

Towards a Methodology for Modeling with Petri Nets

Christine Choppy

Laure Petrucci

LIPN, CNRS UMR 7030
Université Paris 13
Villetaneuse
FRANCE



Motivation

- complex systems to model
- writing formal specifications is difficult



Motivation

- complex systems to model
- writing formal specifications is difficult

⇒ **help** and **guidelines** required



Motivation

- complex systems to model
- writing formal specifications is difficult

⇒ **help** and **guidelines** required

Method proposed in [CR03]:

- for **software development**
- using **algebraic specifications** in CASL
- extended to **dynamic systems** in CASL-LTL



Motivation

- complex systems to model
- writing formal specifications is difficult

⇒ **help** and **guidelines** required

Method proposed in [CR03]:

- for **software development**
- using **algebraic specifications** in CASL
- extended to **dynamic systems** in CASL-LTL

➡ **Develop a similar method for Petri nets**

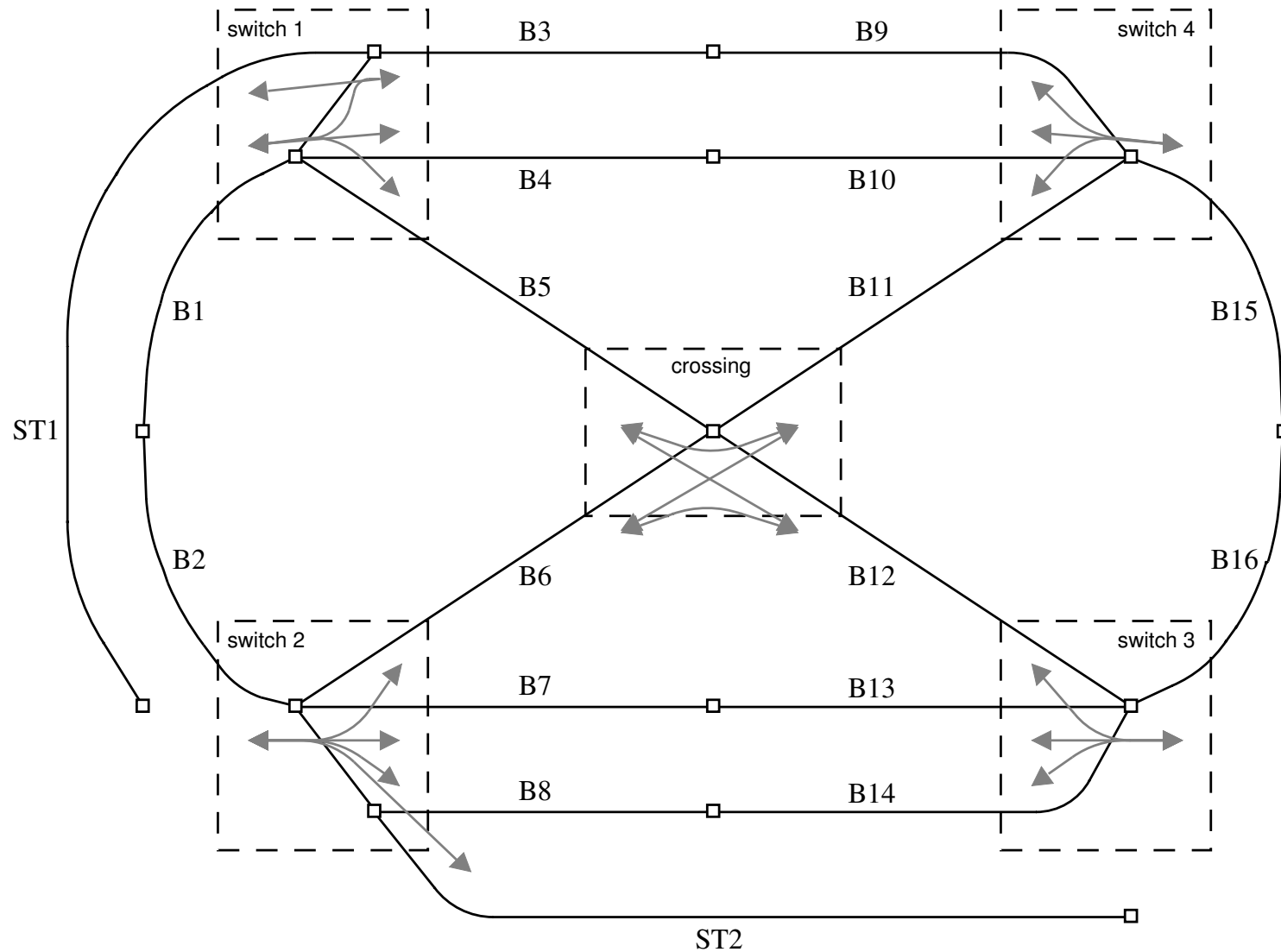


Outline

- Running example: model train system from [BP01]
- Specification method principles
 - systems specifications
 - systems properties
- Applying the specification principles to the train example
 - parts and constituent features
 - system properties
- Derive a coloured Petri net
- Conclusion and future work



Train example



Specification method principles

Design a **software item** :

- simple dynamic system
- structured dynamic system
- data structure



Specification method principles

Design a **software item** :

- simple dynamic system
- structured dynamic system
- data structure

A **simple system item** is composed of:

- parts: data structures
- constituent features:
 - states description
 - elementary interactions definitions



Property-oriented specifications

Provide:

- **abstract structure** of the specification
- **visual** presentation
- **formal** specification
- **properties** expected at this stage

The model must satisfy all properties defined.



Simple systems

No internal components cooperating together



Simple systems

No internal components cooperating together

Labeled Transition System where:

- **states**: relevant intermediate situations in the life of the system
- **transitions**: ability of the system to **evolve** from one state to another
- **transition labels**: elementary interactions



Constituent features and parts

Constituent features :

- elementary interactions types:
 - name
 - arguments: elements of data structures
- states: described by states observers
 - name
 - arguments: elements of data structures
 - observed value



Constituent features and parts

Constituent features :

- elementary interactions types:
 - name
 - arguments: elements of data structures
- states: described by states observers
 - name
 - arguments: elements of data structures
 - observed value

Parts : values of data structures



Simple systems properties

- **labels properties**: incompatibility between elementary interactions
- **state properties**:
 - conditions satisfied by the values returned by state observers
 - properties of paths from/to the state
- **transition properties**: conditions on the state observers applied to the source and target states.



Train specification

Physical system \Rightarrow **state observers**: information on the **layout**

- contiguous track sections
- tracks connected via switches
- train presence
- train direction



Train specification

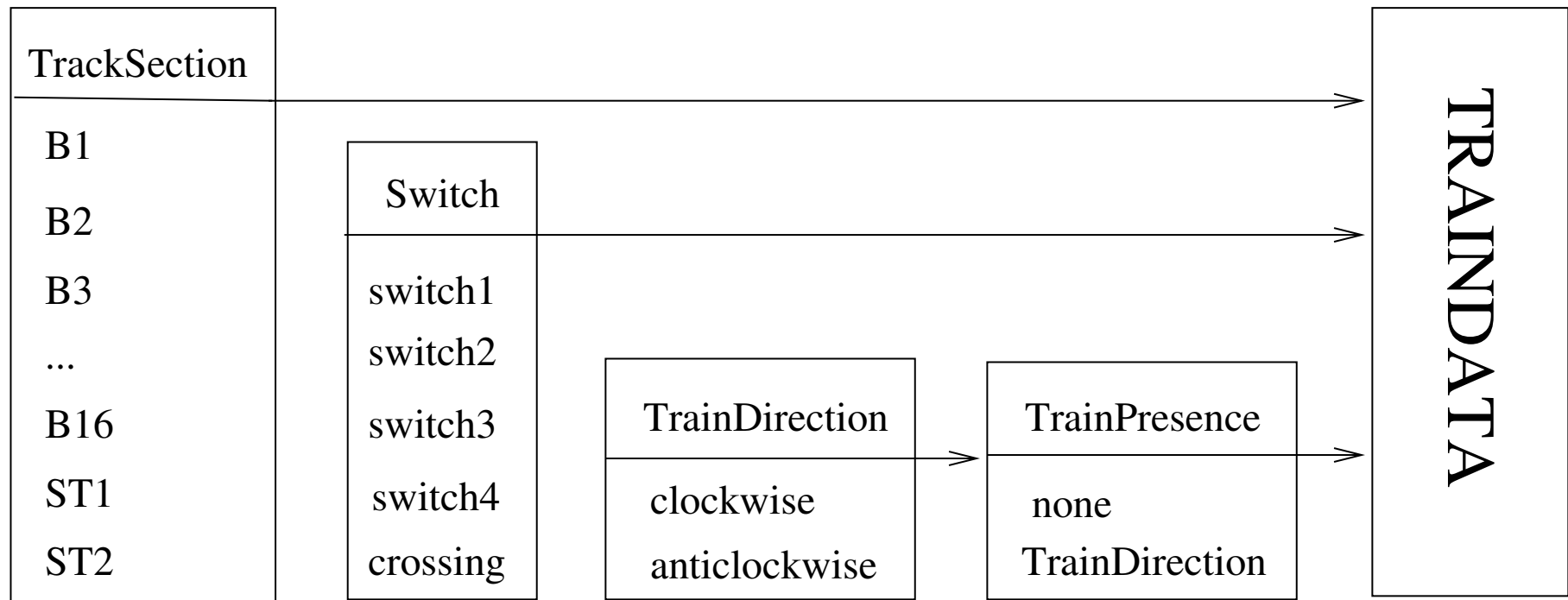
Physical system \Rightarrow **state observers**: information on the **layout**

- contiguous track sections
- tracks connected via switches
- train presence
- train direction

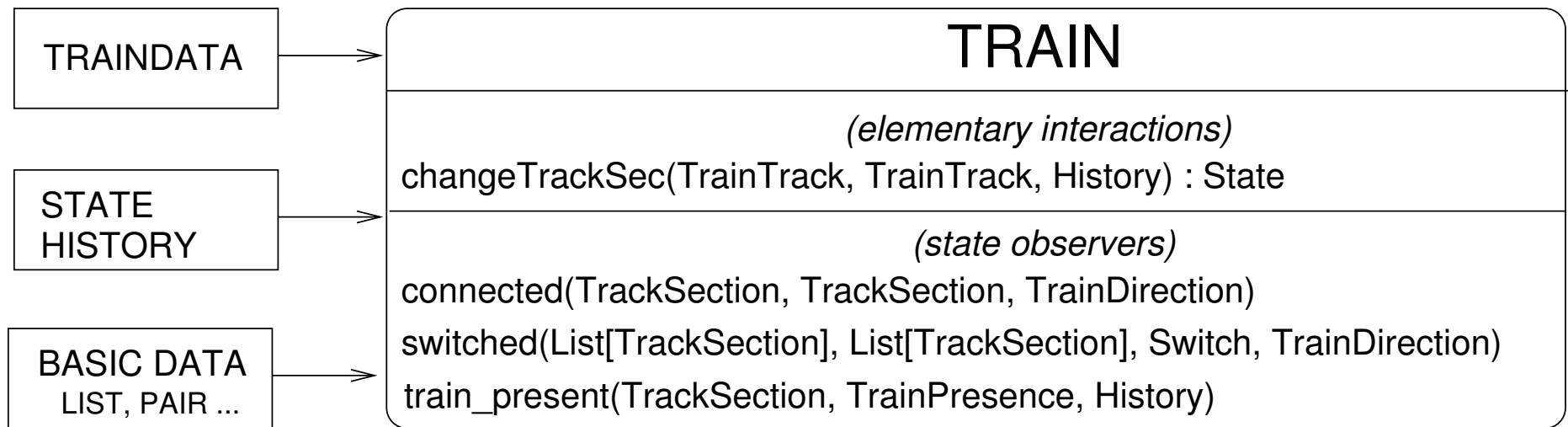
Moves \Rightarrow **elementary interactions**



Data structures



Constituent features



where TrainTrack is an auxiliary type defined as `Pair[TrainPresence, TrackSection]`



STATE & HISTORY

```
spec STATE =  
  sort  State;  
  op    initial : State; %% There is an initial state  
end
```



STATE & HISTORY

spec STATE =

sort *State*;

op *initial* : *State*; %% There is an initial state

end

spec HISTORY = STATE **then**

type *History* ::= *initial* | . (*History*; *State*);

op *last* : *History* → *State*;

vars *h* : *History*; *s* : *State*;

axioms

$last(initial) = initial;$

$last(h.s) = s;$

end



State properties

connected(B1 ,B2,anticlockwise)
connected(B2,B1 ,clockwise)

switched((ST1 ,B1),(B3),switch1 ,clockwise)
switched((B1),(B3,B4,B5),switch1 ,clockwise)
switched((B3),(ST1 ,B1),switch1 ,anticlockwise)
switched((B3,B4,B5),(B1),switch1 ,anticlockwise)

train_present(B1 ,none,initial)



changeTrackSec properties

pre-conditions }
post-conditions } as usual

incompatibility:

simultaneous moves from a same train cannot occur



Associated Petri net

state observers \Rightarrow places, initial marking

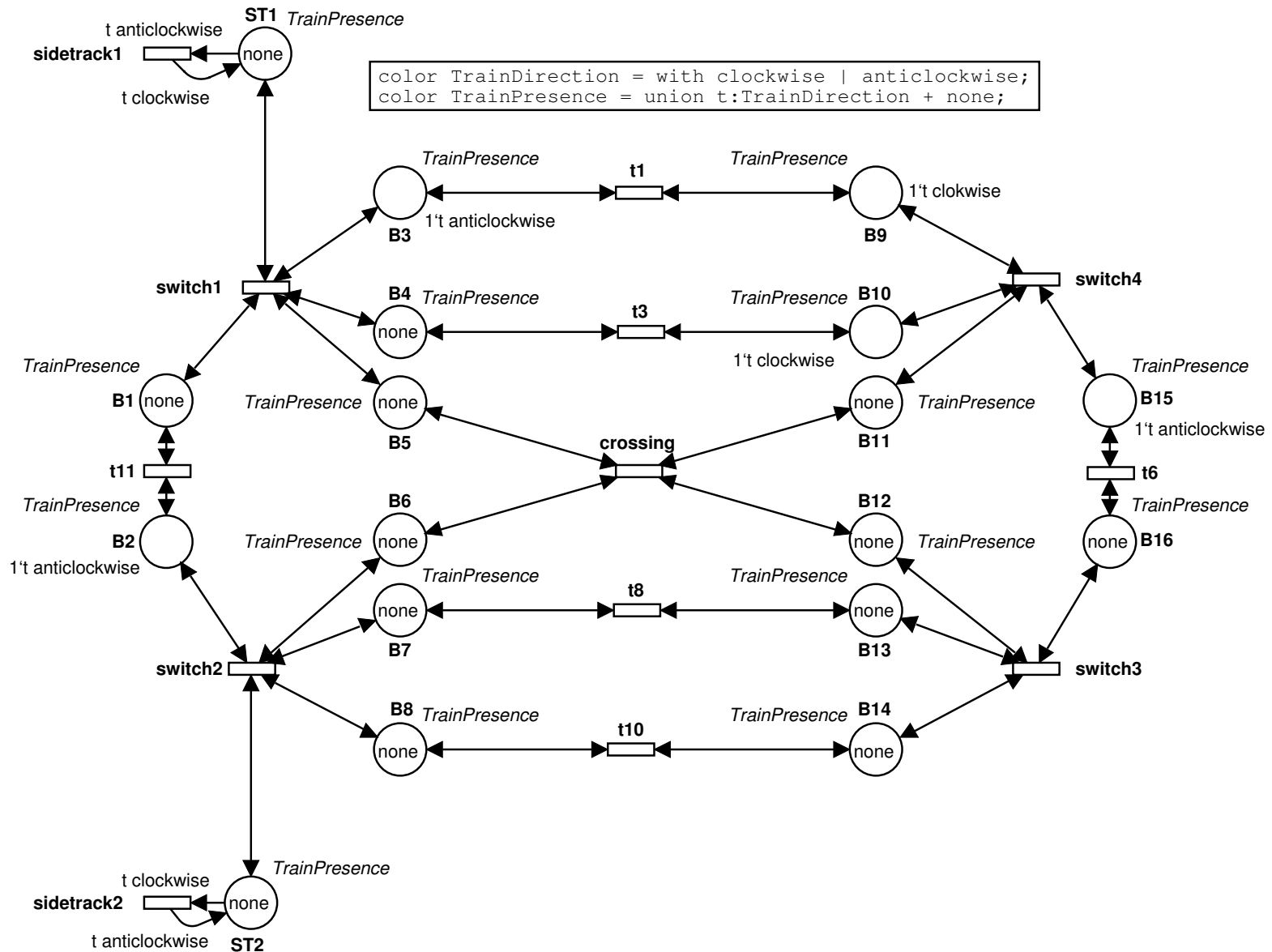
elementary interactions \Rightarrow transitions

train presence \Rightarrow place colours

pre-, post-conditions \Rightarrow arcs



Coloured Petri Net



Conclusion

- provide **guidelines** to specify simple systems using CPNs
 - simple systems **subparts**: **data structures** → e.g. **colours**
 - **state description features** → **layout, place colours**
 - **elementary interactions** → **transitions firing**
- specify **expected properties**
- [CR03] method applies to other formal languages



Conclusion & Future work

- provide guidelines to specify simple systems using CPNs
 - simple systems subparts: data structures → e.g. colours
 - state description features → layout, place colours
 - elementary interactions → transitions firing
- specify expected properties
- [CR03] method applies to other formal languages

- extend the approach to **structured systems** including **communication mechanisms** between modules
- **properties verification**
- complex **case study**



References

- [BP01] G. Berthelot and L. Petrucci. Specification and validation of a concurrent system: An educational project. *Journal of Software Tools for Technology Transfer*, 3(4):372–381, 2001.
- [CR03] C. Choppy and G. Reggio. Towards a Formally Grounded Software Development Method. Technical Report DISI–TR–03–35, DISI, Università di Genova, Italy, 2003. Available at <ftp://ftp.disi.unige.it/person/ReggioG/ChoppyReggio03a.pdf>.

