Deciding Reachability and Coverability in Lossy EOS and its implementations

Tephilla Prince

13/08/2024

Contribution

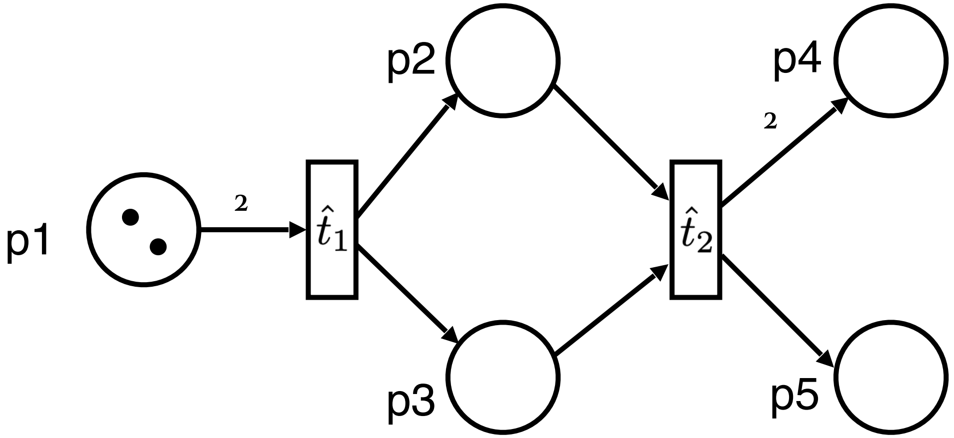
Created a Verification Prototype

For safety, reach, cover, deadlock

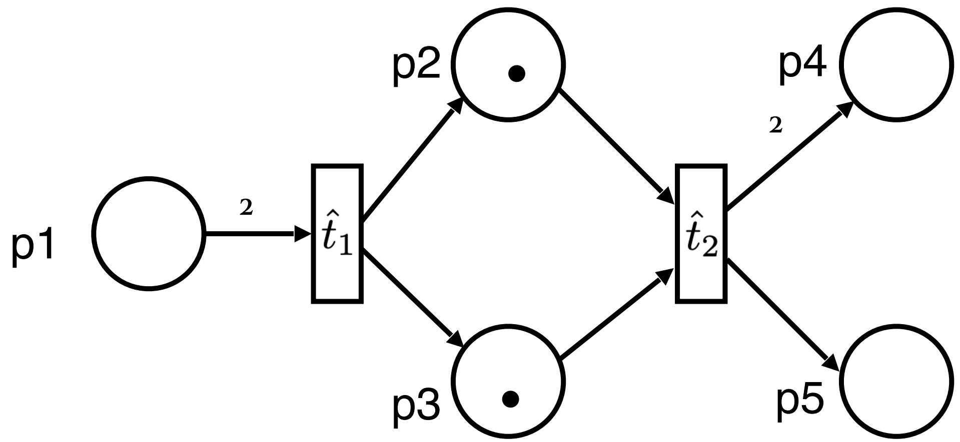
Of lossy PNs and EOSs

Available on Zenodo

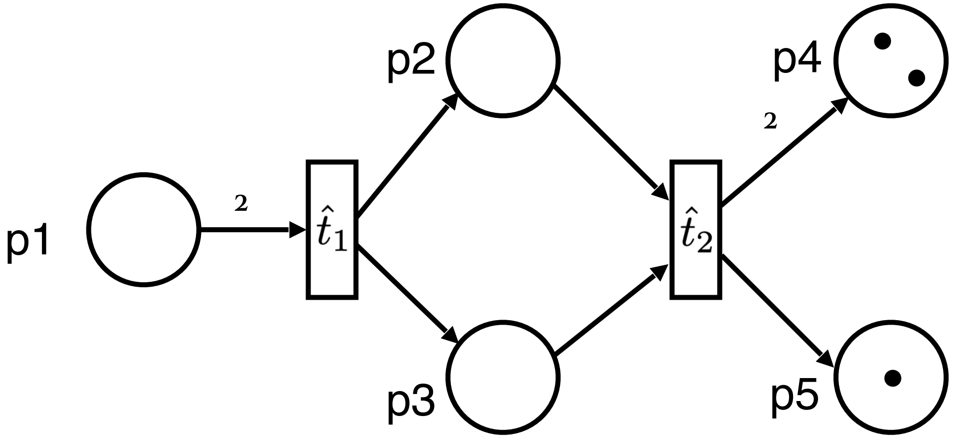
Petri Nets



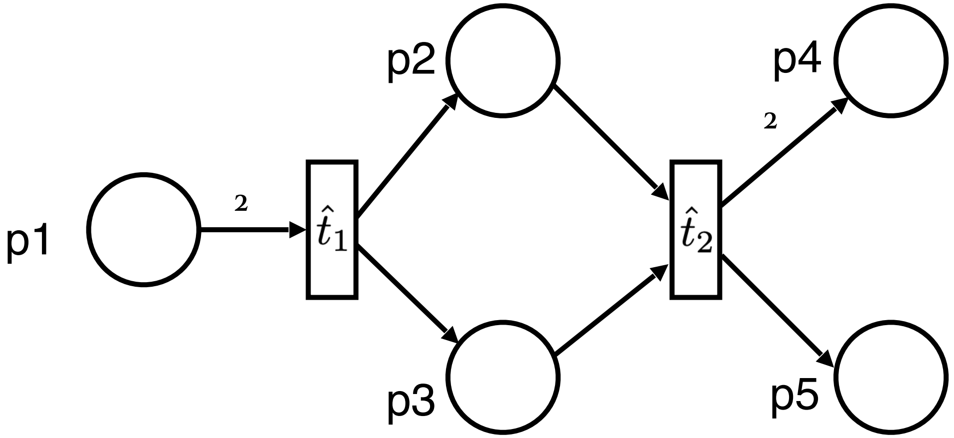
Petri Nets



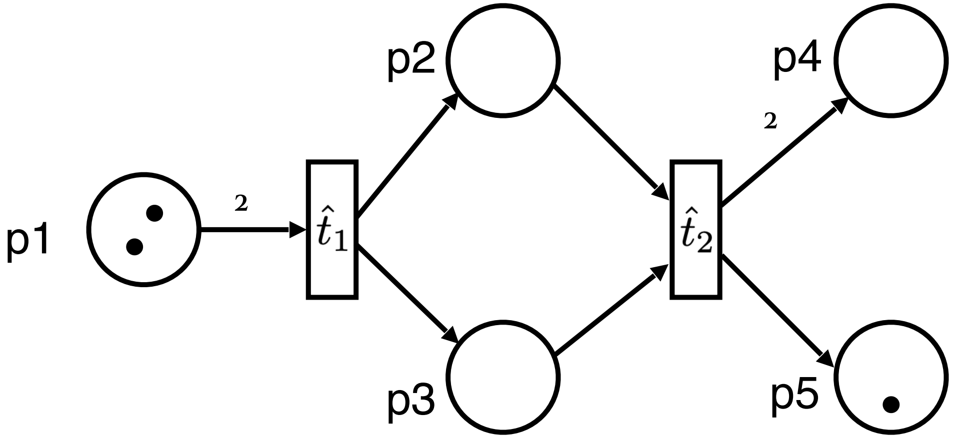
Petri Nets



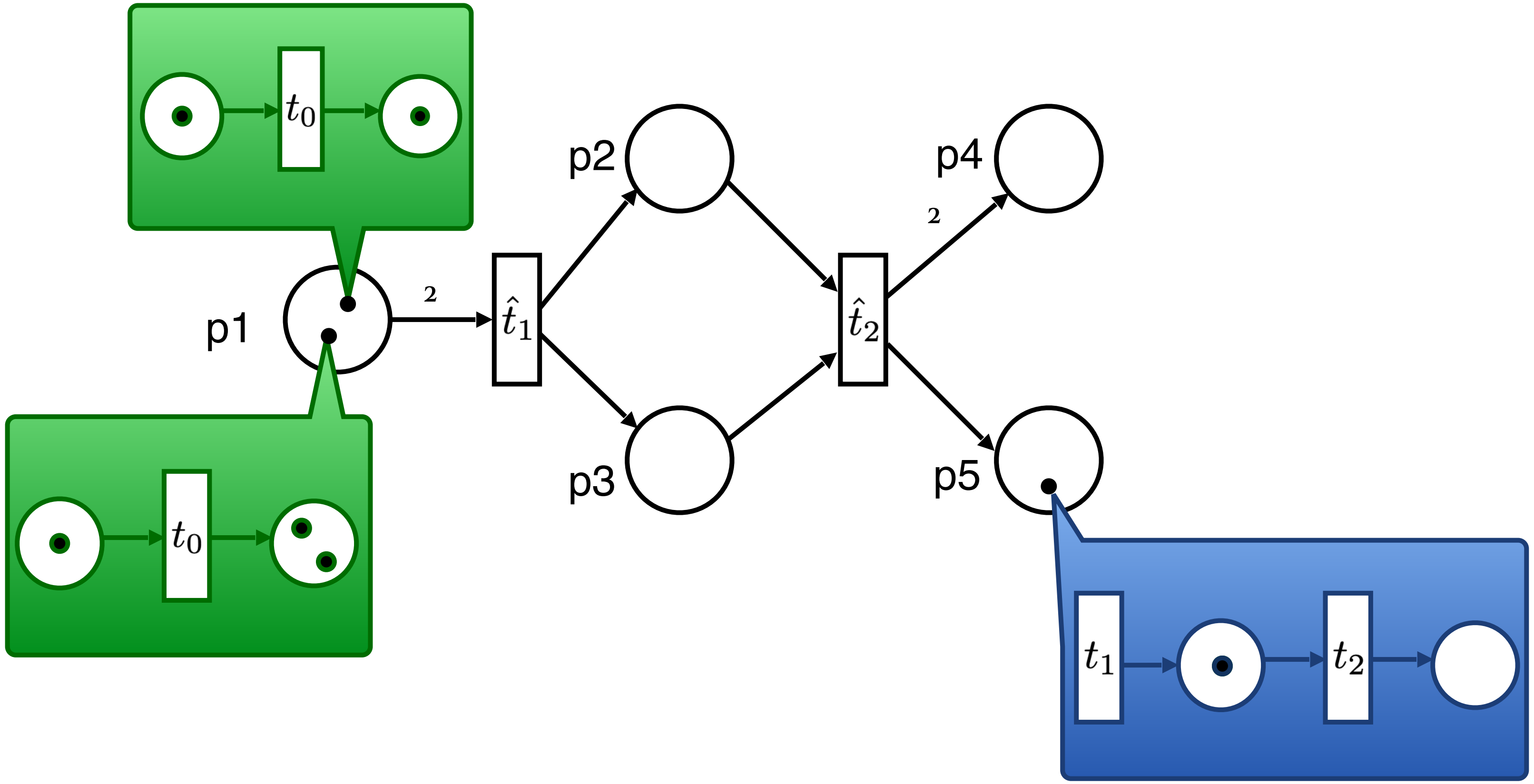
EOS – System Net



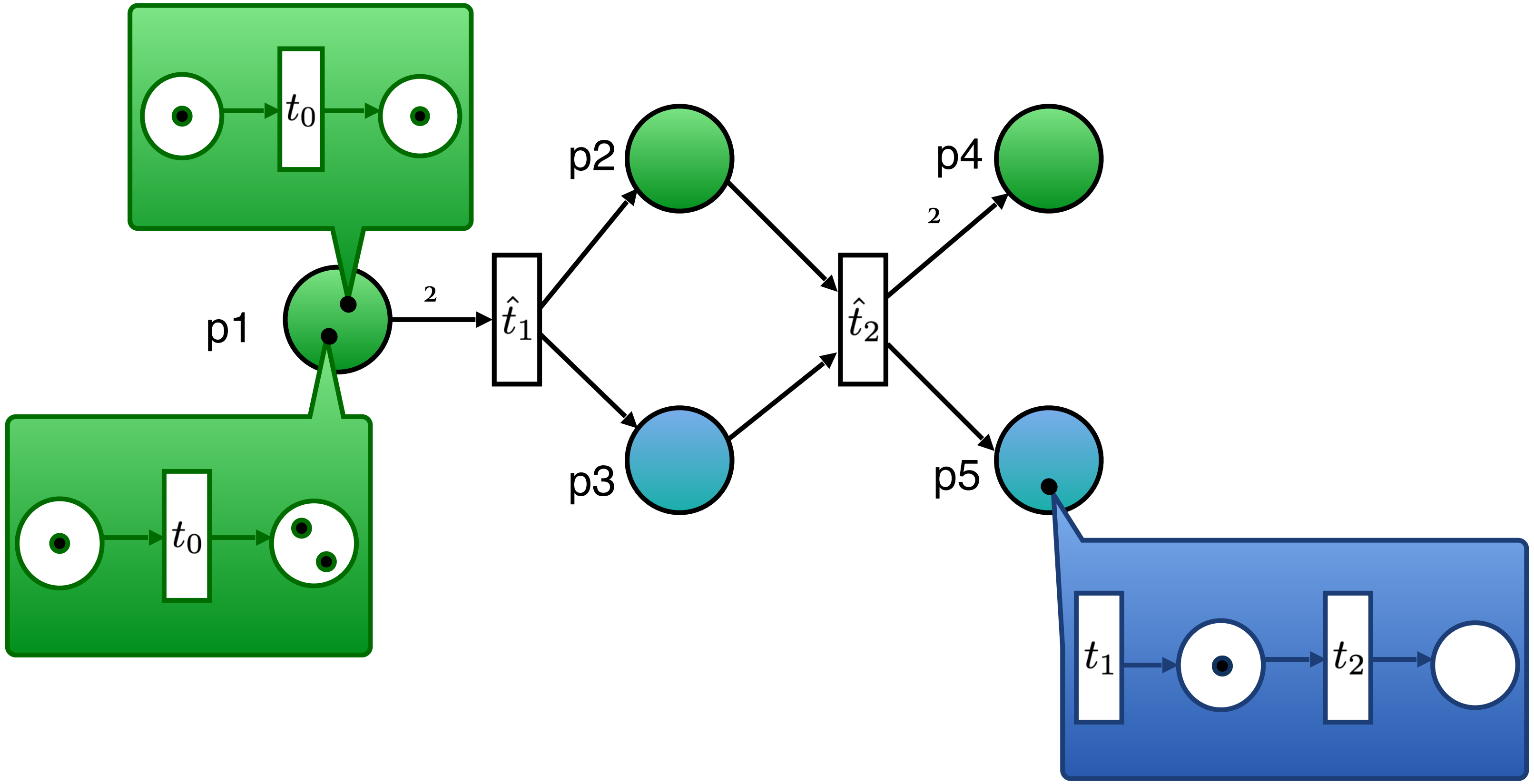
EOS – nested markings



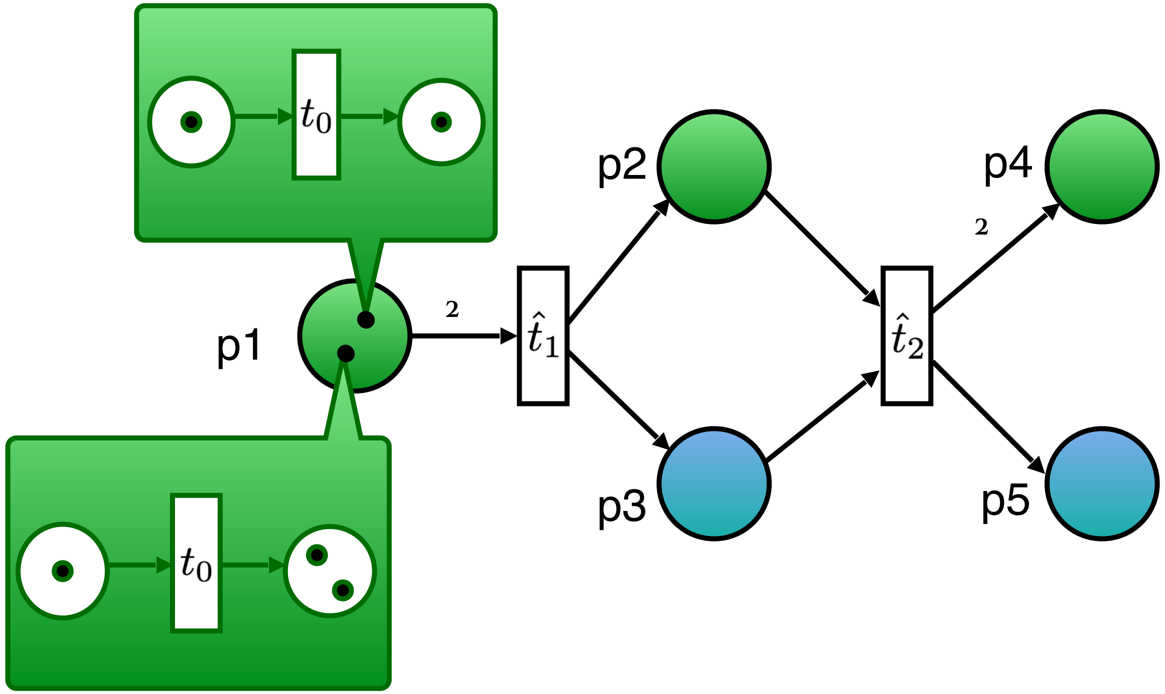
EOS – nested markings



EOS – typed places



EOS – object autonomous events



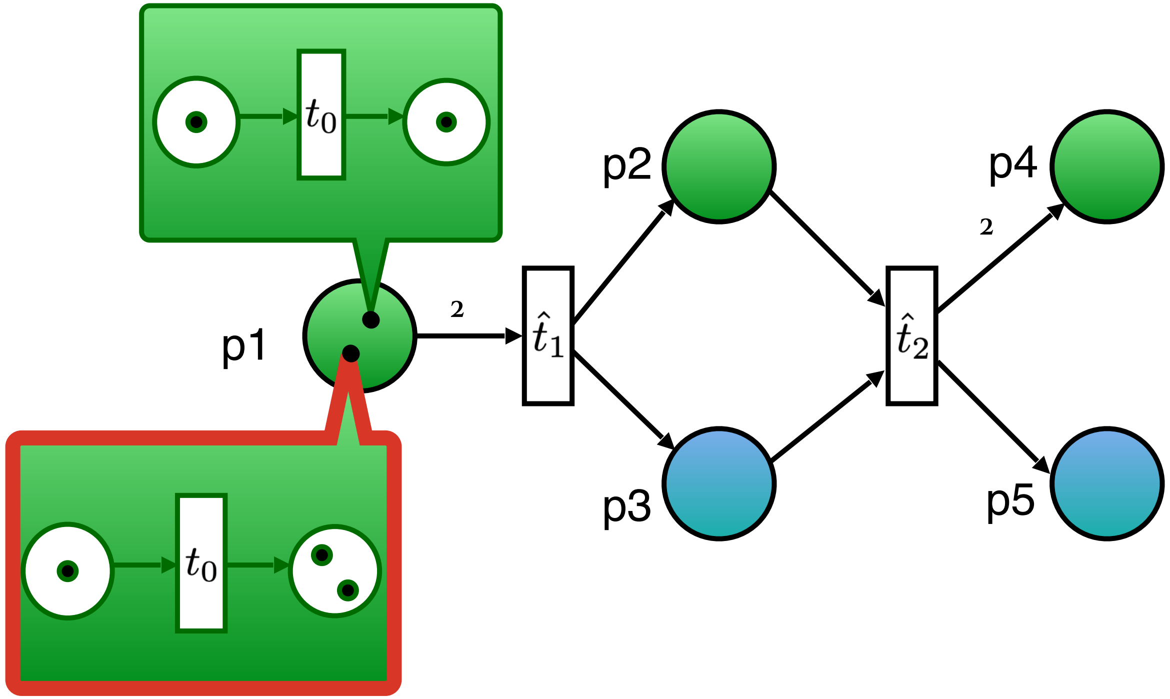
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \emptyset\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – object autonomous events



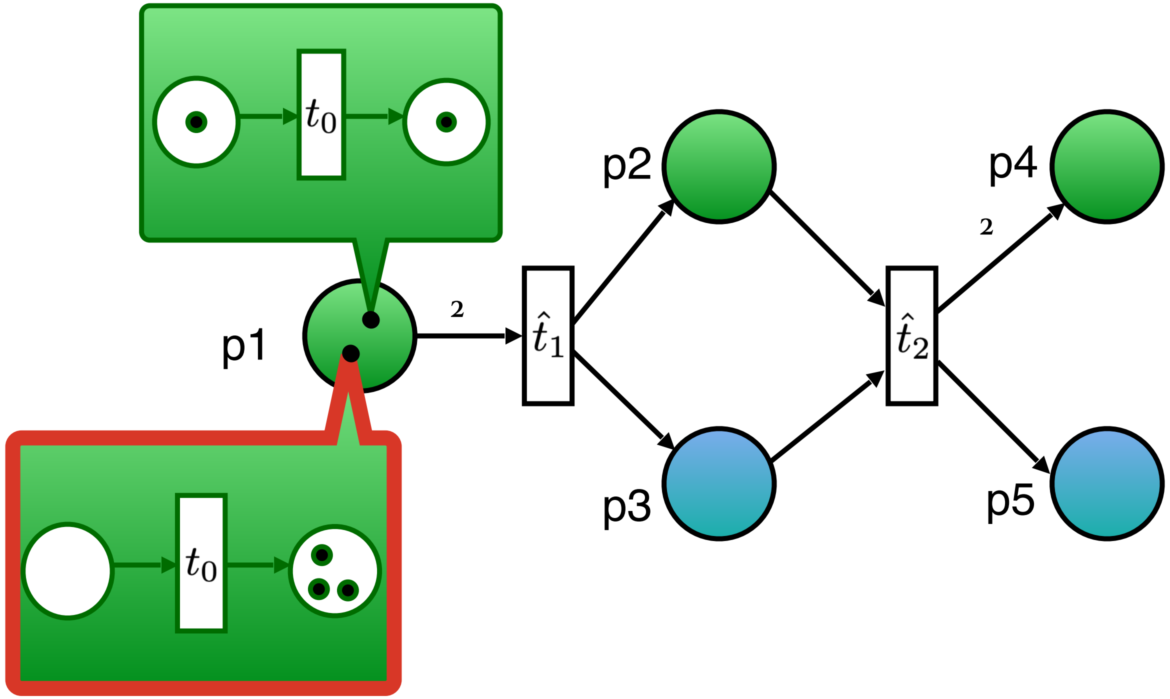
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \varnothing\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – object autonomous events



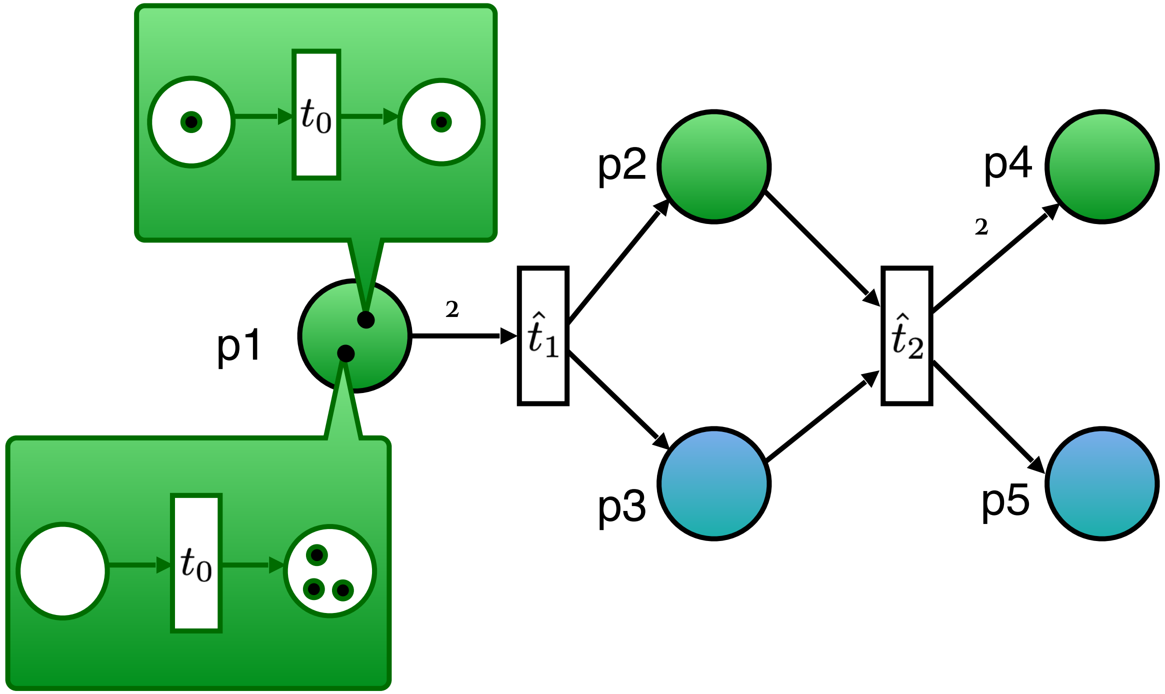
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \varnothing\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – system autonomous events



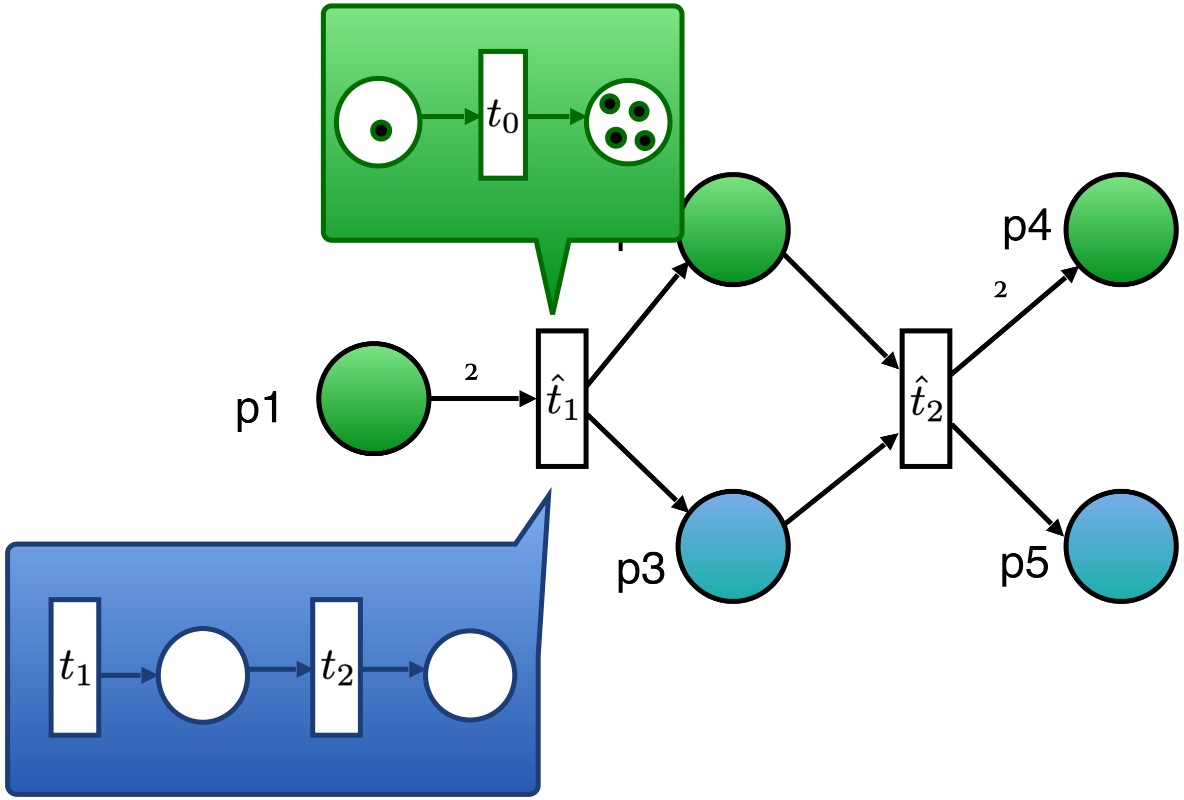
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \varnothing\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – system autonomous events



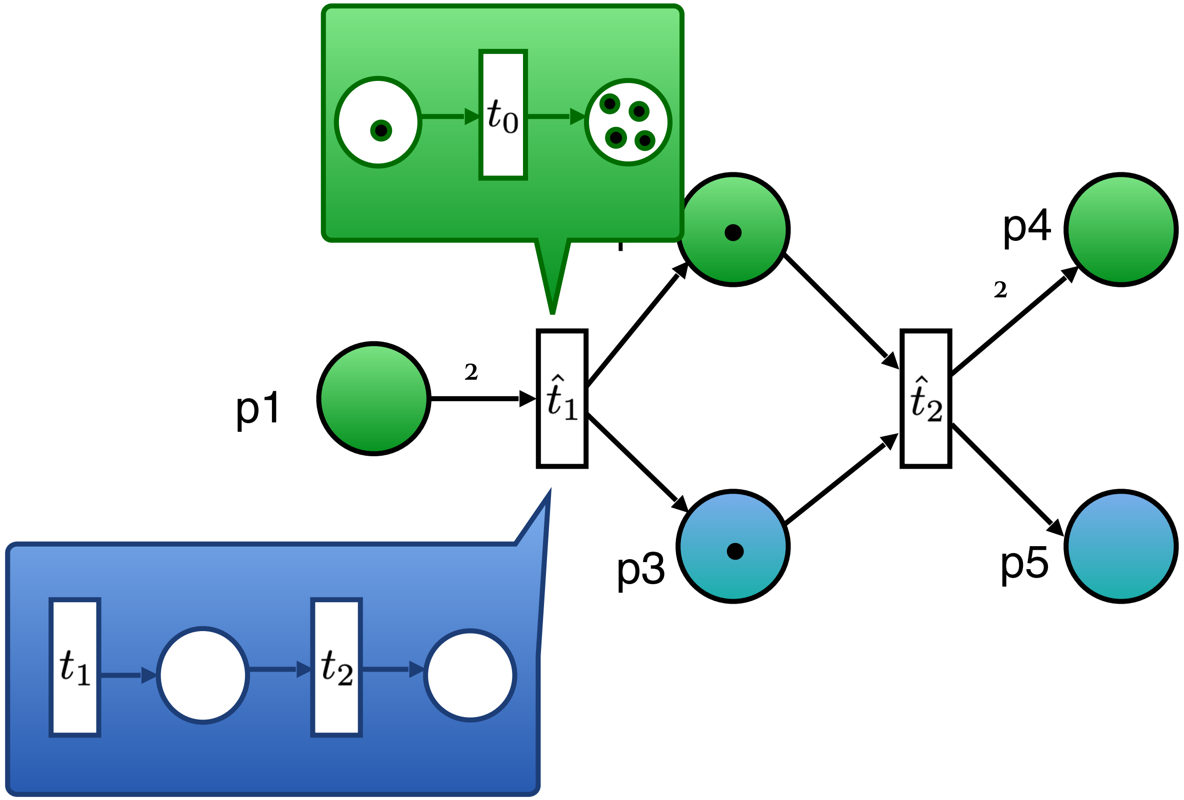
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \varnothing\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – system autonomous events



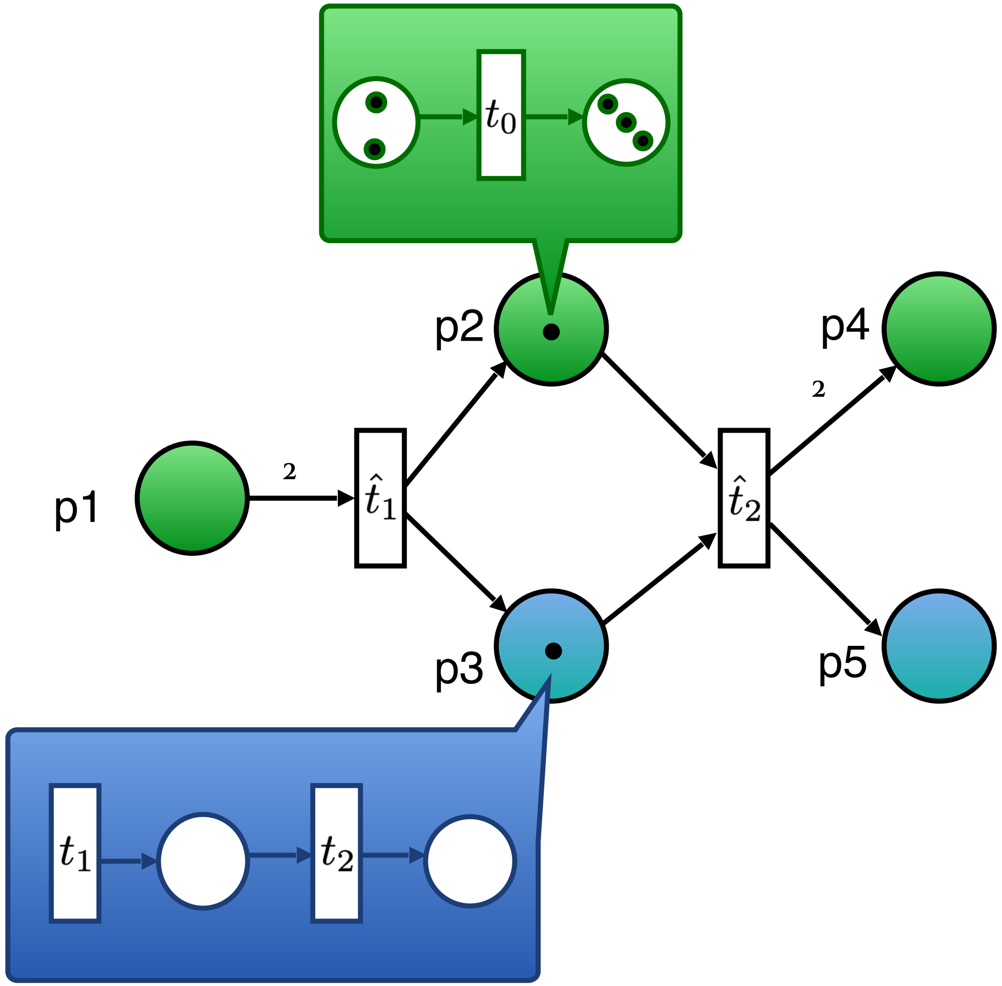
EVENTS

$$e_1 = \left(id_{p_1}, t_0 \right)$$

$$e_2 = \left(\hat{t}_1 , \varnothing \right)$$

$$e_3 = \left(\hat{t}_2 , t_0 t_1 t_1 \right)$$

EOS – system autonomous events



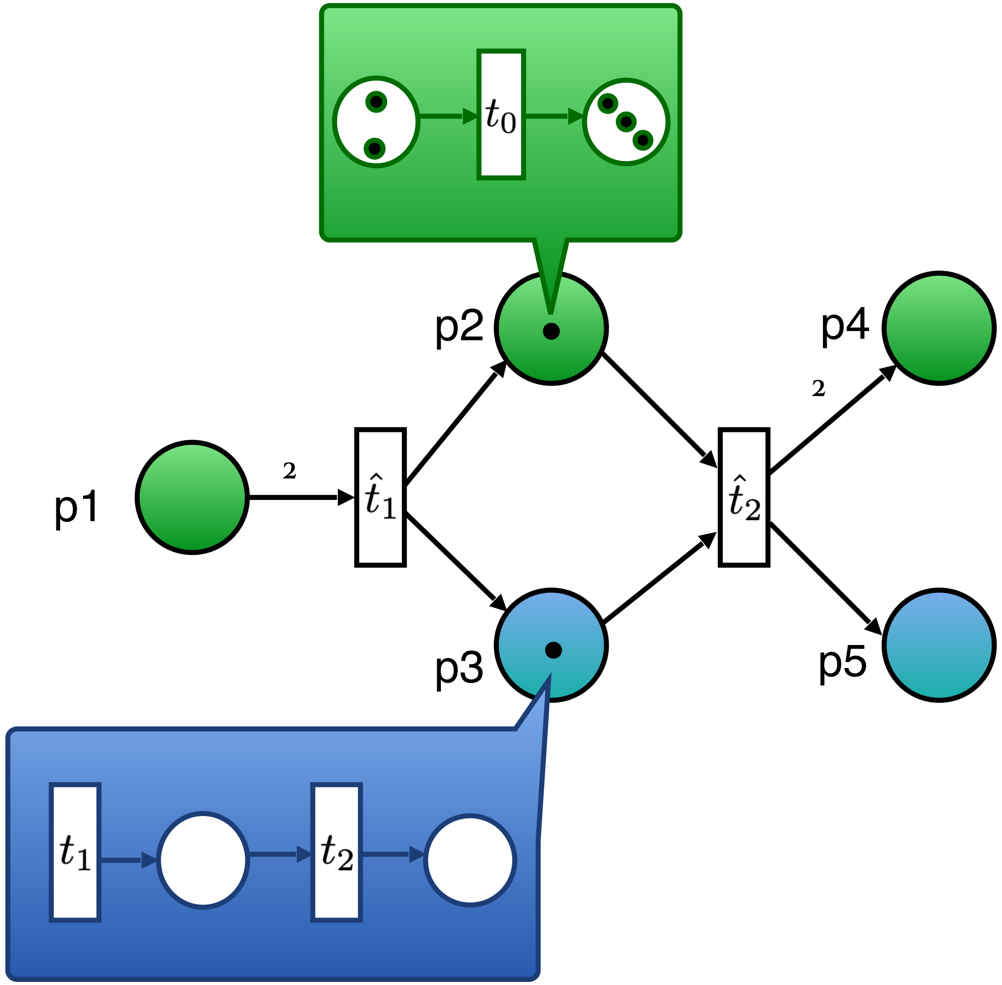
EVENTS

$e_1 = \left(id_{p_1}, t_0\right)$

$e_2 = \left(\hat{t}_1, \varnothing\right)$

$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$

EOS – synchronization events



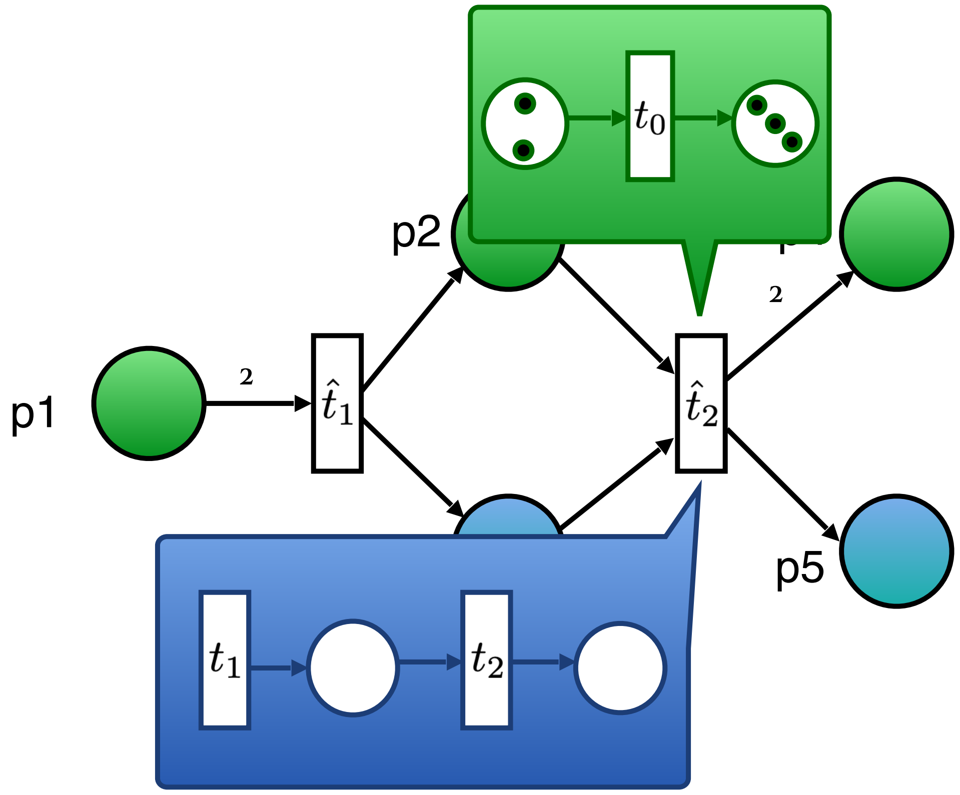
EVENTS

$$e_1 = \left(id_{p_1}, t_0\right)$$

$$e_2 = \left(\hat{t}_1, \varnothing\right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$$

EOS – synchronization events



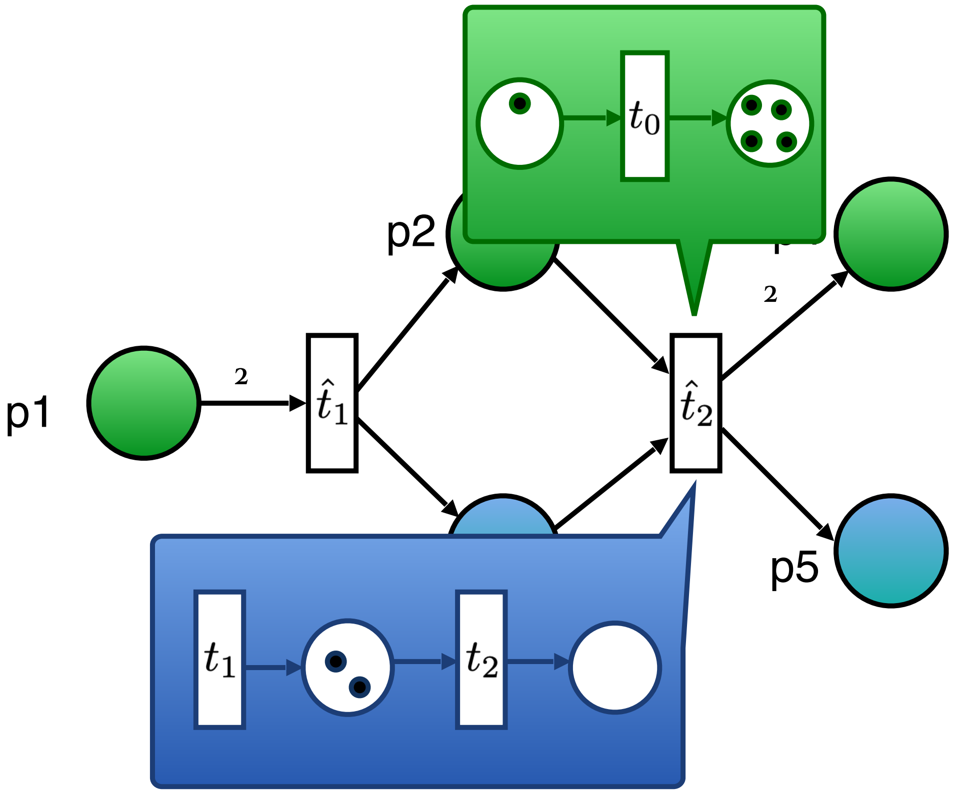
EVENTS

$$e_1 = \left(id_{p_1}, t_0 \right)$$

$$e_2 = \left(\hat{t}_1 , \varnothing \right)$$

$$e_3 = \left(\hat{t}_2 , t_0 t_1 t_1 \right)$$

EOS – synchronization events



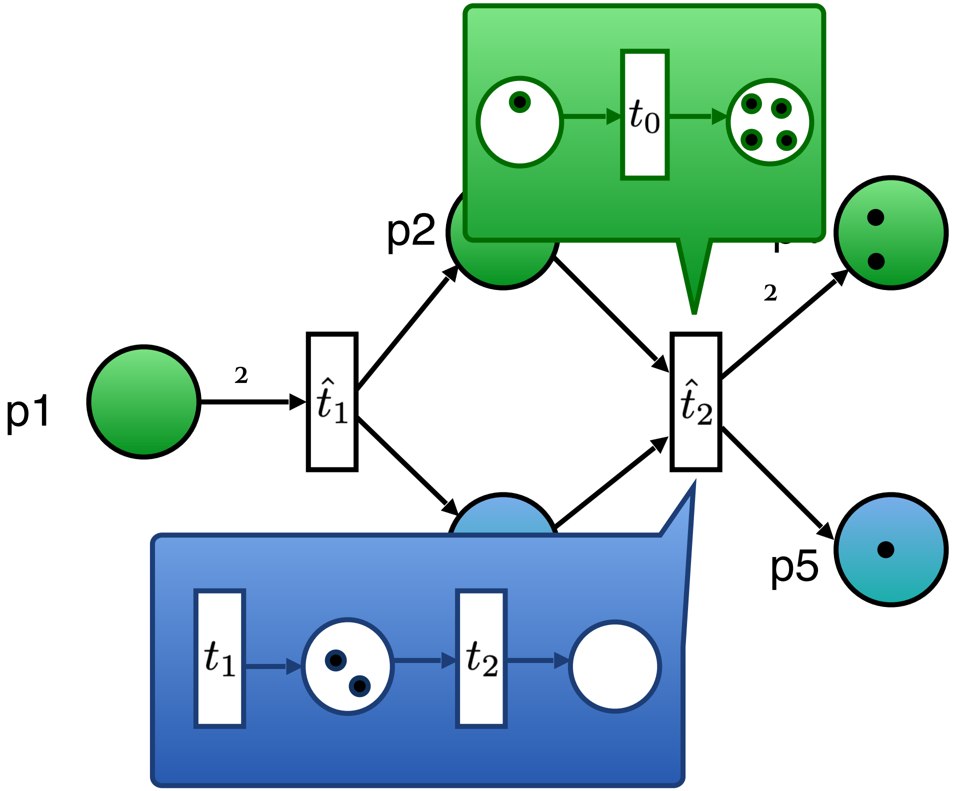
EVENTS

$e_1 = \left(id_{p_1}, t_0 \right)$

$e_2 = \left(\hat{t}_1 , \varnothing \right)$

$e_3 = \left(\hat{t}_2 , t_0 t_1 t_1 \right)$

EOS – synchronization events



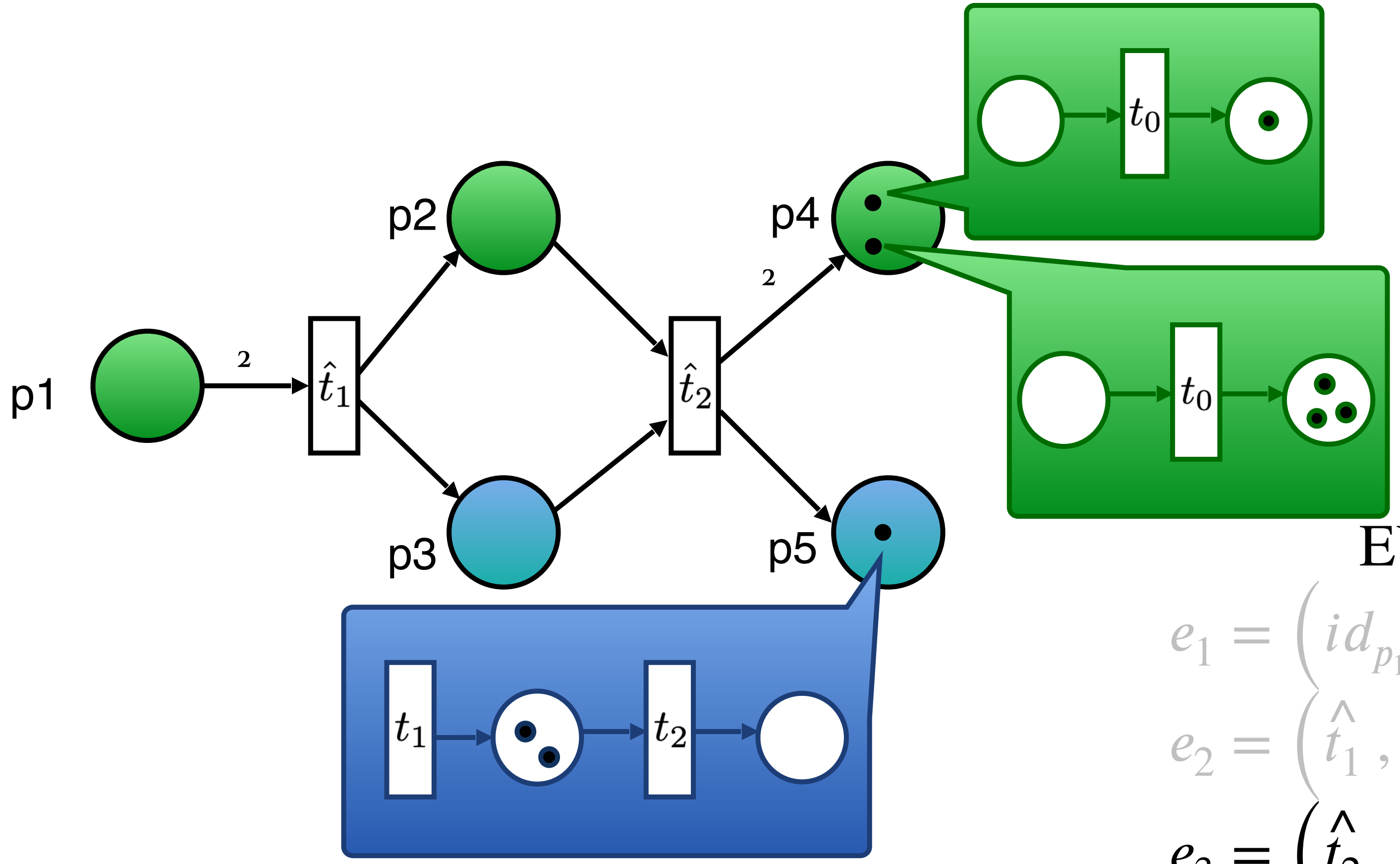
EVENTS

$$e_1 = \left(id_{p_1}, t_0 \right)$$

$$e_2 = \left(\hat{t}_1, \varnothing \right)$$

$$e_3 = \left(\hat{t}_2, t_0 t_1 t_1 \right)$$

EOS – synchronization events



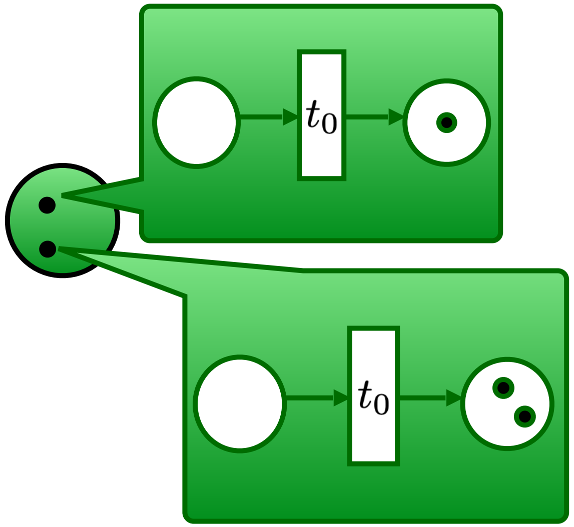
EVENTS

$e_1 = \left(id_{p_1}, t_0\right)$

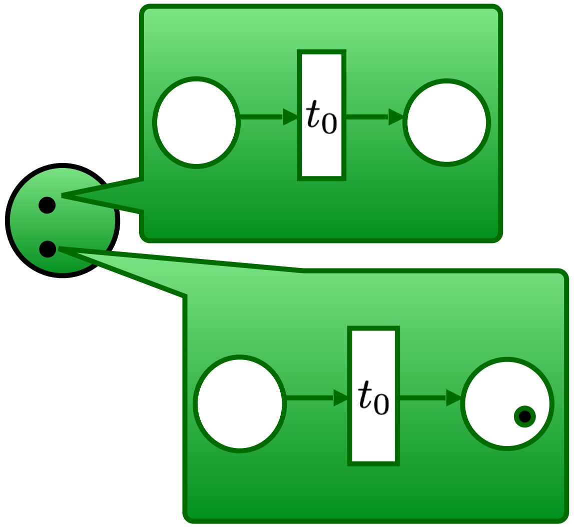
$e_2 = \left(\hat{t}_1, \varnothing\right)$

$e_3 = \left(\hat{t}_2, t_0 t_1 t_1\right)$

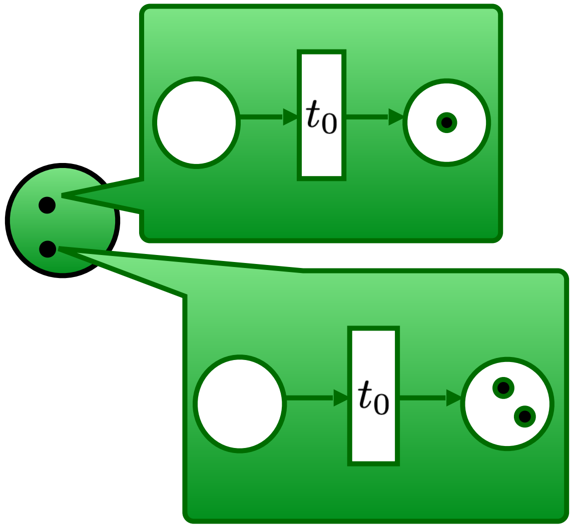
Object lossiness



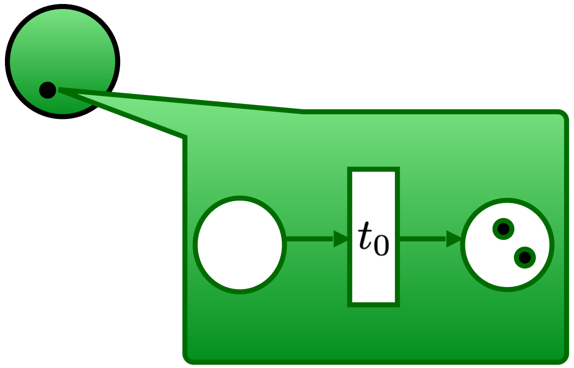
$$\geq_o$$



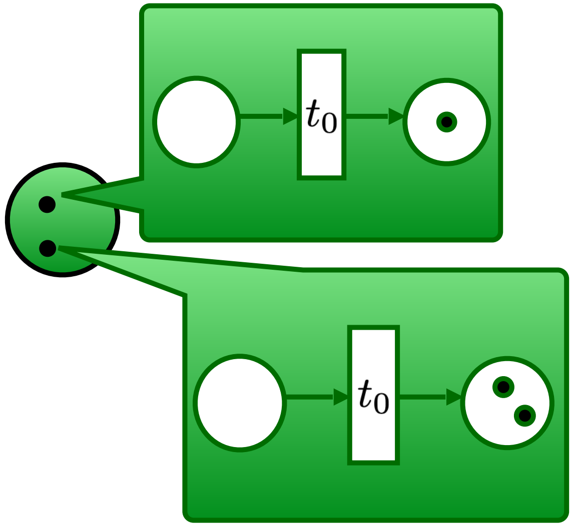
System lossiness



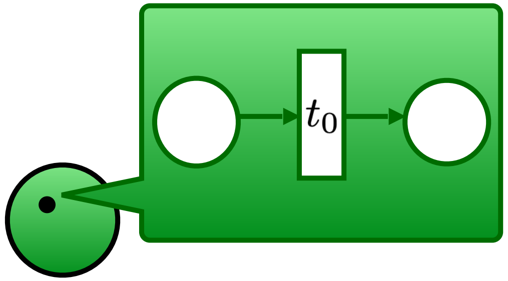
$\geq_s$



Full lossiness



$$\geq_f$$



$(\preccurlyeq, \ell)$-lossy runs

Perfect runs: only standard steps

$$M_1 \to M_2 \to M_3 \to \dots$$

$(\preccurlyeq, \ell)$-runs: at most $\ell \leq |\mathbb{N}|$ steps of type $\preccurlyeq$

$$M_1 \to M_2 \succcurlyeq M'_3 \to M'_4 \succcurlyeq M'_5 \succcurlyeq M'_6 \to M_7 \to \dots$$

$(\preccurlyeq, \ell)$-lossy problems

Is the system **robust** up to ℓ occurrences of $\preccurlyeq$?

$(\preccurlyeq, \ell)$-lossy problems

Is the system **robust** up to ℓ occurrences of $\preccurlyeq$?

$(\preccurlyeq, \ell)$*-deadlock freeness*

Input

An EOS E and an initial marking M.

Output

Is there a $(\preccurlyeq, \ell)$-run from M to a marking where no event is enabled?

Decidability Results (PNSE'24)

	Problems	$\preccurlyeq_f$	$\preccurlyeq_o$	$\preccurlyeq_s$
Conservative EOS	0-reach	**Undec** [Buß14]	**Undec** [Buß14]	**Undec** [Buß14]
	cover	**Dec** [Buß14]	**Undec** (2CM)	**Undec** (2CM)
	ℓ-reach/cover	**Dec** (Comp.)	**Undec** (Comp.)	**Undec** (Comp.)
	ω-reach/cover	**Dec** (Comp.)	**Undec** (Comp.)	**Undec** (Comp.)
EOS	0-reach	**Undec** [Buß14]	**Undec** [Buß14]	**Undec** [Buß14]
	cover	**Undec** [Buß14]	**Undec** (cEOS)	**Undec** (cEOS)
	ℓ-reach/cover	**Undec** (Gadget)	**Undec** (cEOS)	**Undec** (Comp.)
	ω-reach/cover	**Dec** (WSTS)	**Undec** (cEOS)	**Undec** (Comp.)

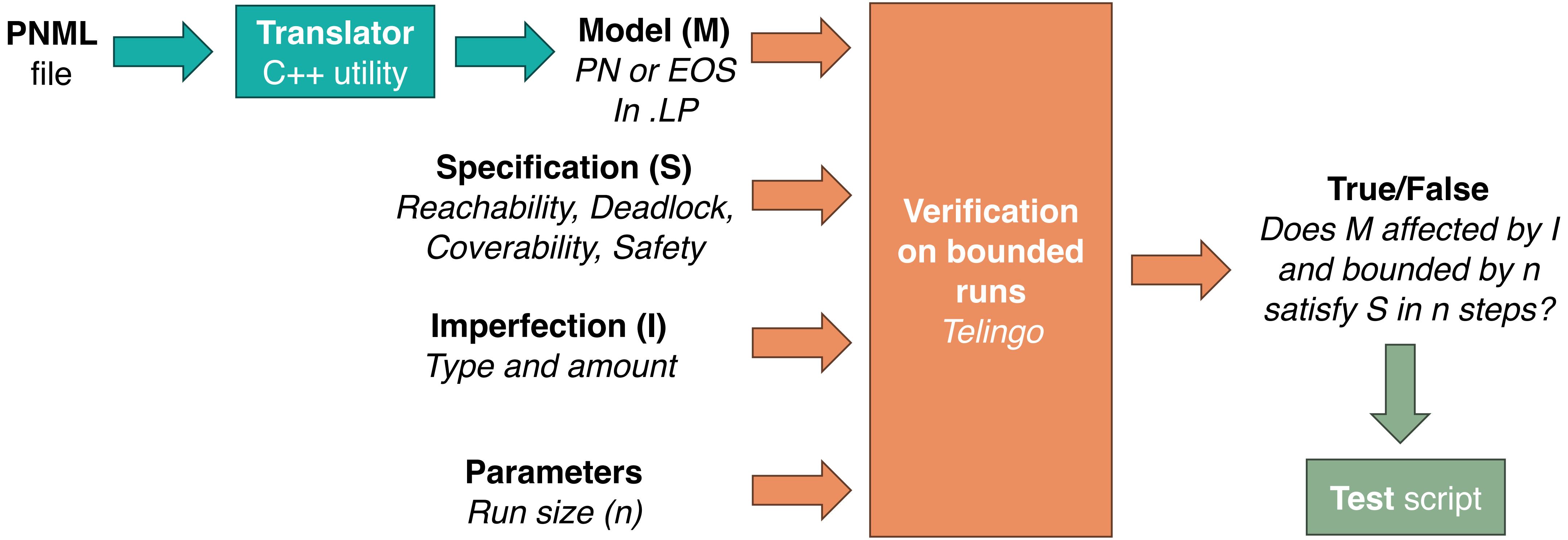
[Buß14] Köhler-Bußmeier, Michael. 'A Survey of Decidability Results for Elementary Object Systems'. 1 Jan. 2014 : 99 – 123.

Motivation

Current PN tools lack of

- EOS support
- Imperfection support
- Robustness analysis
- Partial verification

Prototype (CILC'24)



Why bounded verification?

Problems on general EOS	$\preccurlyeq_f$	$\preccurlyeq_o$	$\preccurlyeq_s$
0-reach	U	U	U
0-cover	U	U	U
ℓ-reach/cover	U	U	U
ω-reach/cover	D	U	U

F. Di Cosmo, S. Mal, T. Prince, *Deciding Reachability and Coverability in Lossy EOS*, PNSE'24

Why Telingo?

Telingo is **declarative** and supports **temporal constraints**

- E.g., `:- &tel(>? (lossy >(>? lossy)` allows at most one lossy step
- The meaning of lossy is declared orthogonally to EOS specification

Telingo **returns finite runs**

- Perfectly matches bounded verification

Encoding of PNs and EOSs is **elegant in ASP**

- E.g., when compared to SMT – R. Phawade, T. Prince, S. Sheerazuddin et al., *Bounded Model Checking for Unbounded Client Server Systems,* Arxiv (2022)

Correctness and performances

Correct answers

- Checked on MCC benchmarks

Slow on PNs

- Compared with Tapaal

Prohibitively slow on EOSs

- Nesting exacerbates grounding

problem	lossiness	−imax =5 (s)	−imax =10 (s)	−imax =20 (s)	TAPAAL (s)
deadlock	none	UNSAT in 0.052	SAT in 0.622	SAT in 82.754	SAT in $5e-6$
deadlock	any	SAT in 0.009	SAT in 0.010	SAT in 0.009	NA
1-safeness	none	UNSAT in 0.010	UNSAT in 0.014	UNSAT in 0.027	UNSAT in 0
1-safeness	any	UNSAT in 0.013	UNSAT in 0.018	UNSAT in 0.032	NA

Table 1
Comparative Results with TAPAAL for the Eratosthenes-PT-010 PN from the MCC benchmarks [12].

Papers:

1. Deciding Reachability and Coverability in Lossy EOS **(PNSE’24)**
2. Bounded Verification of Petri Nets and EOSs using Telingo: An Experience Report **(CILC’24)**

Future Work

1. Tackle undecidability via partial verification. For example:
    a. Runs of bounded length
    b. Probabilistic techniques
2. Consider other problems and other types of lossiness. For example:
    a. Model checking
    b. Reset lossiness
3. Test further applications. For example:
    a. Business processes
    b. Non-monotonic reasoning

35

Artifact: NWN Telingo Analyzer

NWN Telingo Analyzer on Zenodo

Translate PNs from PNML to ASP

Analyze robustness under lossiness

Replicate our tests