A State Class Construction for Computing the Intersection of Time Petri Nets Languages

Eric LUBAT, Silvano DAL ZILIO, Didier LE BOTLAN,
Yannick PENCOLE and Audine SUBIAS

LAAS - CNRS

August 2019 / FORMATS

Index

Introduction

Objective: computing (*efficiently*) the intersection of timed languages to check properties on systems.

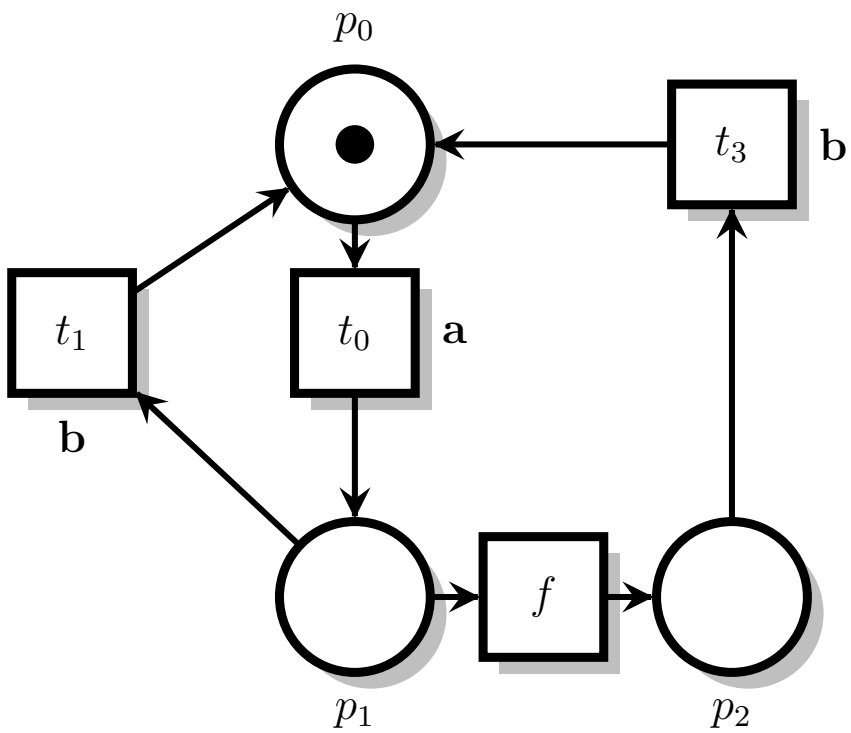
Which properties? Temporal properties; observability; ...

Introduction

Traces: $t_0\ t_1\ t_0\ t_1\ \ldots$
$t_0\ f\ t_3\ t_0\ t_1\ \ldots$

Words and Labels: $\mathcal{L}(t_0) = a$
$\mathcal{L}(t_1) = \mathcal{L}(t_3) = b$
$\mathcal{L}(f) = \epsilon$

Labelled Traces: $a\ b\ a\ b\ \ldots$
$a\ b\ a\ b\ \ldots$

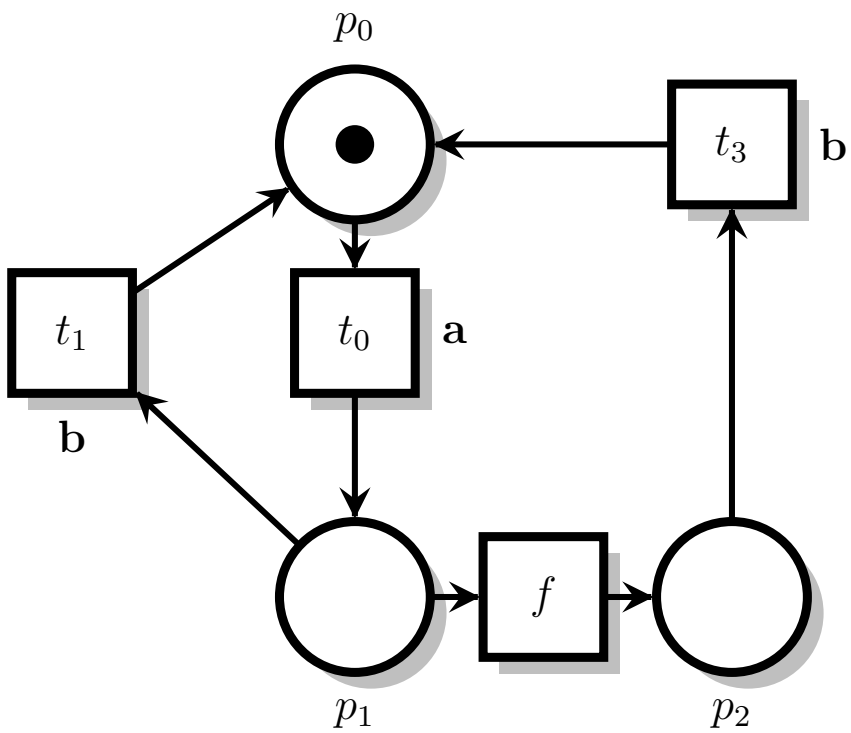


Introduction

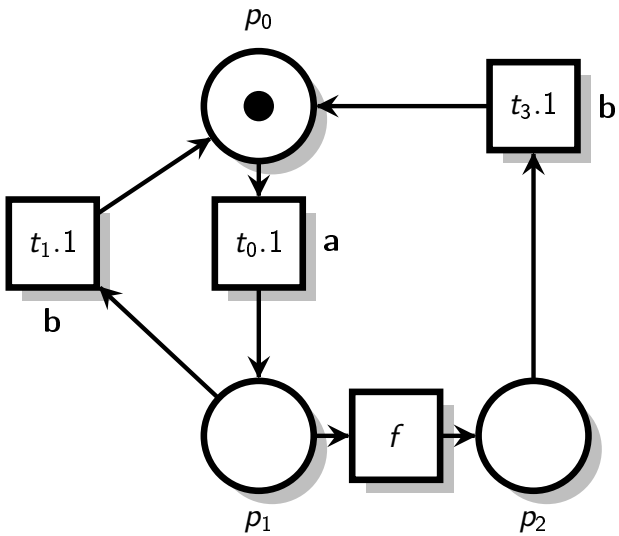
Traces: $t_0\ t_1\ t_0\ t_1\ \ldots \rightarrow a\ b\ a\ b\ \ldots$
$t_0\ f\ t_3\ t_0\ t_1\ \ldots \rightarrow a\ b\ a\ b\ \ldots$

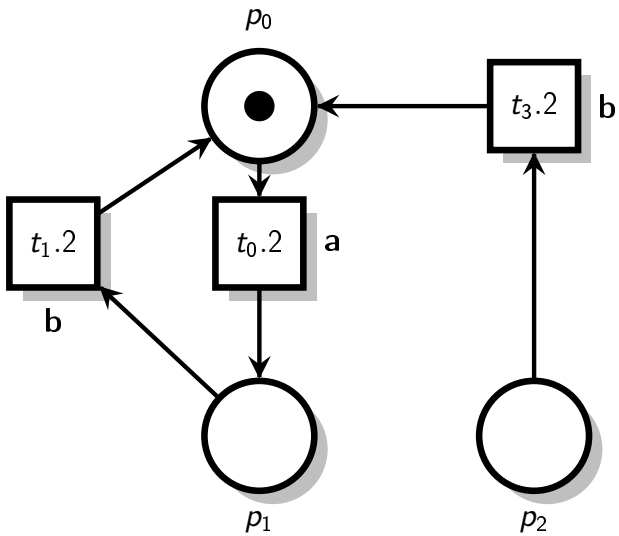
Language: $(ab)^*$

Property: is diagnosable ?



Checking diagnosability





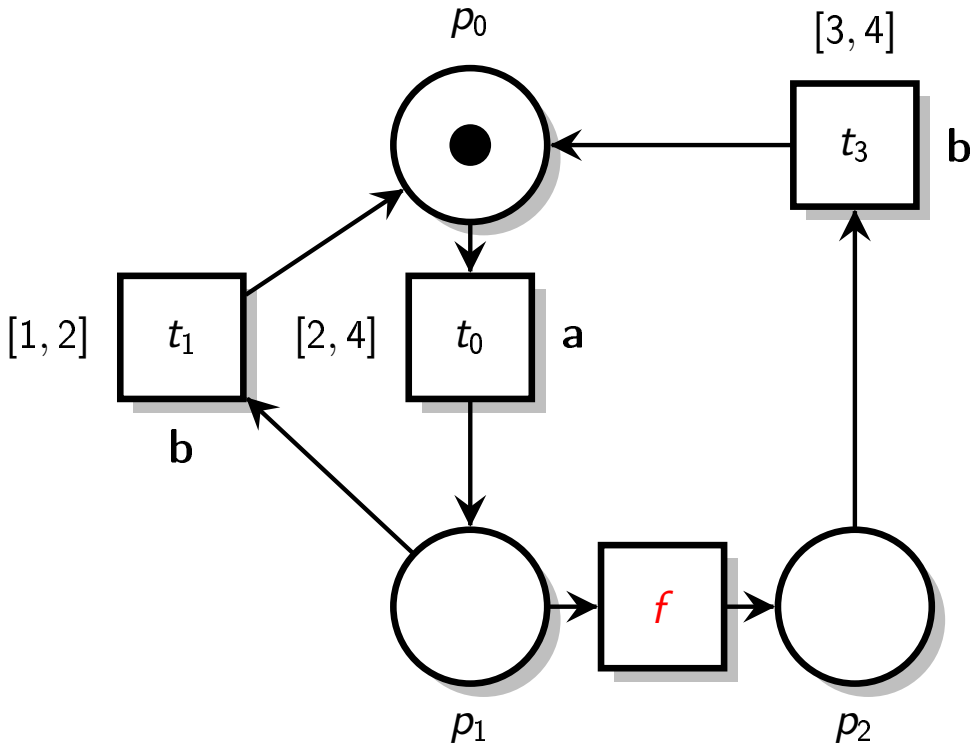
We can use "synchronous product" ($\|$) to check the intersection between languages.

$$[\![\mathrm{Sys}]\!] \cap [\![\mathrm{Sys}_{|f}]\!] \approx [\![\mathrm{Sys} \parallel \mathrm{Sys}_{|f}]\!]$$

Time Petri Nets

TPN are composed of :

- a Petri net; an initial marking
- timing constraints $\mathbf{I}_s(t)$ (*aka* static time interval).



Possible executions:

$2.1\ t_0\ 2\ t_1\ \ldots \quad / \quad 2.1\ a\ 2\ b\ \ldots$
$2.1\ t_0\ 1\ f\ 3\ t_3\ \ldots / \ 2.1\ a\ 4\ b\ \ldots$

Language: $(ab)^*$

Motivation for our work

Fault diagnosis for Discrete Event System $\equiv$ properties on trace languages [Lafortune - 95].

Addition of time constraints.

- Δ-diagnosability $\equiv$ "reachability of (non-Zeno) runs in product of TA" [Tripakis - 02]
- τ-diagnosability $\equiv$ "same" for TPN [Silva - 15, Basile - 17] (but ask for a firing sequence)

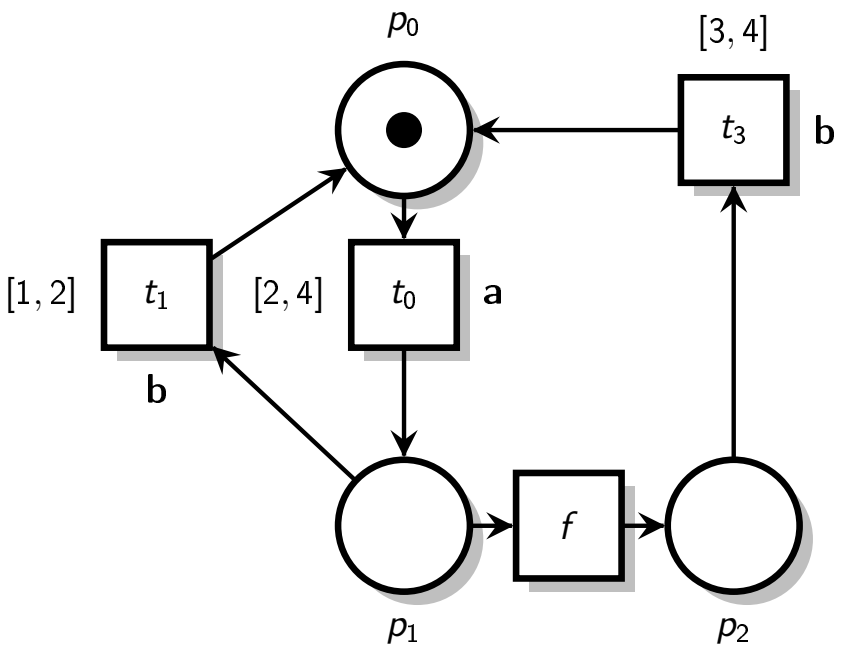
Composition of TPN

Analysing the "intersection" of TPN is hindered by two problems:

1. State Space is infinite
2. *Composability problem*

State Space is infinite

Solution: use abstractions based on *State Classes* [Berthomieu - 83].

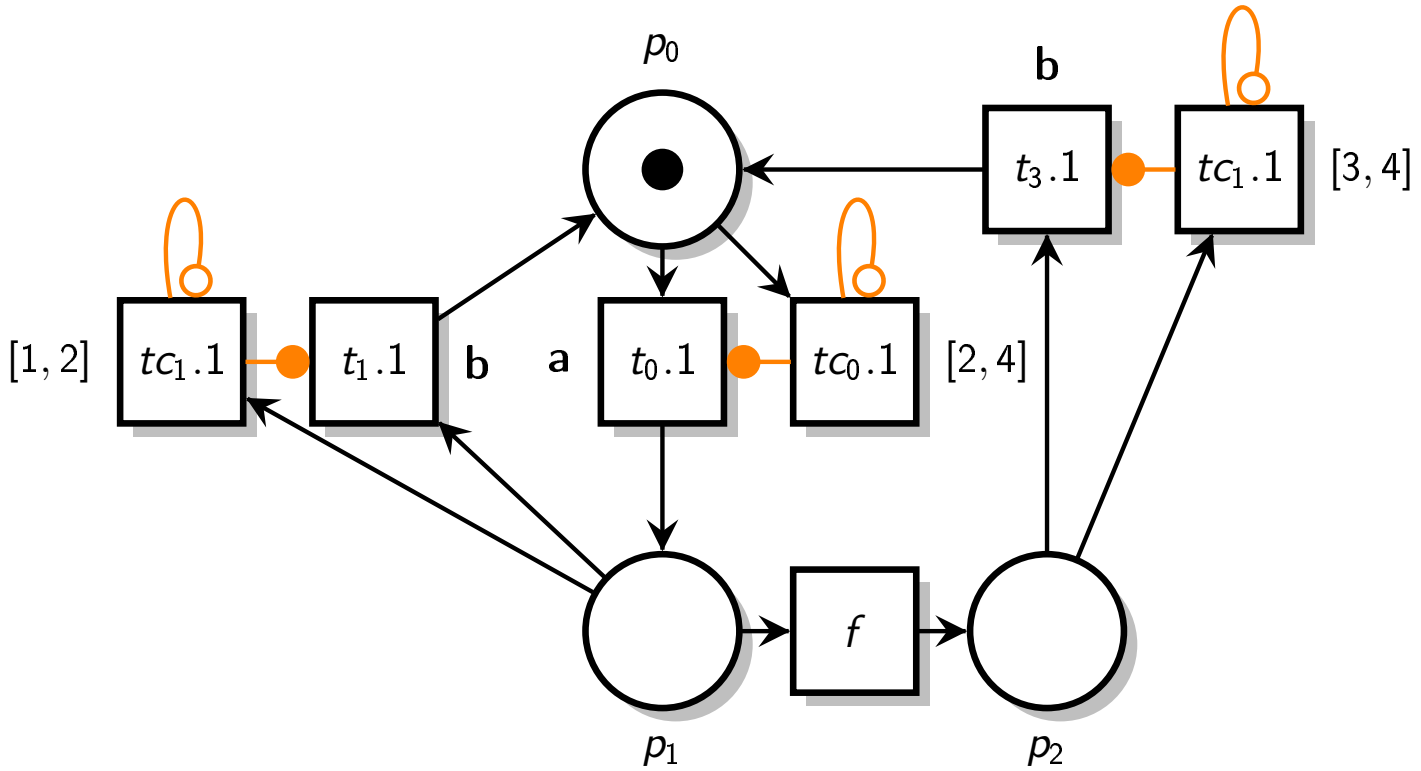



```
# states 3; transitions 4
(class 0) marking: p0
domain: 2 <= t0 <= 4
successors: t0/1

(class 1) marking: p1
domain: 1 <= t1 <= 2
        0 <= f  < w
successors: t1/0 f/2

(class 2)  marking: p2
domain: 3 <= t3 <= 4
successors: t3/0
```

Composability: third solution

IPTPN are TPN with "Inhibit" and "Permit" arcs [Perez - 11].

Adds the possibility to isolate “timing constraints” from “transition enabledness”.



Composability of TPN

Extend the *State Classes* construction to the product of two[1] TPN.

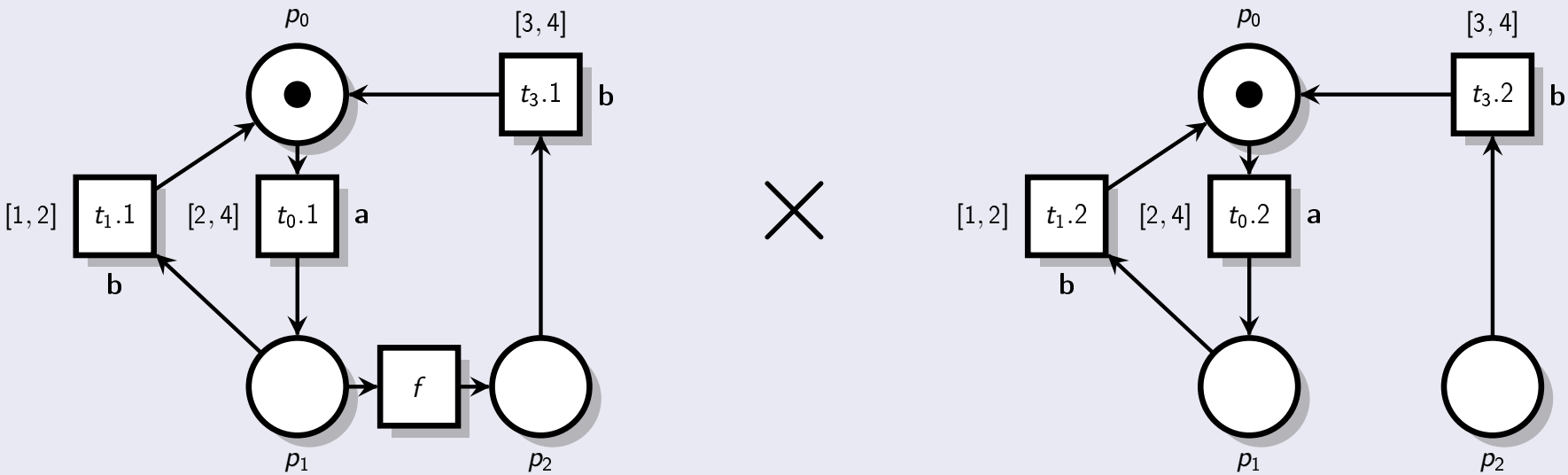
[1] also work with the product of n transitions

Index

PTPN

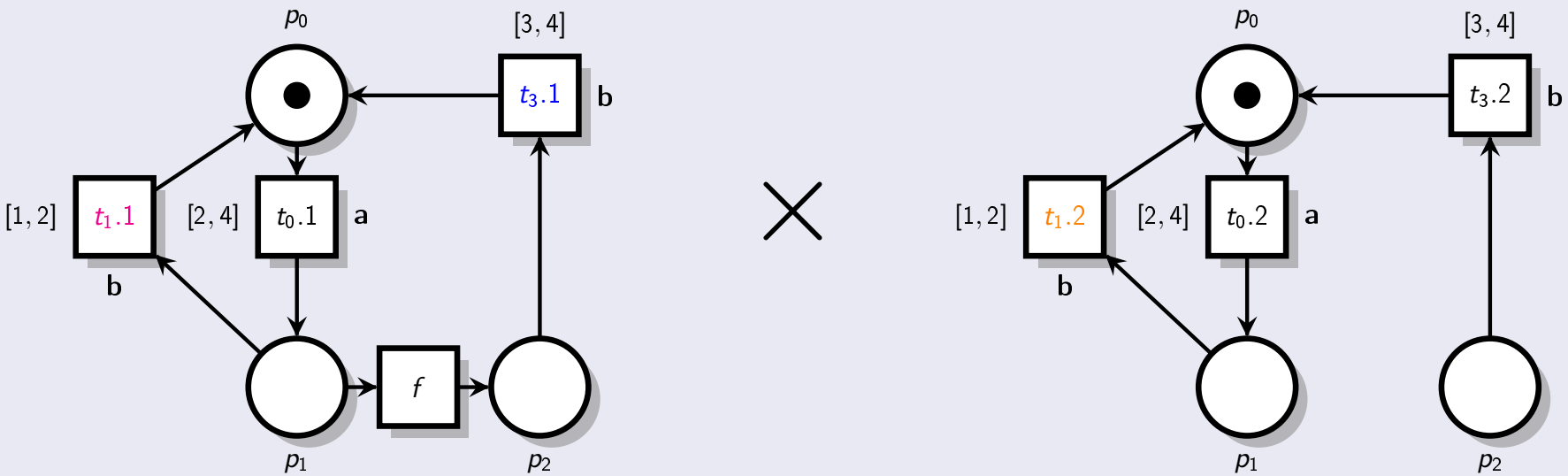
Idea: use a product, $N_1 \times N_2$, and force transitions with same label (e.g. $t_3.1 \in N_1$ and $t_1.2 \in N_2$) to fire "synchronously".

Example



PTPN

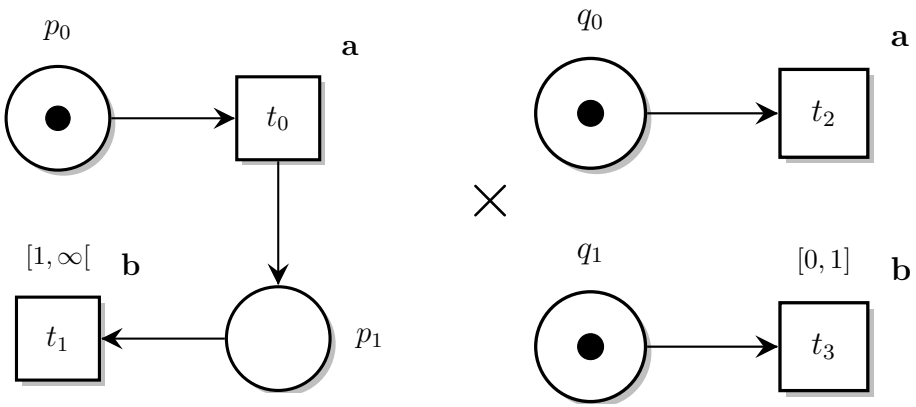
Example

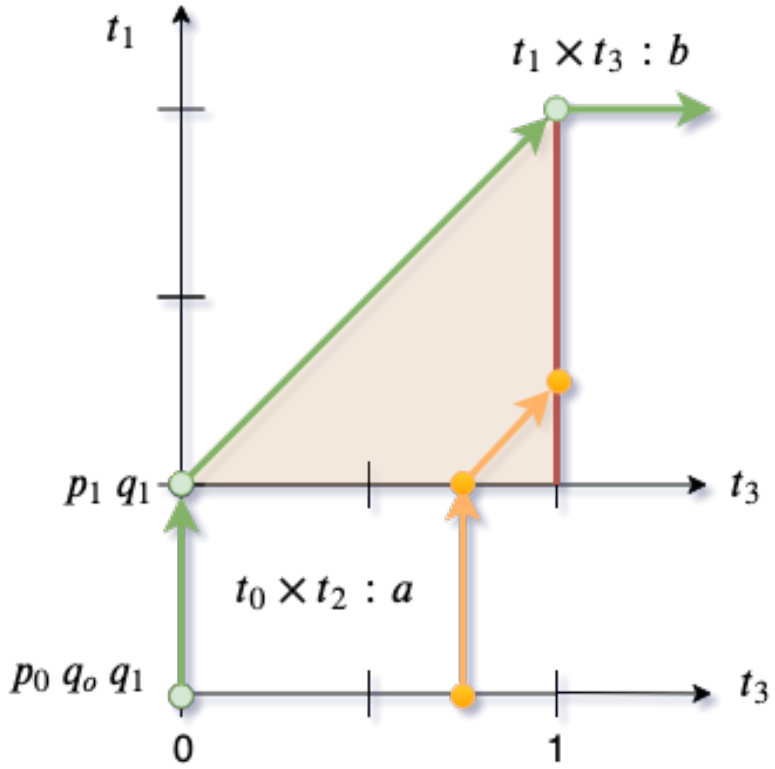


We fire "Synchronous Sets" of transitions at the same time:

$$\{t_0.1, t_0.2\}, \{t_1.1, t_1.2\}, \{t_3.1, t_1.2\}, \{f\}$$

PTPN: example of behaviour





Time elapse as in "classical" TPN $\Rightarrow$ must fire t_0 before 1 initially.

Transitions $\{t_0, t_2\}$ and $\{t_1, t_3\}$ must fire simultaneously $\Rightarrow$ this can create "timelocks".

PTPN: expressiveness

On one side, PTPN semantics introduces new timelocks.

On the other side, you cannot "lose a transition" only by waiting.

Theorem

PTPN are as expressive as TPN (up-to wtb: $\approx$):

$$\forall P \in PTPN\ \exists N \in TPN\ .\ N \approx P$$

[1] Says nothing about the size of $N \propto P$.

PTPN: application to the composability problem

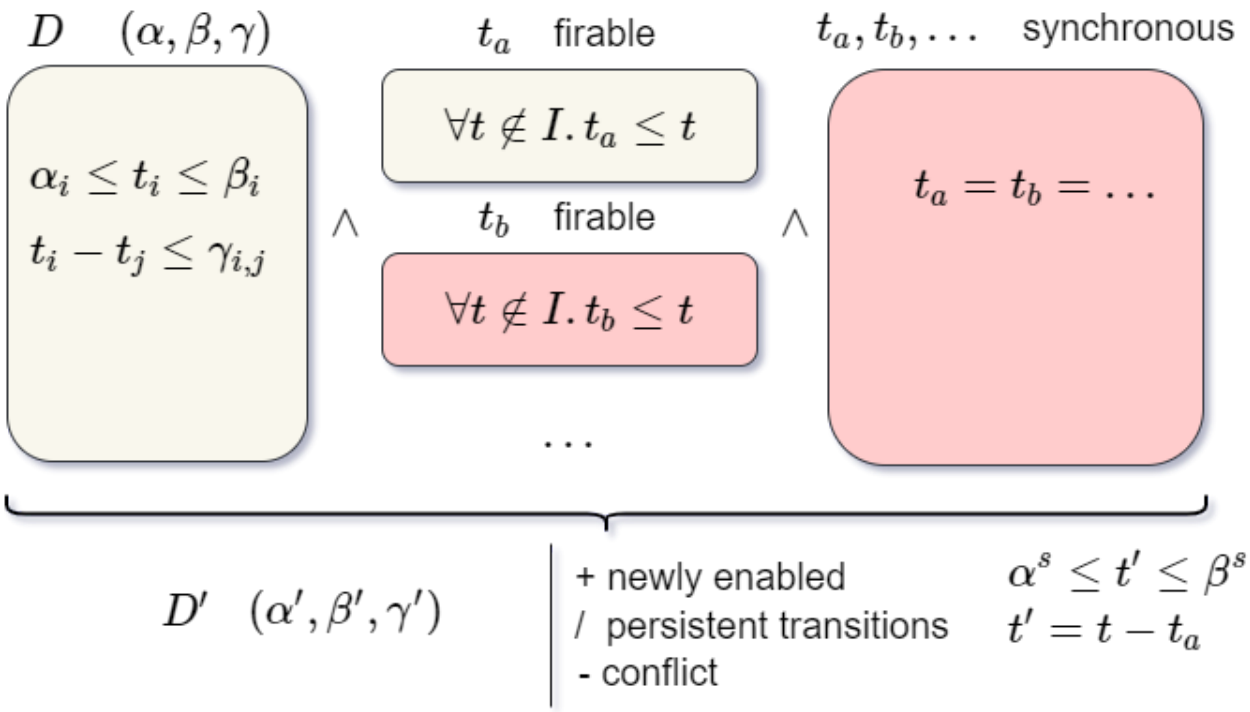
- We prove that "product" is a *congruent composition operator*:

$$[\![N_1 \times N_2]\!] \approx [\![N_1]\!] \parallel [\![N_2]\!]$$

- It is possible to adapt the SCG construction to PTPN
- Hence we can compute the "SCG" for the intersection of two TPN
- This has been implemented in a tool: TWINA (`https://projects.laas.fr/twina/`)

DBM and PTPN

Assume we fire $I = \{t_a, t_b, \dots\}$ synchronously from (m, D)



Index

Comparing size of PTPN and IPTPN

Model	Exp.	Twina Classes	IPTPN/sift Classes	Ratio
jdeds	plain	26	28	$\times 1.1$
jdeds	twin	544	706	$\times 1.3$
jdeds	obs	57	64	$\times 1.1$
train3	plain	$3.10 \cdot 10^3$	$5.05 \cdot 10^3$	$\times 1.6$
train3	twin	$1.45 \cdot 10^6$	$4.02 \cdot 10^6$	$\times 2.8$
train3	obs	$6.20 \cdot 10^3$	$1.01 \cdot 10^4$	$\times 1.6$
train4	plain	$1.03 \cdot 10^4$	$1.68 \cdot 10^4$	$\times 1.6$
train4	twin	$2.10 \cdot 10^7$	$5.76 \cdot 10^7$	$\times 2.7$
train4	obs	$2.06 \cdot 10^4$	$3.37 \cdot 10^4$	$\times 1.6$
plant	plain	$2.70 \cdot 10^6$	$4.63 \cdot 10^6$	$\times 1.7$
plant	twin	$1.30 \cdot 10^3$	$1.63 \cdot 10^3$	$\times 1.3$
plant	obs	$5.72 \cdot 10^6$	$9.79 \cdot 10^6$	$\times 1.7$
wodes	plain	$2.55 \cdot 10^3$	$5.36 \cdot 10^3$	$\times 2.1$
wodes	twin	$5.54 \cdot 10^4$	$1.51 \cdot 10^5$	$\times 2.7$
wodes	obs	$5.77 \cdot 10^3$	$1.47 \cdot 10^4$	$\times 2.5$
wodes232	plain	$2.04 \cdot 10^4$	$3.24 \cdot 10^4$	$\times 1.6$
wodes232	twin	$3.96 \cdot 10^7$	$3.39 \cdot 10^8$	$\times 8.6$
wodes232	obs	$1.06 \cdot 10^5$	$2.26 \cdot 10^5$	$\times 2.1$

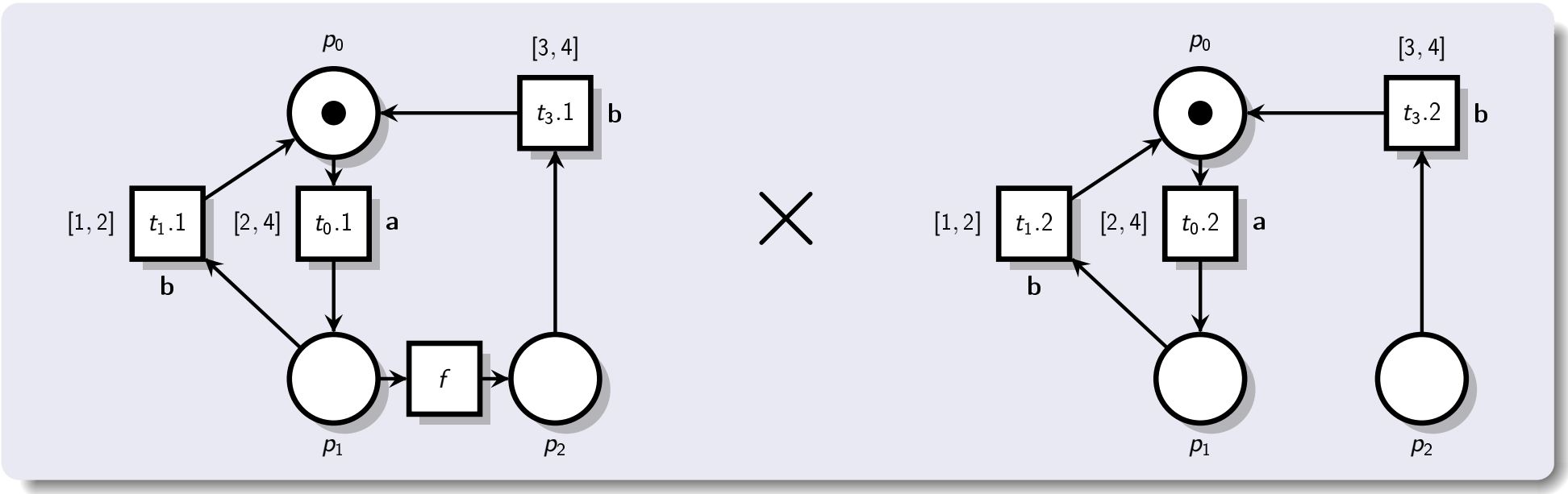
Focus on WODES-2-3-2

Model	Exp.	Twina Classes	IPTPN/sift Classes	Ratio
wodes232	plain	$2.0 \cdot 10^4$	$3.2 \cdot 10^4$	$\times 1.6$
wodes232	twin	$4.0 \cdot 10^7$	$3.4 \cdot 10^8$	$\times 8.6$
wodes232	obs	$1.1 \cdot 10^5$	$2.3 \cdot 10^5$	$\times 2.1$

$$|N \times N_{|f}| \leqslant \mathcal{O}(|N|^2)$$

Comparison with encoding into TPN

Results on our running example; for exp. twin



	TWINA	IPTPN/SIFT	TPN/SIFT
PLACES	6	6	25
TRANS.	7	12 (6 + 6)	211
CLASSES	3	3	1389

Conclusion

- We propose an extension of TPN with "synchronous product" of transitions and extend the (Linear) State Class Graph construction.
- We have implemented this construction in a tool.
- This is a tribute to Bernard Berthomieu's work.
- Future works: experimental comparaison with T.A.; motif detection in traces; ...

Conclusion

Thanks for your attention !

Any questions ?

Composability of TPN: related work

Use encoding of TPN into TA [Cassez - 05, Bérard - 08]
preserves $\approx$ / restricted to "closed" timing constraints
size $\propto$ region graph; size of net with $|T|$ clocks

Structural encoding of TPN into *composable TPN* [Peres - 11]
preserves $\approx$ / restricted to "left-closed" timing constraints
size $\propto |T| \times$ t.c.

Structural encoding of TPN into TPN with "priorities" (IPTPN)
[Peres - 11]; preserves $\approx$ / requires the use of "strong classes"
size $\propto$ size of net