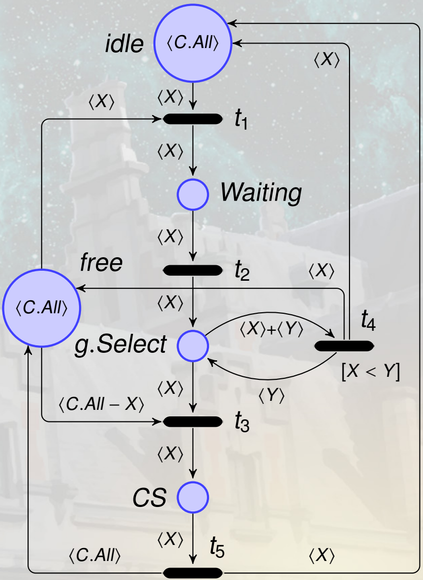
We will now see how to deal with this last case.

Example: critical section with priorities (1/2)



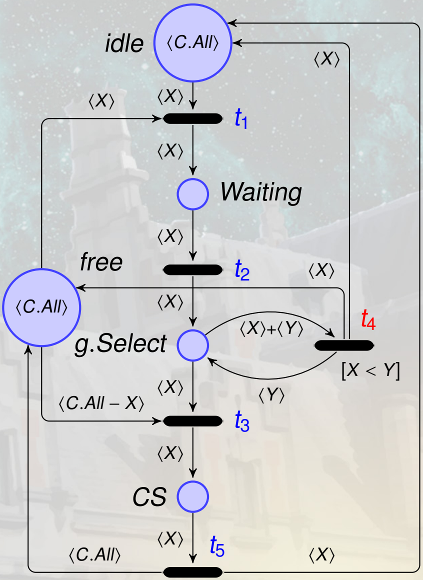
- All places have $C = \{p_1, p_2, p_3\}$ as colour domain.
- Because of the guard $[X < Y]$ on transition t_4, C has to be partitioned into 3 static subclasses: $C = D_1 \cup D_2 \cup D_3$, where $D_i = \{p_i\}$, for $i \in \{1, 2, 3\}$.
- The guard $[X < Y]$ is written: $\bigvee_{i<j} (X \in D_i \wedge Y \in D_j)$

Example: critical section with priorities (2/2)



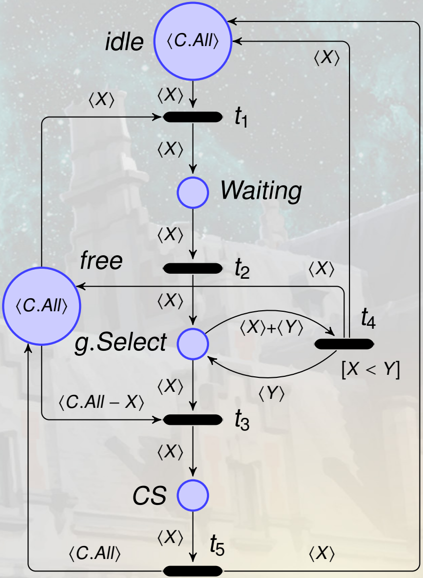
- Since all defined static subclasses are singletons, and
- the symmetries of a SN are defined according to these subclasses (i.e. only objects of the same subclass are symmetrical),
- then, the constructed SRG of this SN has the same size as the RG, i.e. no reduction is possible!
- Is it possible to deal with this problem?

SN and partial symmetries: observation



- The problem (**asymmetry**) comes from a single transition (t_4) and is propagated in the whole net!
- The guard and the firing of t_4 distinguish the objects ⇒ objects are asymmetrical.
- The enabling and the firing of transitions t_1, t_2, t_3 and t_5 do not need information about the identity of the objects ⇒ objects are symmetrical.

SN and partial symmetries: idea



- **Forget** the asymmetries (static subclasses) while not needed to test the enabling of a transition.
- **Reintroduce** the static subclasses while testing the enabling of asymmetric transitions (transitions that refer to static subclasses).
- This way, the propagation of asymmetries will be contained in small parts.

Symmetric Nets with Bags

- SN do not avoid the interleaving inherent to distributed systems (that could be avoided by partial order-based techniques)
- SN do not easily model multiple data association with a single “identifier”

Introduction

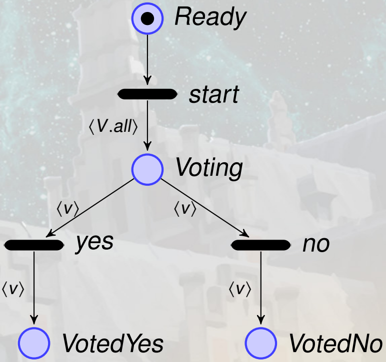
- SN do not avoid the interleaving inherent to distributed systems (that could be avoided by partial order-based techniques)
- SN do not easily model multiple data association with a single “identifier”

SNB (B=bags) bring a solution to these problems

- Suppression of spurious intermediate states
- Possibility to associate items as bags themselves
- Models are even more compact and parametrisable than with SNs

The voting system example (1/2)

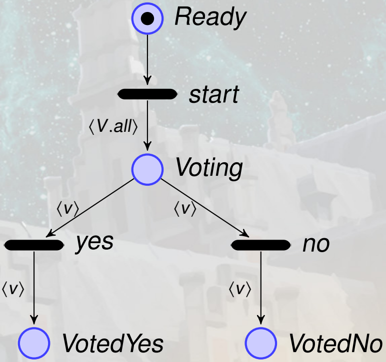
$V = \{v_1, \ldots, v_n\}$
$v \in V$



Voting machine example

The voting system example (1/2)

$V = \{v_1, \ldots, v_n\}$
$v \in V$

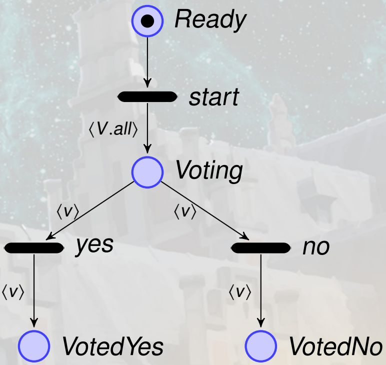


Voting machine example

- Reachability graph shows all possible votes

The voting system example (1/2)

$V = \{v_1, \ldots, v_n\}$
$v \in V$

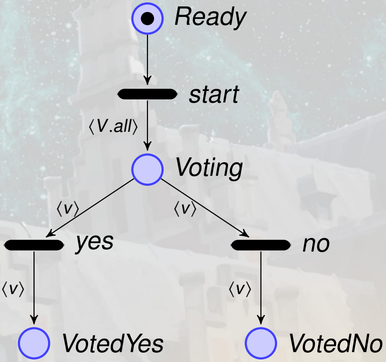


Voting machine example

- Reachability graph shows all possible votes
- High complexity:
  - $3^{|V|} + 1$ states
  - $\binom{|V|+2}{2} + 1$ symbolic states

The voting system example (1/2)

$V = \{v_1, \ldots, v_n\}$
$v \in V$



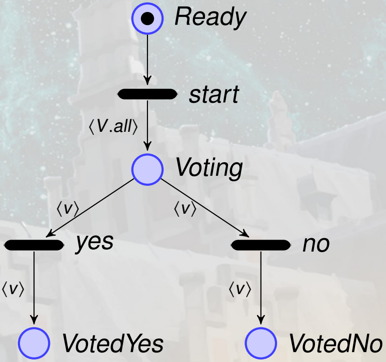
Voting machine example

- Reachability graph shows all possible votes
- High complexity:
  - $3^{|V|} + 1$ states
  - $\binom{|V|+2}{2} + 1$ symbolic states

Incurring problems

The voting system example (1/2)

$V = \{v_1, \dots, v_n\}$
$v \in V$



Voting machine example

- Reachability graph shows all possible votes
- High complexity:
  - $3^{|V|} + 1$ states
  - $\binom{|V|+2}{2} + 1$ symbolic states

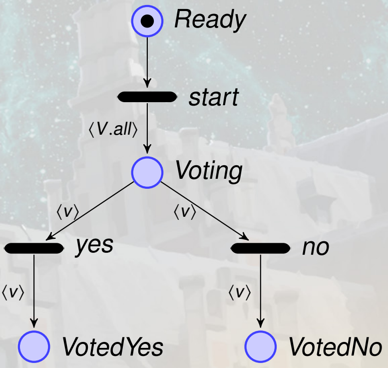
Incurring problems

- $2^{|V|}$ possible vote results

The voting system example (1/2)

$V = \{v_1, \ldots, v_n\}$

$v \in V$



Voting machine example

- Reachability graph shows all possible votes
- High complexity:
  - $3^{|V|} + 1$ states
  - $\binom{|V|+2}{2} + 1$ symbolic states

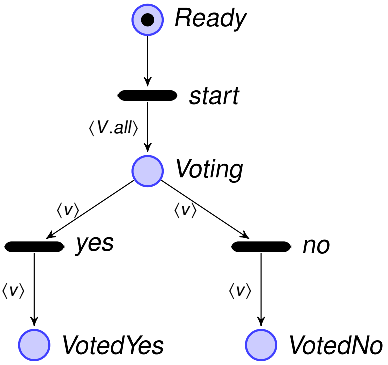
Incurring problems

- $2^{|V|}$ possible vote results
- no symbolic firing to produce all possible votes:
  - Vote categories cannot be computed symbolically
  - Limit of Symmetric Nets

The voting system example (2/2)

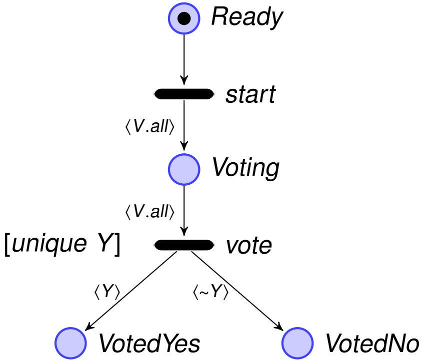
Symmetric Net Model

$V = \{v_1, \ldots, v_n\}$
$v \in V$



Symmetric Net with Bags Model

$V = \{v_1, \ldots, v_n\}$
$Y \in Bag(V)$



Conclusion

At this stage:

- you have seen a basic illustration of SNBs
- you know that SNBs capture bags of values

Conclusion

At this stage:

- you have seen a basic illustration of SNBs
- you know that SNBs capture bags of values

Let's present the functions manipulated in SNBs and the firing rule (next sequence)

Functions used in SNBs and firing rule

Introduction

Now you know:

- the basic underlying features of SNBs
- that SNBs capture bags of values

Introduction

Now you know:

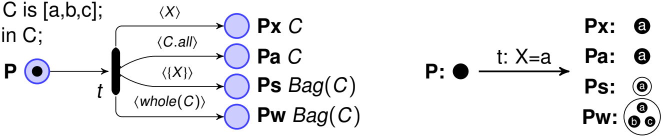
- the basic underlying features of SNBs
- that SNBs capture bags of values

Let's present the functions manipulated in SNBs and the firing rule

Functions and their use in firings

Basic functions (colours and bags)

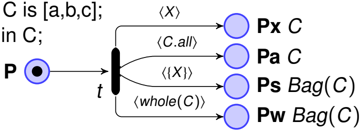
Class C is [a,b,c];
Var X in C;

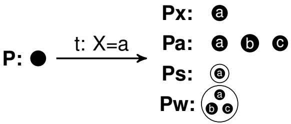


Functions and their use in firings

Basic functions (colours and bags)

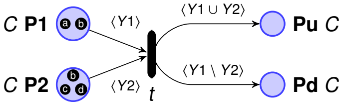
Class C is [a,b,c];
Var X in C;

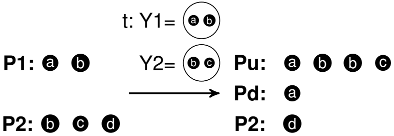




Bag manipulations

Class C is [a,b,c,d];
Var Y1,Y2 in Bag(C);





Bags functions used in guards

ord(x)	: the rank of element x in an ordered set
Unique Y	: true iff elements appear at most once in Y
card(Y)	: the cardinatlity of bag Y

Conclusion

At this stage:

- you know the functions that operate on bags
- you know the additional functions used in guards

Conclusion

At this stage:

- you know the functions that operate on bags
- you know the additional functions used in guards

Let's present a more complete example (next sequence)

Second example of SNB

Introduction

Now you know:

- the functions that operate on bags
- the additional functions used in guards

Introduction

Now you know:

- the functions that operate on bags
- the additional functions used in guards

Let's present a more complete example

The global allocation mechanisms

A way to avoid deadlocks in systems
Make one of the four necessary conditions fail

Principle
When a program enters a *critical section*, it must
own all the resources it will need in this piece of code

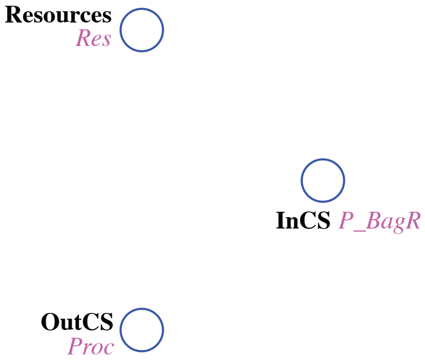
Modeling the problem (1/6)

Entering in the critical section

```
Class
   Proc is [p1, p2, p3];
   Res is 1..6;
Domain
   BagR is Bag(Res);
   P_BagR is <Proc, BagR>;
Var
   p in Proc;
   R, R2 in BagR;
```

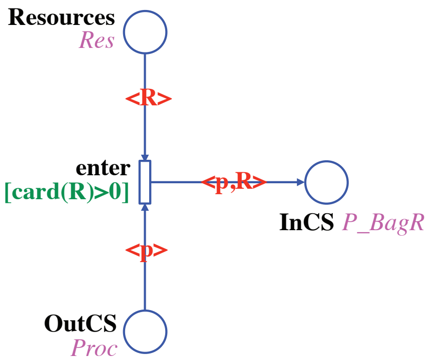
Modeling the problem (2/6)

States of the system



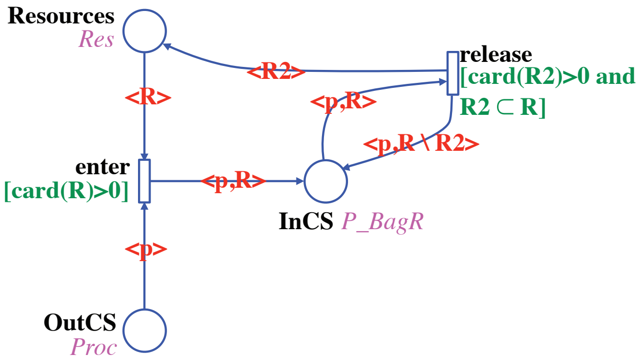
Modeling the problem (3/6)

Entering in the critical section

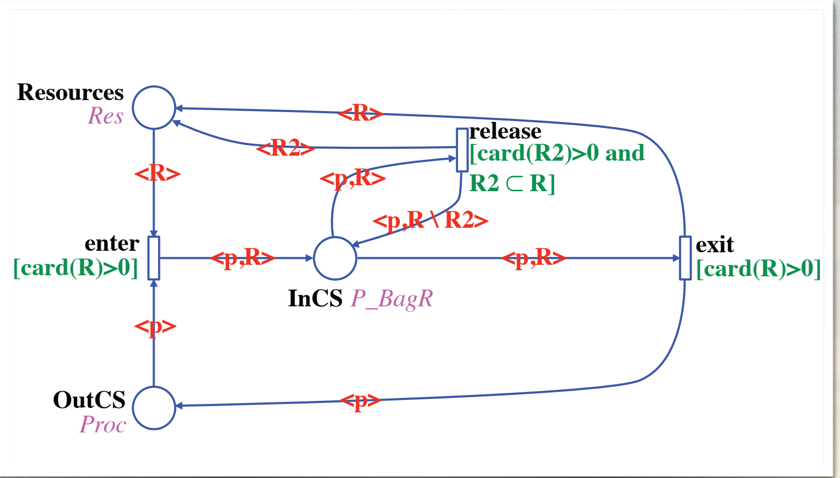


Modeling the problem (4/6)

Releasing some resources (and staying in the critical section)

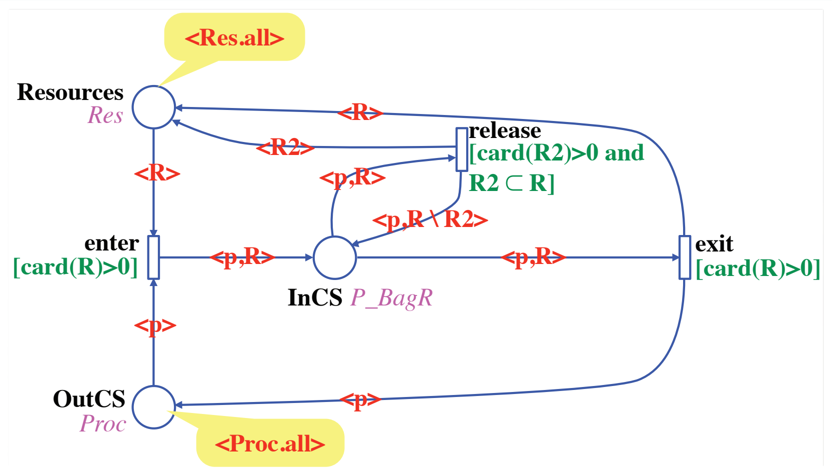


Exiting the critical section



Modeling the problem (6/6)

Initial marking of the system



Conclusion

This tutorial has presented:

- Symmetric Nets with their syntax and semantics
  - how to build the Reachability Graph
  - how to use them for system analysis
- How to use the CosyVerif platform to practice these concepts and formalisms
- The use of global symmetries to reduce the Reachability Graph
  - dynamic and static subclasses
  - the symbolic firing rule
  - the Symbolic Reachability Graph
  - The notion of partial symmetries (the idea of it)
- Symmetric Nets with Bags (SNB)

Conclusion

This tutorial has presented:

- Symmetric Nets with their syntax and semantics
  - how to build the Reachability Graph
  - how to use them for system analysis
- How to use the CosyVerif platform to practice these concepts and formalisms
- The use of global symmetries to reduce the Reachability Graph
  - dynamic and static subclasses
  - the symbolic firing rule
  - the Symbolic Reachability Graph
  - The notion of partial symmetries (the idea of it)
- Symmetric Nets with Bags (SNB)

and next, back to practice (how to model a system with SNB)