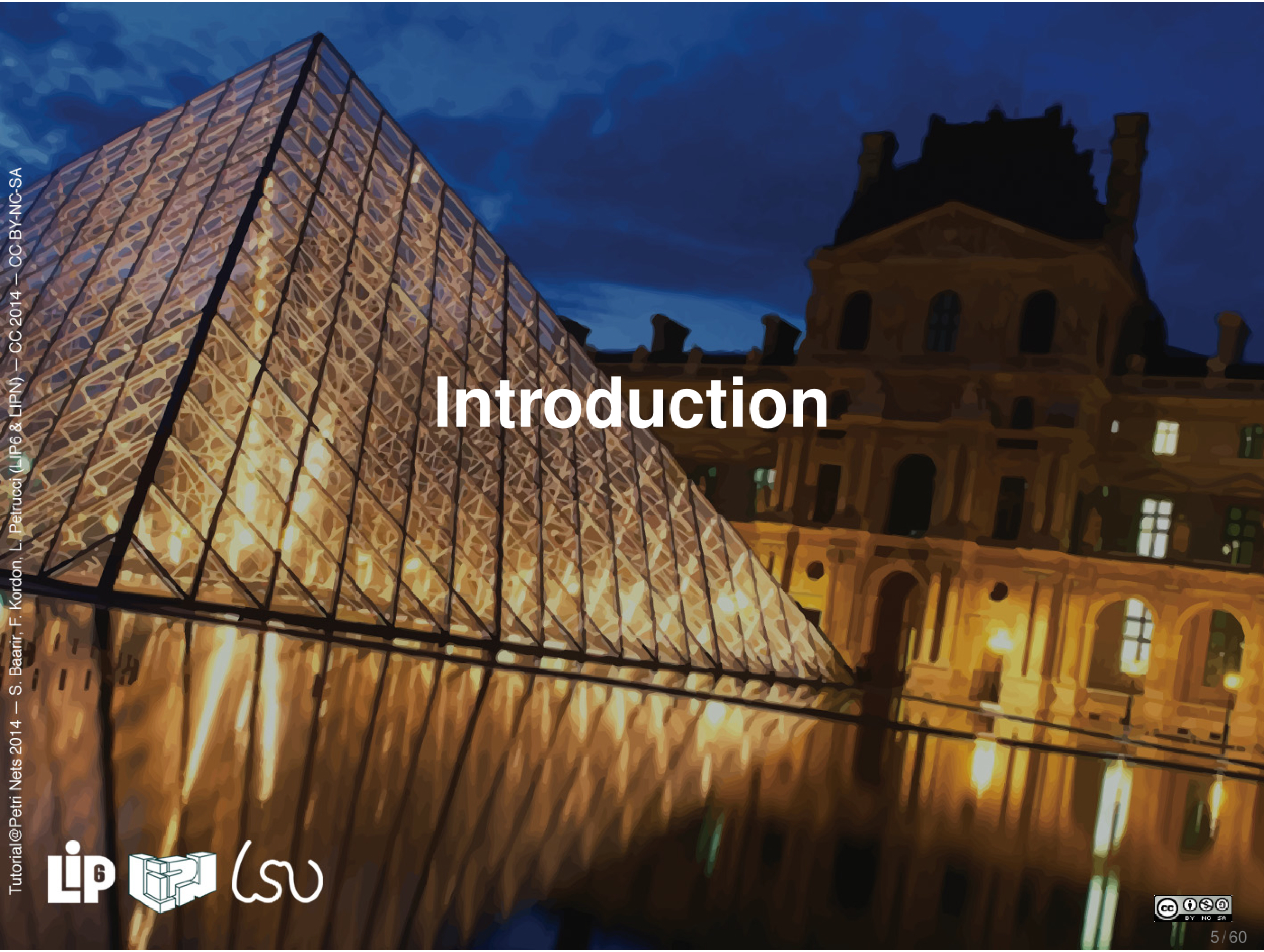
Introduction



First of all...

You know about Place/Transition Petri nets:

- about their structure
- about their marking
- about their reachability graph

First of all...

You know about Place/Transition Petri nets:

- about their structure
- about their marking
- about their reachability graph

Nice notation to model and analyse distributed systems...

...but ...

...complex to handle

- need for parametrisation
- need to represent data
- need for a compact and easy-to-read notation

First of all...

You know about Place/Transition Petri nets:

- about their structure
- about their marking
- about their reachability graph

Nice notation to model and analyse distributed systems. . .

. . . but . . .

. . . complex to handle

- need for parametrisation
- need to represent data
- need for a compact and easy-to-read notation

Let’s have a deeper look on this now

Petri net classes.[1]

- Level 1: PNs characterised by places which can **represent boolean values**,
  - i.e. a place is marked by at most *one unstructured token*,
  - Condition/Event (C/E) Systems, Elementary Net (EN) Systems, 1-safe Systems.

[1] Made by *Monika Trompedeller* in 1995 (based on a survey by *L. Bernardinello and F. De Cindio* from 1992)

Petri net classes.[1]

- Level 1: PNs characterised by places which can **represent boolean values**,
  - i.e. a place is marked by at most *one unstructured token*,
  - Condition/Event (C/E) Systems, Elementary Net (EN) Systems, 1-safe Systems.
- Level 2: PNs characterised by places which can **represent integer values**,
  - i.e. a place is marked by *a number of unstructured tokens*,
  - Place/Transition (P/T) Nets,. . .

[1] Made by *Monika Trompedeller* in 1995 (based on a survey by *L. Bernardinello and F. De Cindio* from 1992)

Petri net classes.[1]

- Level 1: PNs characterised by places which can **represent boolean values**,
  - i.e. a place is marked by at most *one unstructured token*,
  - Condition/Event (C/E) Systems, Elementary Net (EN) Systems, 1-safe Systems.
- Level 2: PNs characterised by places which can **represent integer values**,
  - i.e. a place is marked by *a number of unstructured tokens*,
  - Place/Transition (P/T) Nets,. . .
- Level 3: PNs characterised by places which can **represent high-level values**,
  - i.e. a place is marked by *a multiset of structured tokens*.
  - Coloured Petri Nets, Algebraic Petri Nets, Symmetric Nets (a.k.a Well-Formed Nets),. . .

[1] Made by *Monika Trompedeller* in 1995 (based on a survey by *L. Bernardinello and F. De Cindio* from 1992)

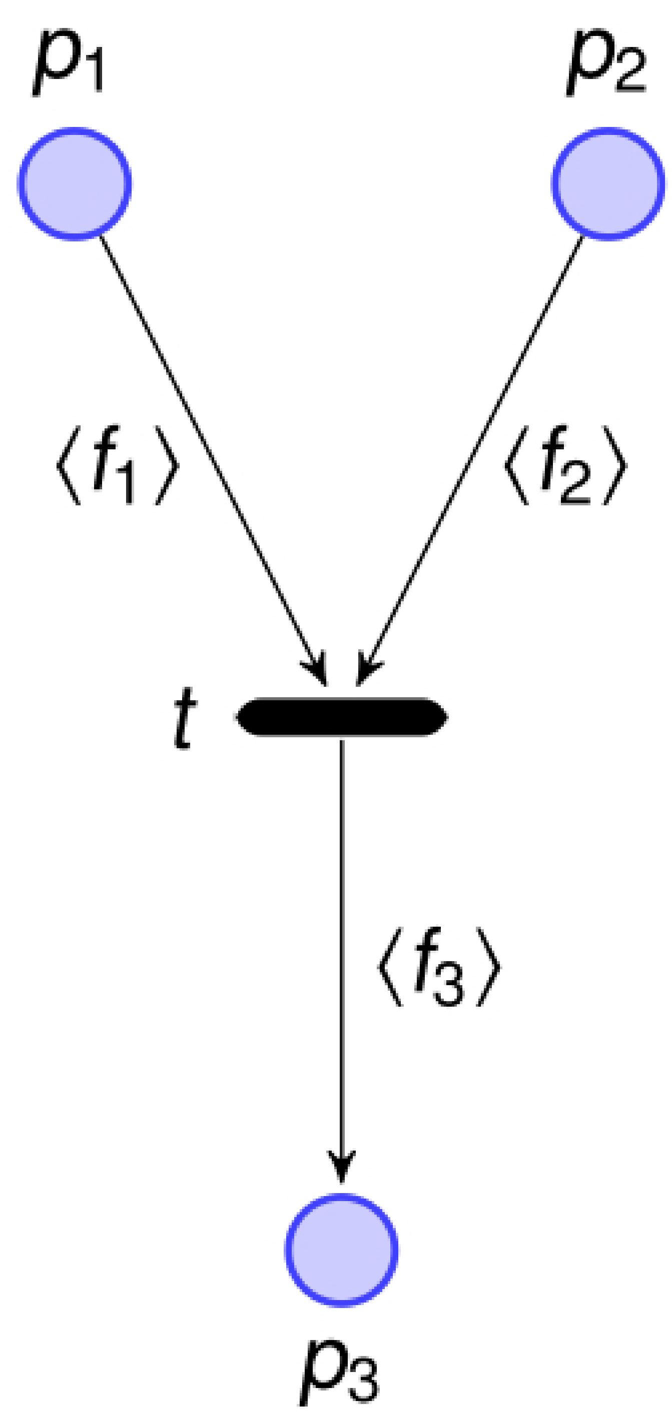
Petri net classes.[1]

- Level 1: PNs characterised by places which can **represent boolean values**,
  - i.e. a place is marked by at most *one unstructured token*,
  - Condition/Event (C/E) Systems, Elementary Net (EN) Systems, 1-safe Systems.
- Level 2: PNs characterised by places which can **represent integer values**,
  - i.e. a place is marked by *a number of unstructured tokens*,
  - Place/Transition (P/T) Nets,. . .
- Level 3: PNs characterised by places which can **represent high-level values**,
  - i.e. a place is marked by *a multiset of structured tokens*.
  - Coloured Petri Nets, Algebraic Petri Nets, Symmetric Nets (a.k.a Well-Formed Nets),. . .

[1] Made by *Monika Trompedeller* in 1995 (based on a survey by *L. Bernardinello and F. De Cindio* from 1992)

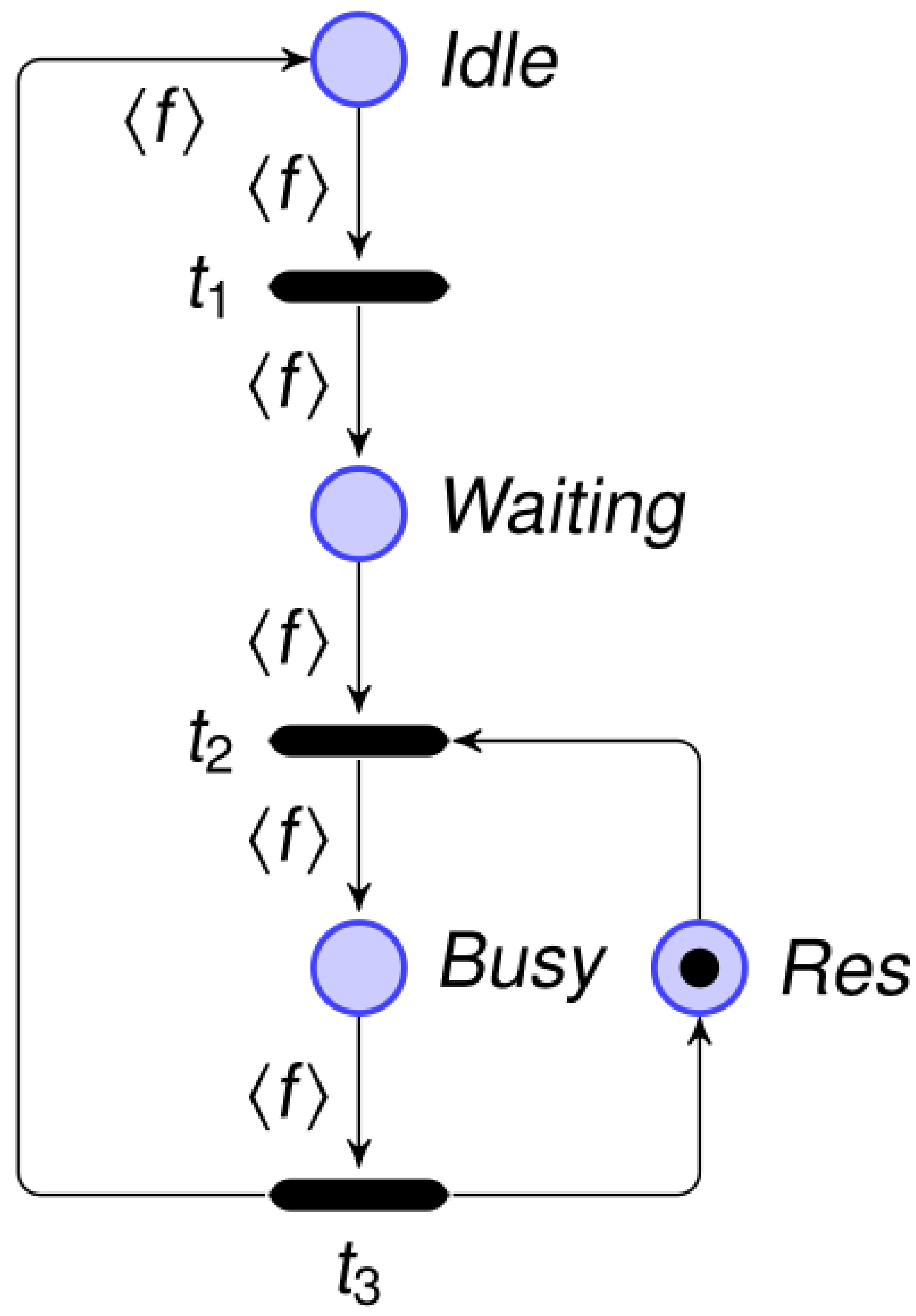
Symmetric Nets (SN): an informal definition

- Each place p is characterised by a colour domain C(p).
- A token of p is an element of C(p).
- Each transition t is characterised by a colour domain C(t).
- The colour domain of a transition characterises the signature of the transition.
- The colour functions on arcs determine the instances of tokens that are consumed and produced during the firing of a transition.



Symmetric Nets (SN): an example

- Processes of class $Cl = \{p_1, ..., p_n\}$, in mutual exclusion on a untyped resource.
- A process is either in an Idle state, or in a Waiting state, or in a Busy state.
- To move from the Waiting state to the Busy state, a process needs the resource.



C(Idle) = C(Waiting) = C(Busy) = Cl
C(Res) = $\{\epsilon\}$
C(t_1) = C(t_2) = C(t_3) = Cl

$f : Cl \rightarrow Cl$
$M_0(Idle) = Cl.All$

Symmetric Nets (SN): another example

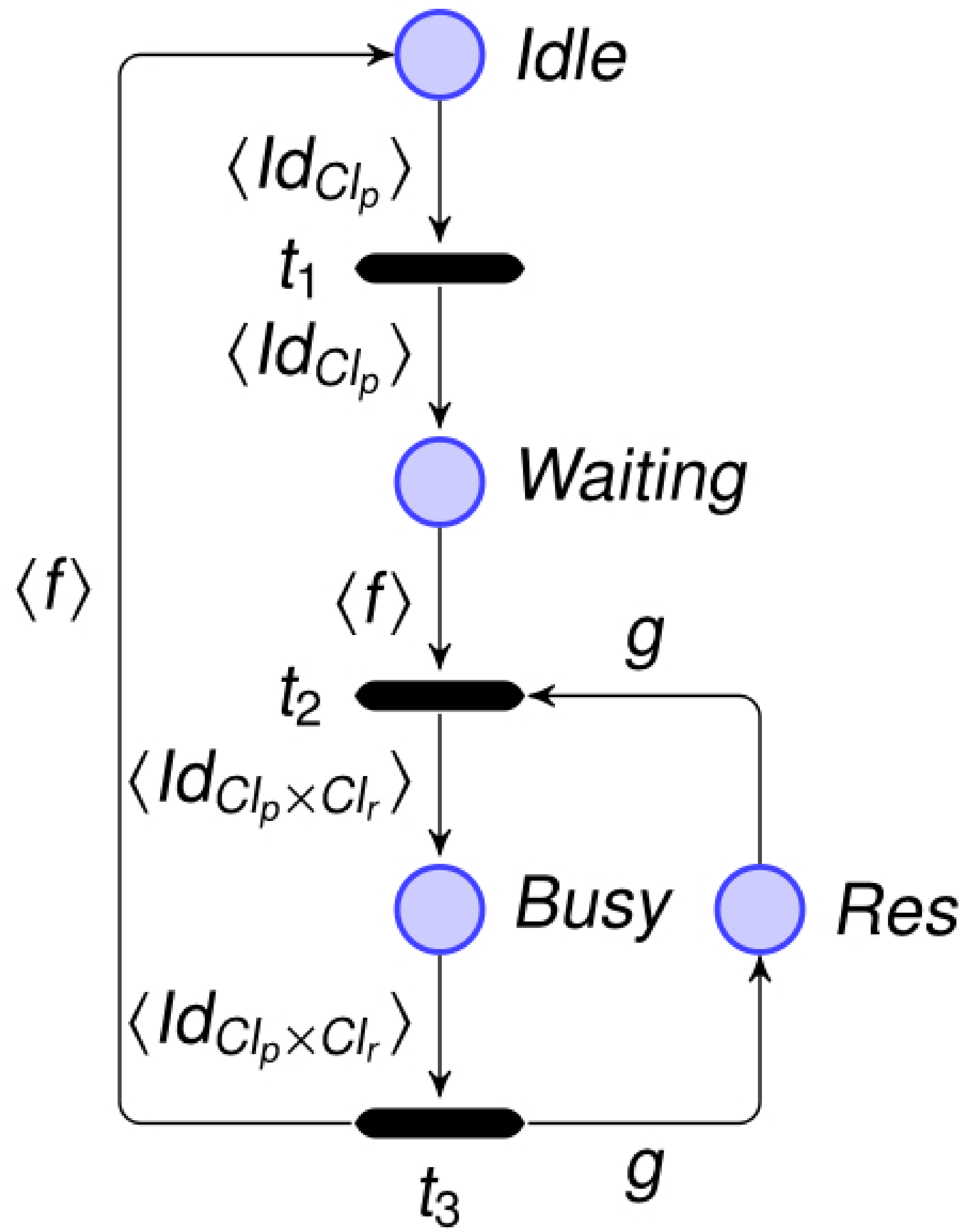
- n_1 processes of class $Cl_p = \{p_1, ..., p_{n_1}\}$, in mutual exclusion on n_2 resources of class $Cl_r = \{r_1, ..., r_{n_2}\}$.
- To move from Waiting to Busy, a process p_i needs a resource r_j.

C(Idle) = C(Waiting) = Cl_p
C(Res) = Cl_r
C(Busy) = $Cl_p \times Cl_r$

C(t_1) = Cl_p
C(t_2) = C(t_3) = $Cl_p \times Cl_r$

$f : Cl_p \times Cl_r \rightarrow Cl_p$
$g : Cl_p \times Cl_r \rightarrow Cl_r$

$M_0(Idle) = Cl_p.All$; $M_0(Res) = Cl_r.All$



Conclusion

At this stage:

- you have an idea on Coloured Nets in general . . .
- . . . and Symmetric Nets in particular

Conclusion

At this stage:

- you have an idea on Coloured Nets in general . . .
- . . . and Symmetric Nets in particular

Let's go for a more precise semantics (next sequence)