How to Use Polyhedral Reduction for the Verification of Petri nets.

Nicolas Amat, Bernard Berthomieu,
Silvano Dal Zilio, Didier Le Botlan

LAAS-CNRS, Toulouse

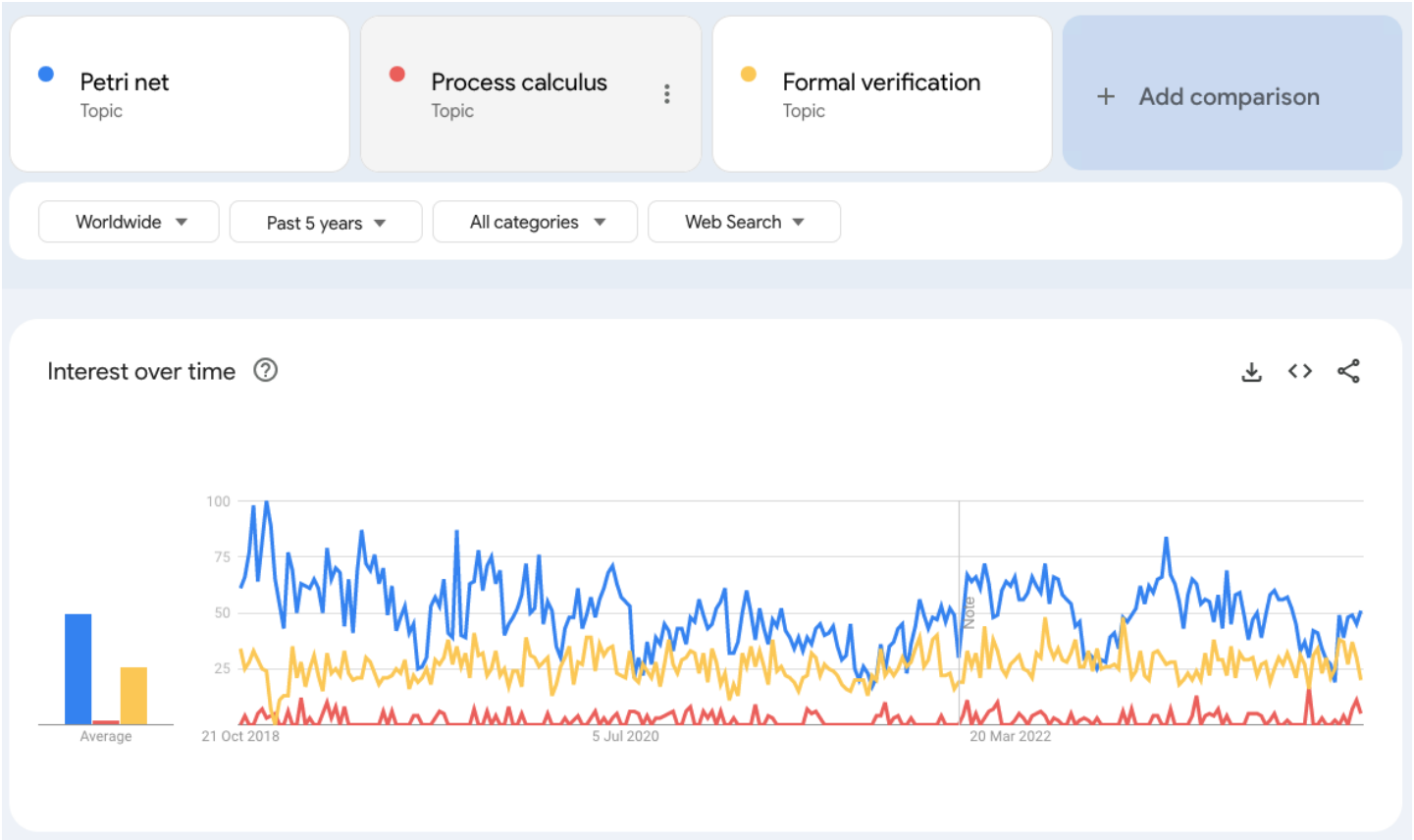
What is Polyhedral Reduction ?

. . . and how we use it to accelerate
the formal verification of Petri nets.

Nicolas Amat, Bernard Berthomieu,
Silvano Dal Zilio, Didier Le Botlan

LAAS-CNRS, Toulouse

Petri net is a trending subject



14ème colloque sur la Modélisation des Systèmes Réactifs (MSR'23)
LAAS-CNRS, Toulouse (France); 22–24 novembre 2023

. . . with theoretical interest

- **Equivalent** to Vector Addition Systems with States (**VASS**)
- Interesting complexity results:
  - Reachability is **decidable** [Mayr, 1981] [Kosaraju, 1982] [Lambert, 1992]
  - Even though the state space can be unbounded (see also the *coverability* problem),
  - But really difficult (**Ackermann-complete**) [Czerwinski et al., 2021] [Leroux,2021]
- Many tools are available: ITS-Tools, LoLA, Tapaal, **Tina**, KReach, FastForward, . . .

Motivation

- A strength of Petri net theory is the ability to reuse results from **linear algebra**, and linear programming techniques, to reason on it [Murata, 1989] [Silva et al., 1996]
  - **Potentially reachable markings**, aka the State Equation
  - **Place invariants**
  - ...
- There are also **structural analysis techniques**: implicit places; siphons and traps; structural boundedness; classes of nets (marked graphs, free-choice, ...)
- **Partial Order Reduction**: reduces the complexity brought by interleavings; can be applied to many different problems: use of symbolic techniques (BDD and SAT); model-checking LTL; ...

Motivation

We want to do something similar with **structural reductions**.

$\Rightarrow$ find an approach for using reductions, seamlessly, with different problems.

Net reduction [Berthelot 86]

A *reduction* is a net transformation which reduces its size such that (for a given set of properties) the reduced net is equivalent to the initial one.

$$(N, m_0) \equiv (N', m_0')$$

A reduction is characterized by:

- Application conditions.
- (Graph) transformation
- The preserved properties: boundedness; deadlock; quasi-liveness; reachability; ...

Polyhedral reduction

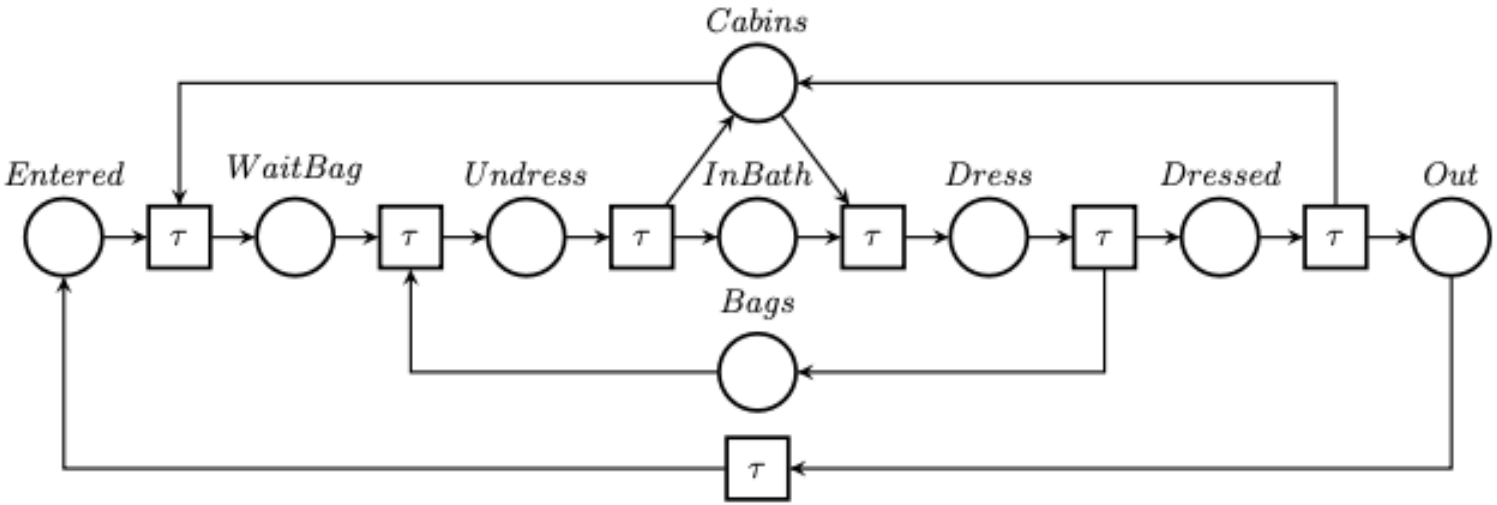
A polyhedral reduction is a net transformation which reduces its size such that we can reconstruct the state space of the initial net from the reduced one.

$$(N, m_0) \equiv_E (N', m_0')$$

A polyhedral reduction is characterized by:

- A set, E, of linear constraints between places.
- Application conditions.
- (Graph) transformation
- The preserved properties: ~~boundedness~~; ~~deadlock~~; ~~quasi-liveness~~; reachability; ...

Probably Reachable ≡ Reachable (PR ≡ R)



$$(N, m_0) \equiv_E \emptyset$$

where

$$E \triangleq \begin{cases} Cabins + Dress + Dressed + Undress + WaitBag = Cabins_0 \\ Dress + Dressed + Entered + InBath \\ \qquad + Out + Undress + WaitBag = Out_0 \\ Bags + Dress + InBath + Undress = Bags_0 \end{cases}$$

$(*)$ *result by Thomas Hujsa*

Outline

We consider three different problems.

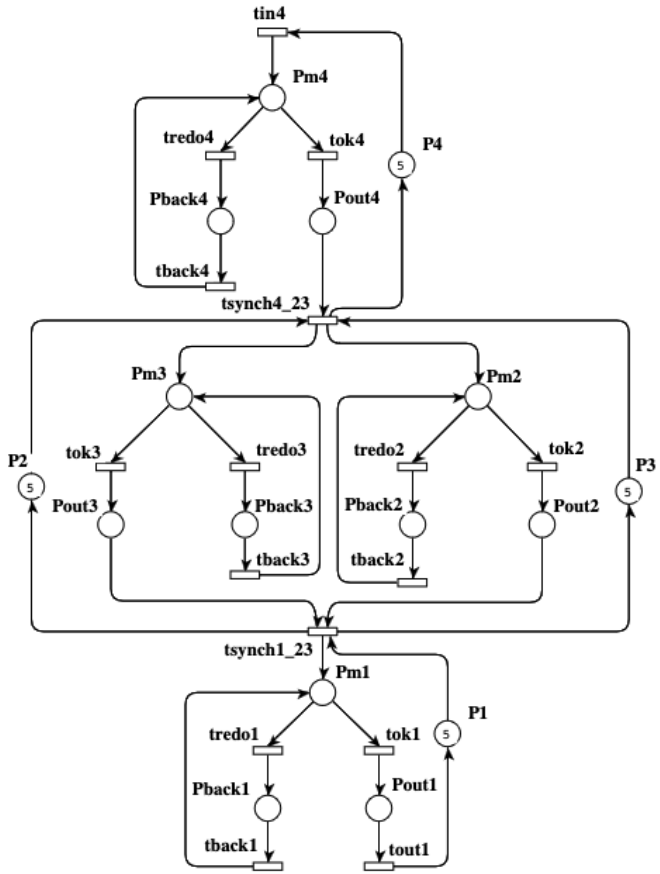
1. Use reductions efficiently with Decision Diagrams, for #SAT
2. Use reductions with SMT-based model checkers
3. What is the geometry of reduction equations

Benchmarks (Model Checking Contest)

Nothing would have been possible without the Model Checking Contest (MCC, https://mcc.lip6.fr/).

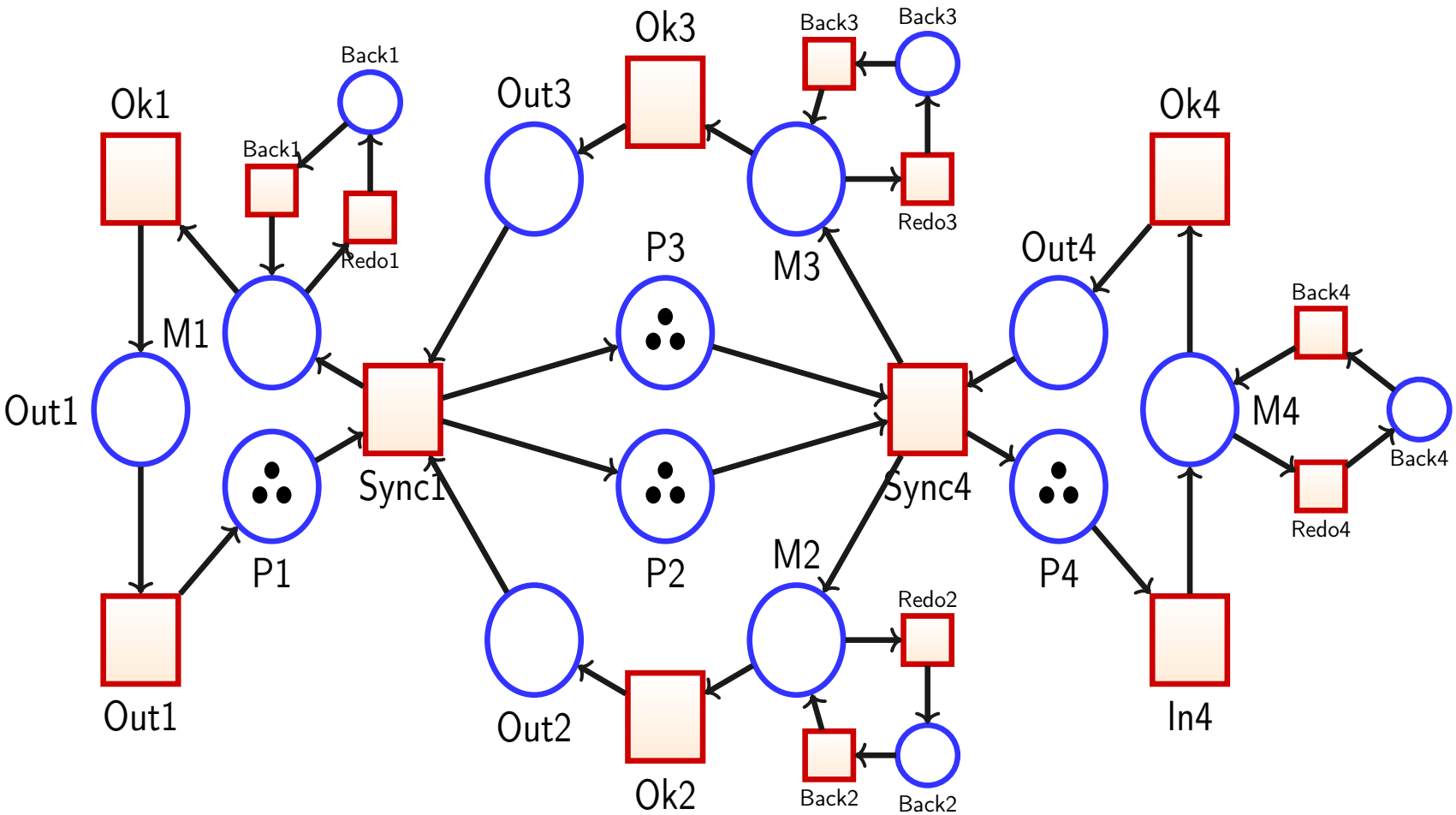
- The reason we got here.
- The reason we have tool-oriented results.
- A great source of model instances, $\approx$ 130 instances / 1 700 nets
- Also a source of reachability formulas, $\approx$ 50 000 queries
  . . . but random (☹)

Reduction equations

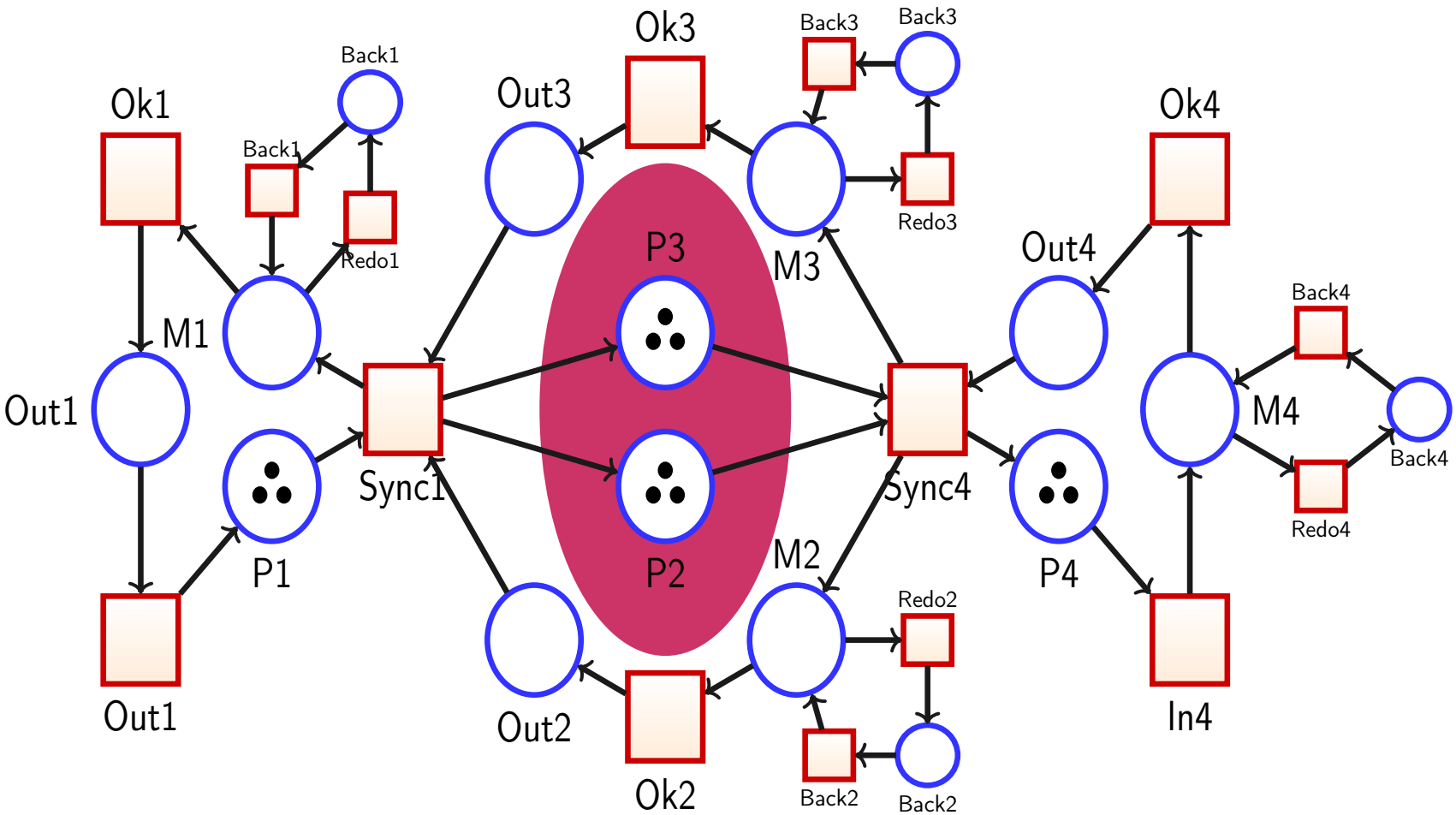


Kanban-PT-0003 (benchmark for // model-checking)

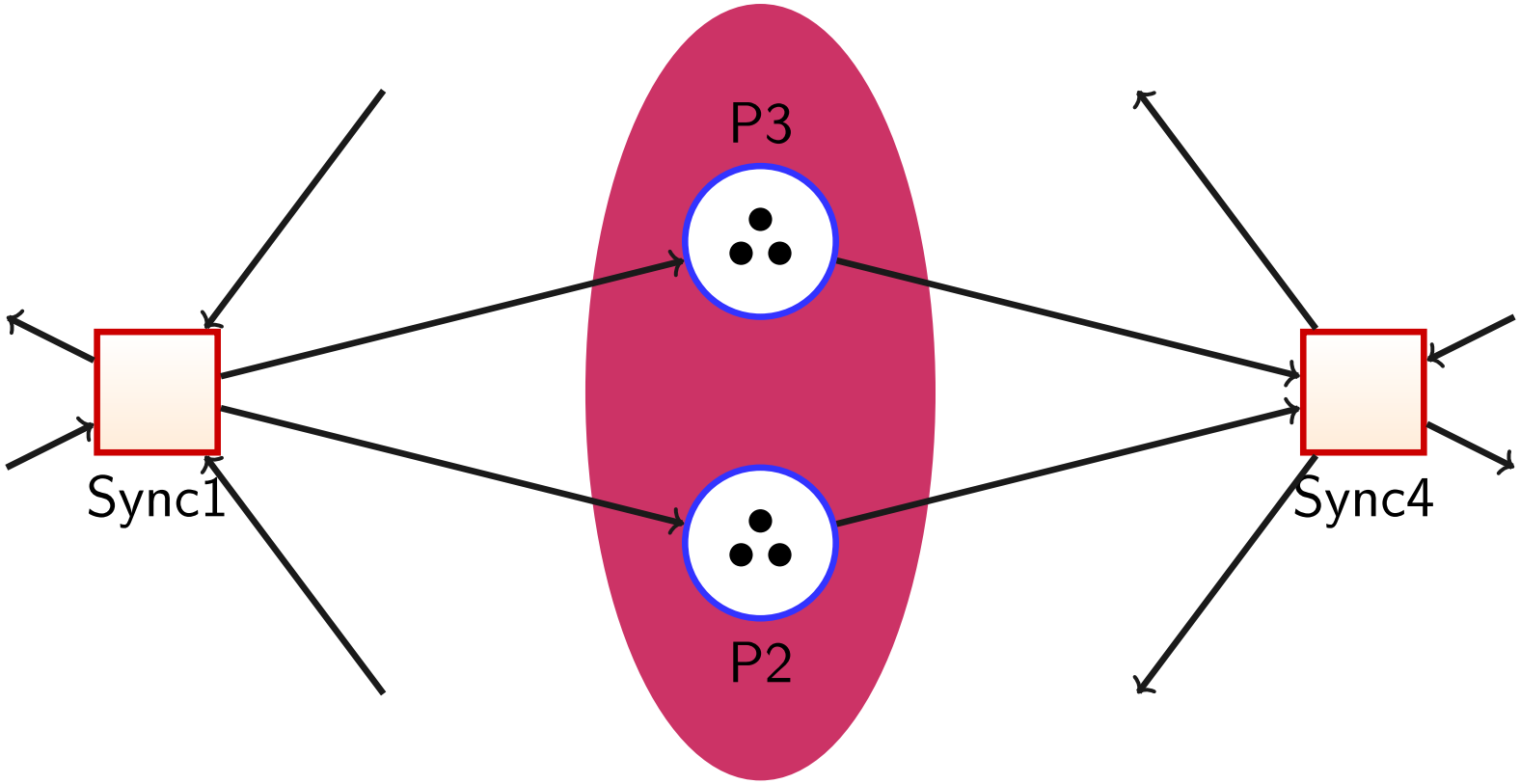
Running example: model Kanban with $n = 3$



Running example: model Kanban with $n = 3$

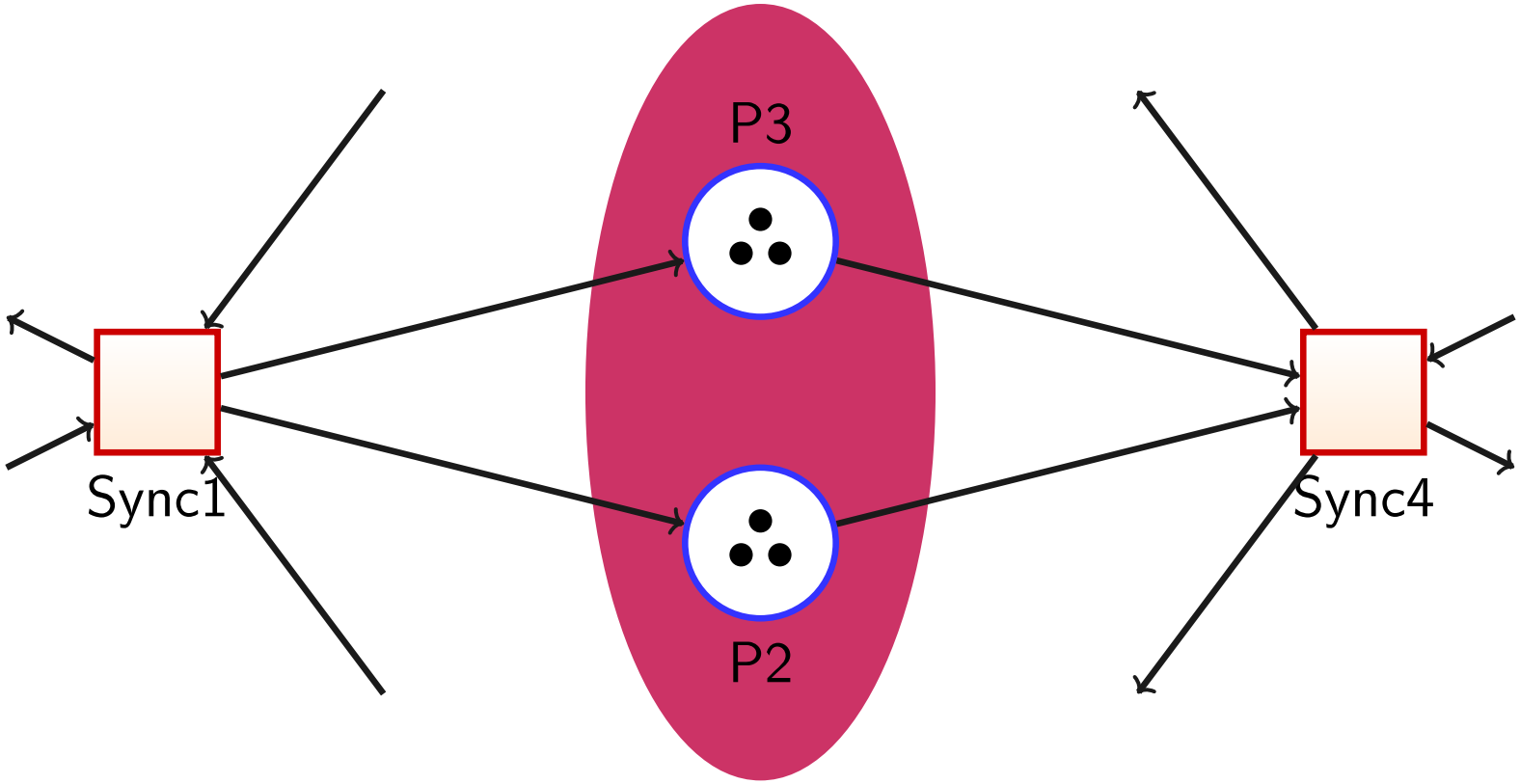


Redundant places (see [Berthelot 86])



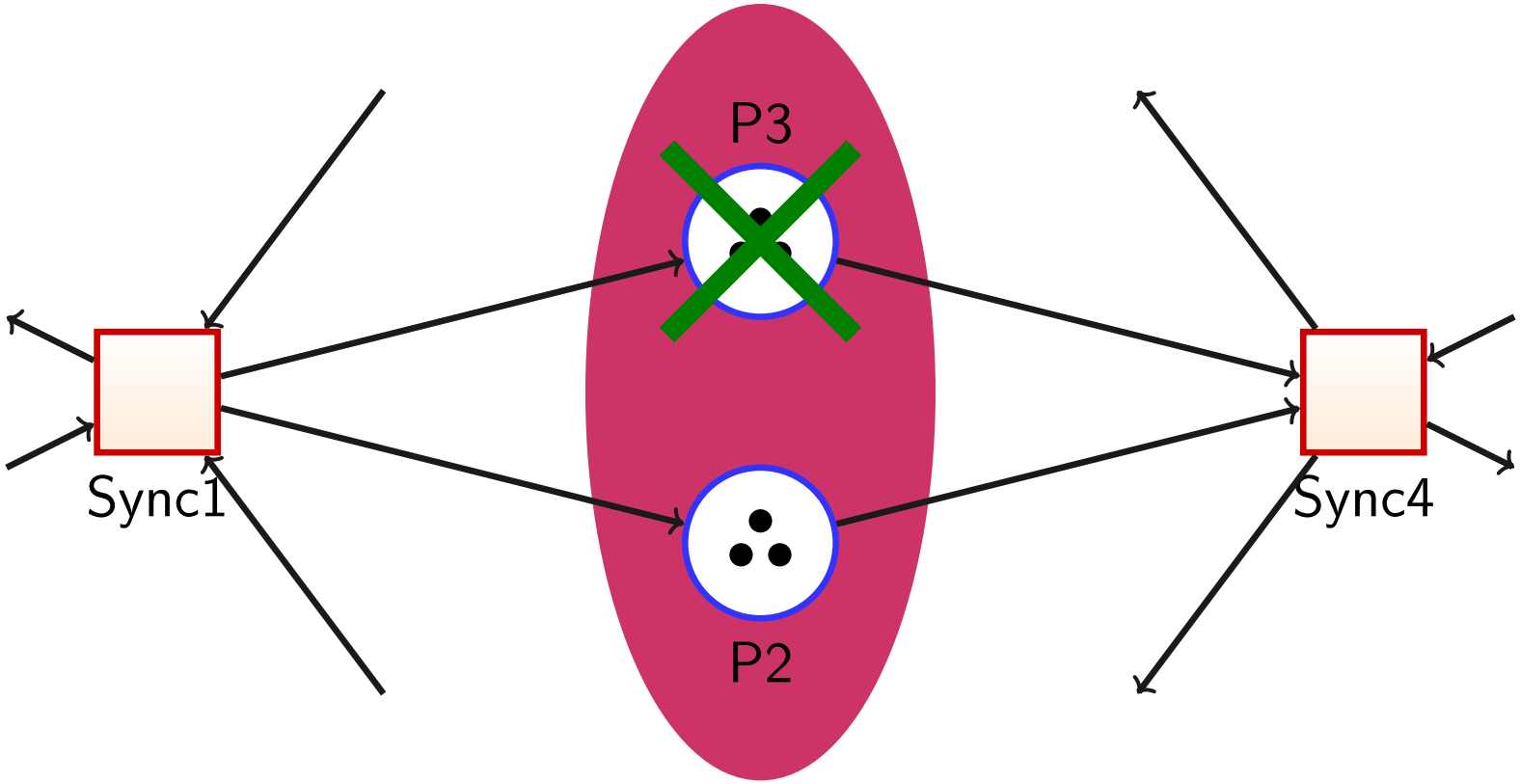
- P_2 and P_3 have the same incoming arcs and outgoing arcs, to the same transitions.

Redundant places (see [Berthelot 86])



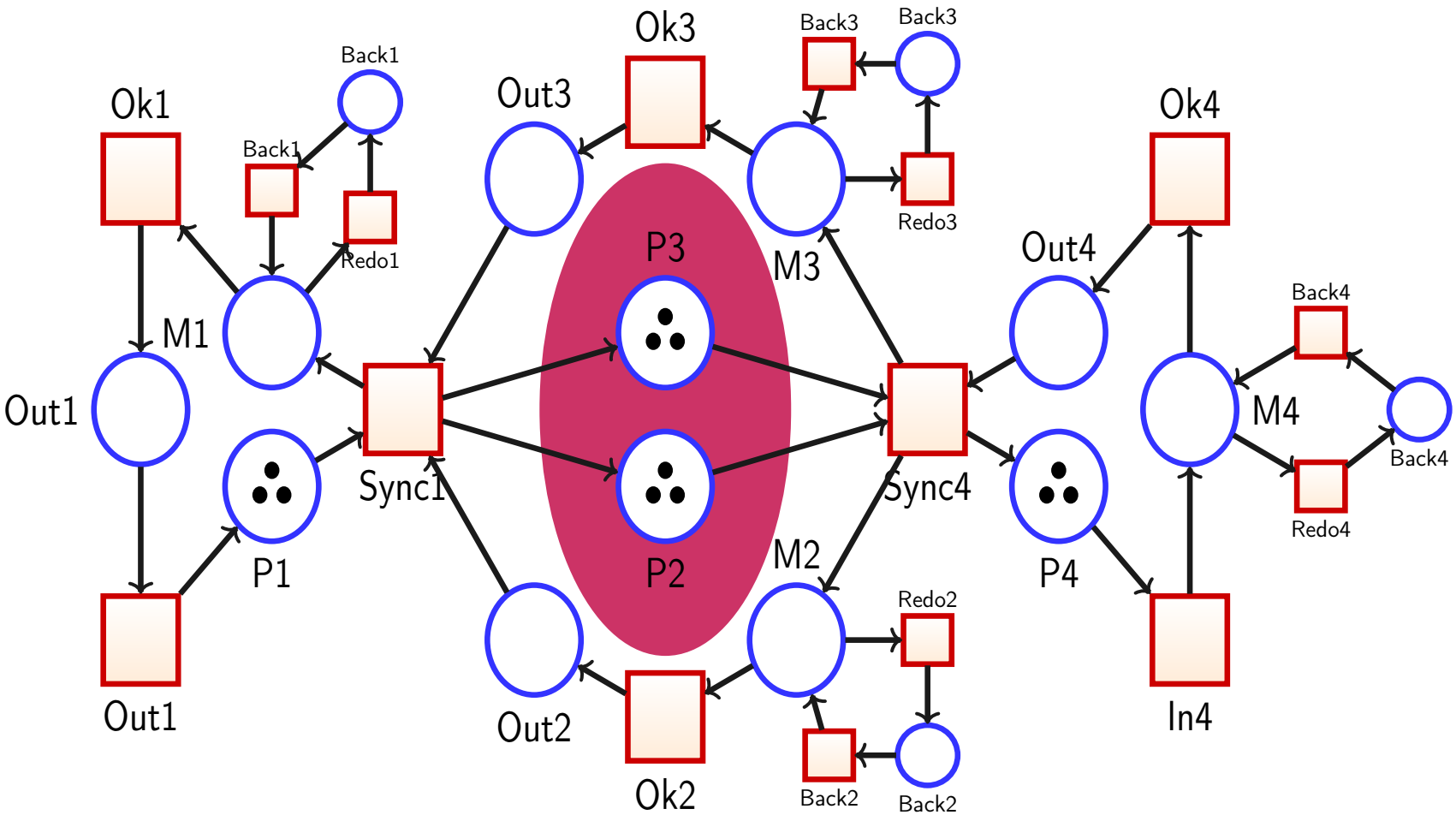
- P_2 and P_3 have the same incoming arcs and outgoing arcs, to the same transitions.
- $P_3 = P_2$ in the initial marking.

Redundant places (see [Berthelot 86])

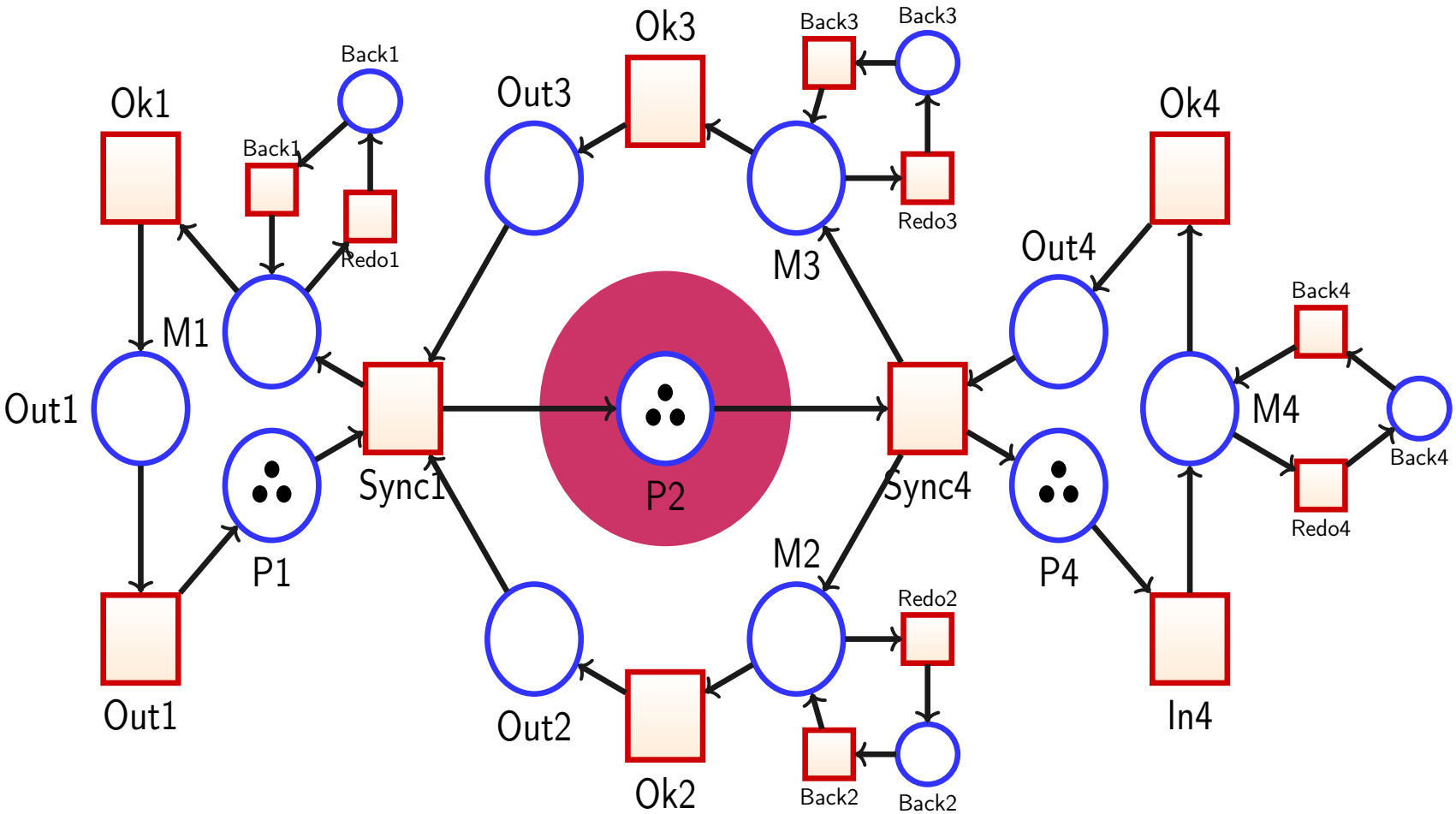


- P_2 and P_3 have the same incoming arcs and outgoing arcs, to the same transitions.
- $P_3 = P_2$ in the initial marking.
- Hence $P_3 = P_2$ is an invariant.

Running example: model Kanban with $n = 3$



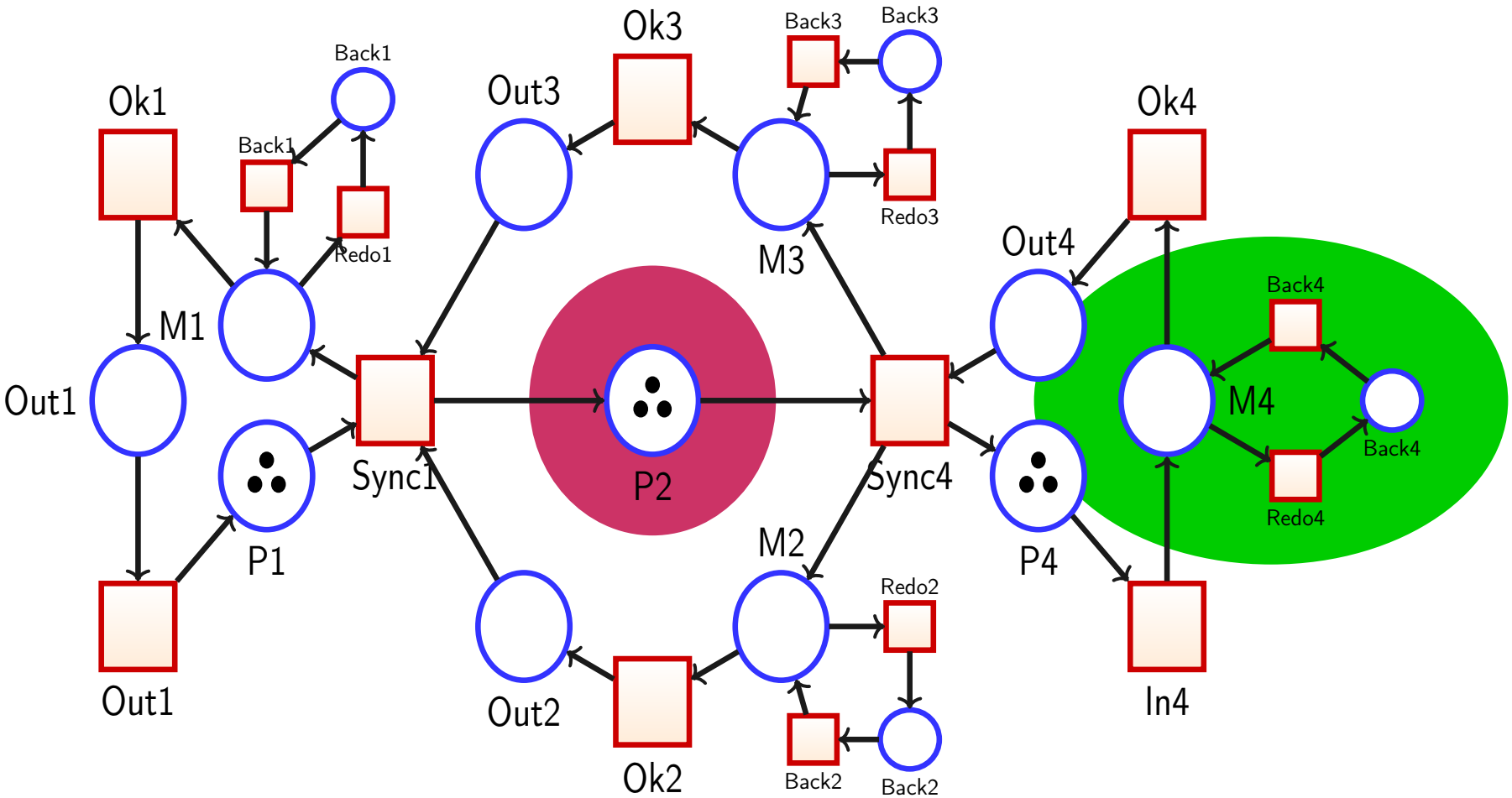
Removing redundant places



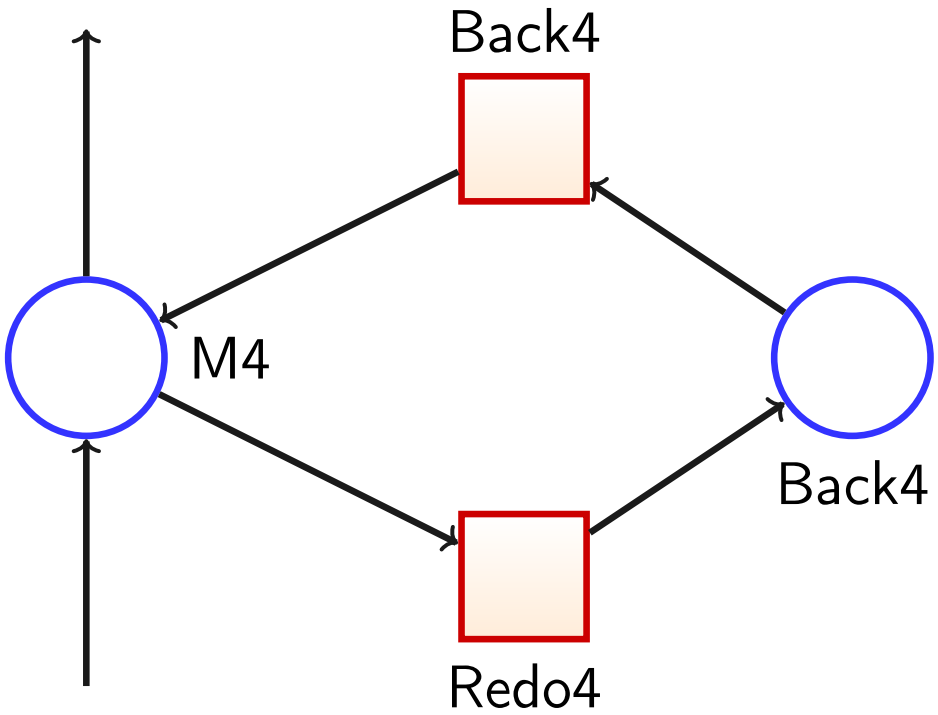
We record the relation $R \vdash P_3 = P_2$

We remove place P_3 from the net.

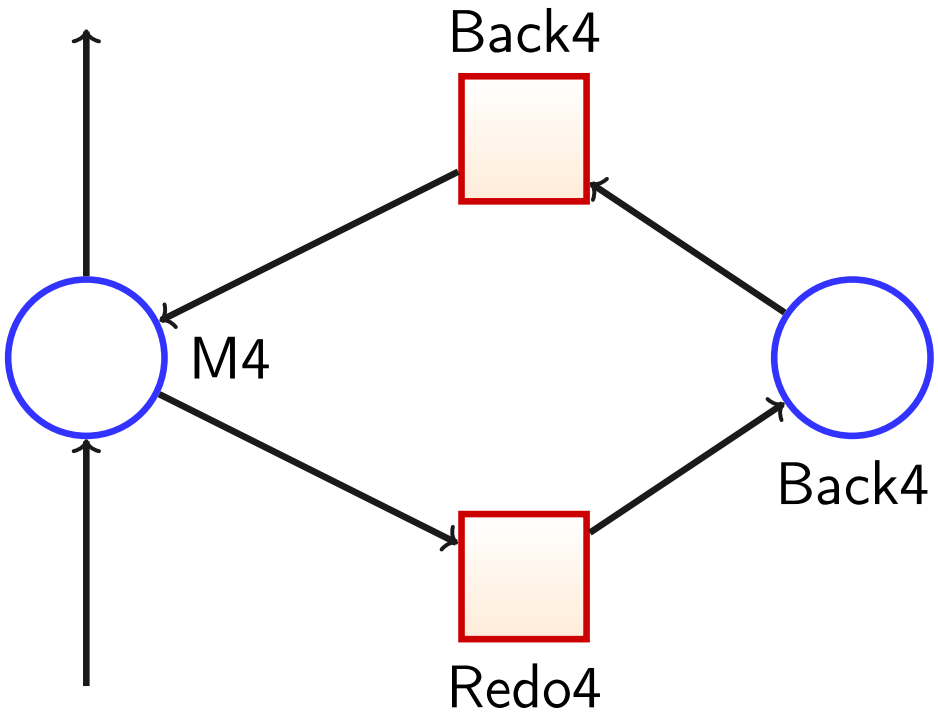
Removing redundant places



Loop agglomerations



Loop agglomerations

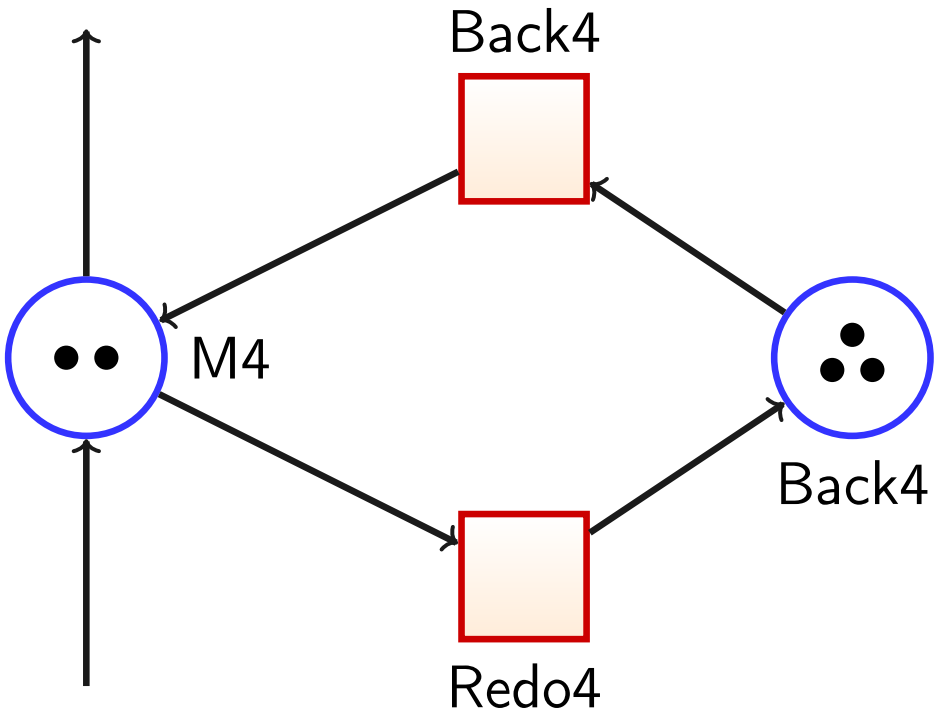


Loop agglomeration :

- A loop of n places and n transitions.
- Each transition has exactly one input and one output.

  (No constraint on the places)

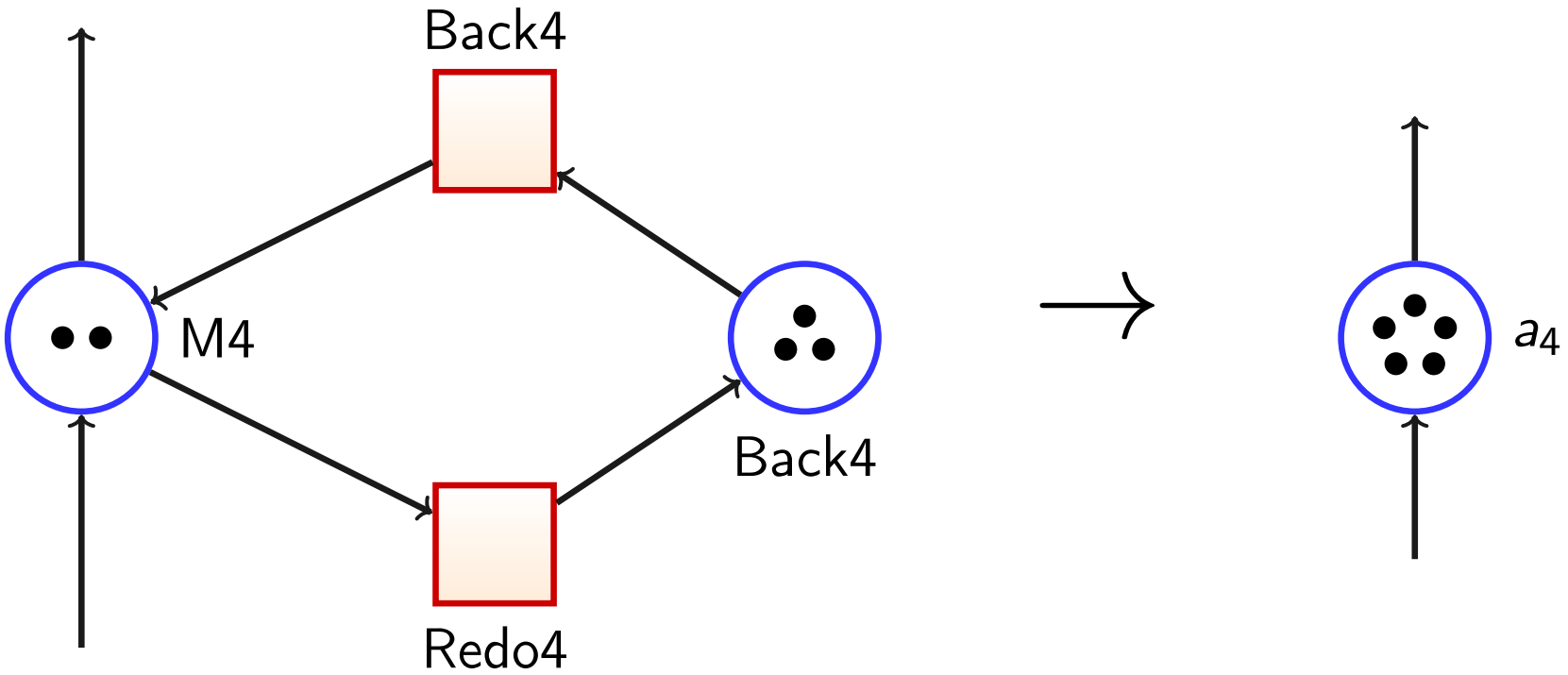
Loop agglomerations



Loop agglomeration :

- When the transitions of the loop are fired, the amount $x = M_4 + Back_4$ is unchanged.
- Given x, all the solutions of $x = M_4 + Back_4$ are reachable.

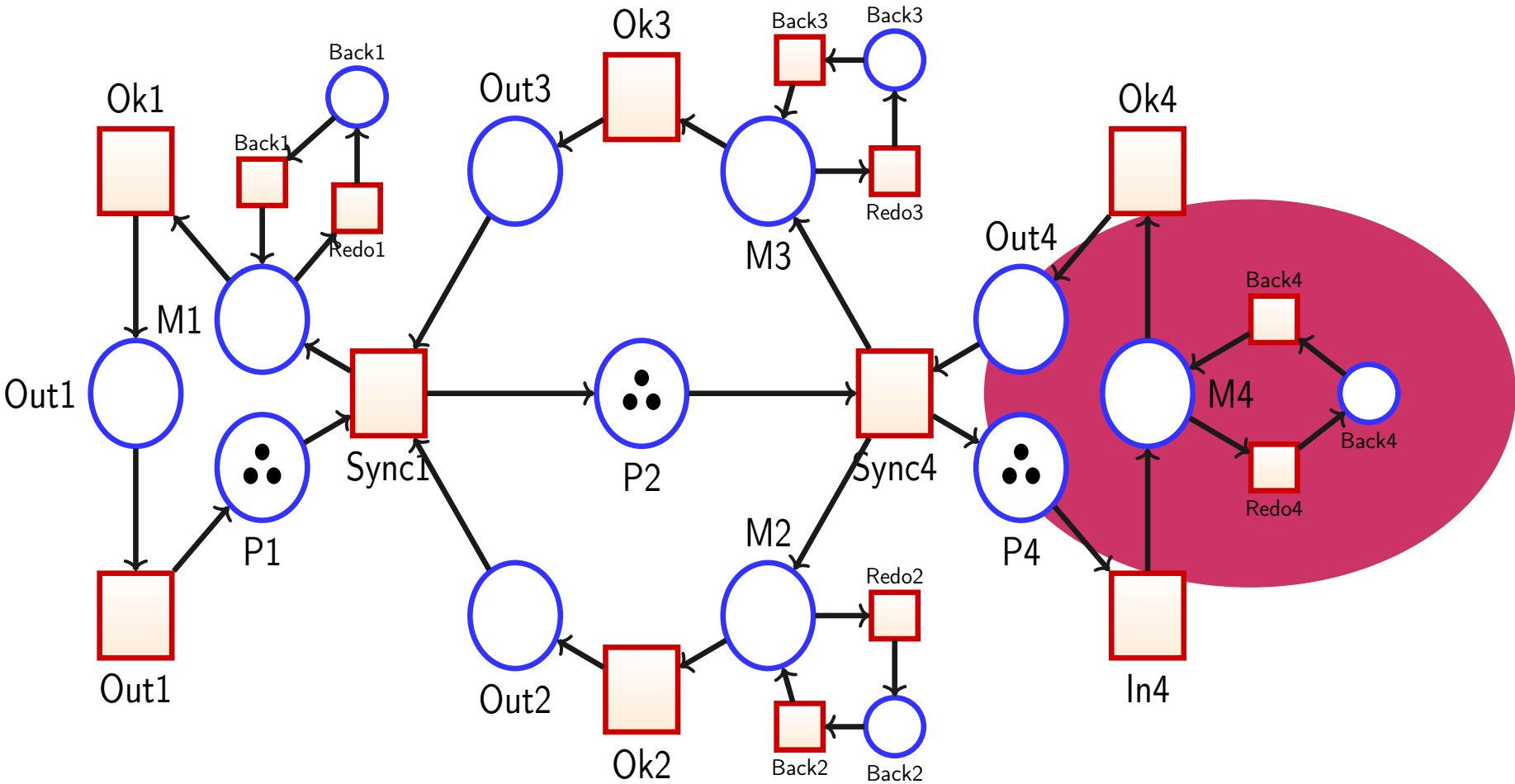
Loop agglomerations



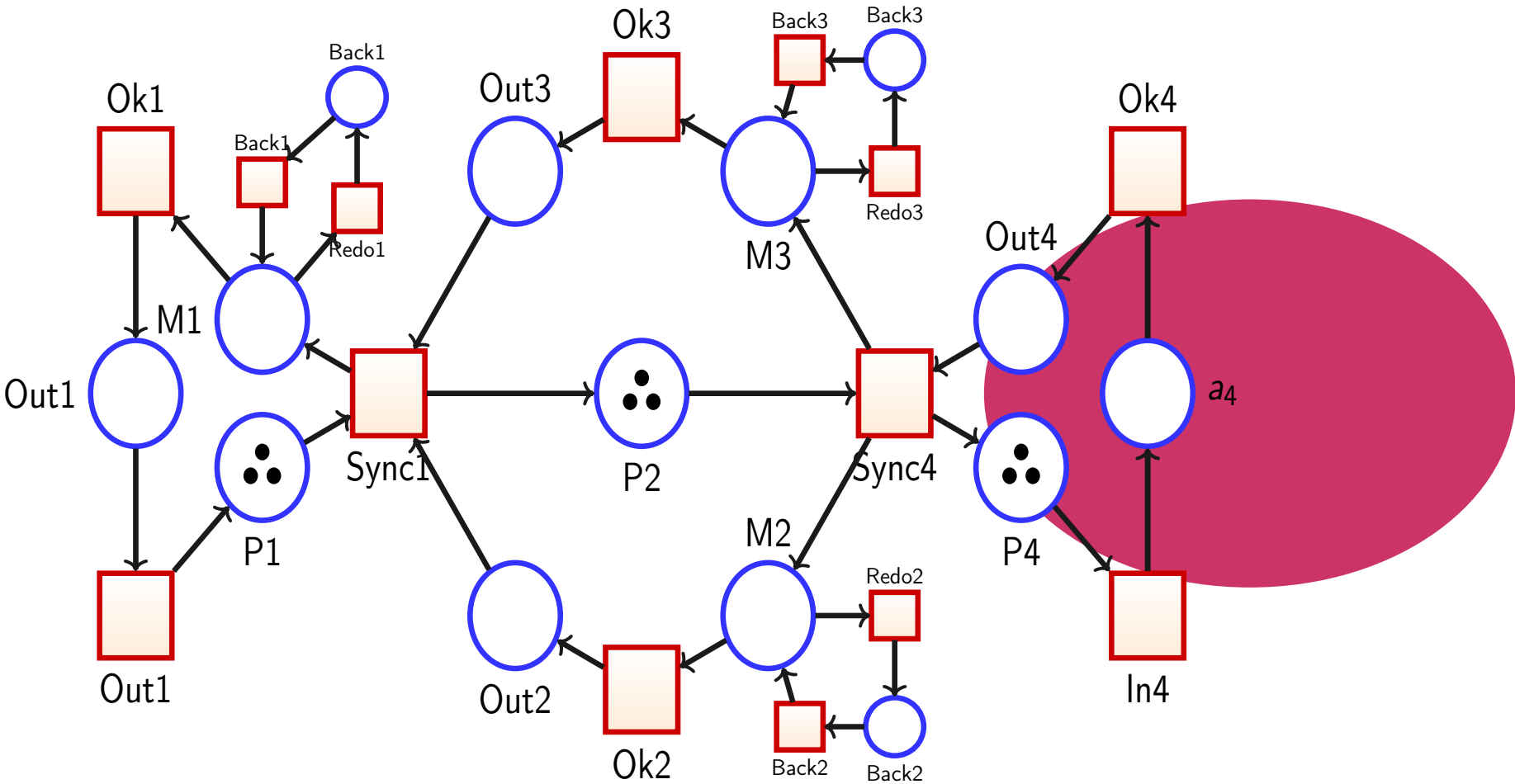
Loop agglomeration :

- The loop is agglomerated into a single place a_4.
- We keep the equation $A \vdash a_4 = M_4 + Back_4$
  (the annotation $A \vdash$ means Agglomeration)

Loop agglomerations

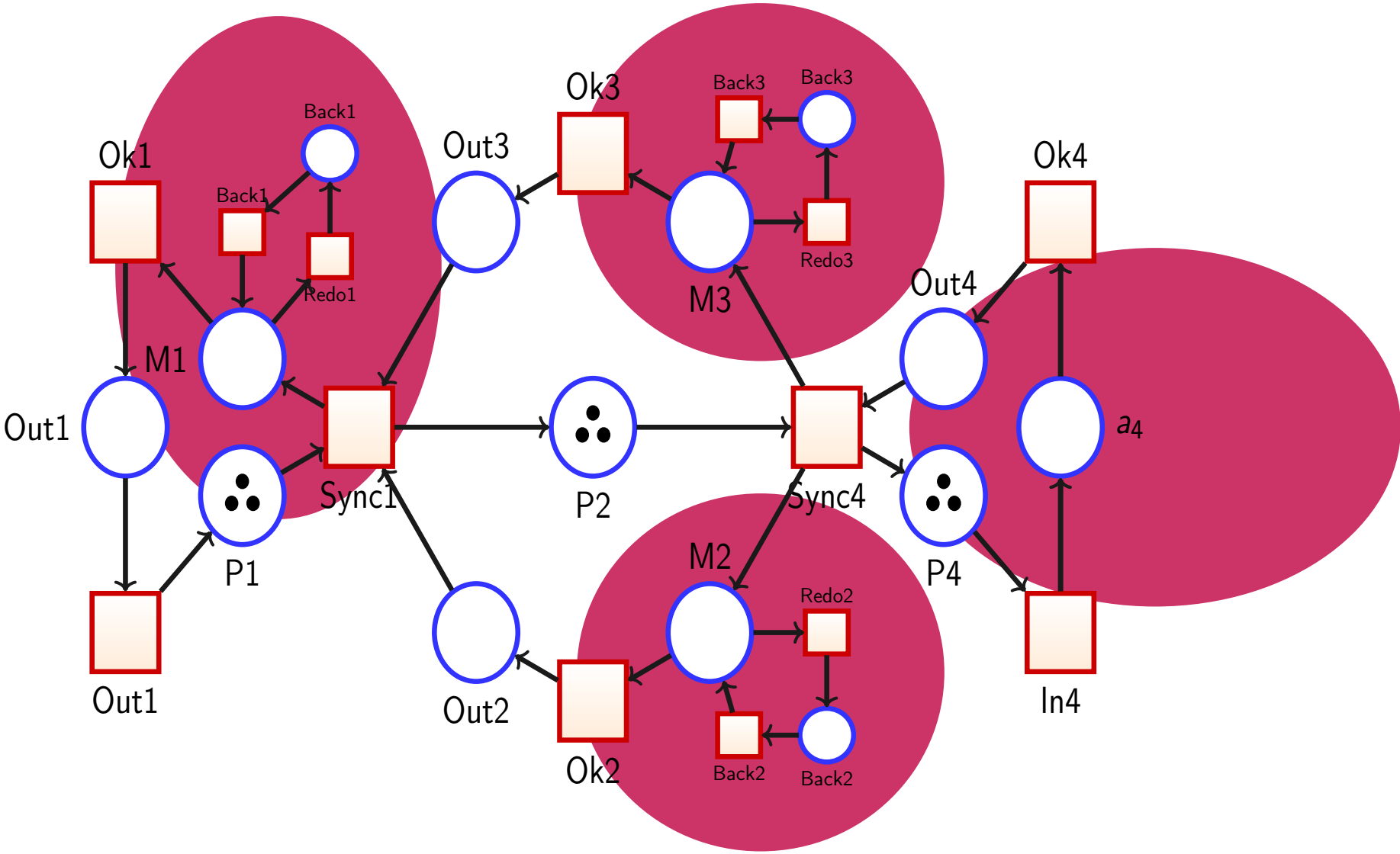


Loop agglomerations



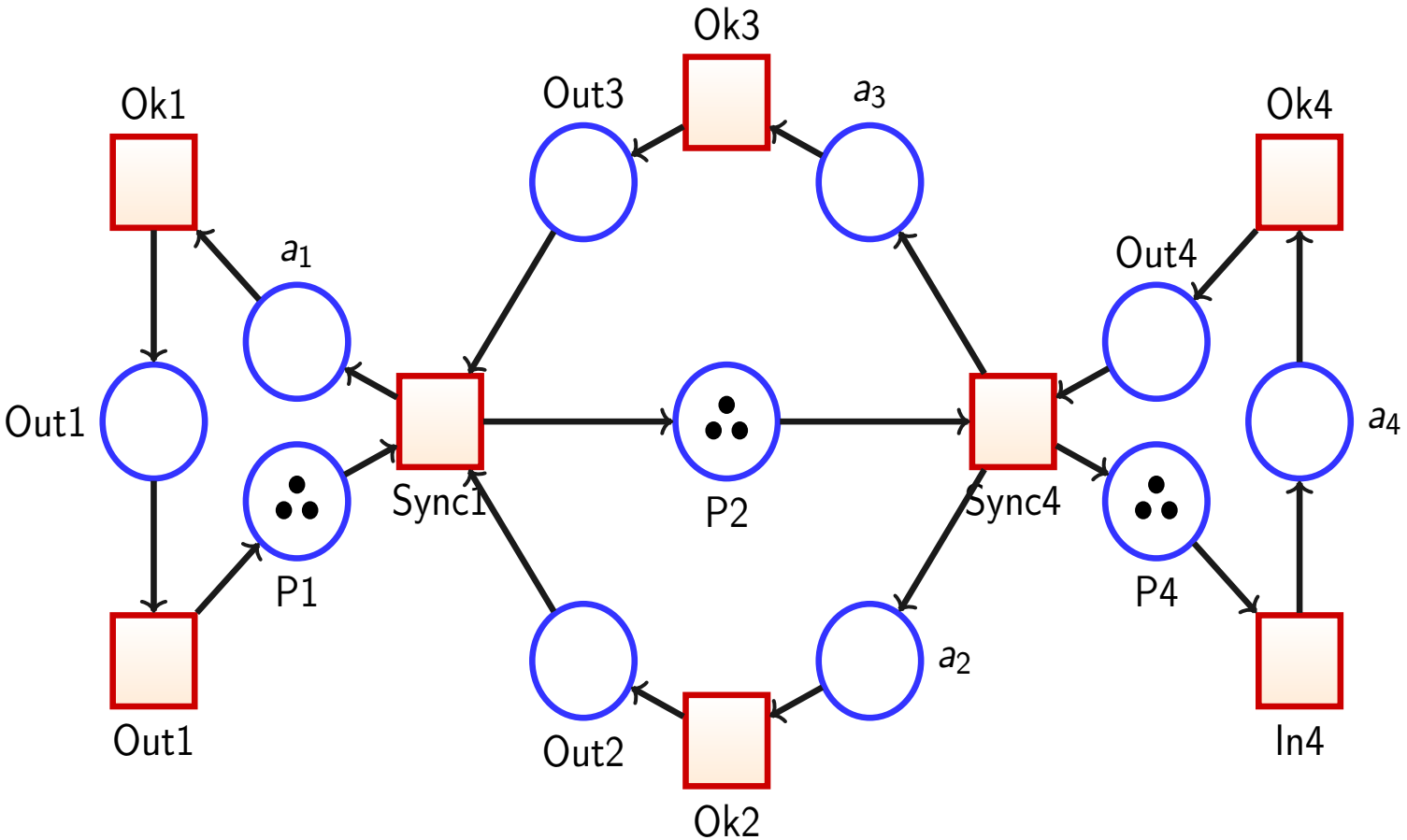
$$A \vdash a_4 = M_4 + Back_4$$

Loop agglomerations



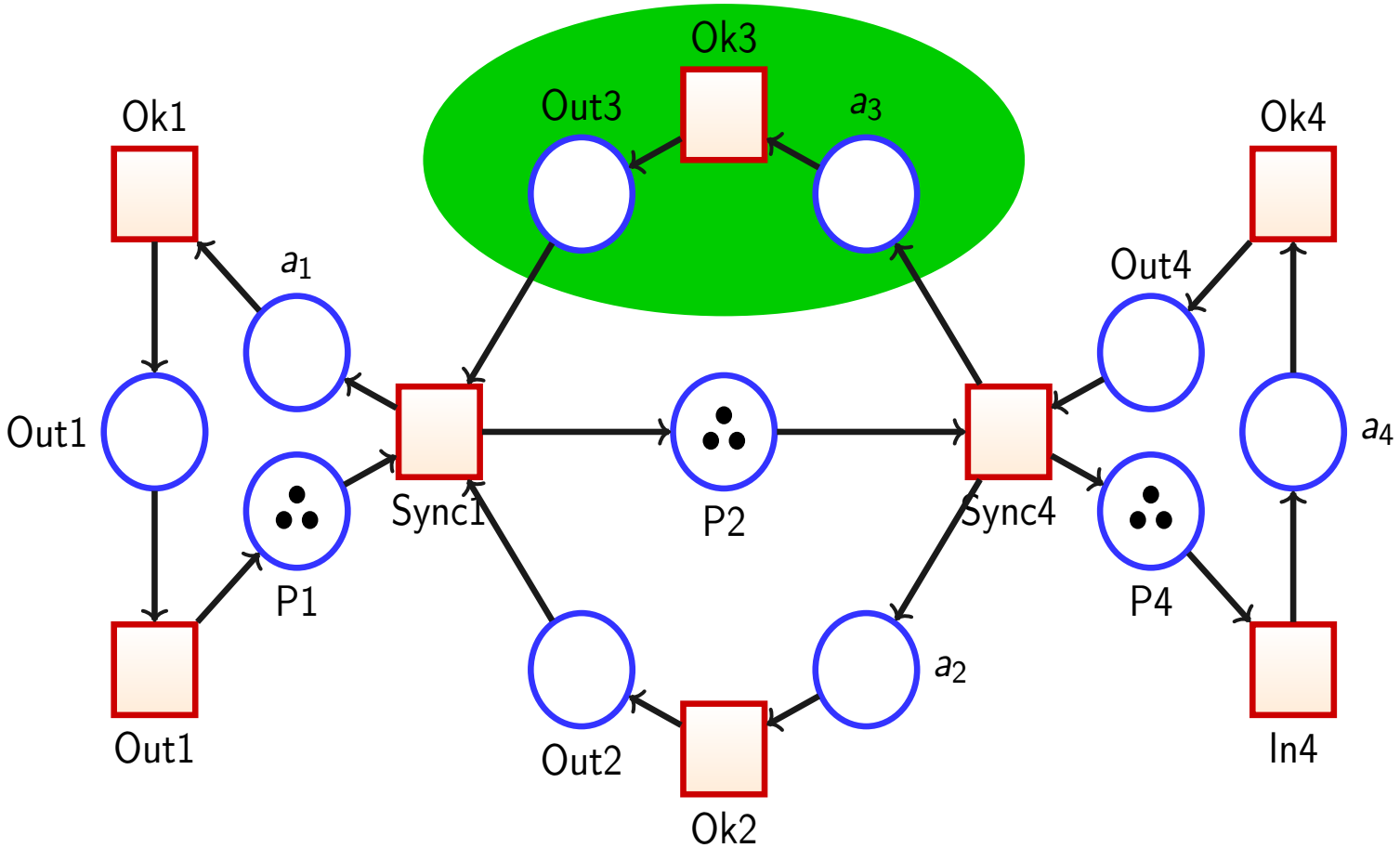
$$A \vdash a_4 = M_4 + Back_4$$

Loop agglomerations



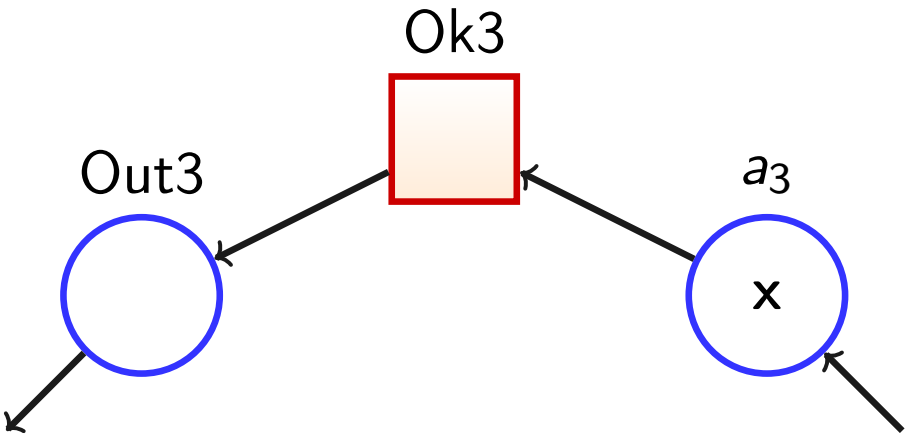
$$A \vdash a_4 = M_4 + Back_4 \qquad A \vdash a_2 = M_2 + Back_2$$
$$A \vdash a_1 = M_1 + Back_1 \qquad A \vdash a_3 = M_3 + Back_3$$

Loop agglomerations



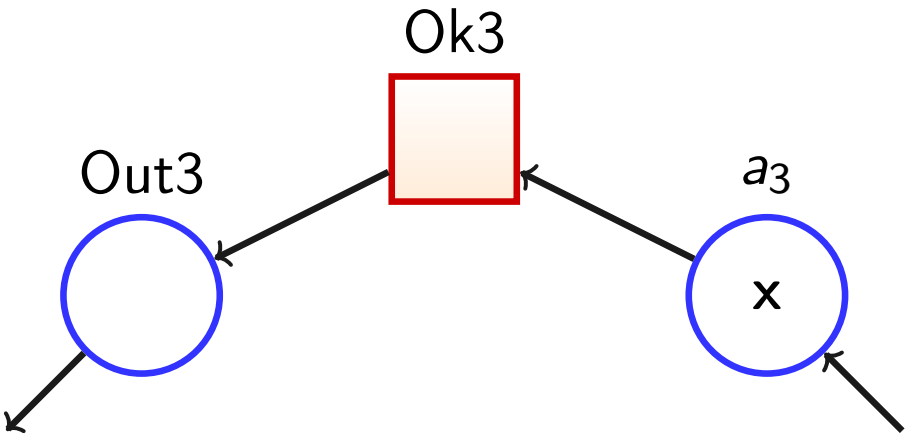
$$A \vdash a_4 = M_4 + Back_4 \qquad A \vdash a_2 = M_2 + Back_2$$
$$A \vdash a_1 = M_1 + Back_1 \qquad A \vdash a_3 = M_3 + Back_3$$

Chain agglomerations



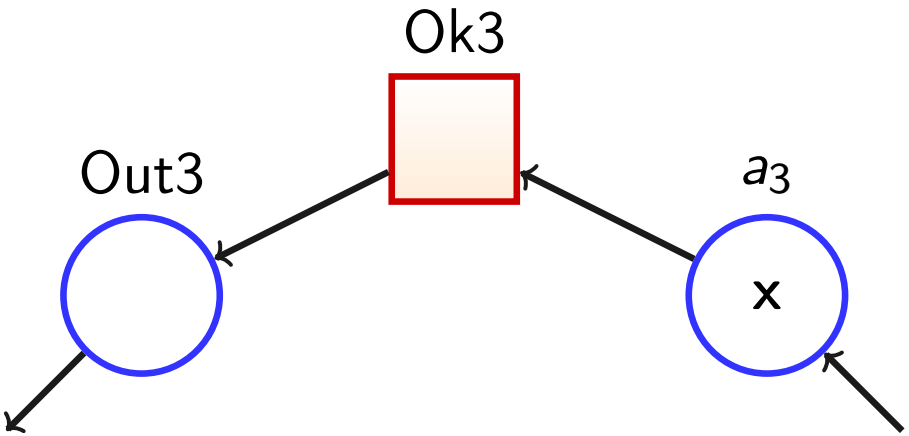
- In the initial state, Out_3 is unmarked.

Chain agglomerations



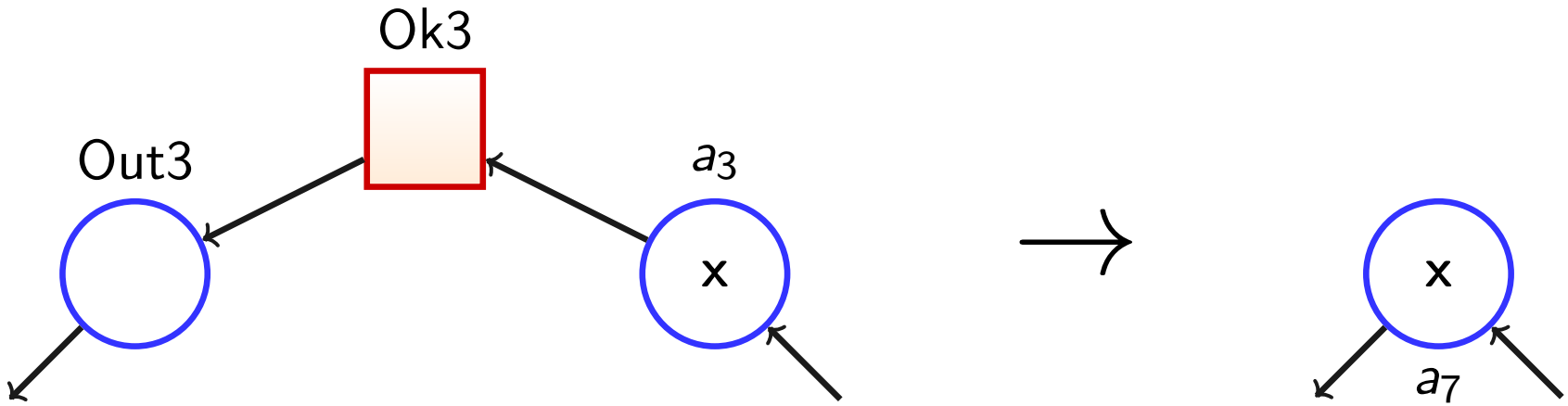
- In the initial state, Out_3 is unmarked.
- Transition Ok_3 only transfers tokens from a_3 to Out_3.

Chain agglomerations



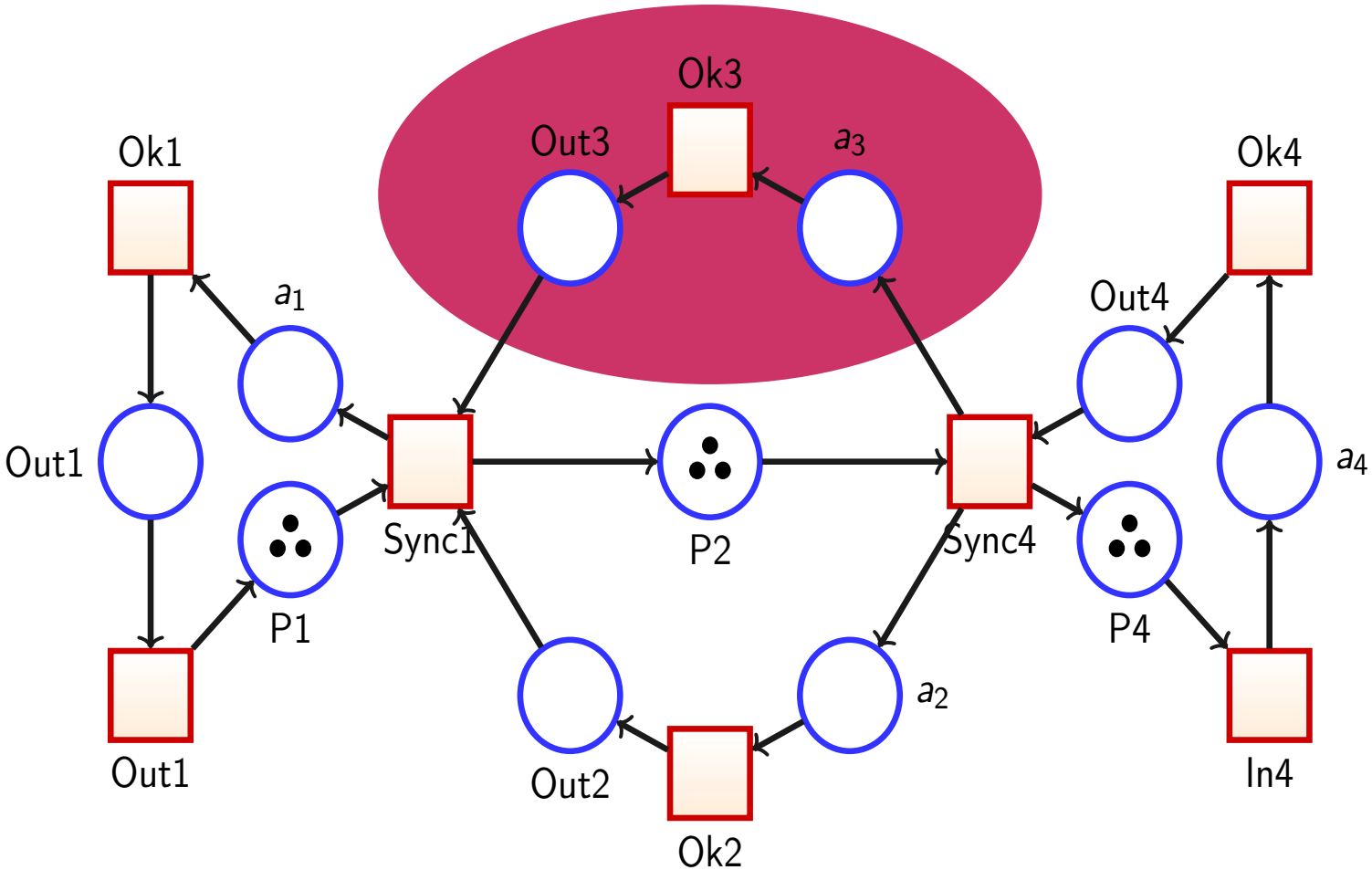
- In the initial state, Out_3 is unmarked.
- Transition Ok_3 only transfers tokens from a_3 to Out_3.
- OK_3 is the only incoming transition of Out_3.

Chain agglomerations

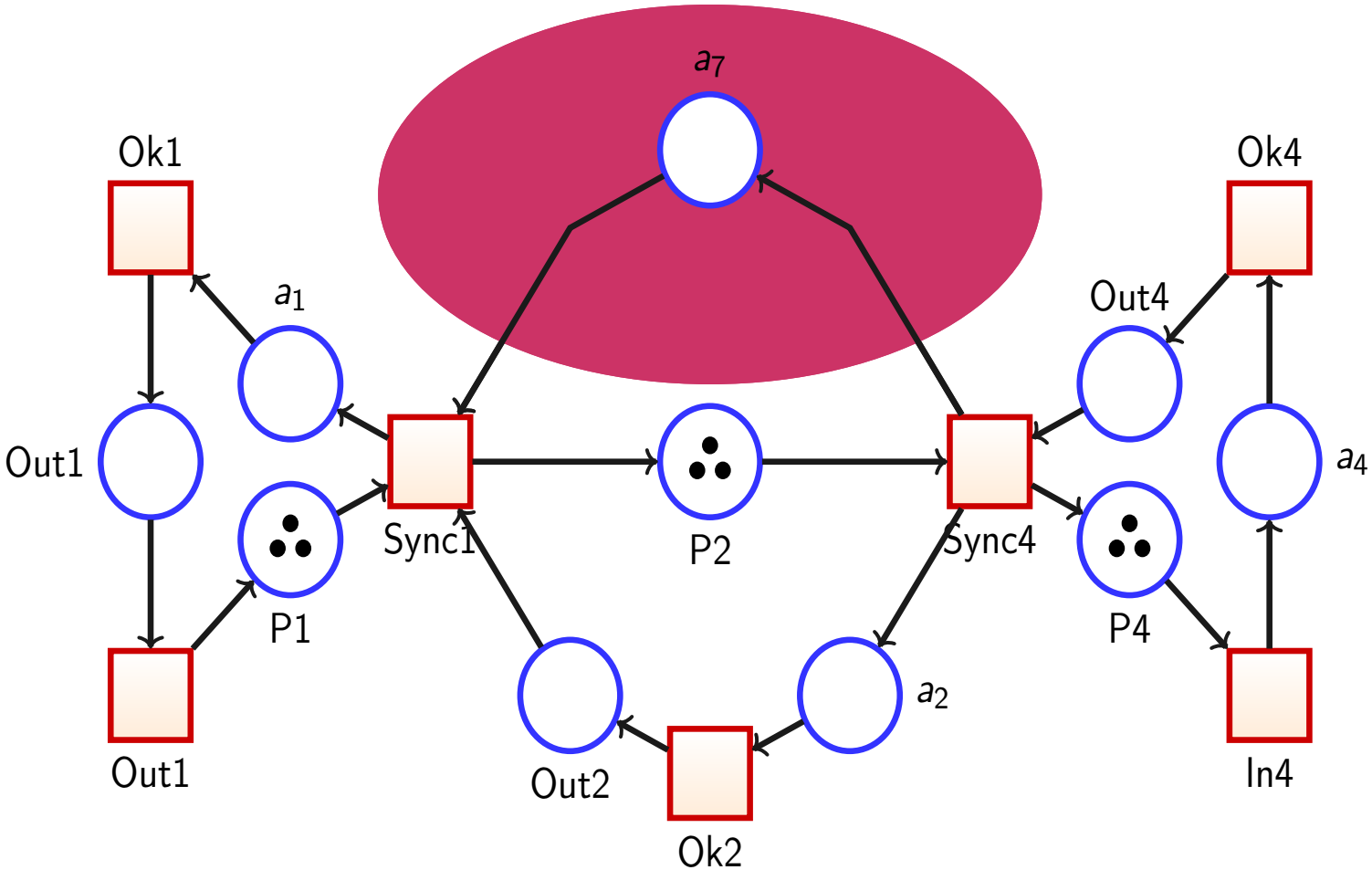


- We agglomerate Out_3 and a_3 into a_7.
- and keep the equation $A \vdash a_7 = Out_3 + a_3$

Chain agglomerations

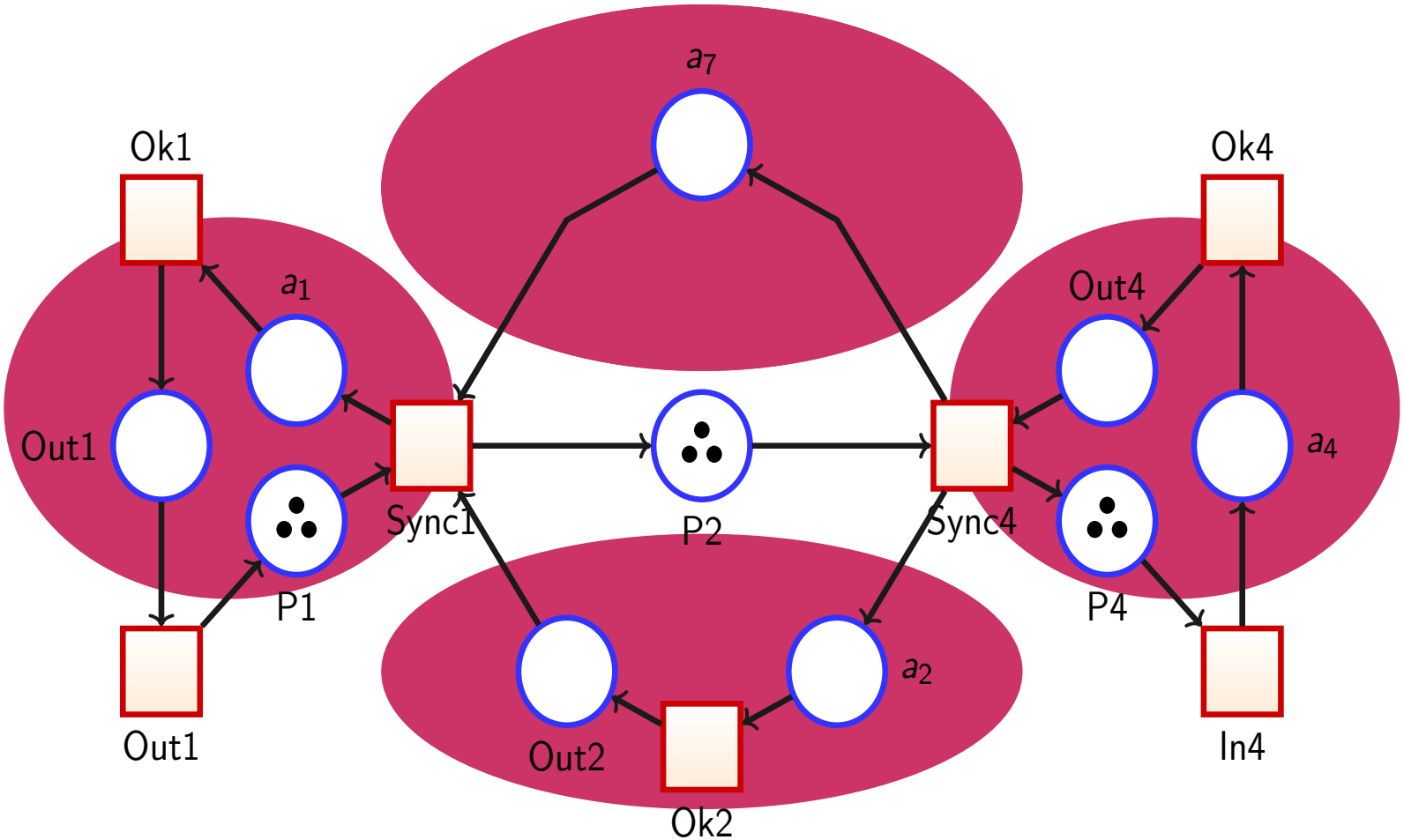


Chain agglomerations



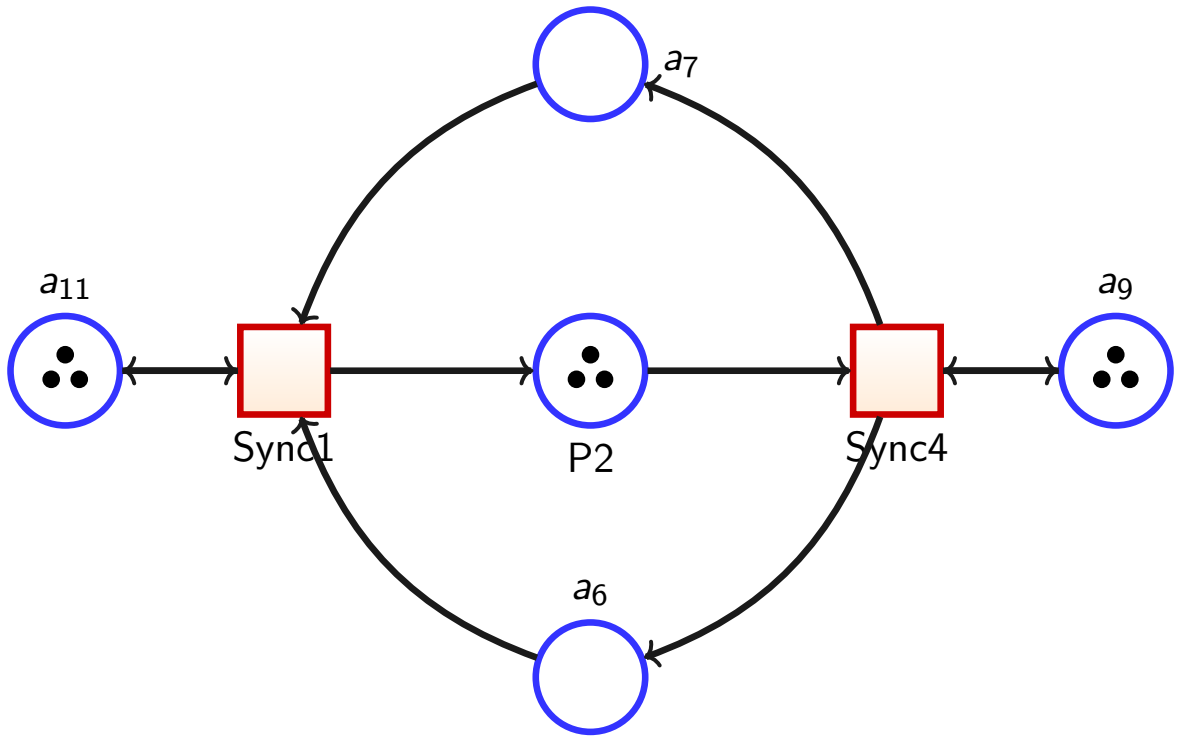
$$A \vdash a_7 = Out_3 + a_3$$

Chain agglomerations



$$A \vdash a_7 = Out_3 + a_3$$

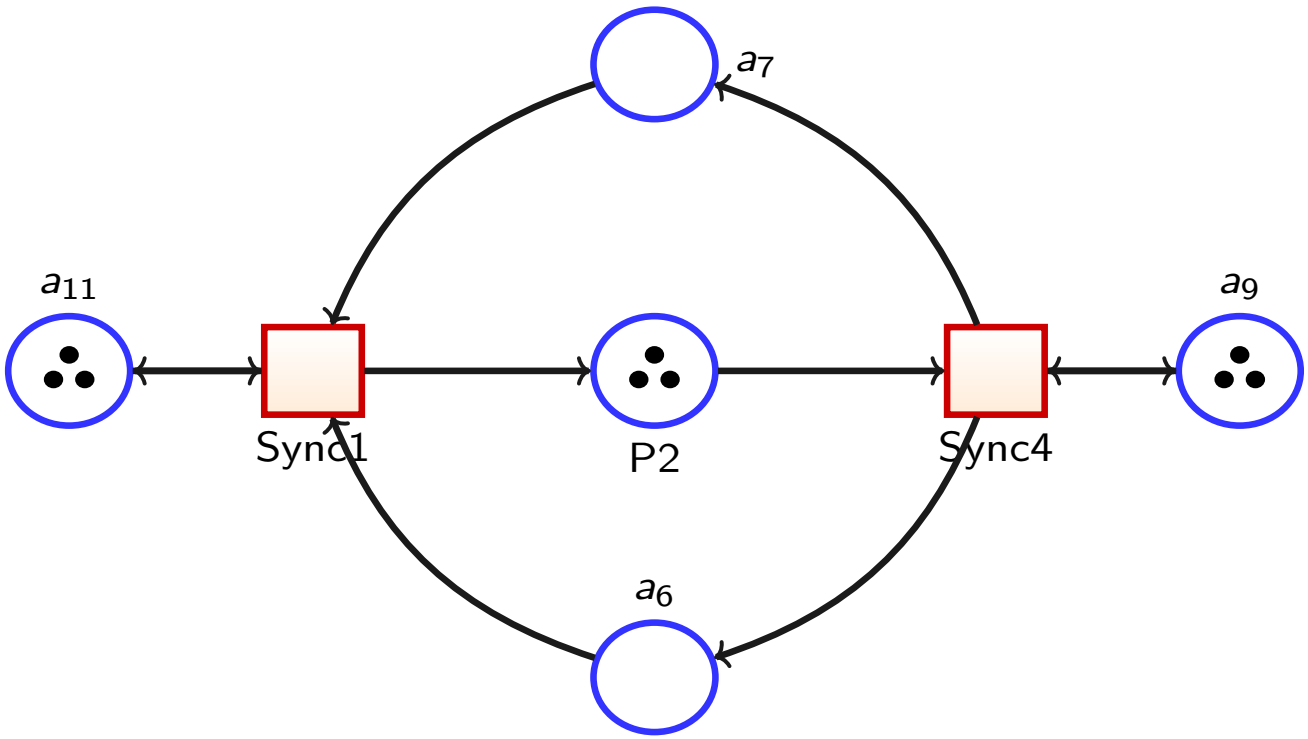
Chain agglomerations



$$A \vdash a_7 = Out_3 + a_3$$
$$A \vdash a_5 = Out_1 + a_1$$
$$A \vdash a_9 = a_8 + P_4$$

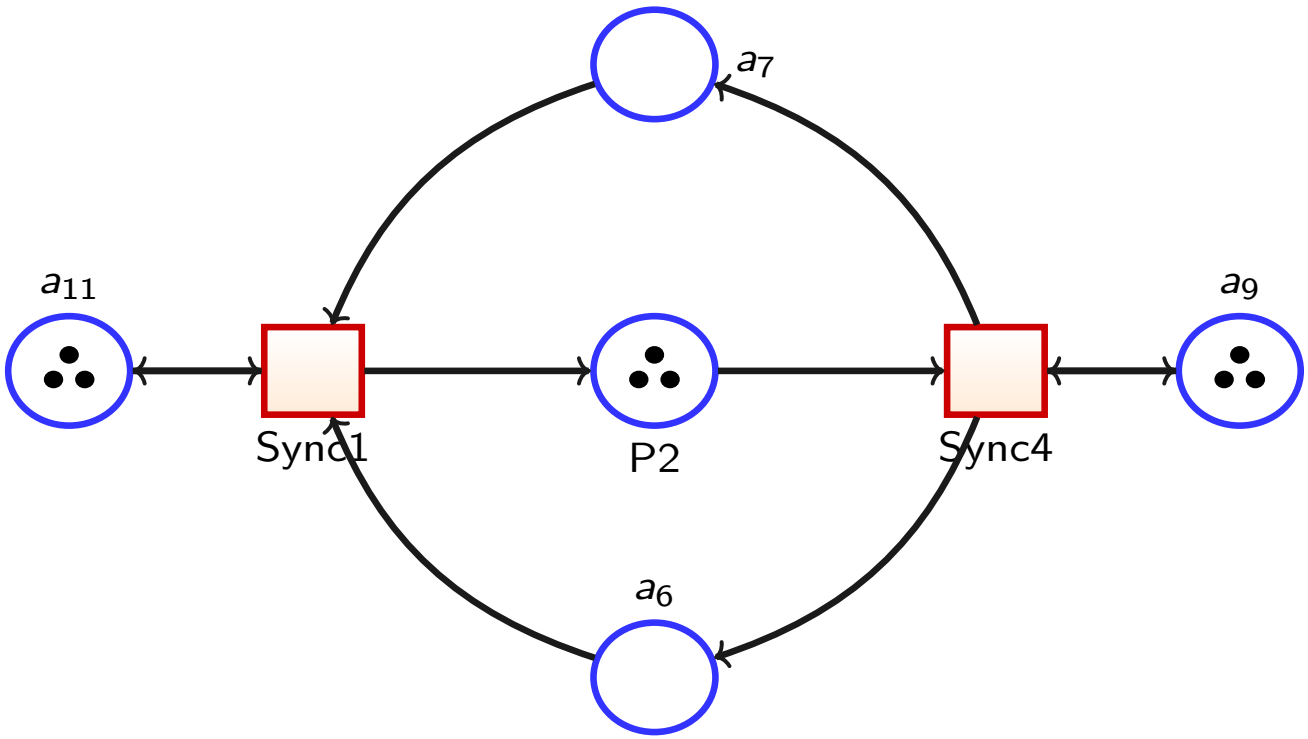
$$A \vdash a_6 = Out_2 + a_2$$
$$A \vdash a_8 = Out_4 + a_4$$
$$A \vdash a_{11} = a_5 + P_1$$

Reductions



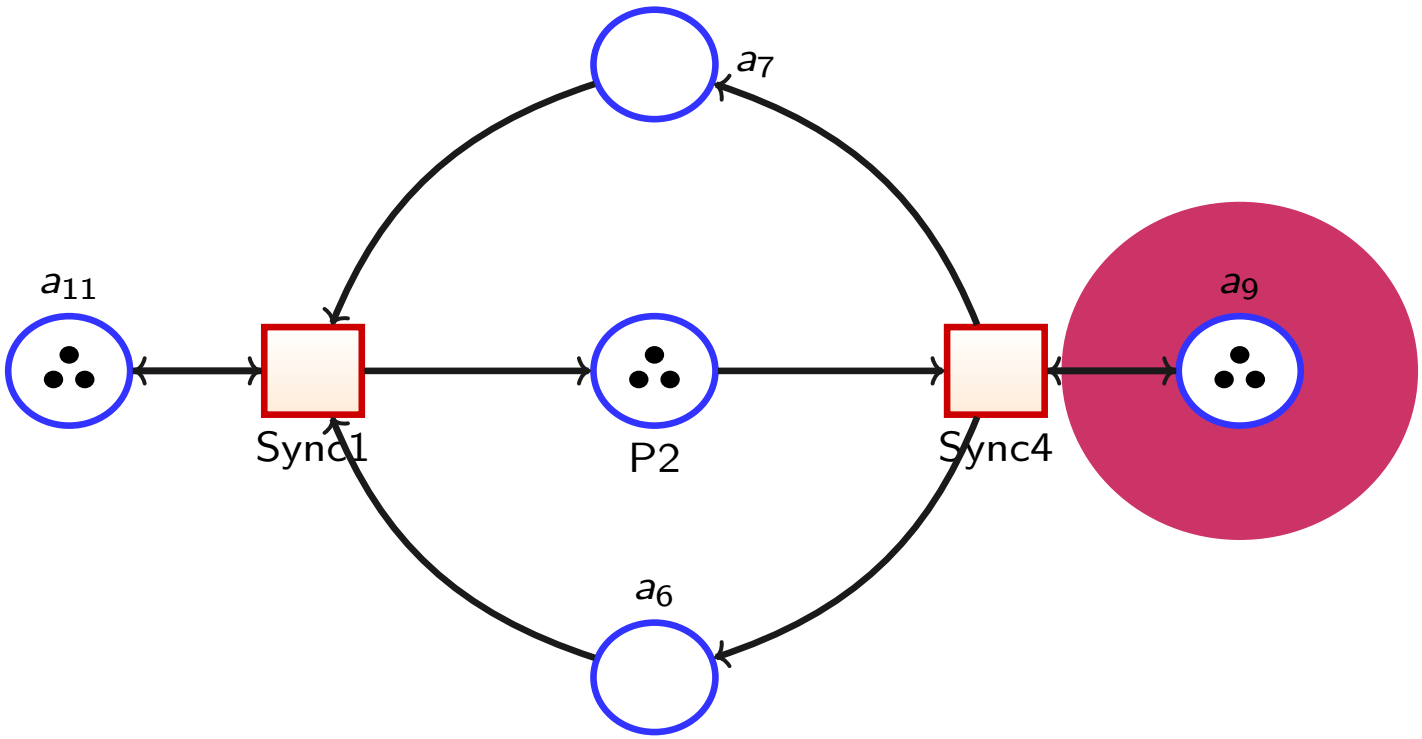
No more chain agglomerations.

Reductions



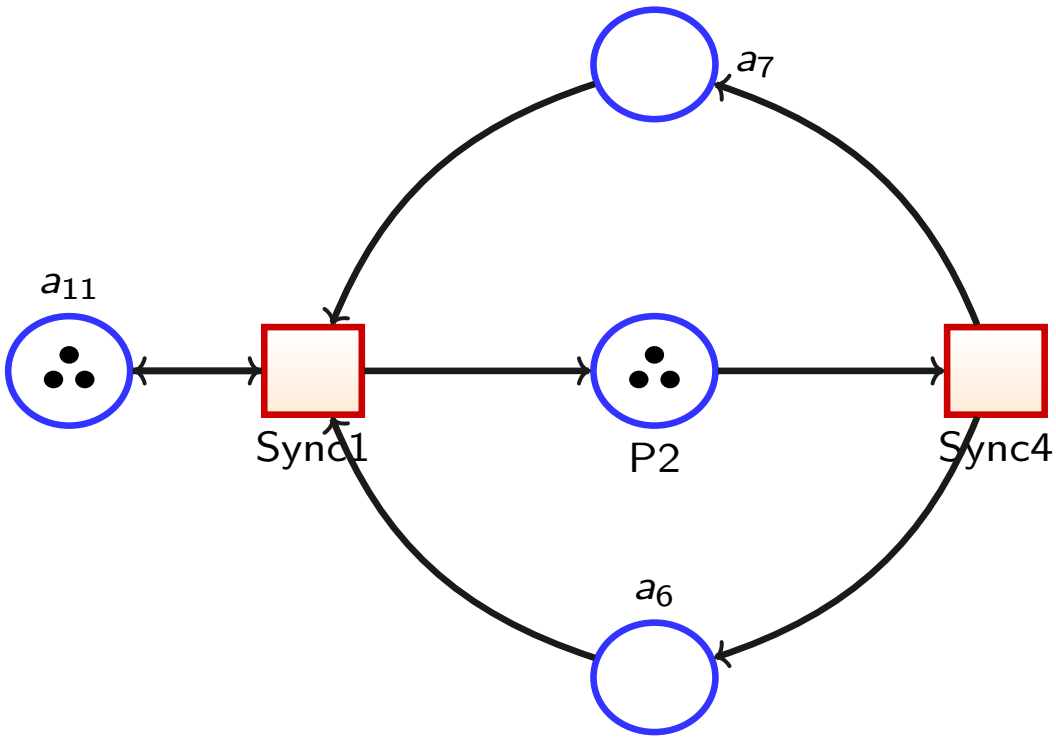
No more chain agglomerations.
But three new redundancies have appeared.

Reductions



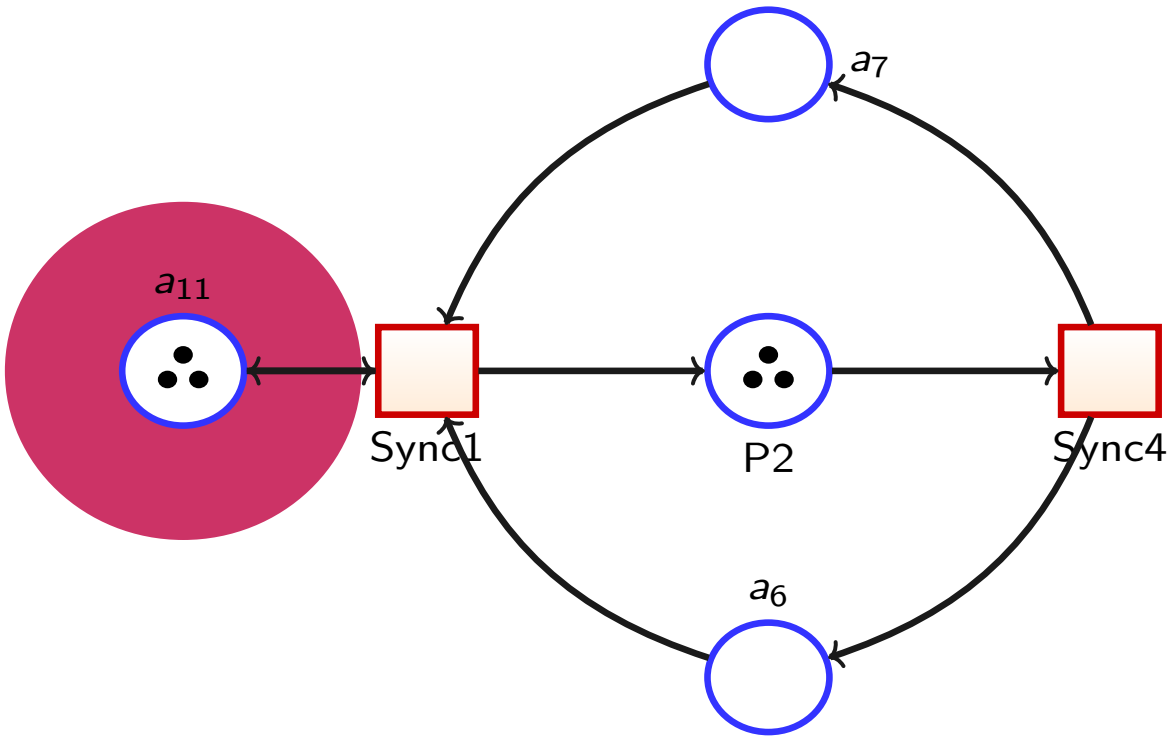
$R \vdash a_9 = 3$ a_9 is a **constant place**

Reductions



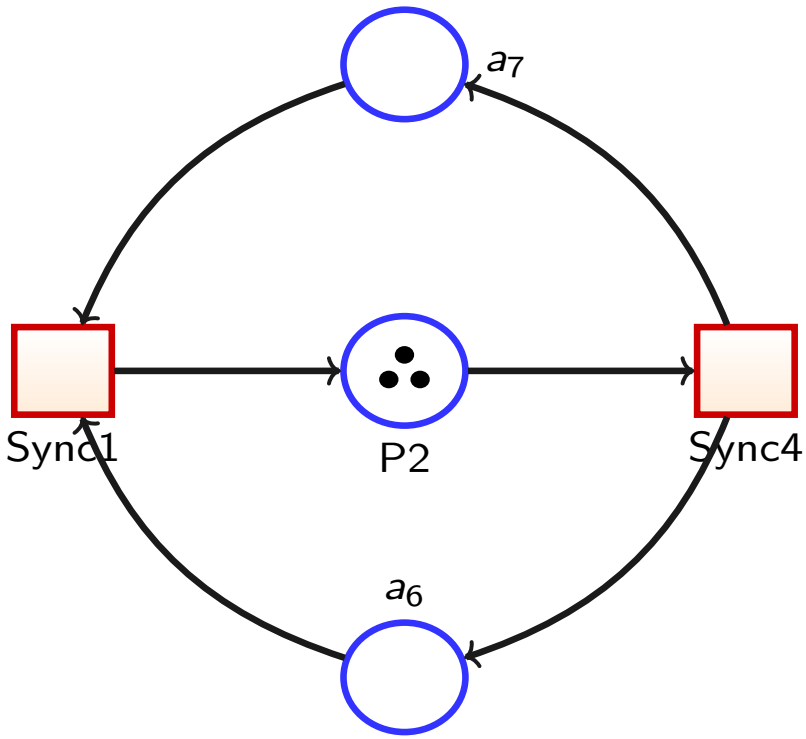
$$R \vdash a_9 = 3$$

Reductions



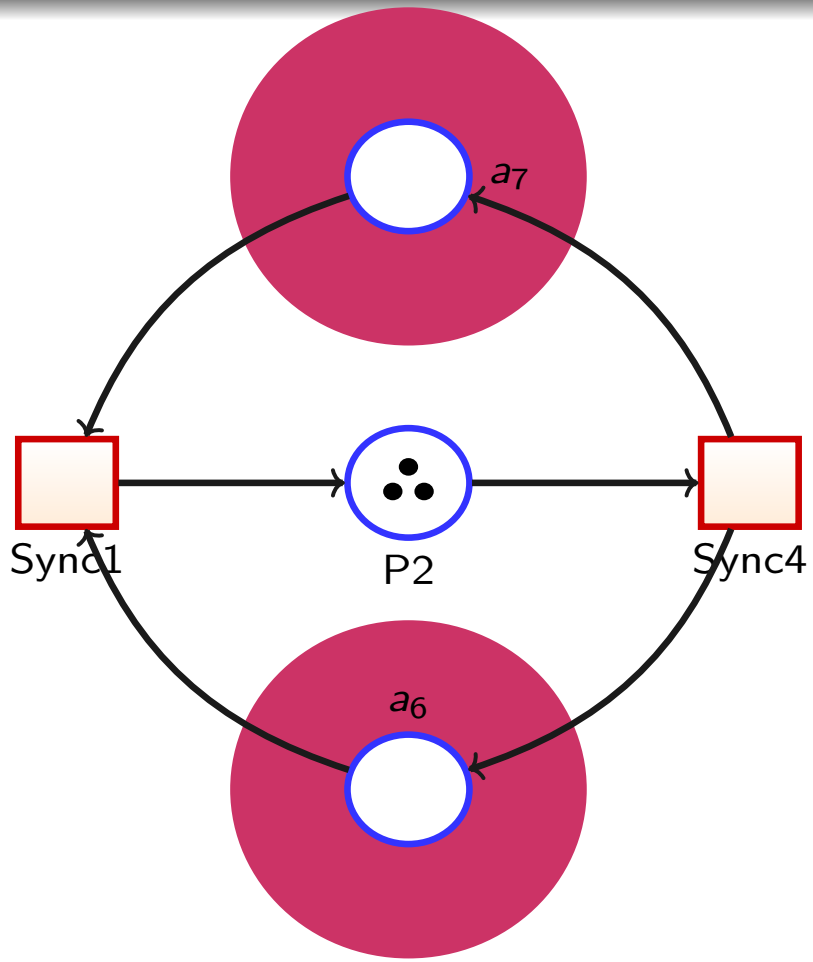
$R \vdash a_9 = 3$ $\qquad$ $R \vdash a_{11} = 3$

Reductions



$R \vdash a_9 = 3$ $\qquad$ $R \vdash a_{11} = 3$

Reductions

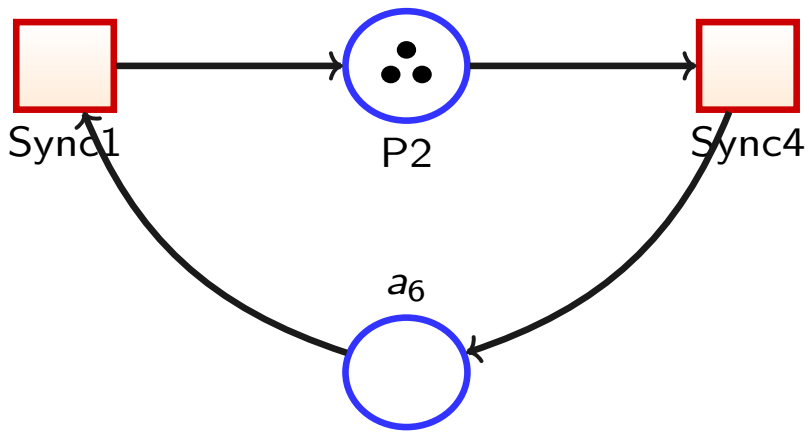


$R \vdash a_9 = 3$

$R \vdash a_7 = a_6$

$R \vdash a_{11} = 3$

Reductions

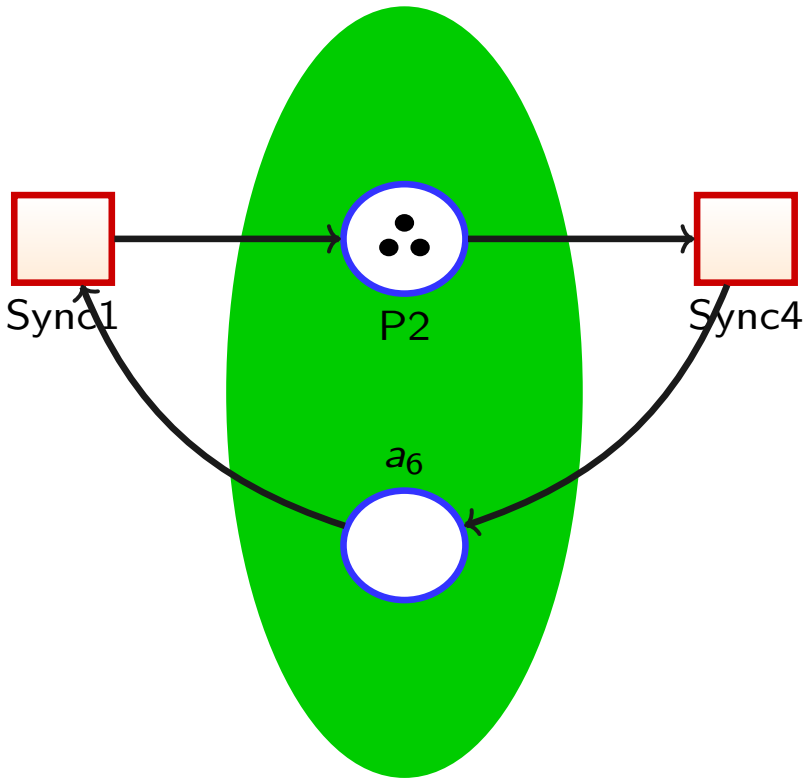


$R \vdash a_9 = 3$

$R \vdash a_7 = a_6$

$R \vdash a_{11} = 3$

Reductions



$R \vdash a_9 = 3$

$R \vdash a_7 = a_6$

$R \vdash a_{11} = 3$

$A \vdash a_{10} = a_6 + P_2$

Reductions

$$R \vdash a_9 = 3 \qquad R \vdash a_{11} = 3$$
$$R \vdash a_7 = a_6 \qquad A \vdash a_{10} = a_6 + P_2$$

Reductions

$$R \vdash a_9 = 3$$
$$R \vdash a_7 = a_6$$

$$R \vdash a_{11} = 3$$
$$A \vdash a_{10} = a_6 + P_2$$
$$R \vdash a_{10} = 3$$

Kanban equations

- Kanban is **totally reducible** using only four simple rules.
- The associated system of equations describes **exactly** the reachable states:

$$
\begin{aligned}
R \vdash P_3 &= P_2 \\
A \vdash a_2 &= Pback_2 + Pm_2 \\
A \vdash a_4 &= Pback_4 + Pm_4 \\
A \vdash a_6 &= Pout_2 + a_2 \\
A \vdash a_8 &= Pout_4 + a_4 \\
R \vdash a_7 &= a_6 \\
A \vdash a_{10} &= a_6 + P_2 \\
A \vdash a_{11} &= a_5 + P_1
\end{aligned}
\qquad
\begin{aligned}
A \vdash a_1 &= Pback_1 + Pm_1 \\
A \vdash a_3 &= Pback_3 + Pm_3 \\
A \vdash a_5 &= Pout_1 + a_1 \\
A \vdash a_7 &= Pout_3 + a_3 \\
A \vdash a_9 &= a_8 + P_4 \\
R \vdash a_9 &= n \\
R \vdash a_{10} &= n \\
R \vdash a_{11} &= n
\end{aligned}
$$

Each (non-negative, integer) solution to this set of linear equations corresponds to a reachable marking.

Kanban equations

Given these equations, one may

- Check if a state is **reachable** (use a solver)
- Count the number of reachable states (count lattice points)

- Compute some bounds : maximum marking for each place, ...

Kanban equations

Given these equations, one may

- Check if a state is **reachable** (use a solver)
- Count the number of reachable states (count lattice points)

$$\begin{aligned}\#K(n) = & \frac{1}{720}n^{11} + \frac{11}{360}n^{10} + \frac{649}{2160}n^9 + \frac{209}{120}n^8 \\ & + \frac{4757}{720}n^7 + \frac{413}{24}n^6 + \frac{7531}{240}n^5 + \frac{14413}{360}n^4 \\ & + \frac{37813}{1080}n^3 + \frac{1199}{60}n^2 + \frac{67}{10}n + 1\end{aligned}$$

- Compute some bounds : maximum marking for each place, ...

Food for thoughts

- This is interesting !
- We just solved an infinite state / parametrized model-checking problem.
- Each step is simple, yet combining / iterating them is very effective. Reductions have a **cumulative effect**: new redundancies appear after applying agglomerations, and conversely.
  $\Rightarrow$ the power of composition

Our (monolithic) reduction system

Redundant place $R \vdash a_{10} = 5$ or $R \vdash a_7 = a_6$

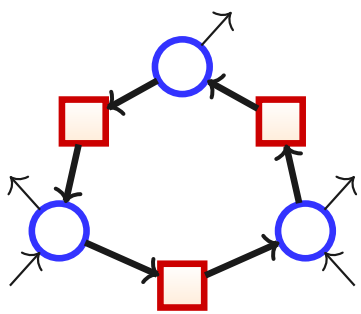
Any place equal to a linear combination of other places.

Redundant transition $\emptyset$

Any transition equivalent to a sequence of other transitions

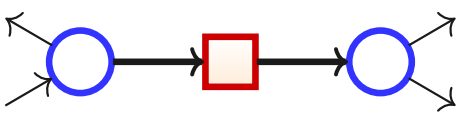
Loop agglomeration $A \vdash a_4 = M_4 + Back_4$

A loop of places linked by 1-1 transitions



Chain agglomeration $A \vdash a_{11} = a_5 + P_1$

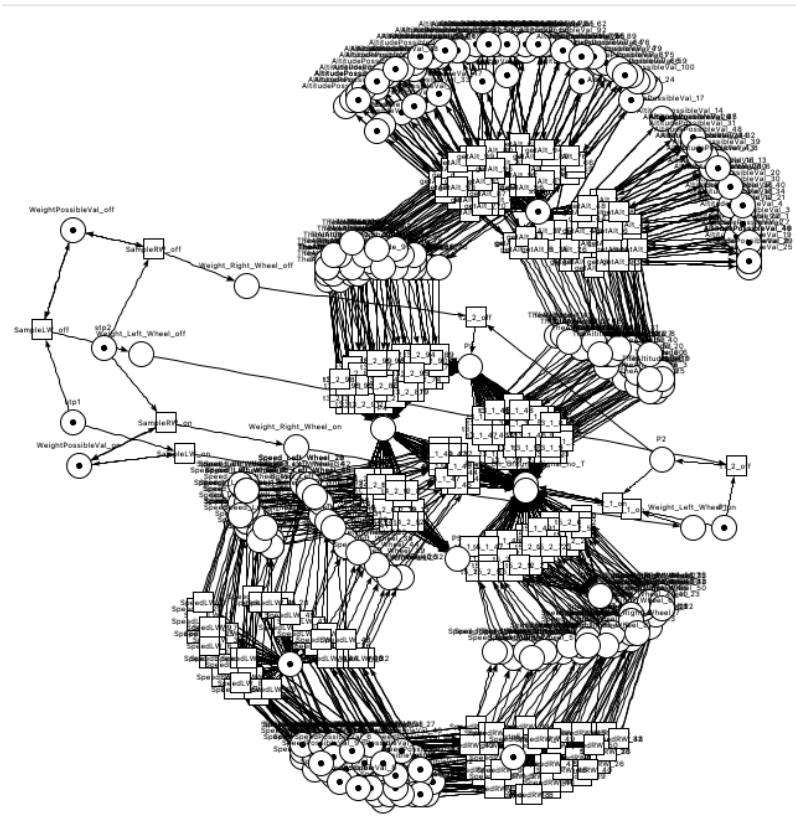
A place reachable from another place by a single 1-1 transition



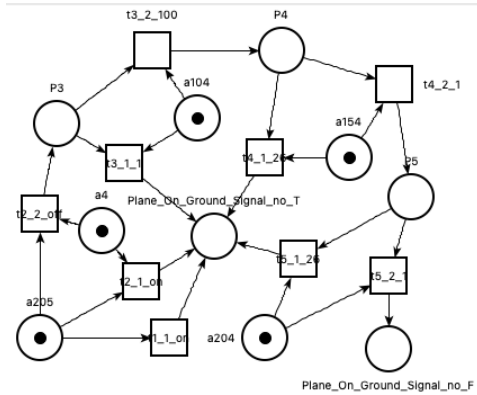
Opens lots of interesting questions

- What about "partially reduced" nets ?
- Are reductions common in practice ?
- What are the nets that can be abstracted using a *Linear Integer Arithmetic* (LIA) formula ?
  - ...with relatively few variables ?
  - What about unbounded nets whose reduced net is bounded ?
- What if we allow Presburger formulas instead

Experimental Results

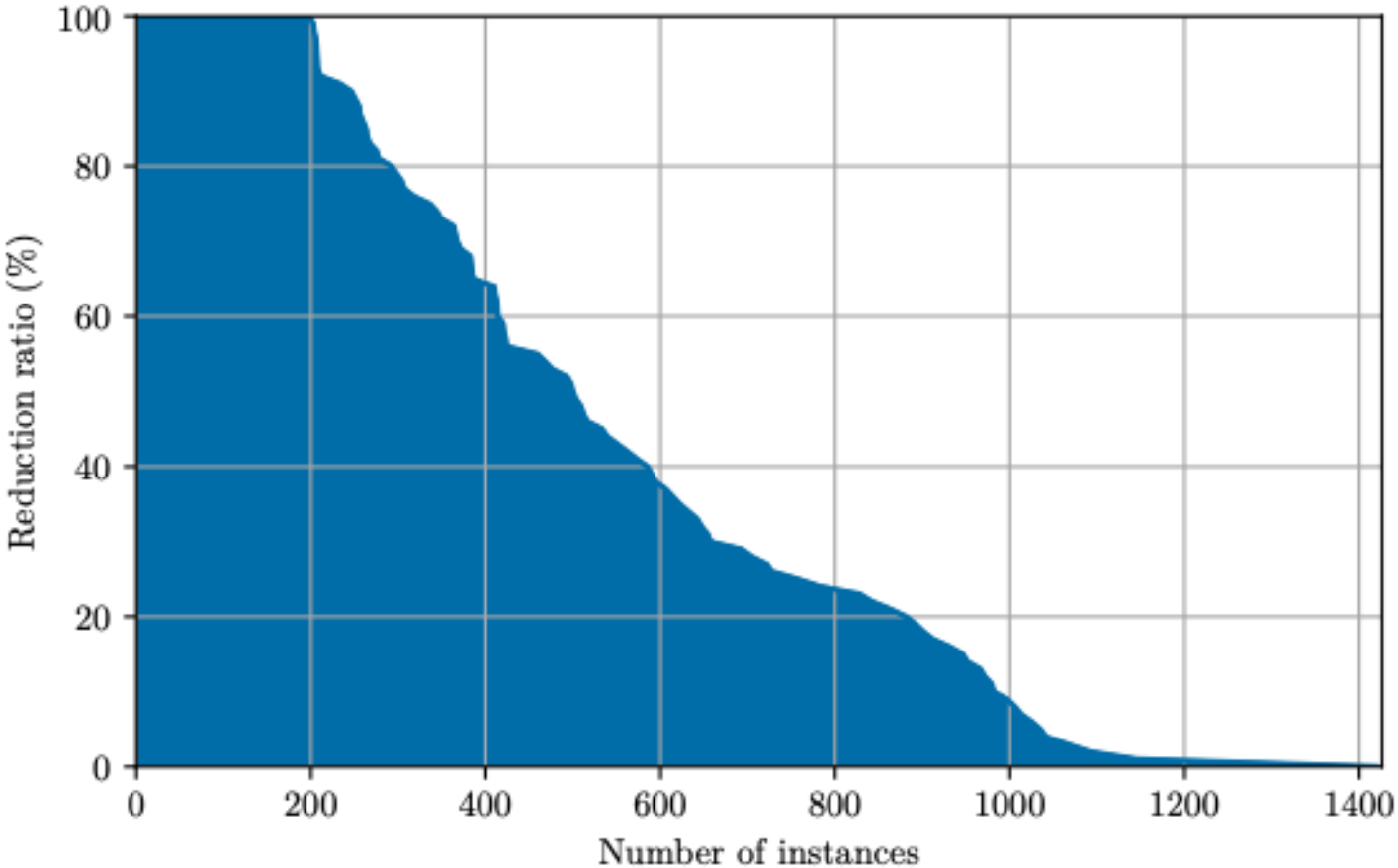


$\equiv_E$



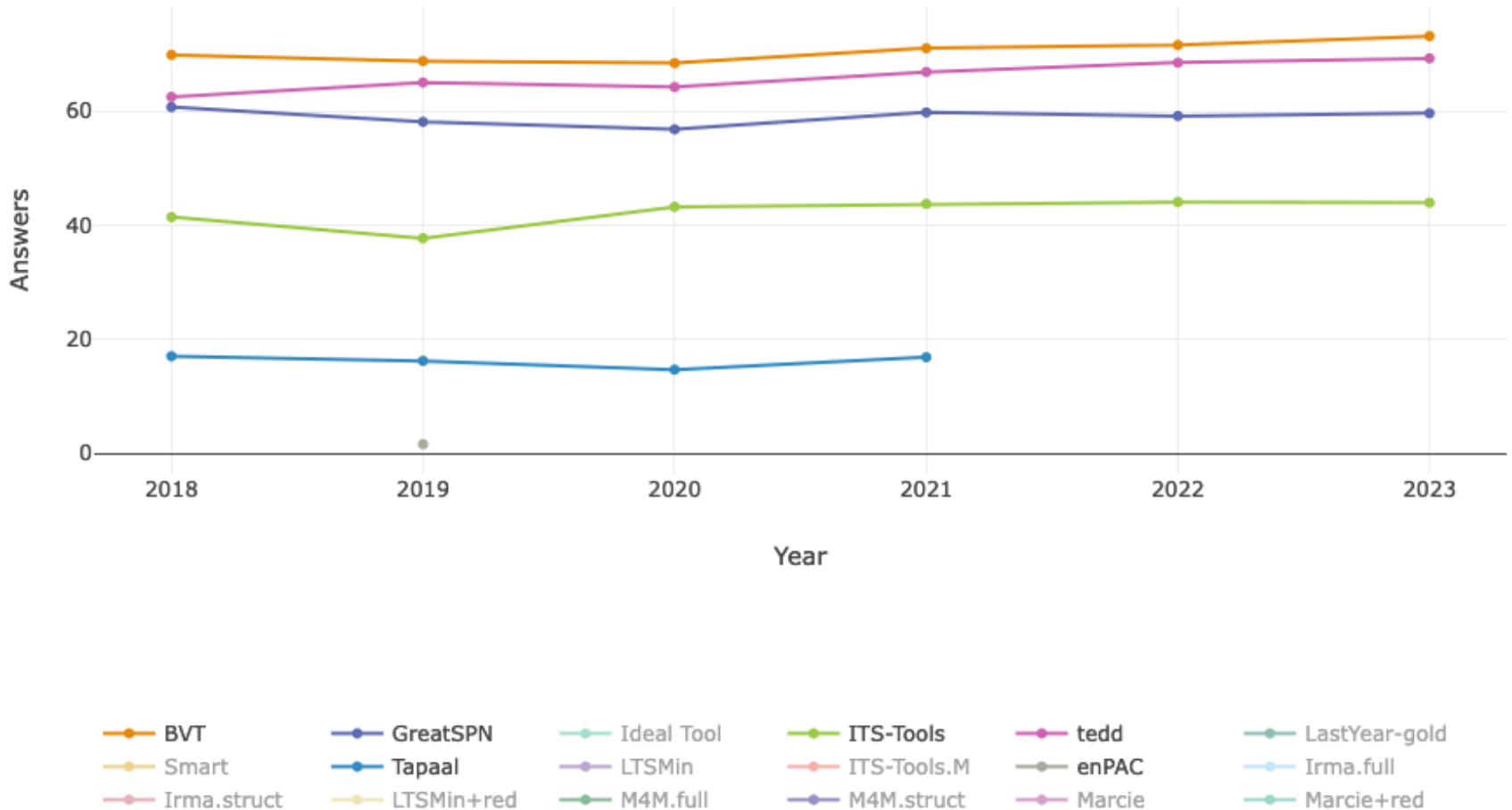
AirplaneLD-50 (landing detector with max. speed 50)

Prevalence of reductions on the MCC instances



- 14% of instances are **fully reducible** (like Kanban)
- Half of them are significantly reduced (reduction ratio $> 30\%$)
- 80% are reduced by $> 1\%$

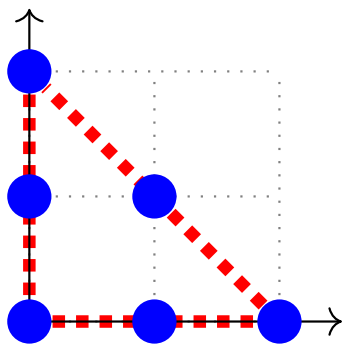
StateSpace examination at the MCC



Counting solutions

How to count the number of solutions of E?

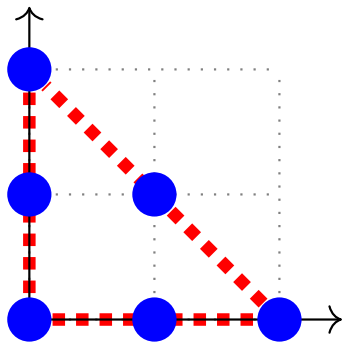
- Off-the-shelf tool: LaTTe *[De Loera et al.]*
  - Computes the number of integral points within convex polytopes.
  - Geometric method.
  - Handles equations having up to ≈ 40 variables.



Counting solutions

How to count the number of solutions of E?

- Off-the-shelf tool: LaTTe *[De Loera et al.]*
  - Computes the number of integral points within convex polytopes.
  - Geometric method.
  - Handles equations having up to ≈ 40 variables.



- Our ad-hoc tool: *polycount*
  - Less general than LaTTe, but more efficient here.
  - Takes advantage of the particular "form" of our equations.
  - Handles equations having hundreds or thousands of variables *(Its complexity depends mostly on the dependencies between variables.)*

Tools and references

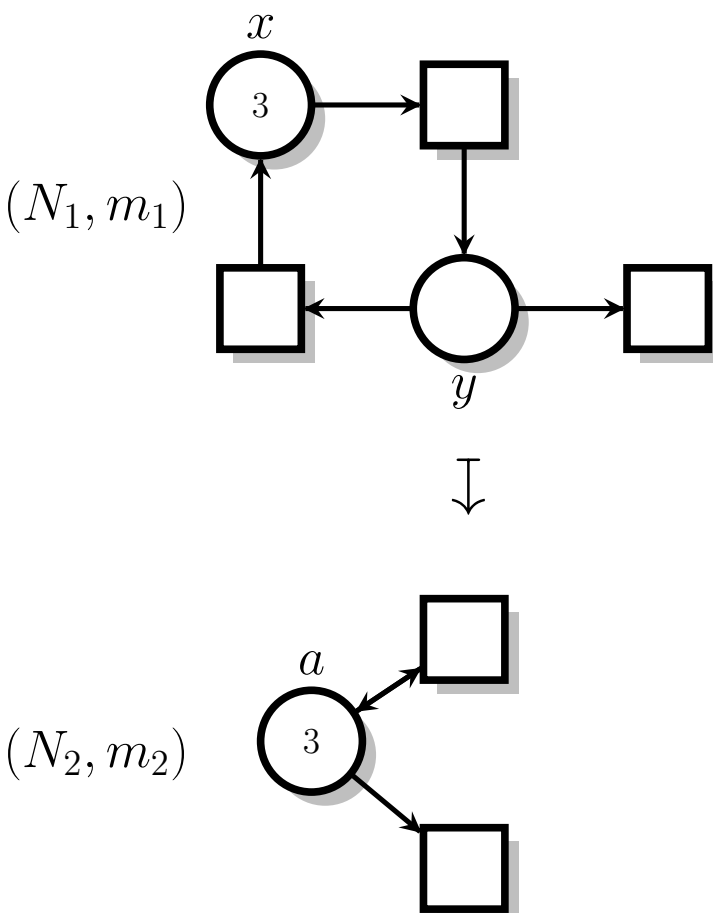
reduce and **tedd** are available at projects.laas.fr/tina/.

[SPIN18] Bernard Berthomieu, Didier Le Botlan, Silvano Dal Zilio.
Petri Net Reductions for Counting Markings.
Model Checking Software (SPIN), 2018.
10.1007/978-3-319-94111-0_4

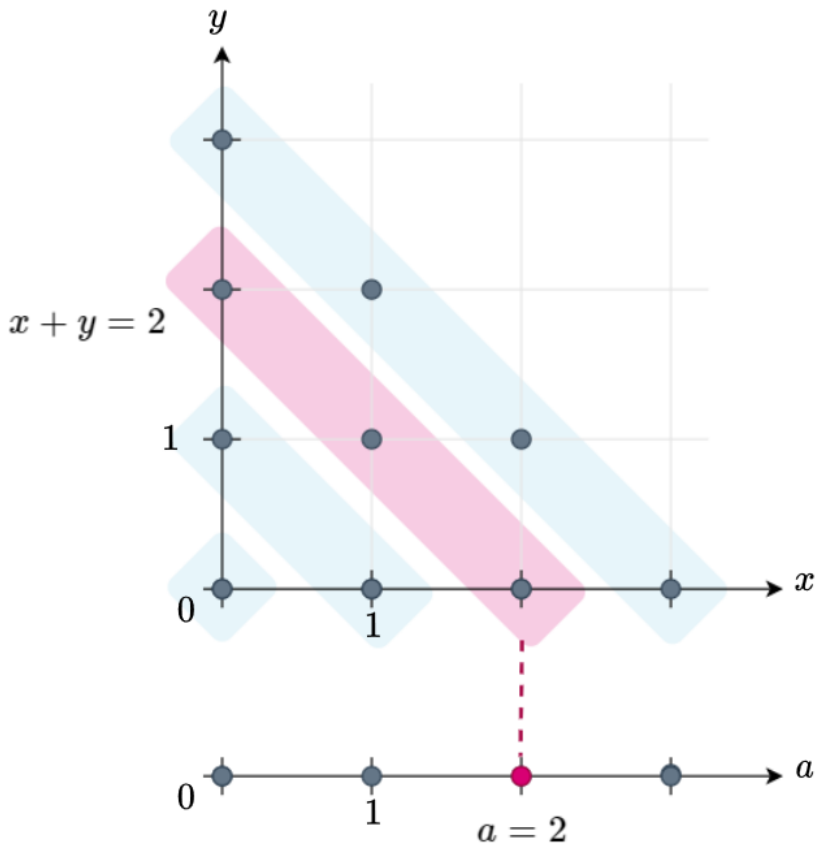
[STTT19] Bernard Berthomieu, Didier Le Botlan, and Silvano Dal Zilio.
Counting Petri net markings from reduction equations.
Int. Jour. on Software Tools for Technology Transfer, 22, 2019.
10.1007/s10009-019-00519-1

Polyhedral Abstraction

Polyhedral Reductions

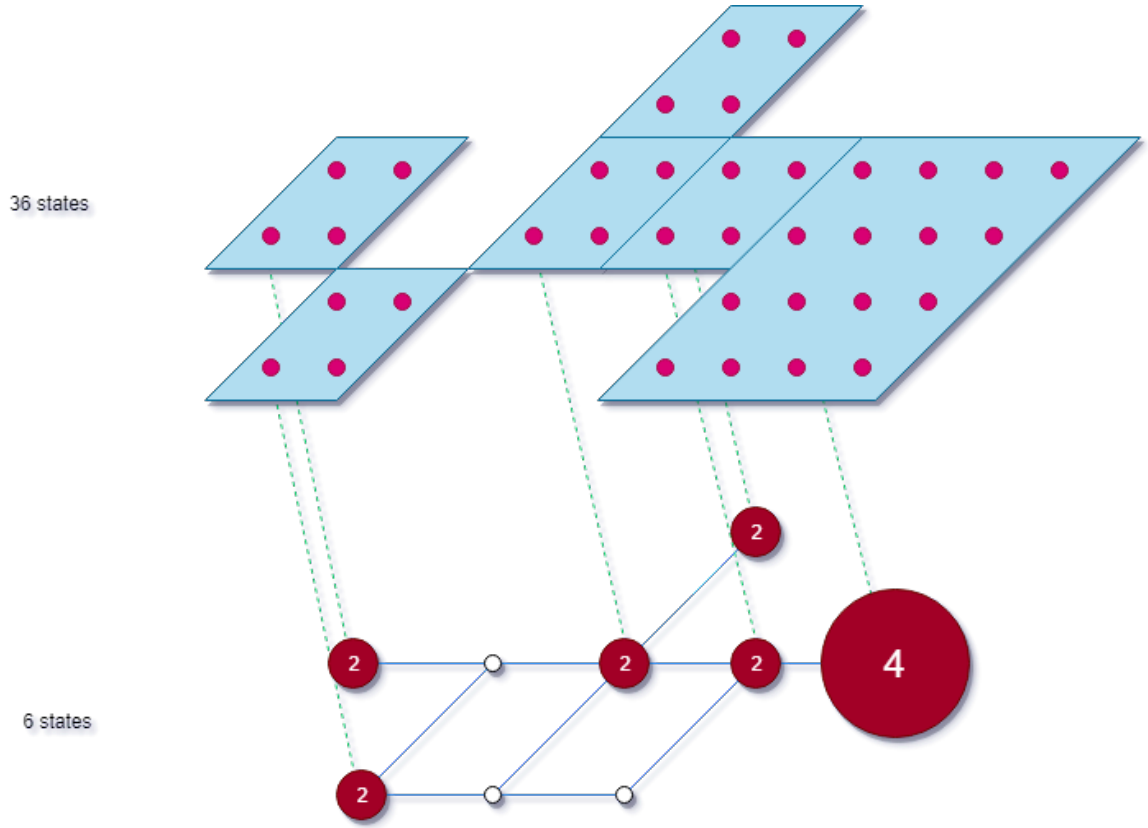


Net reduction example, with equation $E : a = x + y$



Relation between state-spaces

Polyhedral Abstraction



State-space abstraction based on a "polyhedral relation" E

Relation E includes: places/variables from N_2 ; variables from N_1 (and no longer in N_2) ; extra ($\exists$) variables

Petri net theory as LIA formulas

- A **marking** is a formula (a **cube**), with variables in $\vec{x}$, that is only "satisfiable at m": $\underline{m}(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$

$$\underline{m_0}(\vec{p}) \equiv (p_0 = 5) \wedge (p_1 = 0) \wedge \ldots$$

Petri net theory as LIA formulas

- A **marking** is a formula (a **cube**), with variables in $\vec{x}$, that is only "satisfiable at m": $\underline{m}(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$

$$\underline{m_0}(\vec{p}) \equiv (p_0 = 5) \wedge (p_1 = 0) \wedge \ldots$$

- A linear system E is **satisfiable** for marking m if the system $E \wedge \underline{m}$ has solutions, also written $m \models E$.

Petri net theory as LIA formulas

- A **marking** is a formula (a **cube**), with variables in $\vec{x}$, that is only "satisfiable at m": $\underline{m}(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$

$$\underline{m_0}(\vec{p}) \equiv (p_0 = 5) \wedge (p_1 = 0) \wedge \ldots$$

- A linear system E is **satisfiable** for marking m if the system $E \wedge \underline{m}$ has solutions, also written $m \models E$.

- Two markings m_1 and m_2 are **compatible** when $m_1(p) = m_2(p)$ for all p in $P_1 \cap P_2$.
  We use $m_1 \uplus m_2$ for the "combined marking".

Petri net theory as LIA formulas

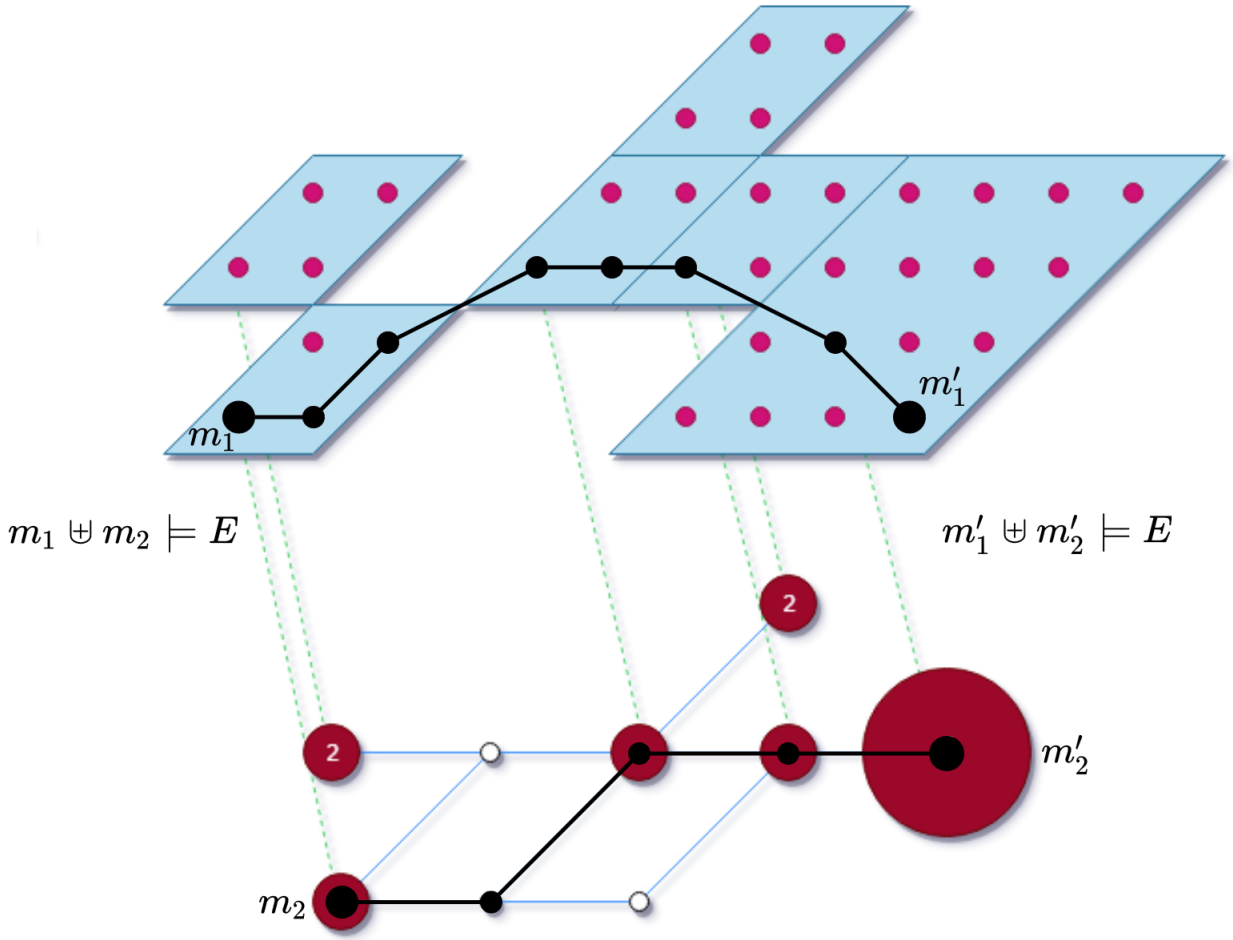
- A **marking** is a formula (a **cube**), with variables in $\vec{x}$, that is only "satisfiable at m": $\underline{m}(\vec{x}) \equiv \bigwedge_{i \in 1..n}(x_i = m(p_i))$

$$\underline{m_0}(\vec{p}) \equiv (p_0 = 5) \wedge (p_1 = 0) \wedge \ldots$$

- A linear system E is **satisfiable** for marking m if the system $E \wedge \underline{m}$ has solutions, also written $m \models E$.
- Markings m_1 and m_2 are "matched" in a reduction when $m_1 \wedge m_2 \wedge E$ is SAT. We simply say $m_1 \equiv_E m_2$.

($*$) *we could choose E to be any Presburger-definable relation. Not necessarily a linear equation system.*

E-Abstraction Equivalence



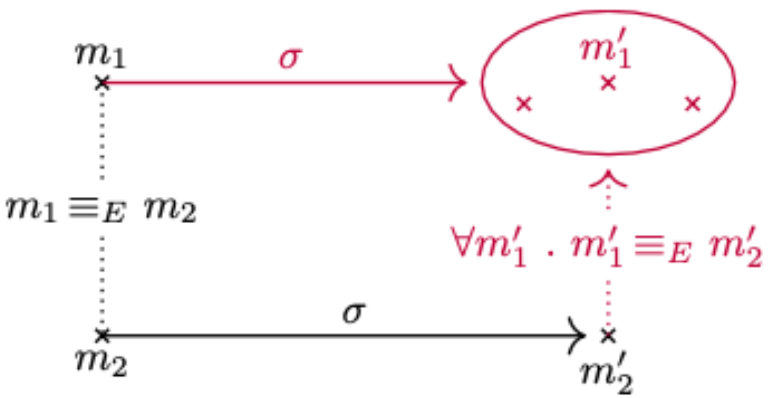
$(*)$ *under condition that* E *solvable for* (N_1, N_2)

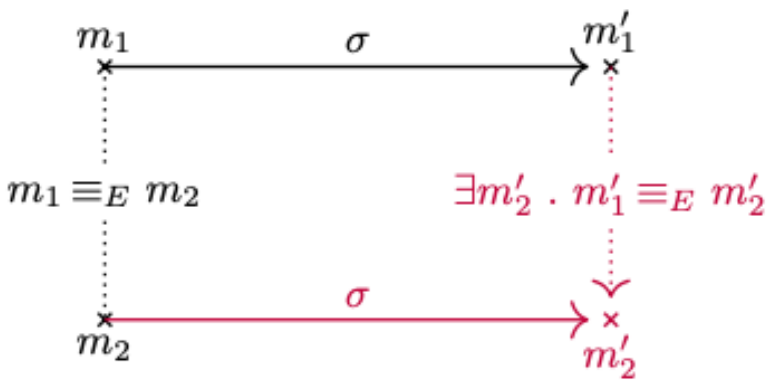
E-Abstraction: $(N_1, m_1) \sqsupseteq_E (N_2, m_2)$

(A1) initial markings are compatible with E, meaning $m_1 \equiv_E m_2$

(A2) for all firing sequence $(N_1, m_1) \stackrel{\sigma_1}{\Longrightarrow} (N_1, m_1')$

- there is at least one marking m_2' over P_2 such that $m_1' \equiv_E m_2'$,
- and for all markings m_2' such that $m_1' \equiv_E m_2'$ we must have an (observably equivalent) firing sequence $(N_2, m_2) \stackrel{\sigma_2}{\Longrightarrow} (N_2, m_2')$





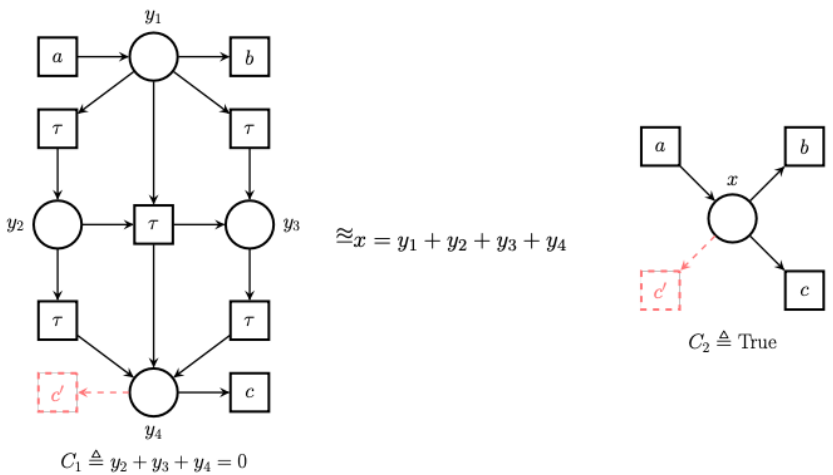
Algebraic laws of reductions

Axioms: Reduction Rules (RED, CONCAT, etc.)
Hence structural reductions appear as **one** method (among others ?) for generating polyhedral abstractions.

Composability: congruence for $\|$-composition

Transitivity

Relabeling



Algebraic laws of reductions

- Checking polyhedral equivalence is undecidable, even for the simple case where $E \stackrel{\triangle}{=} \top$ ($\approx$ marking equivalence problem [Hack, 1976]).
- But we have a procedure to **automatically** prove polyhedral equivalence between two **parametric** nets $(N_1, C_1) \equiv_E (N_2, C_2)$.
- What we achieve with polyhedral abstraction $\equiv$ capture "**flattable**" sub-parts of nets (nets whose reachable sets are Presburger-definable, see [Leroux, 2013])

Application to SMT-based Model-Checking

Properties of Interest

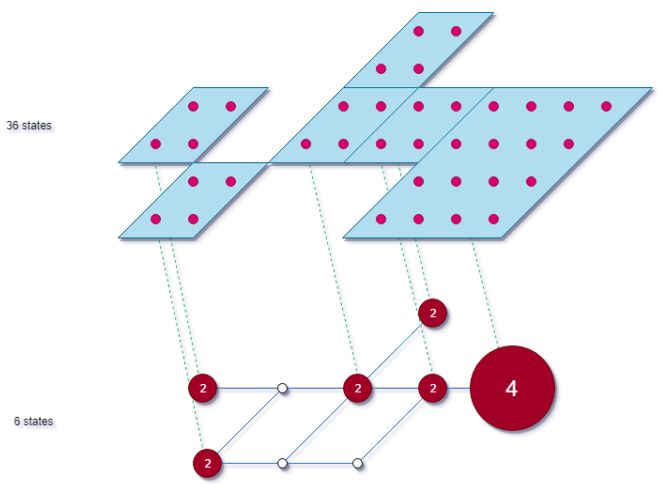
- ϕ **reachable**: $\exists m \in R(N, m_0)$ s.t. $\phi(\vec{x}) \wedge \underline{m}(\vec{x})$ *SAT*
- ϕ **invariant**: $\forall m \in R(N, m_0)$ we have $\neg\phi(\vec{x}) \wedge \underline{m}(\vec{x})$ *UNSAT*

- **Coverability**: $\mathrm{COVER}(p, k) \equiv m(p) \geq k$
- **Reachability**: $\mathrm{PLIVE}(p) \equiv m(p) \geq 1$
- **Quasi-liveness**: $\mathrm{TLIVE}_t \equiv \bigwedge_{p \in {}^\bullet t} \mathrm{COVER}(p, \mathbf{pre}(t, p))$
- **Deadlock**: $\mathrm{DEAD} \equiv \bigwedge_{t \in T} \neg\mathrm{TLIVE}_t$

Combination with Polyhedral Abstraction

We can express that a marking m'_1 of N_1 is "related" with the marking m'_2 of N_2 through E ($m_1 \equiv_E m_2$), using the following formula

$$\underline{m'_1}(\vec{x}) \wedge \underline{m'_2}(\vec{y}) \wedge \tilde{E}(\vec{x}, \vec{y})$$



(∗) *$\tilde{E}$ is E added with $x_i = y_j$ if places i in N_1 and j in N_2 are the same*

Combination with Polyhedral Abstraction

Definition (E-transform Formula)

Formula $F_2(\vec{y}) \triangleq \exists \vec{x}. \left(\tilde{E}(\vec{x}, \vec{y}) \wedge F_1(\vec{x}) \right)$ is the E-transform of F_1

Theorem (Reachability Conservation)

F_1 is reachable in N_1 if and only if its E-tranform formula is reachable in N_2

Theorem (Invariant Conservation)

$\neg F_1$ is invariant on N_1 if and only if $\neg F_2$ is invariant on N_2

$(*)$ *$\neg F_2$ has $\forall$ quantification*

SMT-based Algorithms

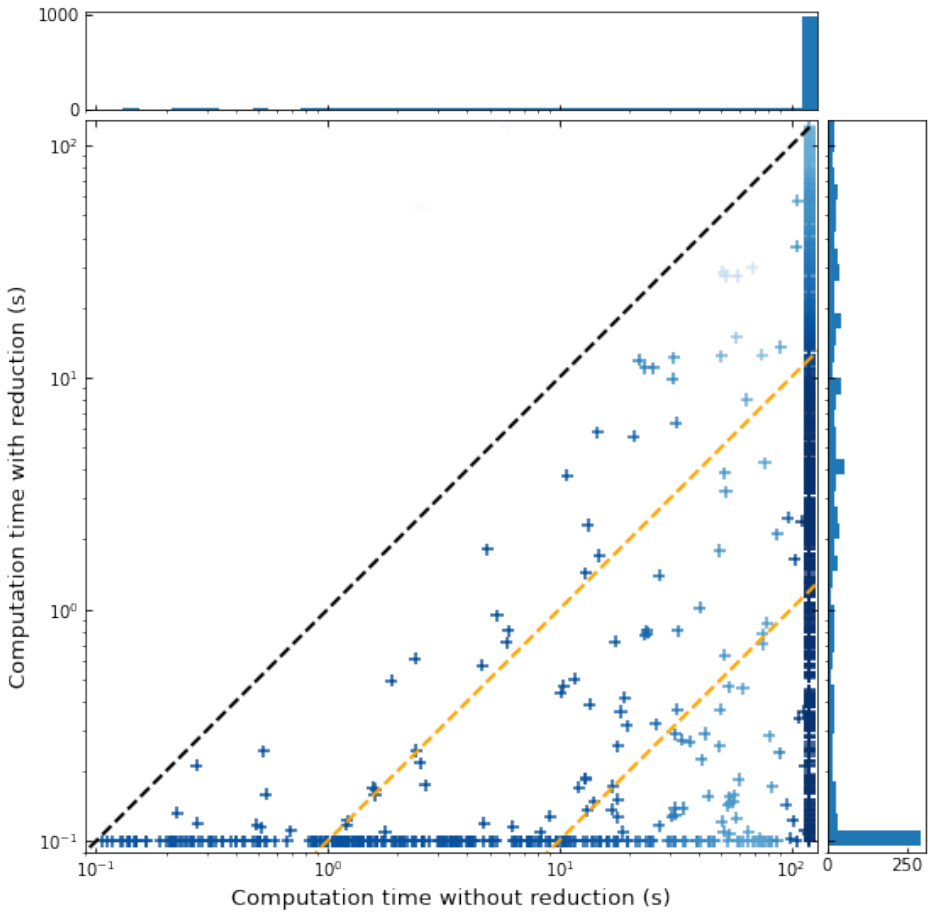
We implemented several symbolic model-checking algorithms for generalized Petri nets and extended them to support reductions ($\tilde{E}$)

- **Bounded Model Checking** (BMC): counterexample finder
- k-induction
- **Property Directed Reachability** (PDR): invariant prover
- PDR-COV: for coverability properties
- ...

Computation time

Experimental Results, medium to high reduction ratio

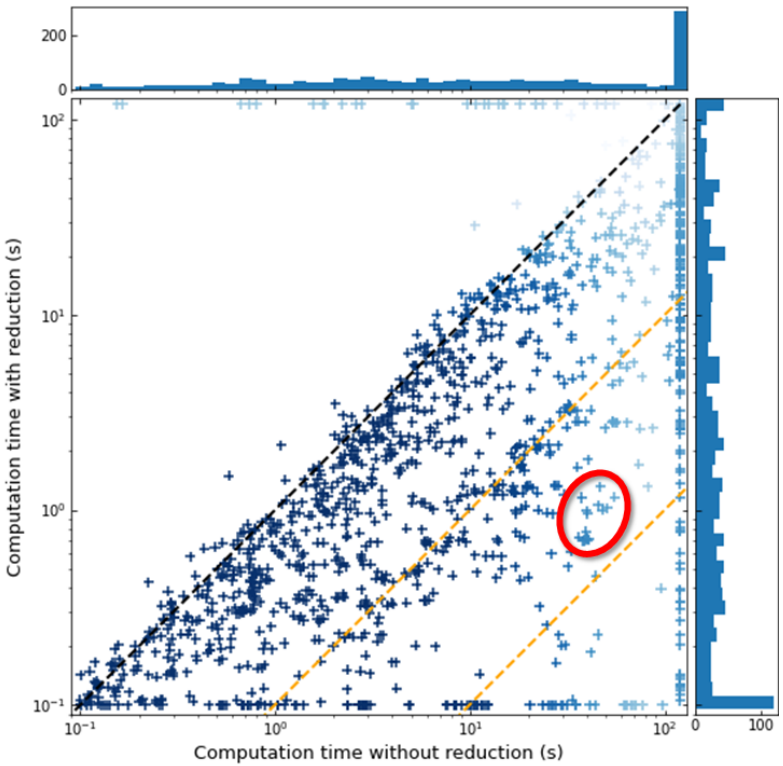
Computation time with (y-axis) vs without (x-axis) reduction (s)



Reduction ratio $\in [0.5, 1[$

A look at some concrete instances

Experimental Results

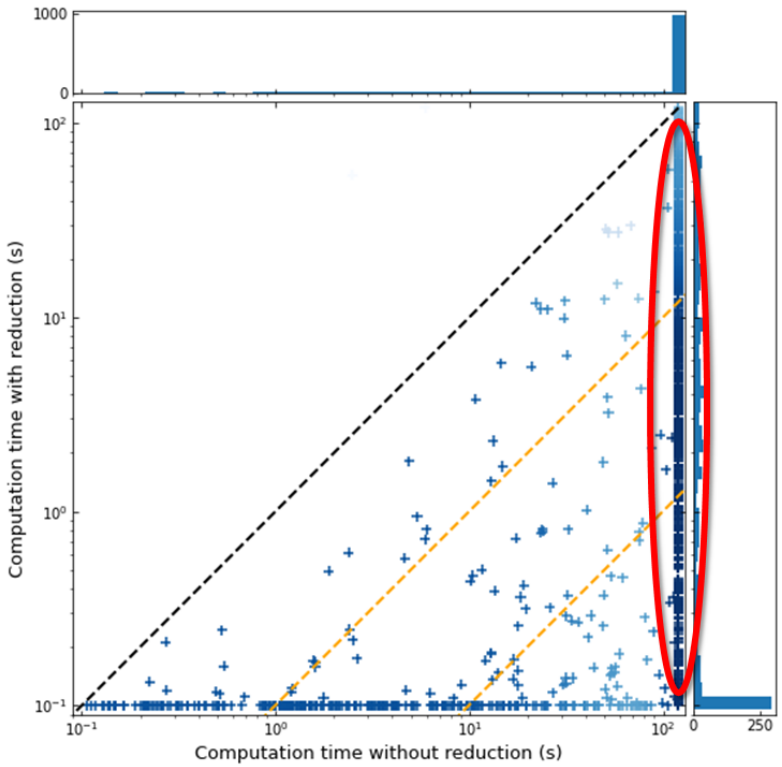


Reduction ratio $\in]0, 0.25[$

Instance	ARMCacheCoherence
State Space	3.206e+8
Red Ratio	17%
$\mathbb{E}_{red}(\theta)$	1 s
$\mathbb{E}_{\overline{red}}(\theta)$	20 s

A Look at Concrete Instances

Experimental Results



Reduction ratio $\in [0.5, 1[$

Instance	AirplaneLD-1000
State Space	?
Red Ratio	99%
# Props with red	14
# Props without red	0

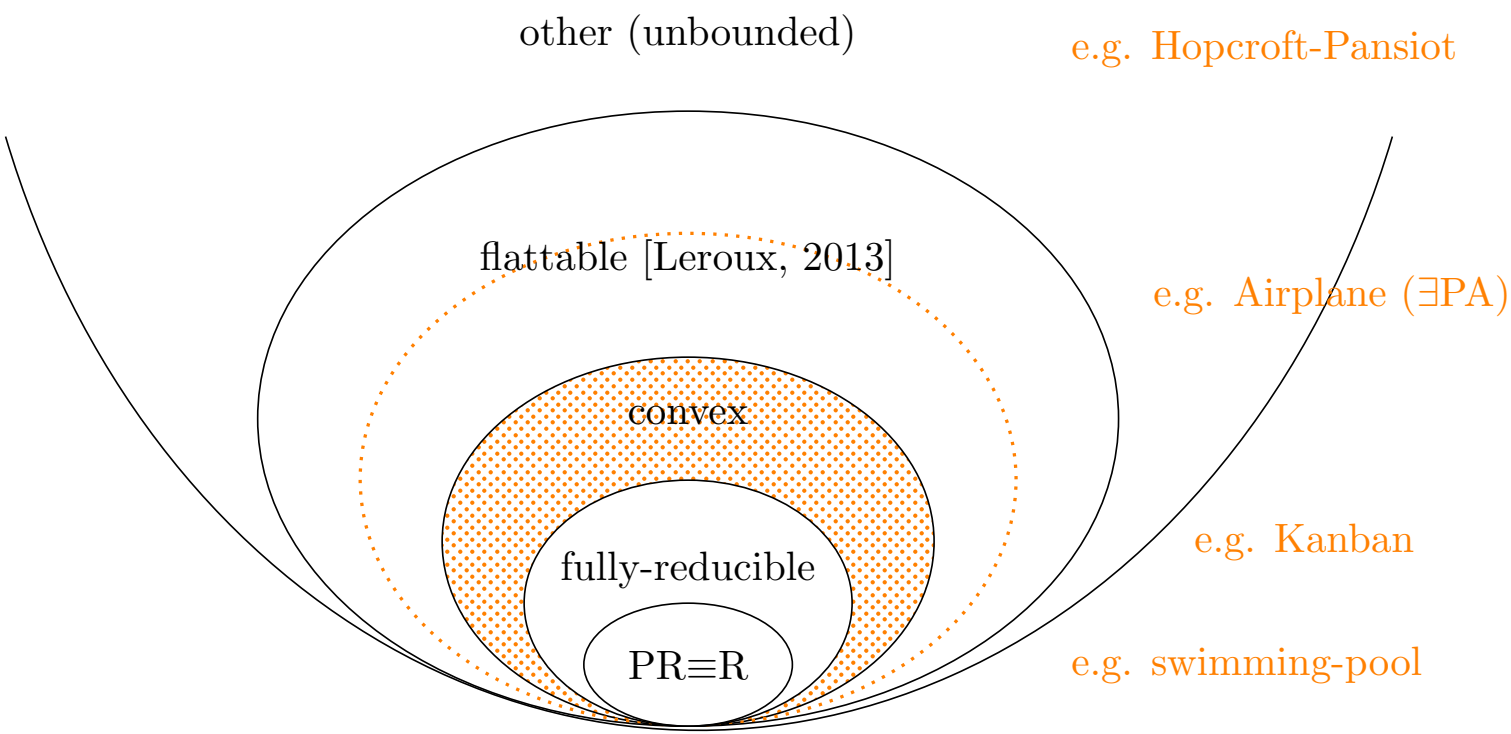
Tools and references

SMPT is Open Source and available on at
github.com/nicolasAmat/SMPT.

[PN21] Nicolas Amat, Bernard Berthomieu, and Silvano Dal Zilio.
On the combination of Polyhedral Abstraction and
SMT-based model checking for Petri nets.
In *A. T. Petri Nets and Concurrency (Petri Nets)*, 2021.
10.1007/978-3-030-76983-3_9

[TACAS22] Nicolas Amat, Silvano Dal Zilio, and Thomas Hujsa.
Property Directed Reachability for generalized Petri nets.
In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2022.
10.1007/978-3-030-99524-9_28

Geometry of the reachability sets of Petri nets



Conclusion

- We use a set of "*simple*" reductions, which are surprisingly **efficient** to reduce the net size when used together.
- Reductions generate linear equations which **characterize the state space** (partially or totally). This is different/orthogonal to slicing
- We define methods, and data structures, to transfer problems between the initial and the reduced net.

Questions ?

Thank you for your attention !

Any questions ?