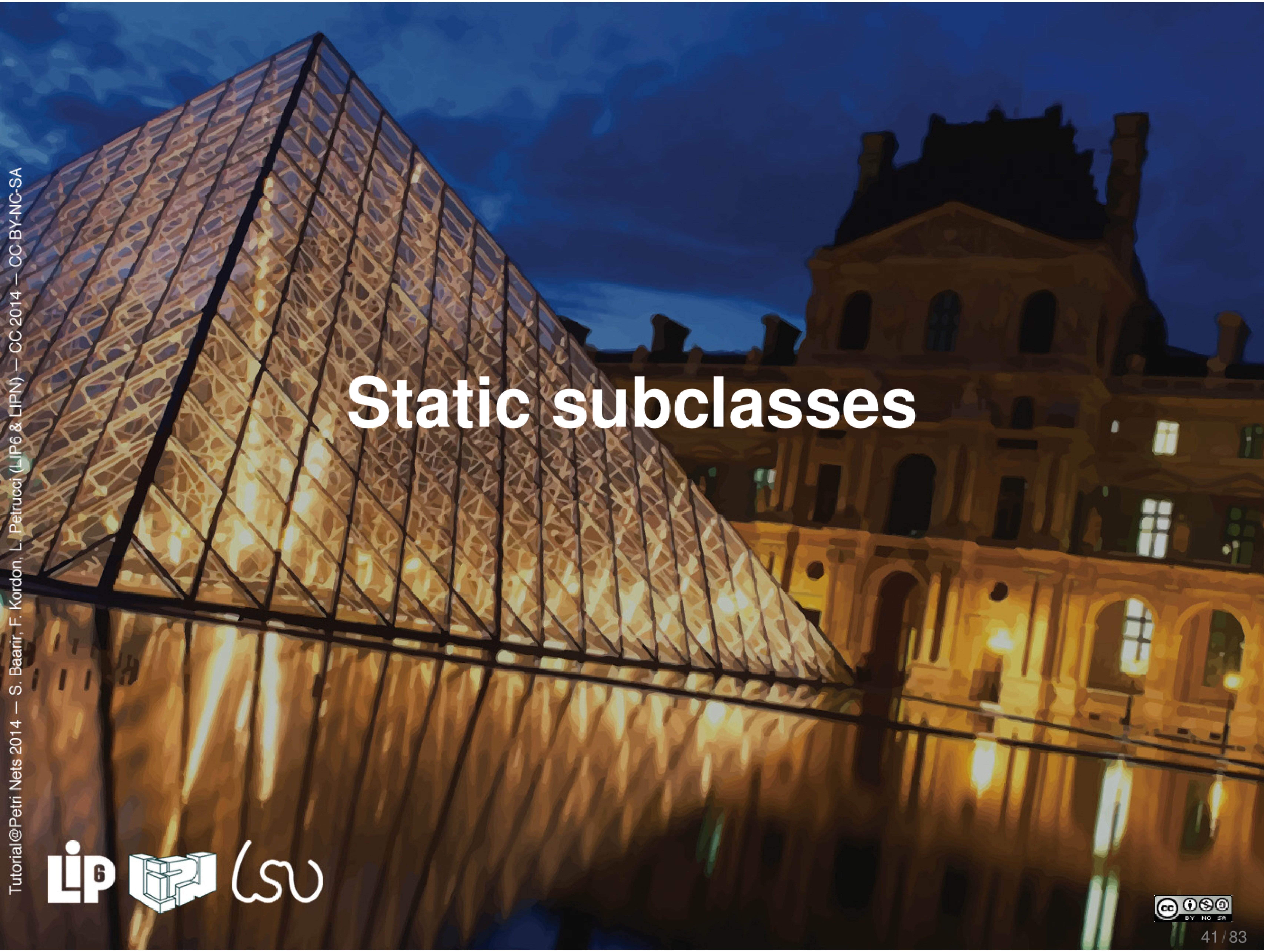


Static subclasses

- 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 net extends easily 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.

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

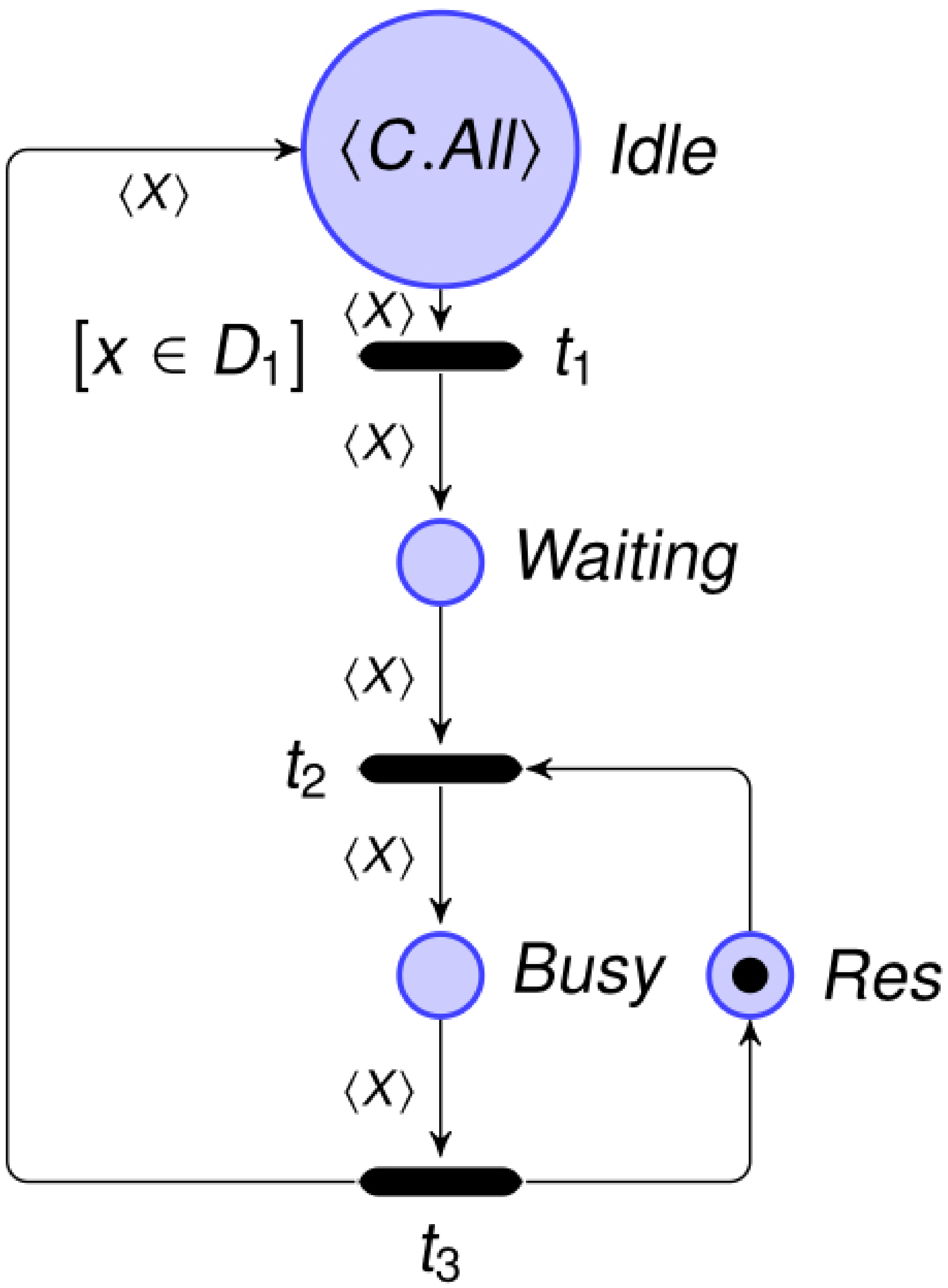
  - $D_{i,j}$ is called a static subclass.

- 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

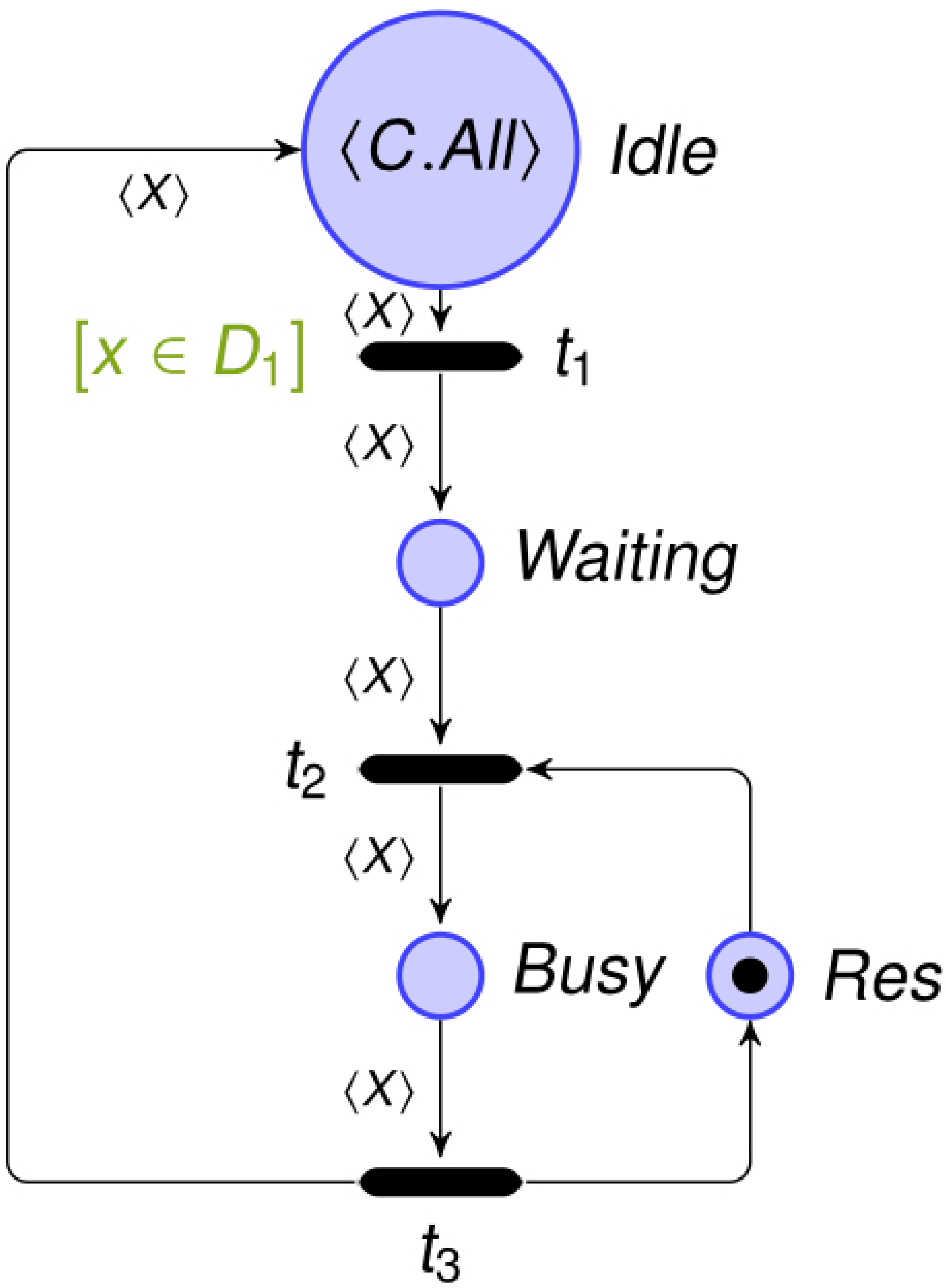
$C = D_1 \cup D_2$ where
$D_1 = \{c_1, c_2\}$, $D_2 = \{c_3, c_4\}$



- Colour class C is partitioned in 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

t_1(Z) ↓

Idle(Z^1) + Wait(Z^2) + Res
| Z^1 | = 3, | Z^2 | = 1

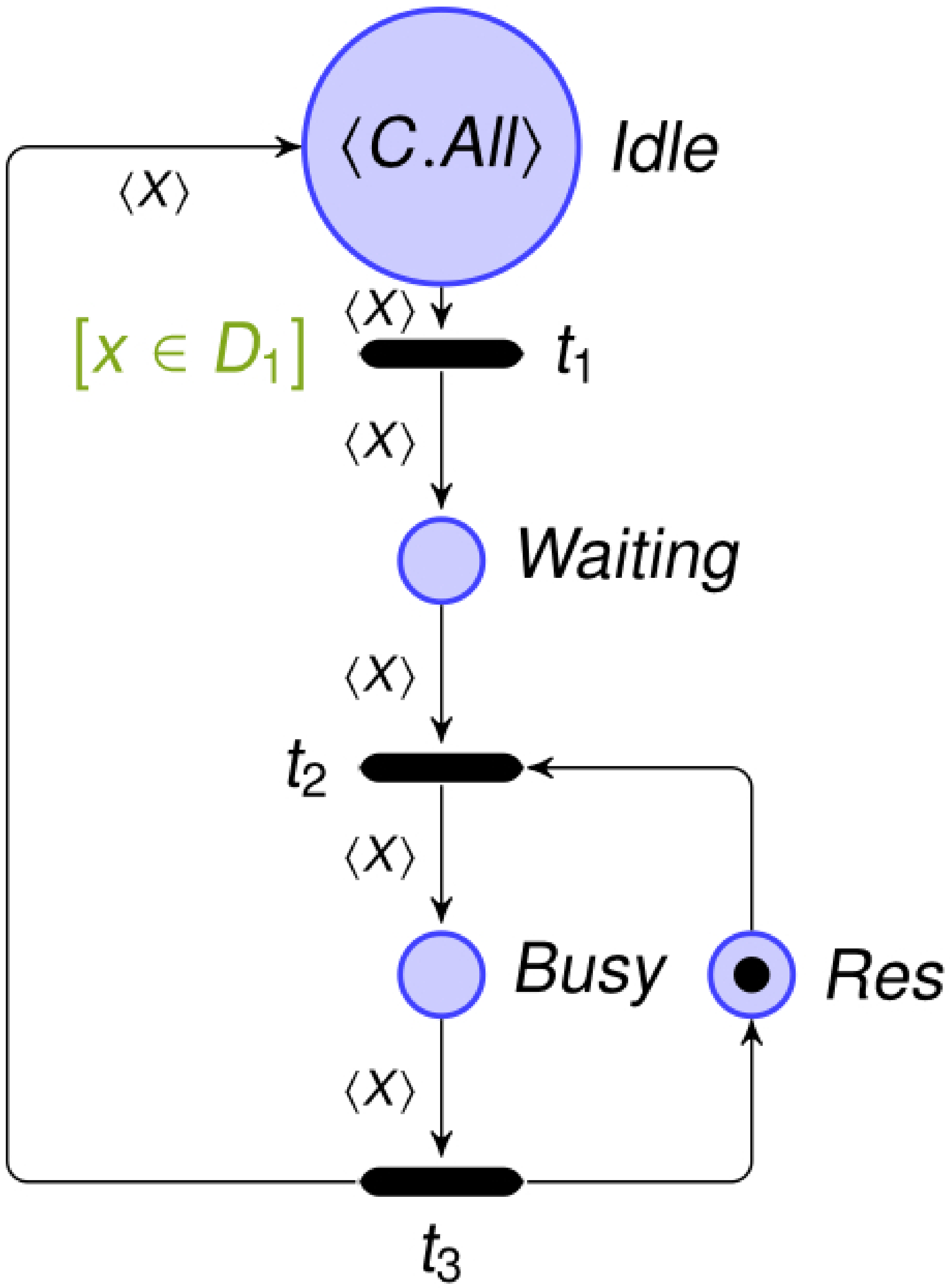
Idle(c_1 + c_3 + c_4) + Wait(c_2) + Res
Idle(c_2 + c_3 + c_4) + Wait(c_1) + Res
Idle(c_1 + c_2 + c_3) + Wait(c_4) + Res
Idle(c_1 + c_2 + c_4) + Wait(c_3) + Res

- The defined symbolic marking supposes 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

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



$Idle(Z^1 + Z^2) + Res$

$|Z^1| = 2\ ,\ \ |Z^2| = 2$

$Z^1 \subseteq D_1\ ,\ Z^2 \subseteq D_2$

↓ **$t_1(Z^1)$**

$Idle(Z^1 + Z^3) + Wait(Z^2) + Res$

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

$Z^1, Z^2 \subseteq D_1\ ,\ Z^3 \subseteq D_2$

$Idle(c_1 + c_3 + c_4) + Wait(c_2) + Res$

$Idle(c_2 + c_3 + c_4) + Wait(c_1) + Res$

- 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).