The complexity of soundness in workflow nets

Filip Mazowiecki

University of Warsaw

FI_t 2022

The complexity of soundness in workflow nets

Filip Mazowiecki

University of Warsaw

FI_t 2022

Plan

1. Petri nets and reachability

2. Workflow nets and soundness

3. Some proofs

4. Implementation

Almost Petri nets

Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

Almost Petri nets

Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

Almost Petri nets

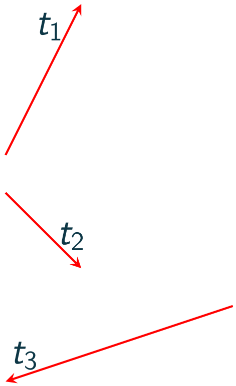
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$



Almost Petri nets

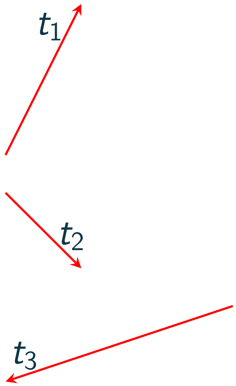
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$



Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

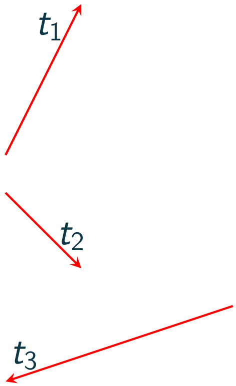
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

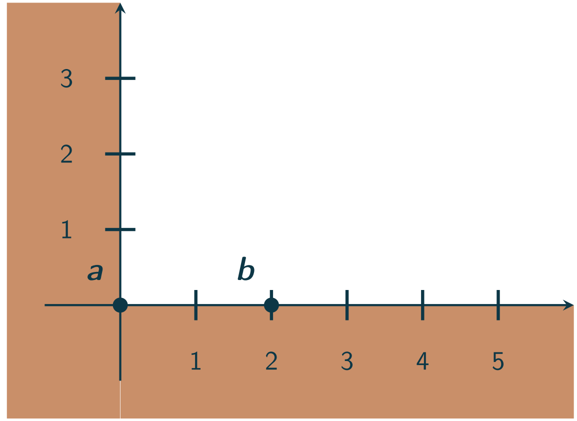
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

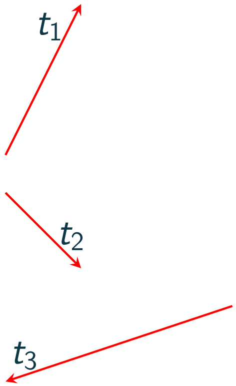
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

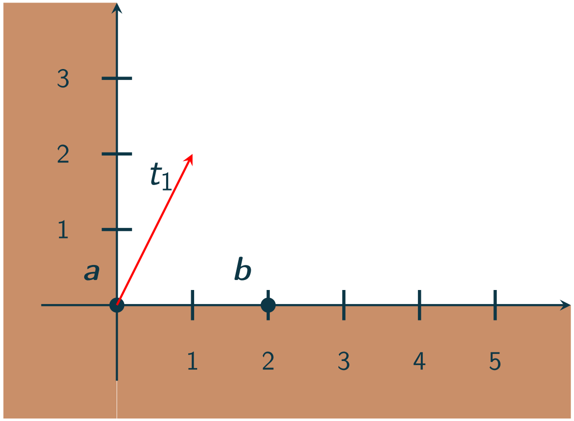
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

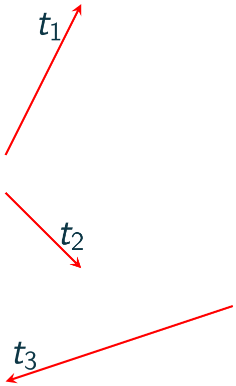
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

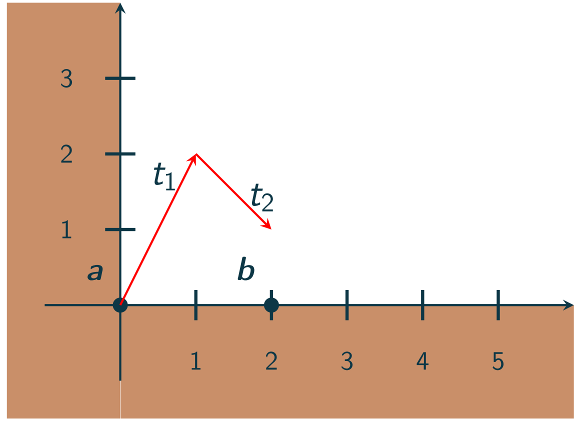
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

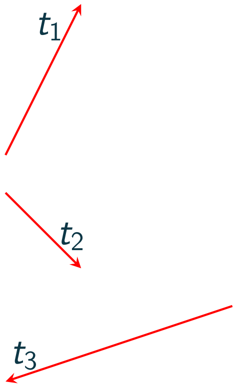
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

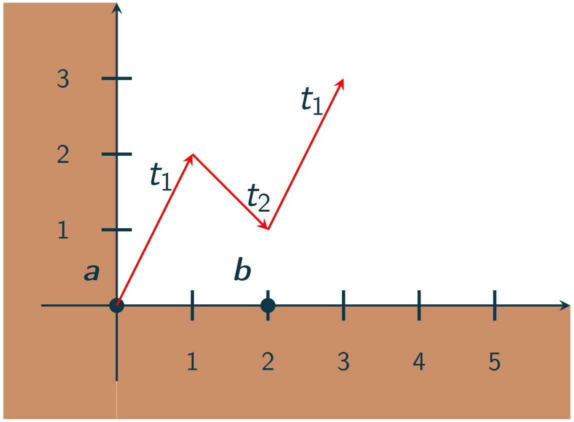
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

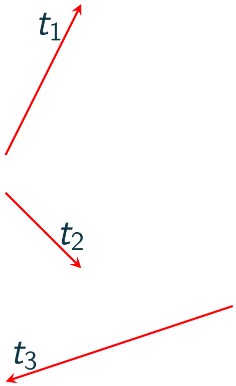
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

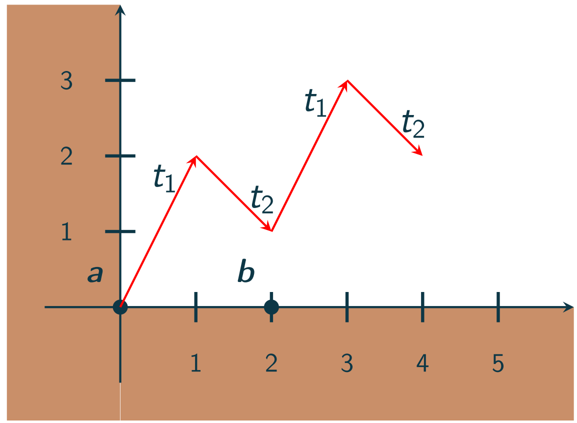
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

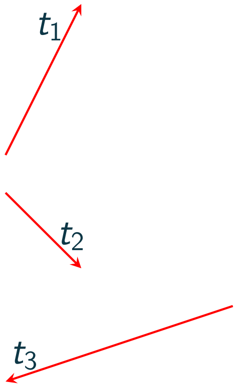
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

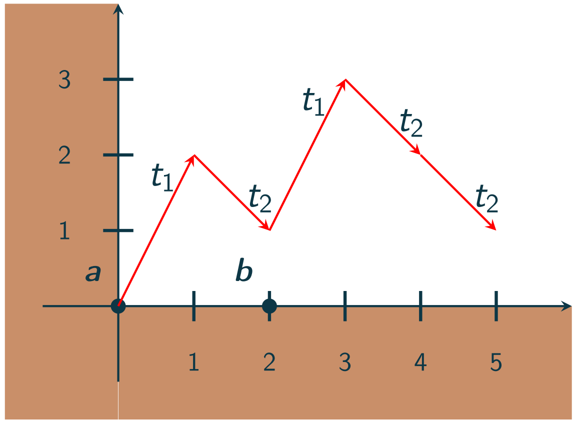
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

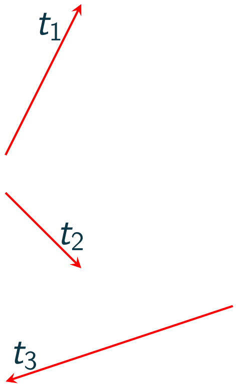
Almost Petri net (d, T): $\quad d$ – dimension, $\quad T \subseteq \mathbb{Z}^d$ (finite)

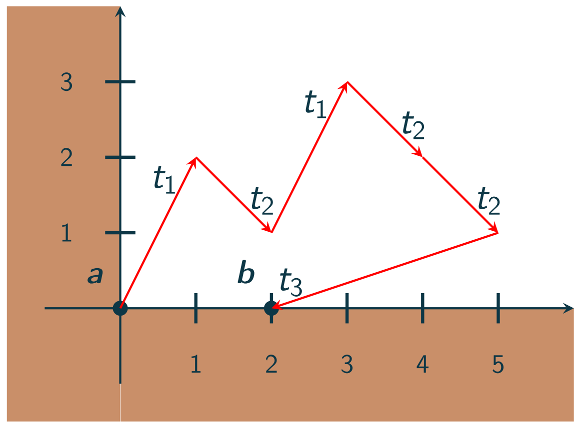
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

Almost Petri nets

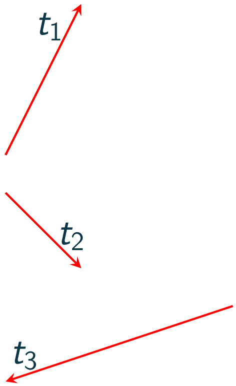
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

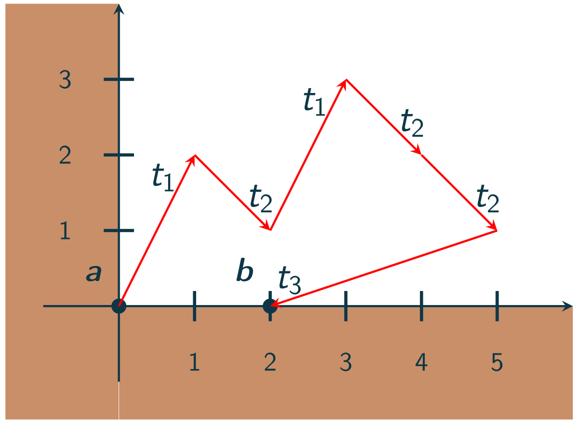
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

(without t_3 it's not possible)

Almost Petri nets

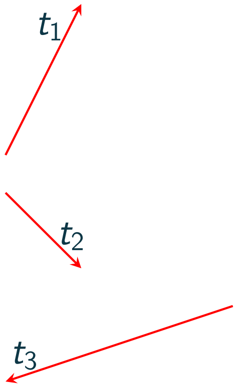
Almost Petri net (d, T): $\quad d$ – dimension, $\quad T \subseteq \mathbb{Z}^d$ (finite)

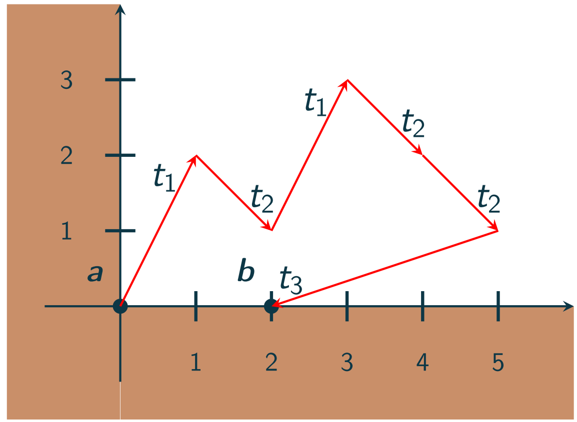
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





Reachability problem: given (d, T) and two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{N}^d$

Determine if one can go from $\boldsymbol{a}$ to $\boldsymbol{b}$ remaining in $\mathbb{N}^d$?

(without t_3 it's not possible)

Notation: $\boldsymbol{a} \rightarrow^* \boldsymbol{b}, \quad \boldsymbol{a} \not\rightarrow^* \boldsymbol{b}$

Petri nets

(d, T): $\quad d$ – dimension, $\quad T \subseteq \cancel{\mathbb{Z}^d} \quad \mathbb{N}^d \times \mathbb{N}^d$ (finite)

Petri nets

(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2$, $T = \{t'_1, t'_2, t'_3\}$

Petri nets

(d, T): $\quad d$ – dimension, $\quad T \subseteq \cancel{\mathbb{Z}^d} \quad \mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2,\ T = \{t'_1, t'_2, t'_3\}$

instead of $t_1 = (1, 2) \qquad t'_1 = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1) \qquad t'_2 = (0, 1) \times (1, 0)$

instead of $t_3 = (-3, -1) \qquad t'_3 = (3, 1) \times (0, 0)$

Petri nets

(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2$, $T = \{t'_1, t'_2, t'_3\}$

instead of $t_1 = (1, 2)$ $t'_1 = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1)$ $t'_2 = (0, 1) \times (1, 0)$ $(,) =$ -$(,) + (,)$

instead of $t_3 = (-3, -1)$ $t'_3 = (3, 1) \times (0, 0)$

Petri nets

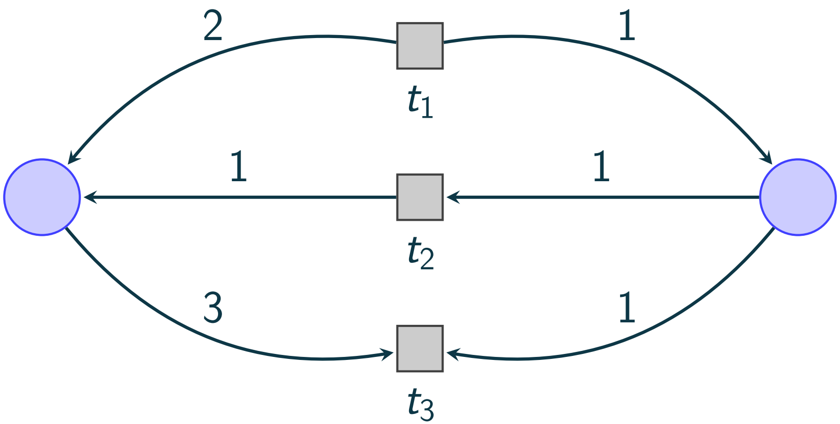
(d, T): d – dimension, $T \subseteq \cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2$, $T = \{t_1', t_2', t_3'\}$

instead of $t_1 = (1, 2)$ $t_1' = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1)$ $t_2' = (0, 1) \times (1, 0)$ $(,) =$ -$(,) + (,)$

instead of $t_3 = (-3, -1)$ $t_3' = (3, 1) \times (0, 0)$



Petri nets

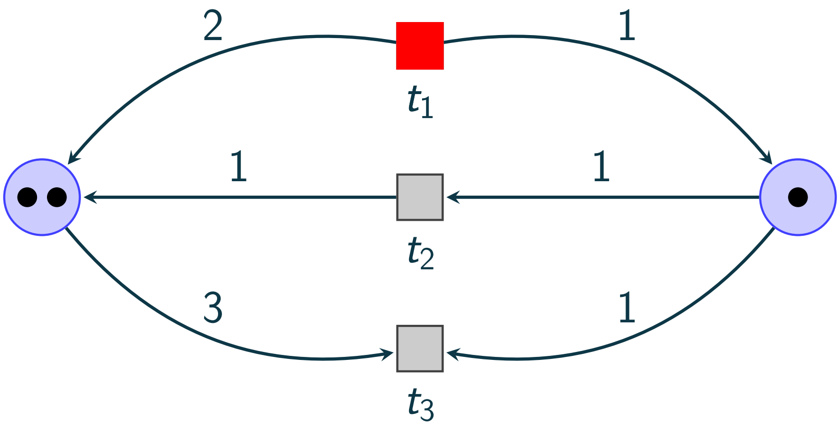
(d, T): $\quad d$ – dimension, $\quad T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2,\ T = \{t_1', t_2', t_3'\}$

instead of $t_1 = (1, 2) \quad t_1' = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1) \quad t_2' = (0, 1) \times (1, 0) \qquad (,) = \text{-}(,) + (,)$

instead of $t_3 = (-3, -1) \quad t_3' = (3, 1) \times (0, 0)$



Petri nets

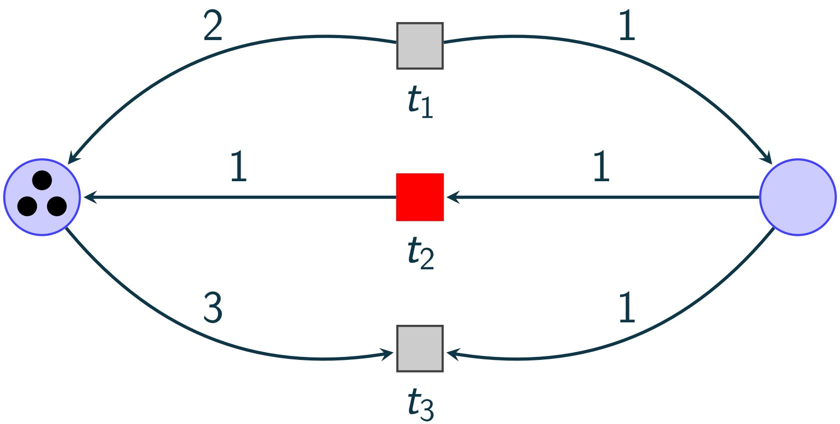
(d, T): d – dimension, $T \subseteq \cancel{\mathbb{Z}^d}\ \mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2,\ T = \{t'_1, t'_2, t'_3\}$

instead of $t_1 = (1, 2)$ $t'_1 = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1)$ $t'_2 = (0, 1) \times (1, 0)$ $(,) =$ -$(,) + (,)$

instead of $t_3 = (-3, -1)$ $t'_3 = (3, 1) \times (0, 0)$



Petri nets

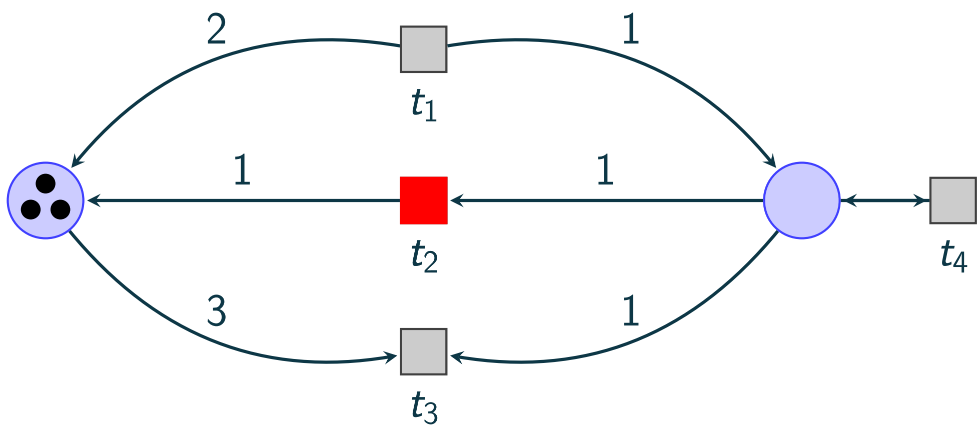
(d, T): d – dimension, $T \subseteq \cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2,\ T = \{t'_1, t'_2, t'_3\}$

instead of $t_1 = (1, 2)$ $t'_1 = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1)$ $t'_2 = (0, 1) \times (1, 0)$ $(,) =$ -$(,) + (,)$

instead of $t_3 = (-3, -1)$ $t'_3 = (3, 1) \times (0, 0)$



Petri nets

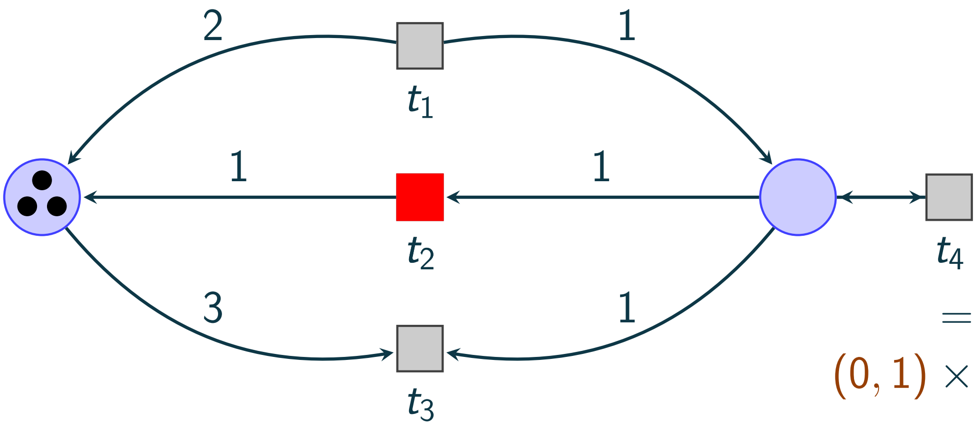
(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}}^d$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

Previous example: $d = 2$, $T = \{t'_1, t'_2, t'_3\}$

instead of $t_1 = (1, 2)$ $t'_1 = (0, 0) \times (1, 2)$

instead of $t_2 = (1, -1)$ $t'_2 = (0, 1) \times (1, 0)$ $(,) =$ -$(,) + (,)$

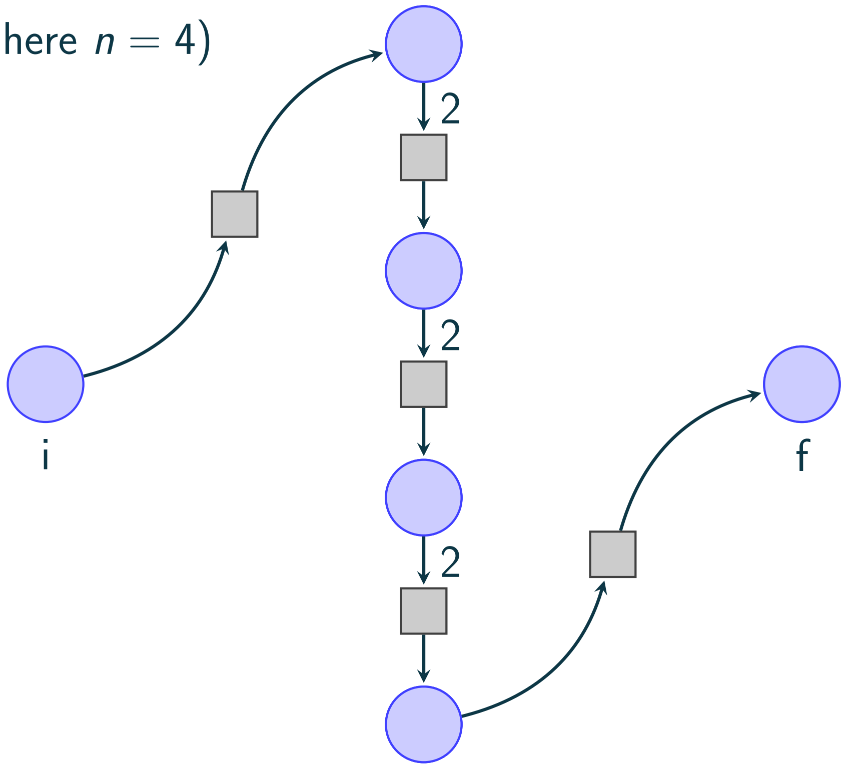
instead of $t_3 = (-3, -1)$ $t'_3 = (3, 1) \times (0, 0)$



$t_4 = (0, 1) \times (0, 1)$

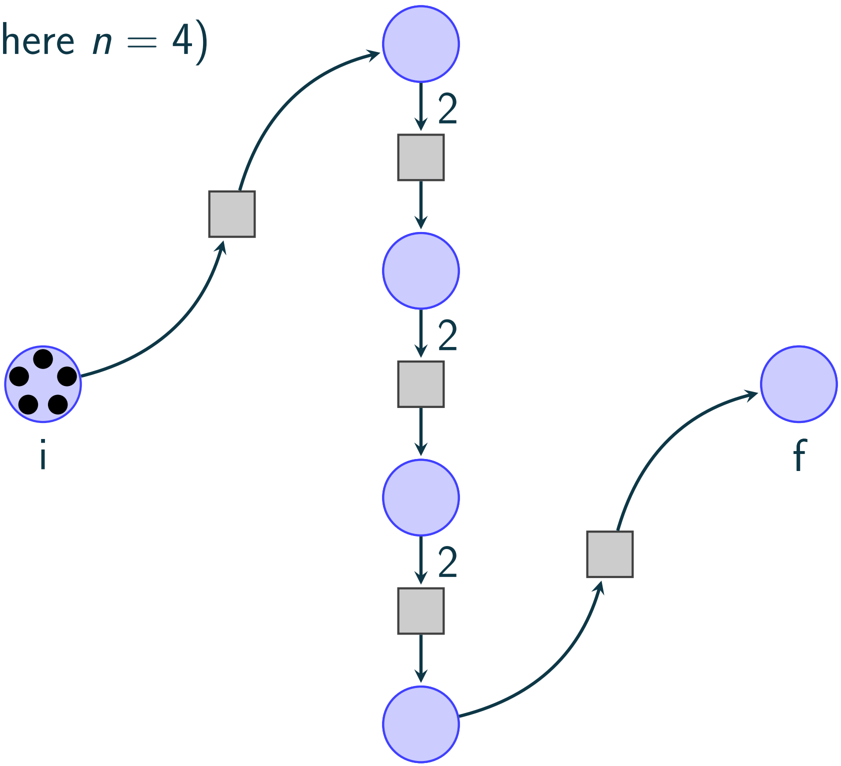
A more interesting example

$d = 2 + n \quad$ (here $n = 4$)



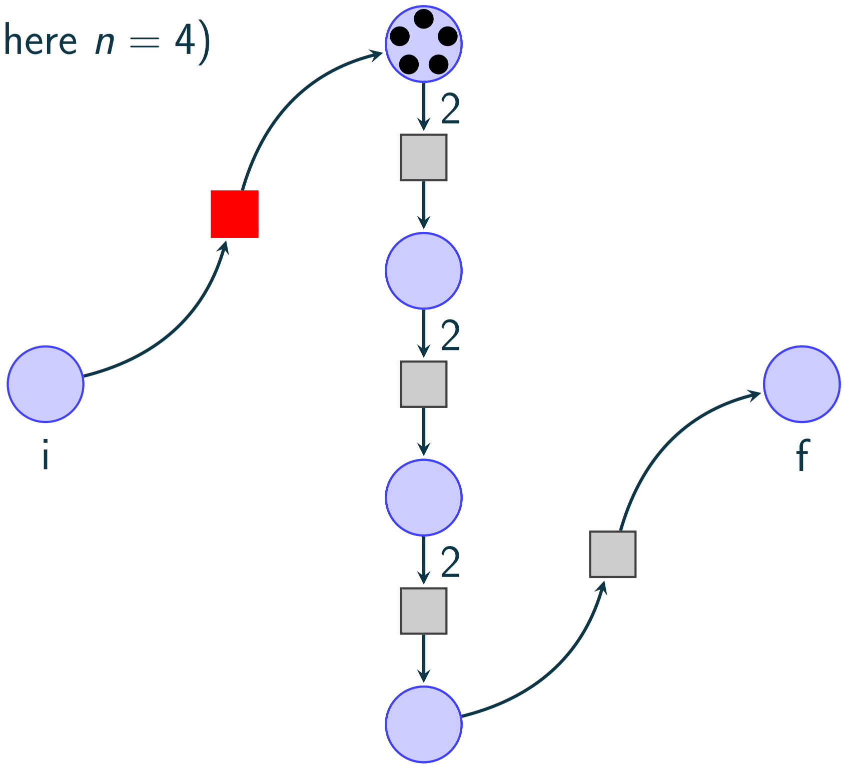
A more interesting example

$d = 2 + n \quad$ (here $n = 4$)



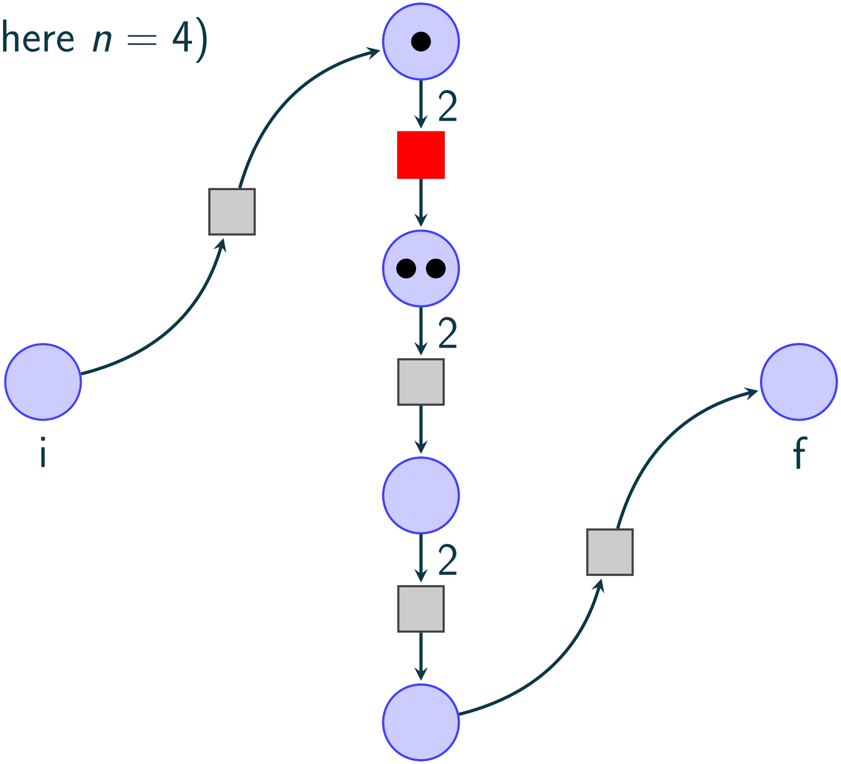
A more interesting example

$d = 2 + n \quad$ (here $n = 4$)



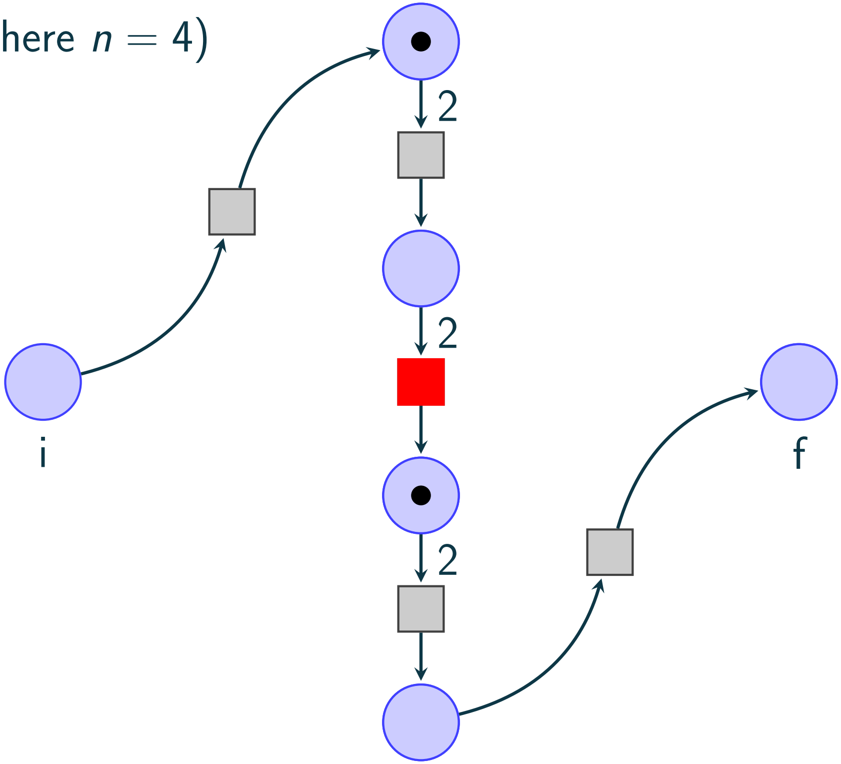
A more interesting example

$d = 2 + n \quad$ (here $n = 4$)



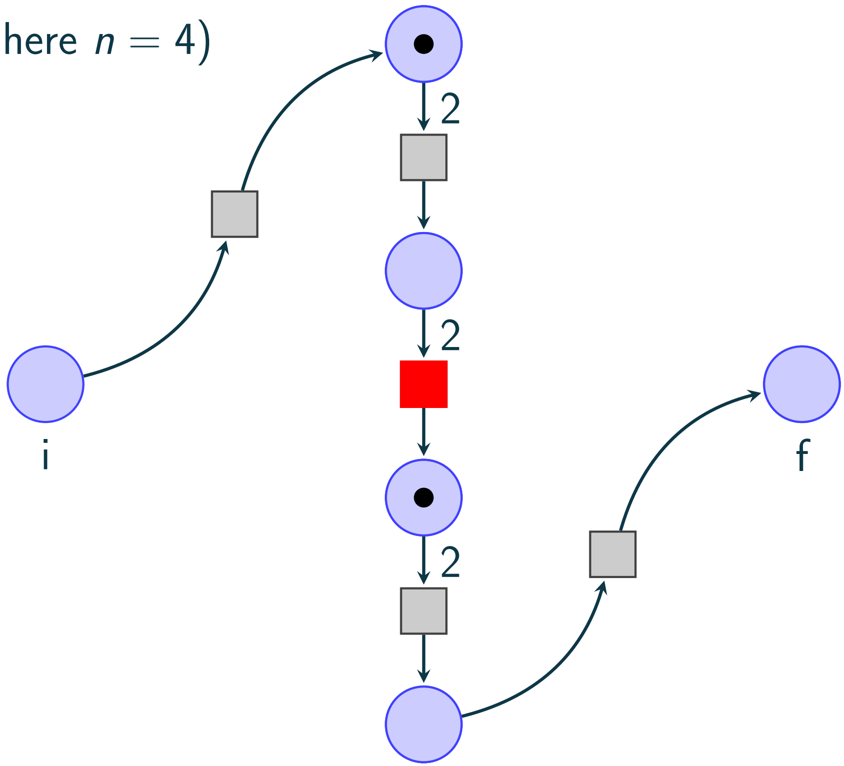
A more interesting example

$d = 2 + n \quad \text{(here } n = 4\text{)}$



A more interesting example

$d = 2 + n \quad$ (here $n = 4$)



If we start with k tokens in i

then we reach f only if $k \geq 2^{n-1}$

Reachability and coverability problems

Reachability problem: determine $\boldsymbol{a} \rightarrow^* \boldsymbol{b}$?

Reachability and coverability problems

Reachability problem: determine $\boldsymbol{a} \to^* \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Reachability and coverability problems

Reachability problem: determine $\boldsymbol{a} \rightarrow^* \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \rightarrow^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $\boldsymbol{a} \rightarrow^* \boldsymbol{b}' \geq \boldsymbol{b}$?

Reachability and coverability problems

Reachability problem: determine $a \to^* b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $a \to^* b' \geq b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

Reachability and coverability problems

Reachability problem: determine $\boldsymbol{a} \to^* \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $\boldsymbol{a} \to^* \boldsymbol{b}' \geq \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

- Both problems are EXPSPACE-hard [Lipton, 1976]

Reachability and coverability problems

Reachability problem: determine $\boldsymbol{a} \rightarrow^* \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \rightarrow^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $\boldsymbol{a} \rightarrow^* \boldsymbol{b}' \geq \boldsymbol{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

- Both problems are EXPSPACE-hard [Lipton, 1976]
- Coverability is in EXPSPACE [Rackoff, 1978]

Reachability and coverability problems

Reachability problem: determine $a \to^* b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $a \to^* b' \geq b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

- Both problems are EXPSPACE-hard [Lipton, 1976]
- Coverability is in EXPSPACE [Rackoff, 1978]
- Reachability is decidable [Mayr, 1981]

Reachability and coverability problems

Reachability problem: determine $\mathbf{a} \to^* \mathbf{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $\mathbf{a} \to^* \mathbf{b}' \geq \mathbf{b}$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

- Both problems are EXPSPACE-hard [Lipton, 1976]
- Coverability is in EXPSPACE [Rackoff, 1978]
- Reachability is decidable [Mayr, 1981]
- Reachability is in Ackermann [Leroux and Schmitz, 2019]

Reachability and coverability problems

Reachability problem: determine $a \to^* b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \to^* (0, 0, 0, 0, 0, 1) \text{ iff } k = 2^{n-1}$$

Coverability problem: determine $a \to^* b' \geq b$?

In the previous example

$$(k, 0, 0, 0, 0, 0) \text{ can cover } (0, 0, 0, 0, 0, 1) \text{ iff } k \geq 2^{n-1}$$

- Both problems are EXPSPACE-hard [Lipton, 1976]
- Coverability is in EXPSPACE [Rackoff, 1978]
- Reachability is decidable [Mayr, 1981]
- Reachability is in Ackermann [Leroux and Schmitz, 2019]
- Reachability is Ackermann-hard [Czerwiński and Orlikowski 2021], [Leroux, 2021]

Plan

1. Petri nets and reachability

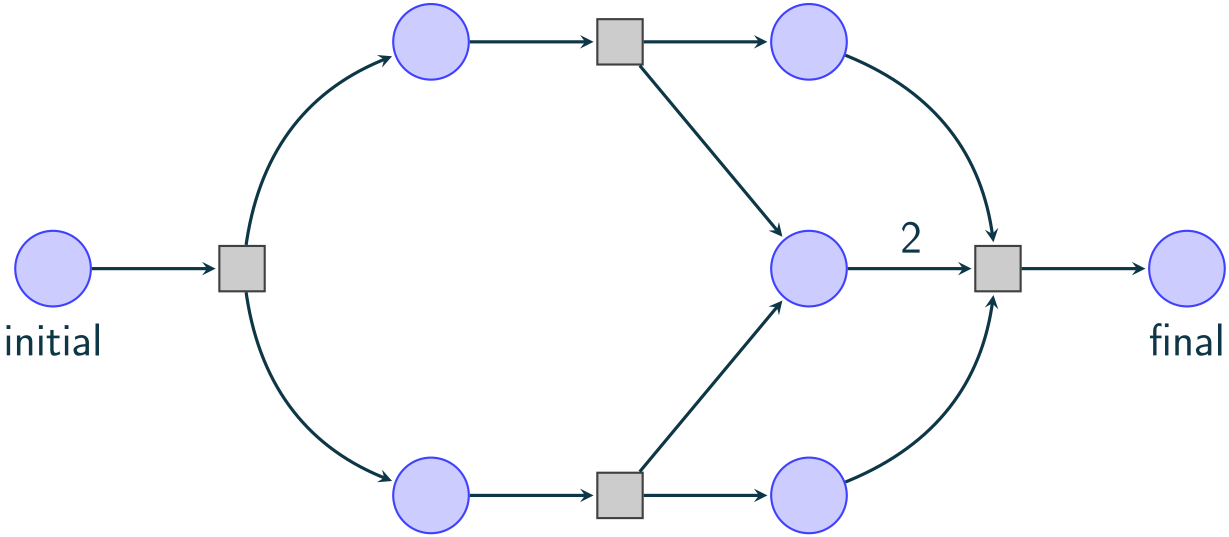
2. Workflow nets and soundness

3. Some proofs

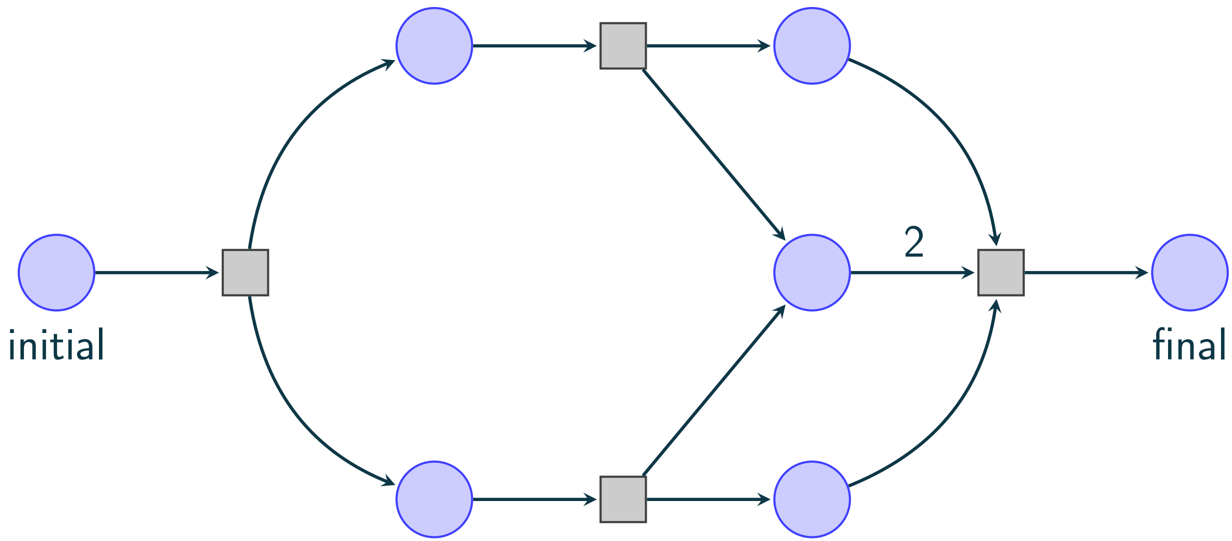
4. Implementation

Workflow nets

Workflow nets



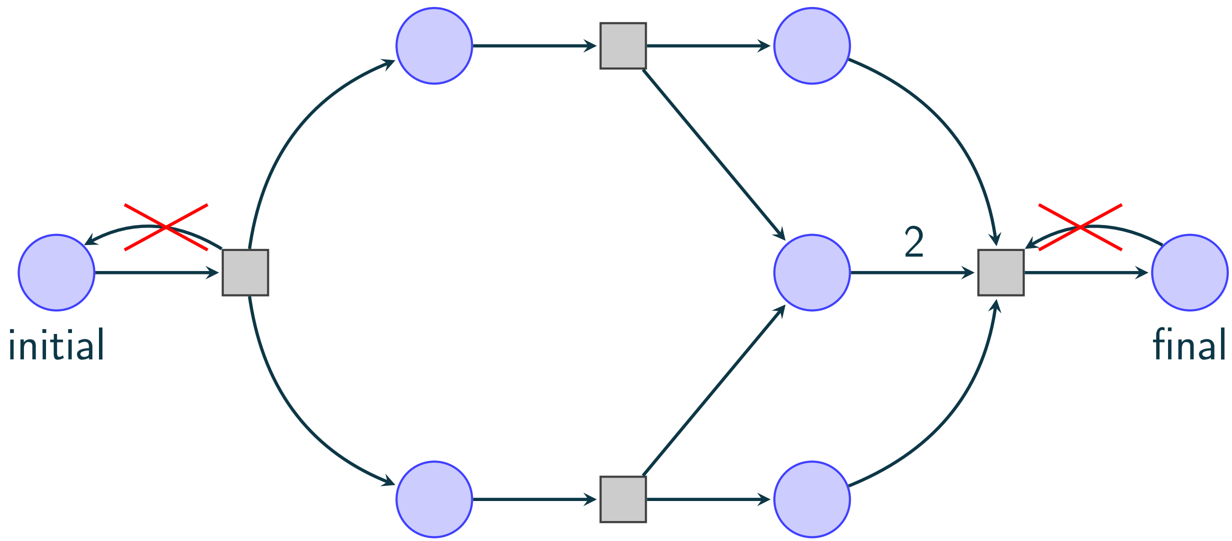
Workflow nets



Workflow nets are Petri nets such that:

- Two places are distinguished: initial and final

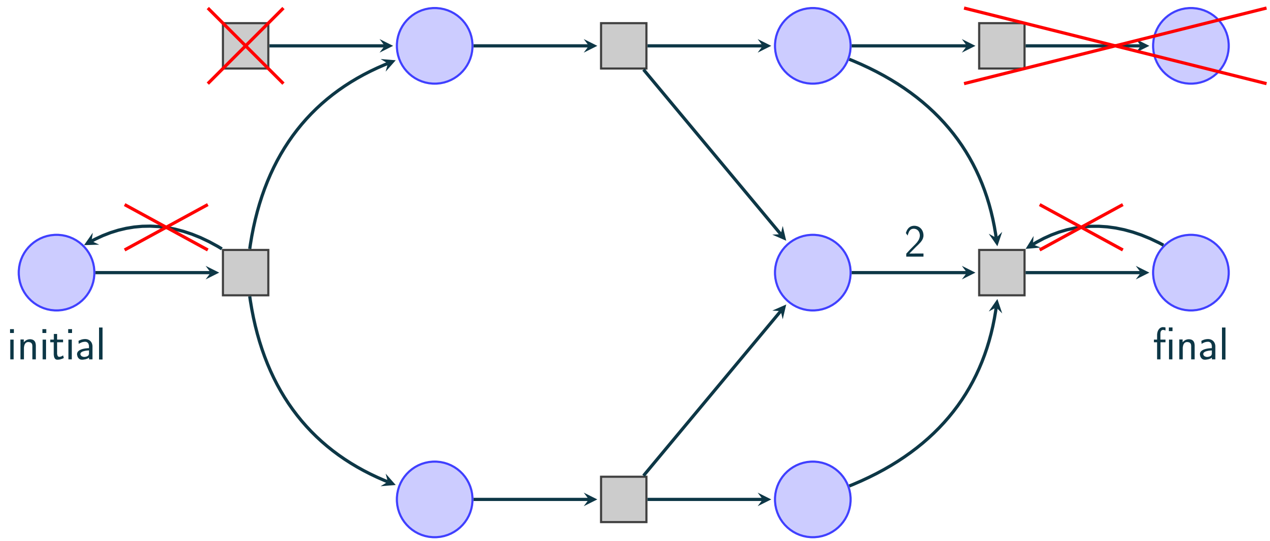
Workflow nets



Workflow nets are Petri nets such that:

- Two places are distinguished: initial and final
- No ingoing edges to initial and no outgoing edges from final

Workflow nets



Workflow nets are Petri nets such that:

- Two places are distinguished: initial and final
- No ingoing edges to initial and no outgoing edges from final
- All places and transitions are on a path from initial to final

Business processes

Suppose a professor wants to hire a student

Business processes

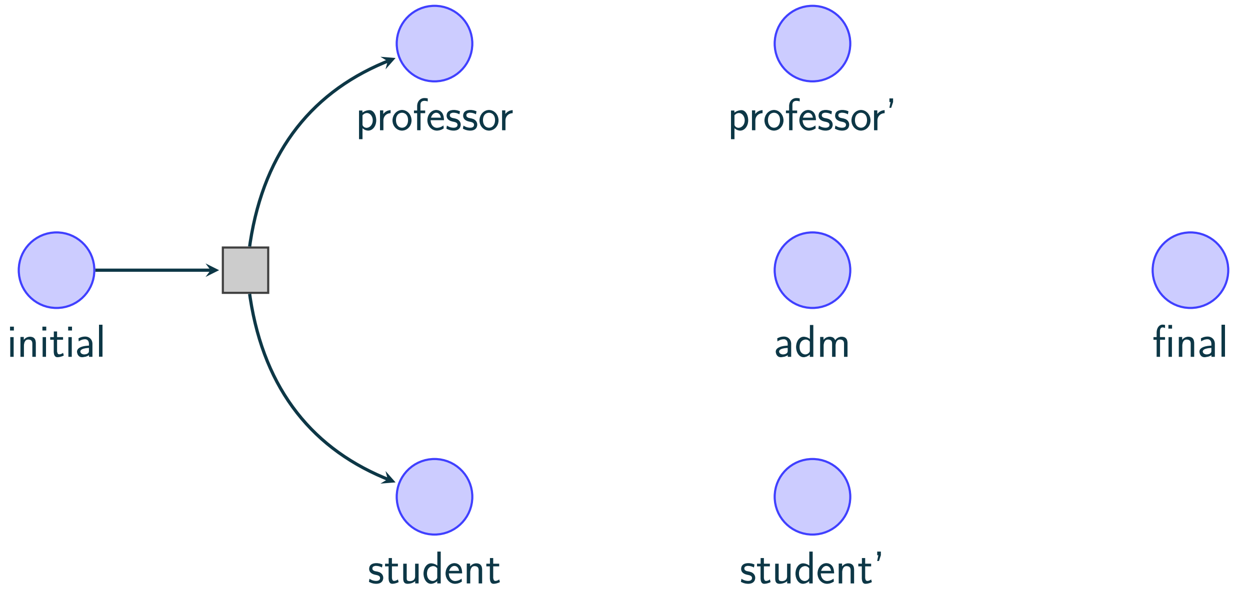
Suppose a professor wants to hire a student

- The student and the professor need to deal with the administration

Business processes

Suppose a professor wants to hire a student

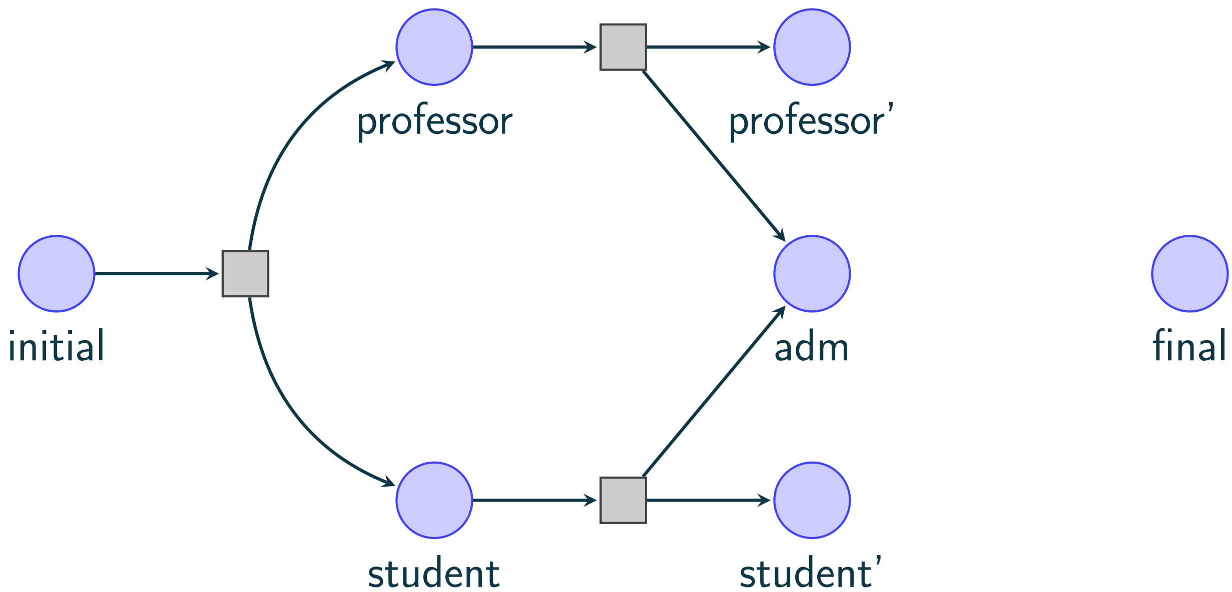
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

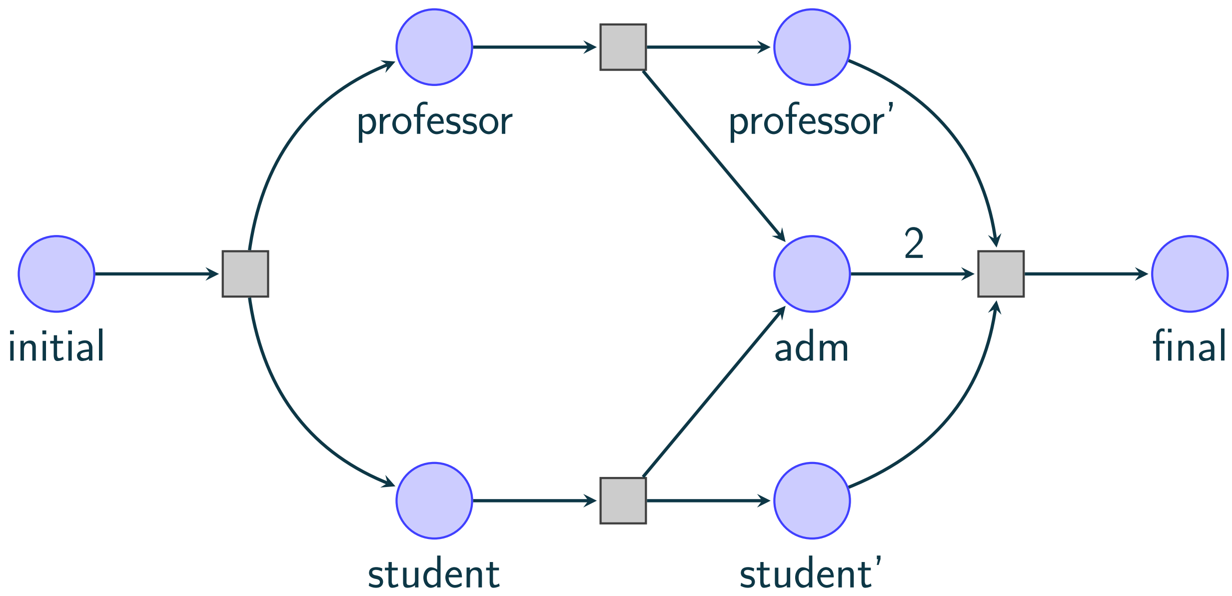
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

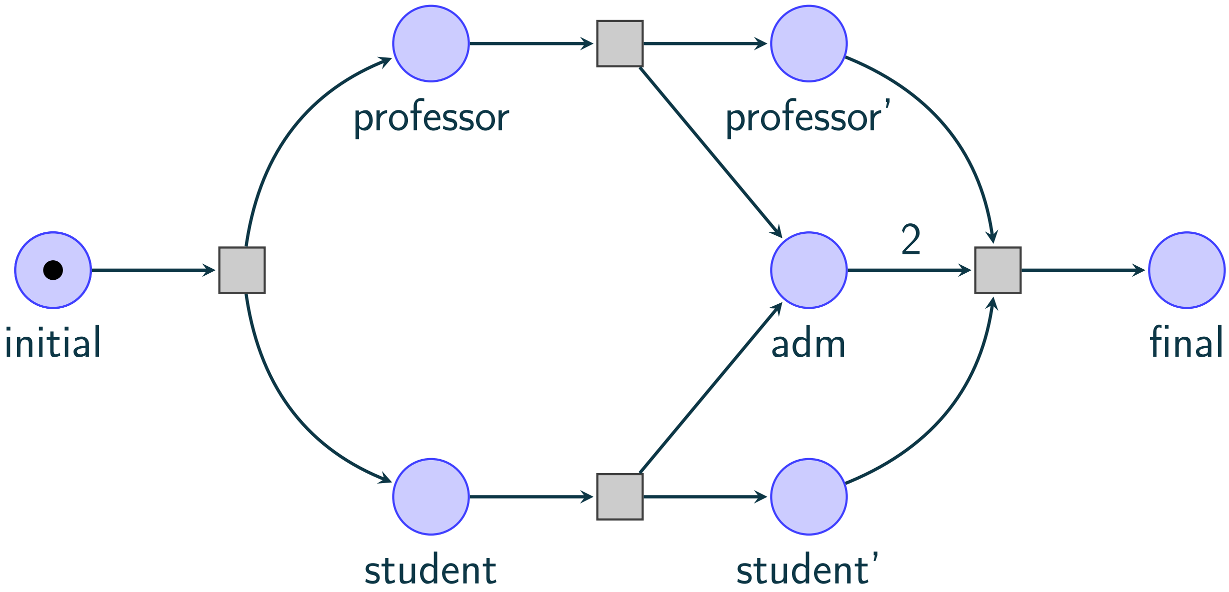
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

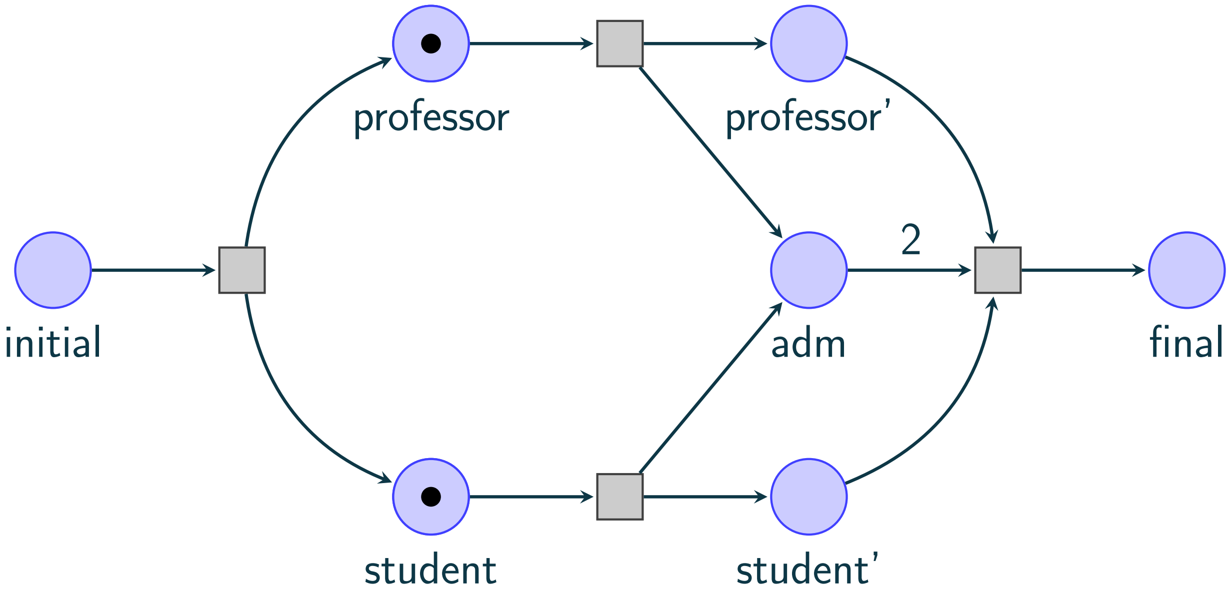
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

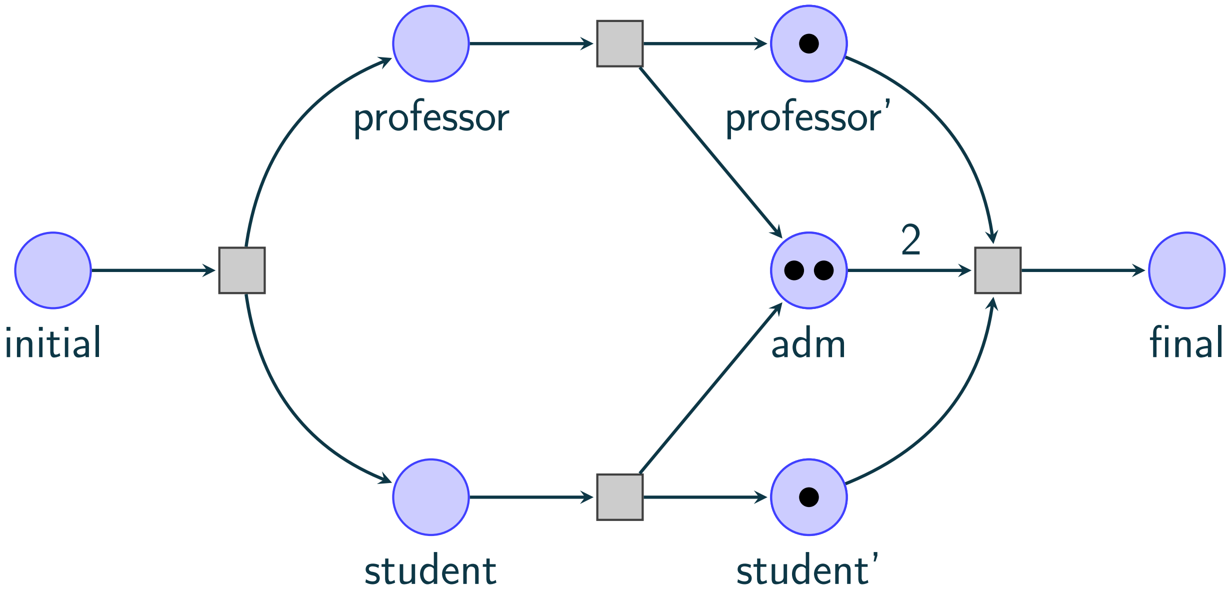
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

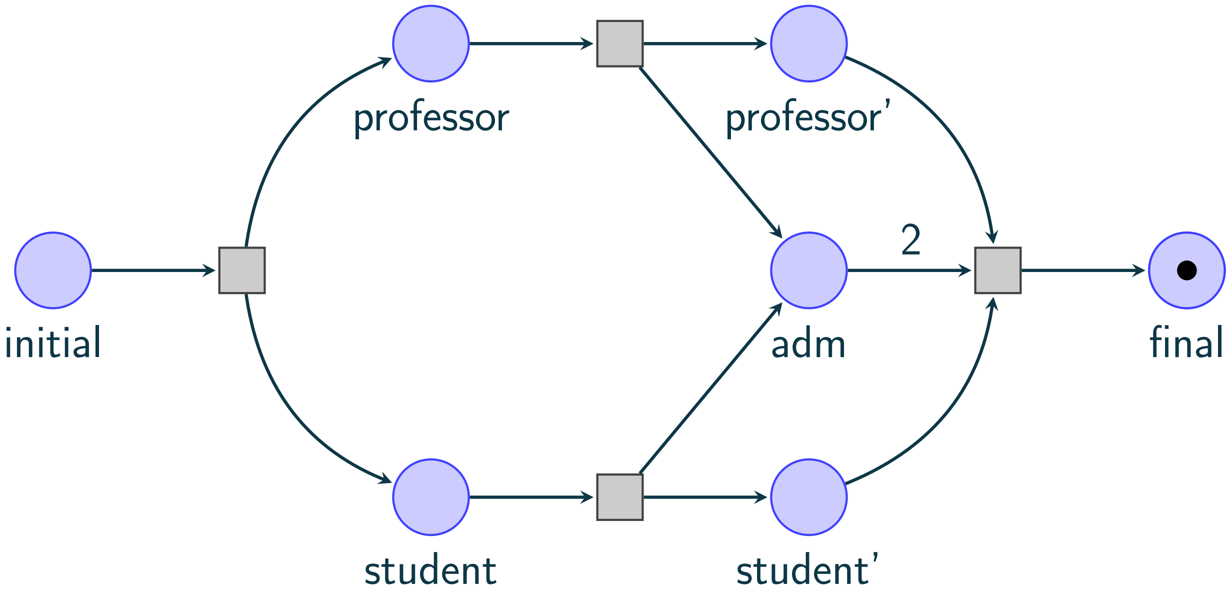
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

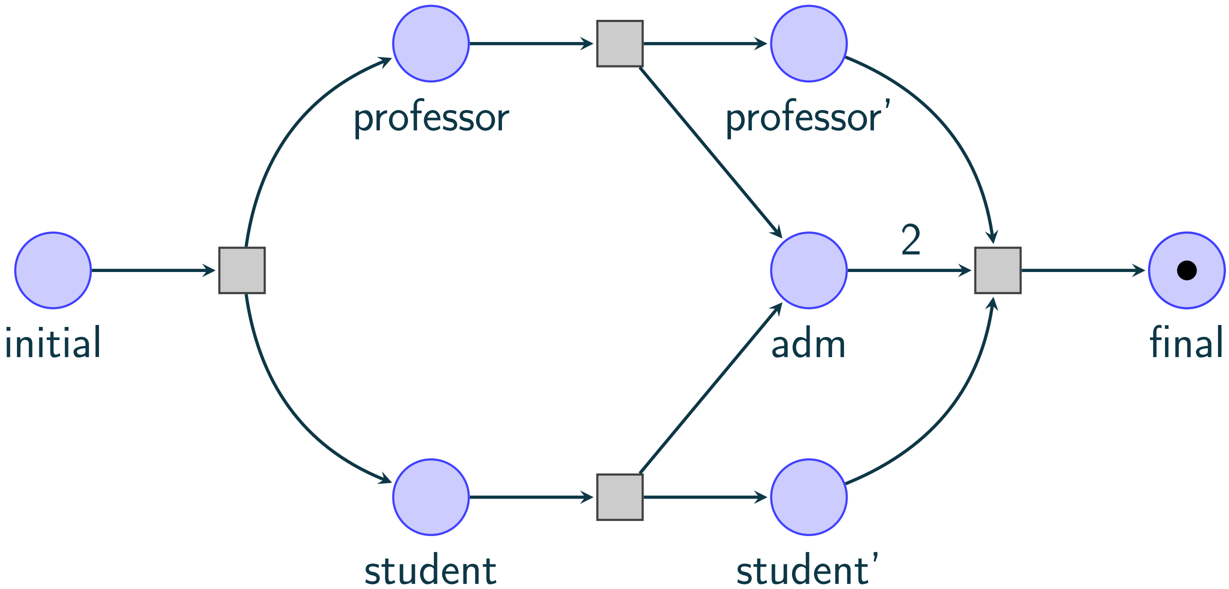
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

- The student and the professor need to deal with the administration



Behaves good even with many students at once

Soundness problems

What would we like to verify in the previous example?

Soundness problems

What would we like to verify in the previous example?

Given:

- workflow net (d, T)
- initial: $\{i : 1\} = (1, 0, 0, 0, 0, 0, 0)$ and final $\{f : 1\} = (0, 0, 0, 0, 0, 0, 1)$

Soundness problems

What would we like to verify in the previous example?

Given:

- workflow net (d, T)
- initial: $\{i : 1\} = (1, 0, 0, 0, 0, 0, 0)$ and final $\{f : 1\} = (0, 0, 0, 0, 0, 0, 1)$

Questions:

- Wherever we get from initial can we reach final?

Soundness problems

What would we like to verify in the previous example?

Given:

- workflow net (d, T)
- initial: $\{i : 1\} = (1, 0, 0, 0, 0, 0, 0)$ and final $\{f : 1\} = (0, 0, 0, 0, 0, 0, 1)$

Questions:

- Wherever we get from initial can we reach final?
- What if we change initial and final to $(k, 0, 0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0, 0, k)$?

Soundness problems

What would we like to verify in the previous example?

Given:

- workflow net (d, T)
- initial: $\{i : 1\} = (1, 0, 0, 0, 0, 0, 0)$ and final $\{f : 1\} = (0, 0, 0, 0, 0, 0, 1)$

Questions:

- Wherever we get from initial can we reach final?
- What if we change initial and final to $(k, 0, 0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0, 0, k)$?

These are k-soundness problem and soundness problem (1-soundness problem)

Soundness problems

What would we like to verify in the previous example?

Given:

- workflow net (d, T)
- initial: $\{i : 1\} = (1, 0, 0, 0, 0, 0, 0)$ and final $\{f : 1\} = (0, 0, 0, 0, 0, 0, 1)$

Questions:

- Wherever we get from initial can we reach final?
- What if we change initial and final to $(k, 0, 0, 0, 0, 0, 0)$ and $(0, 0, 0, 0, 0, 0, k)$?

These are k-soundness problem and soundness problem (1-soundness problem)

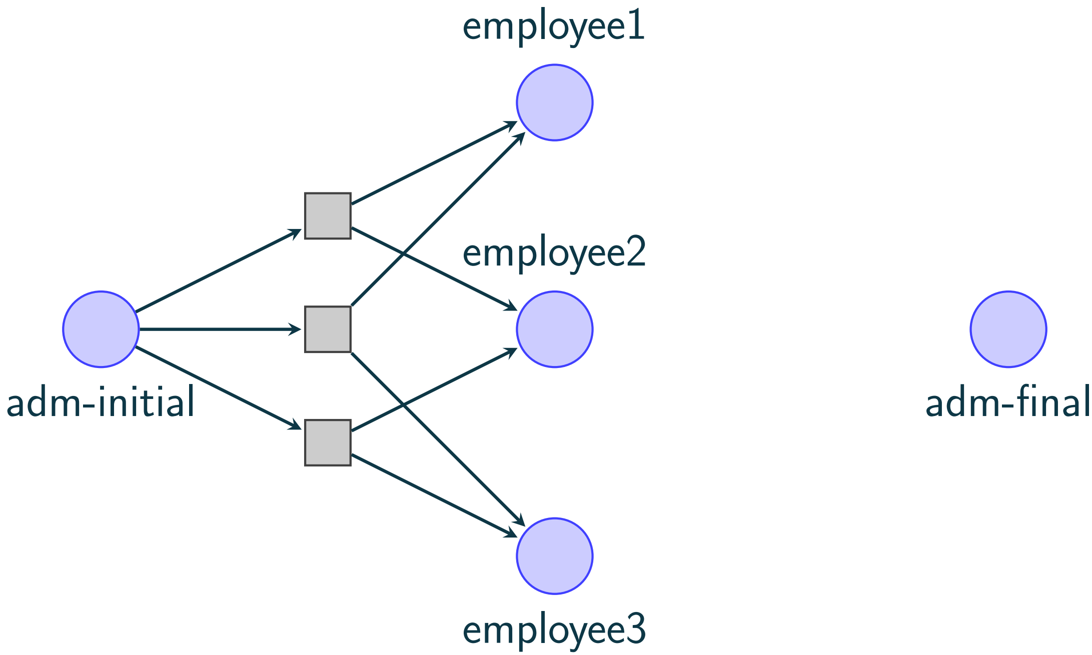
The previous example is sound and even k-sound for every $k > 0$

A more difficult example

Suppose the administration has it's own processes

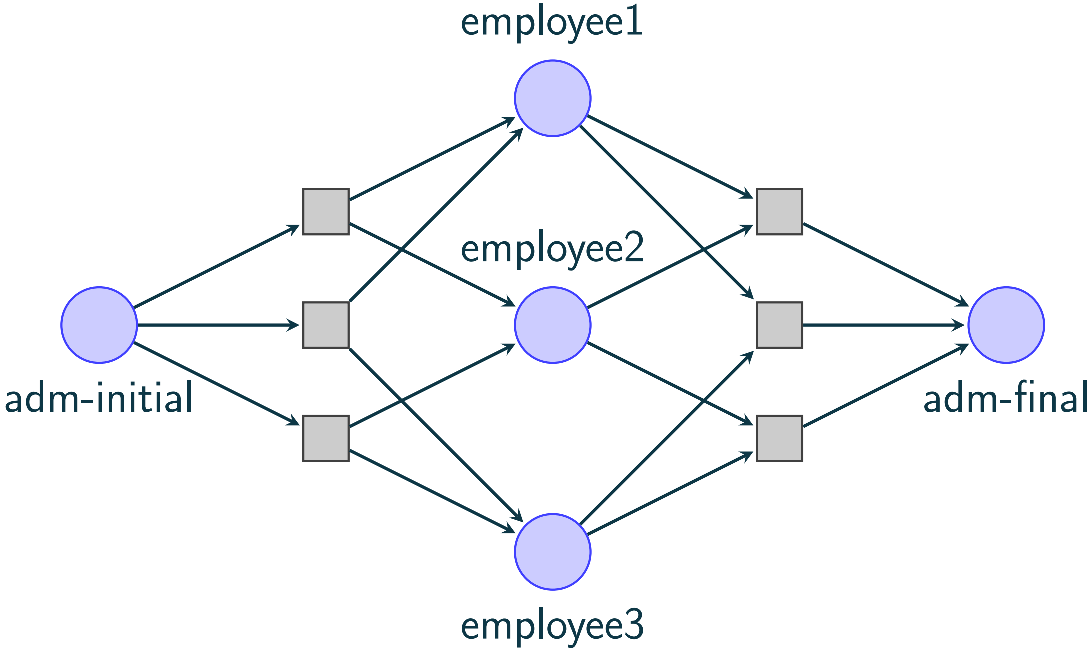
A more difficult example

Suppose the administration has it's own processes



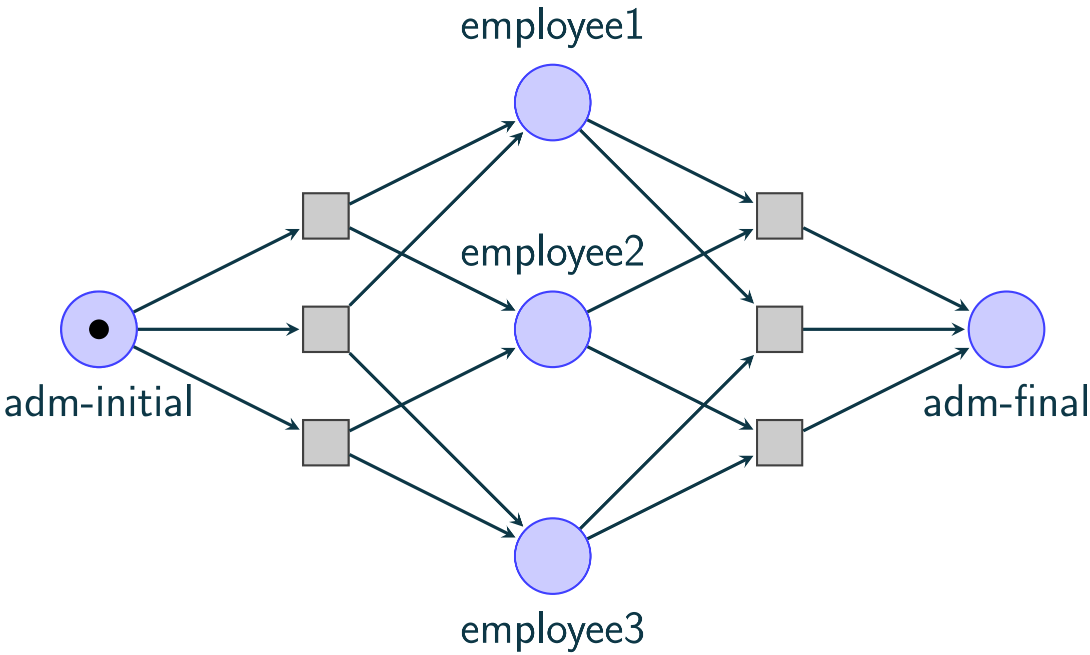
A more difficult example

Suppose the administration has it's own processes



A more difficult example

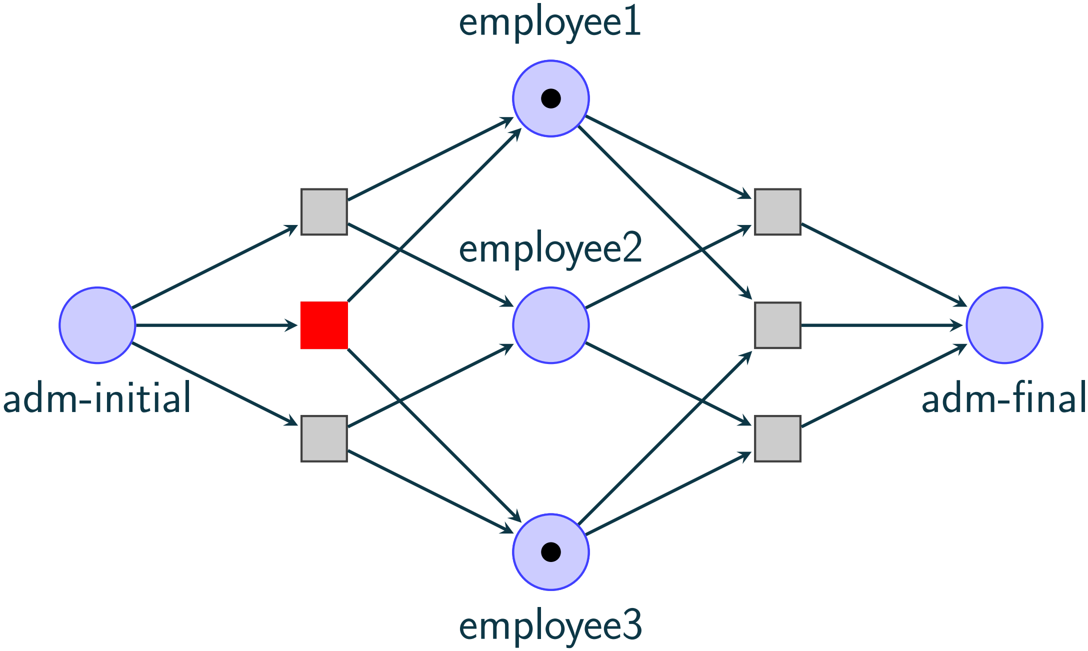
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

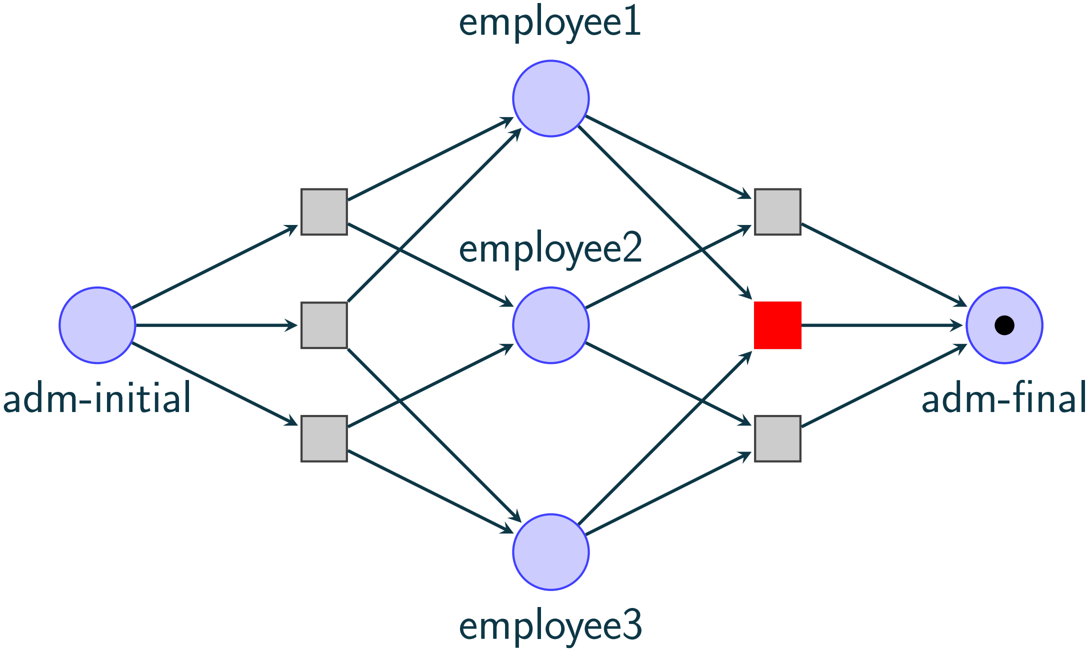
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

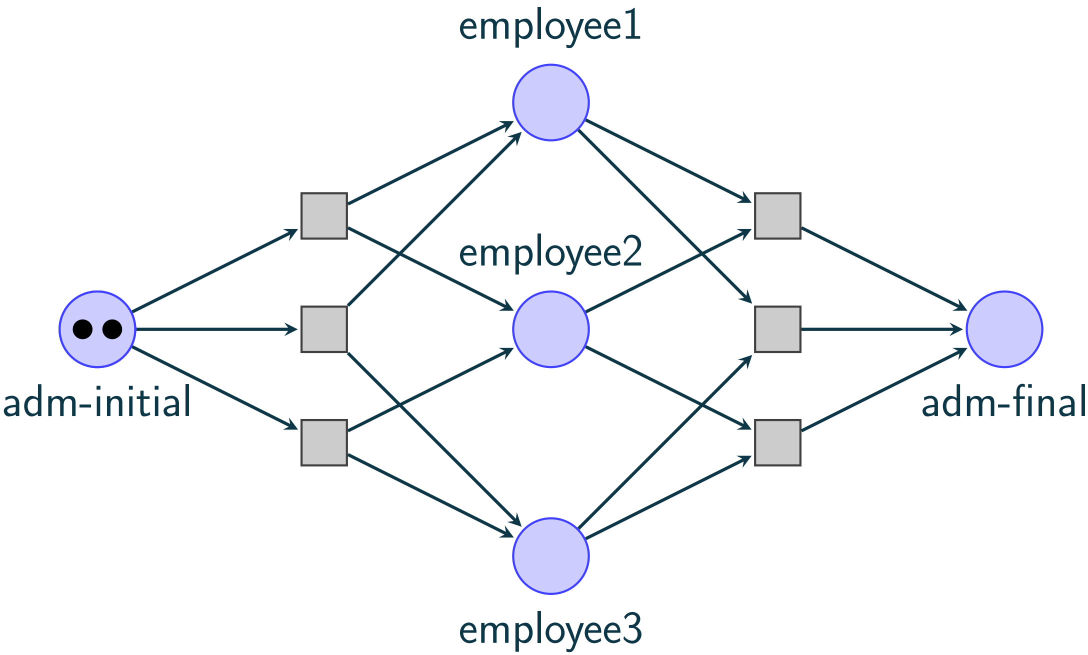
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

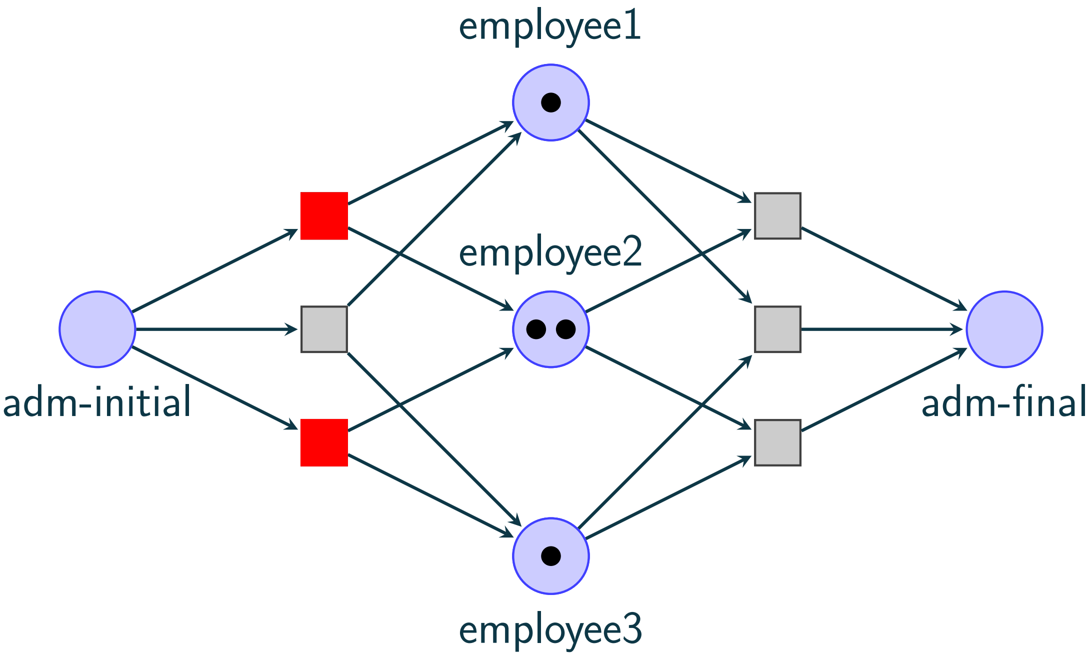
Suppose the administration has it's own processes



- One can verify it's sound
- But not 2-sound

A more difficult example

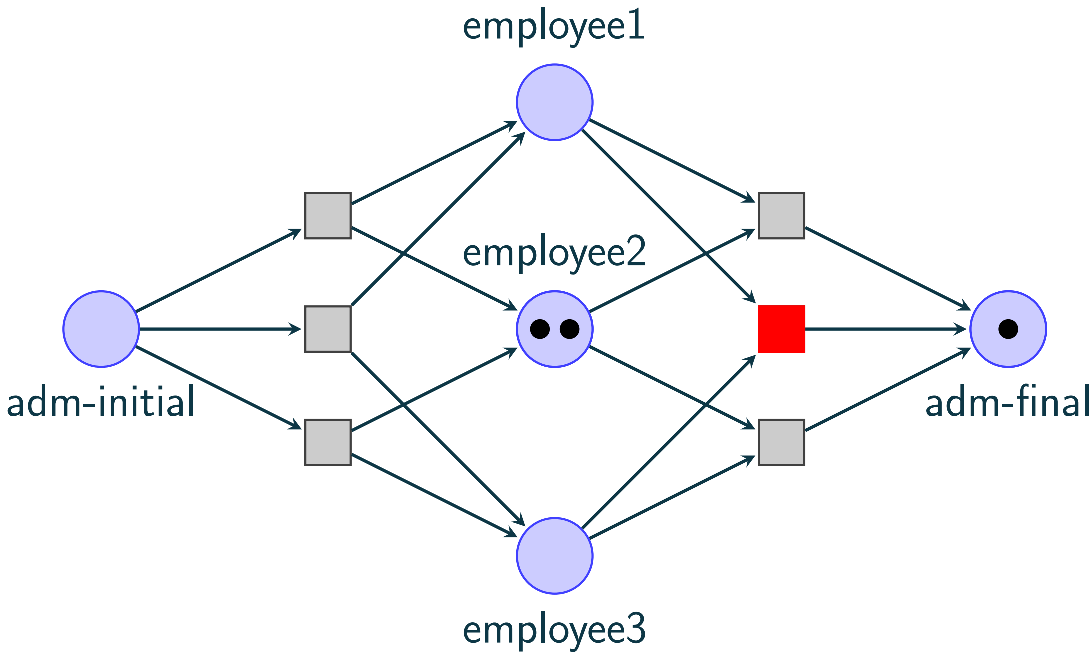
Suppose the administration has it's own processes



- One can verify it's sound
- But not 2-sound

A more difficult example

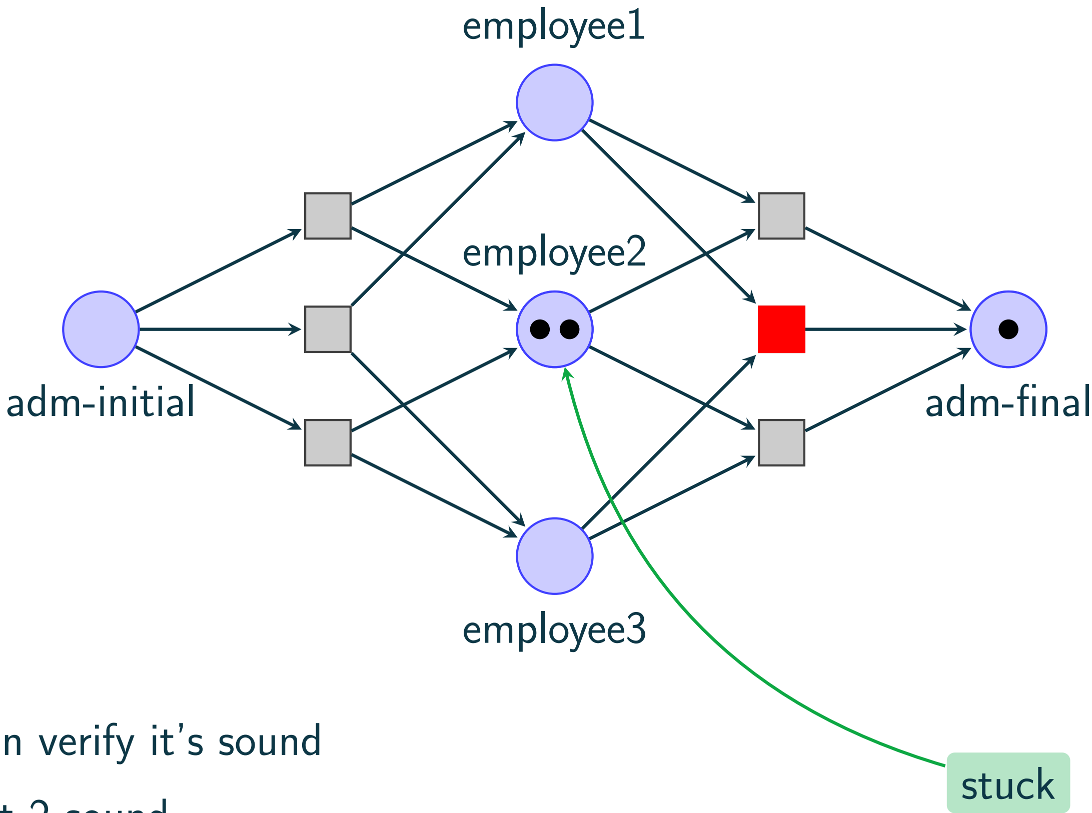
Suppose the administration has it's own processes



- One can verify it's sound
- But not 2-sound

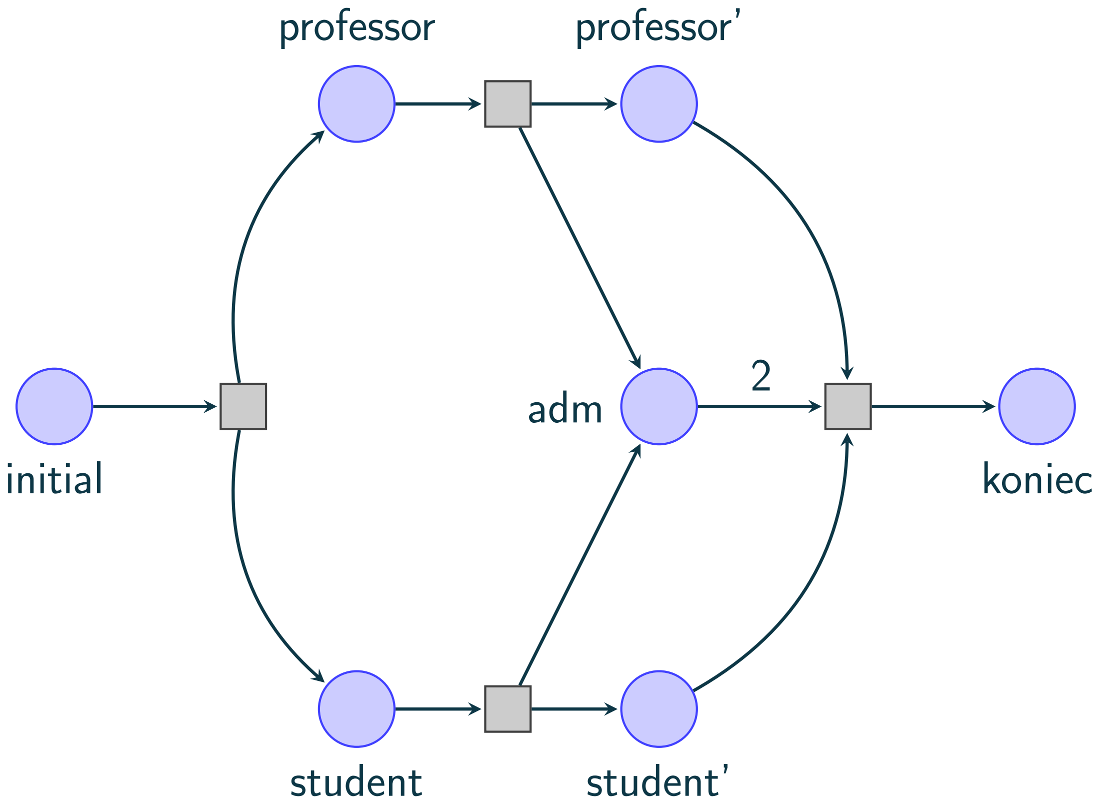
A more difficult example

Suppose the administration has it's own processes

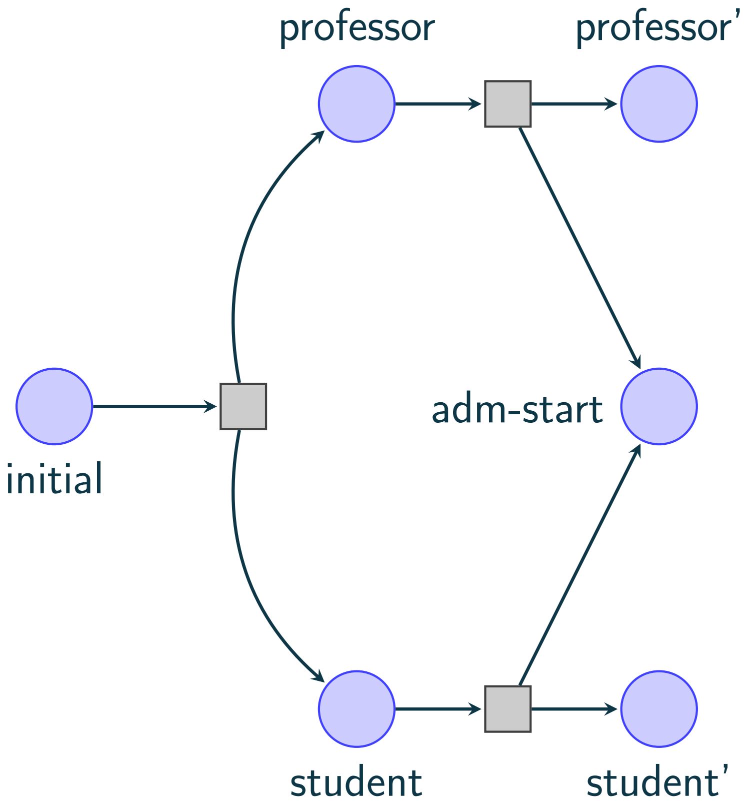


- One can verify it's sound
- But not 2-sound

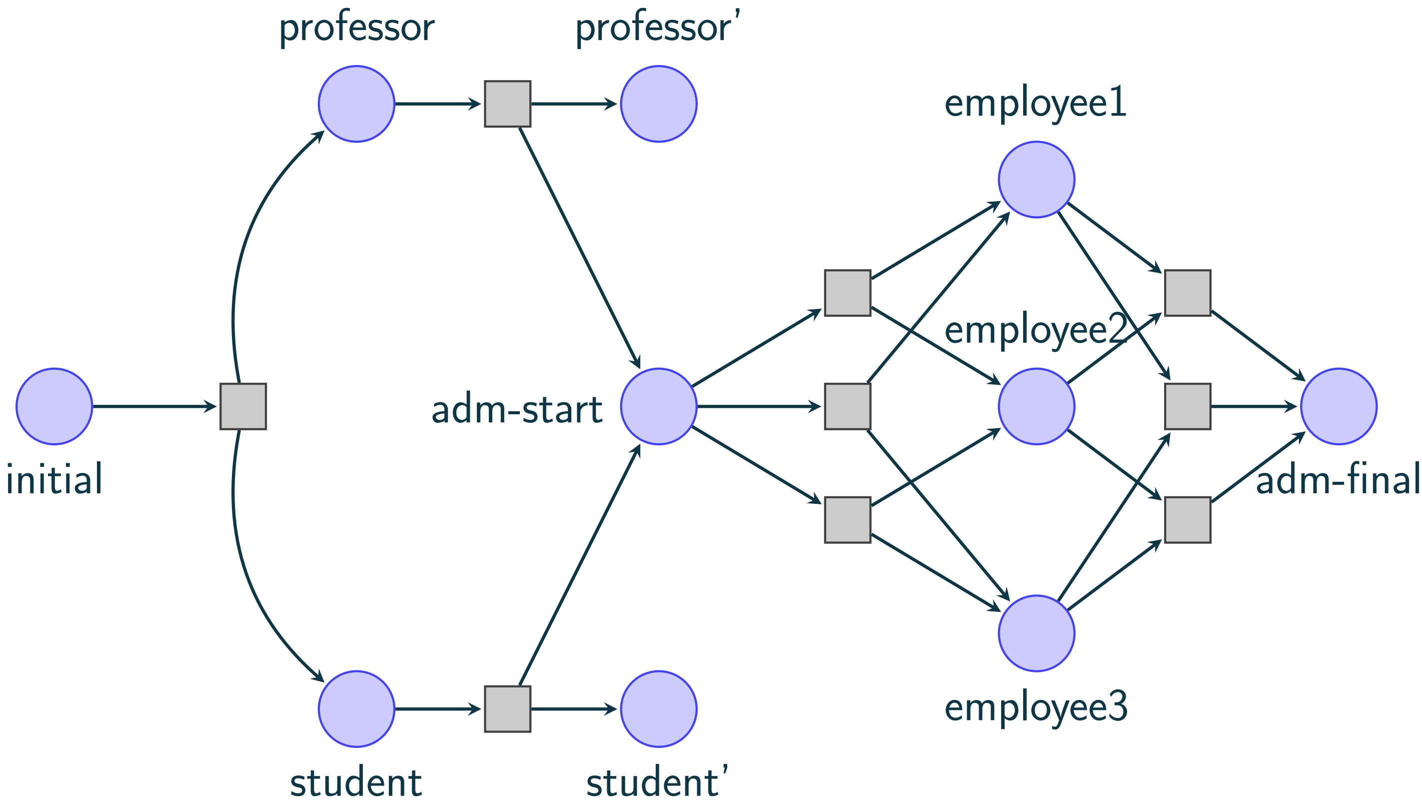
Combining the examples



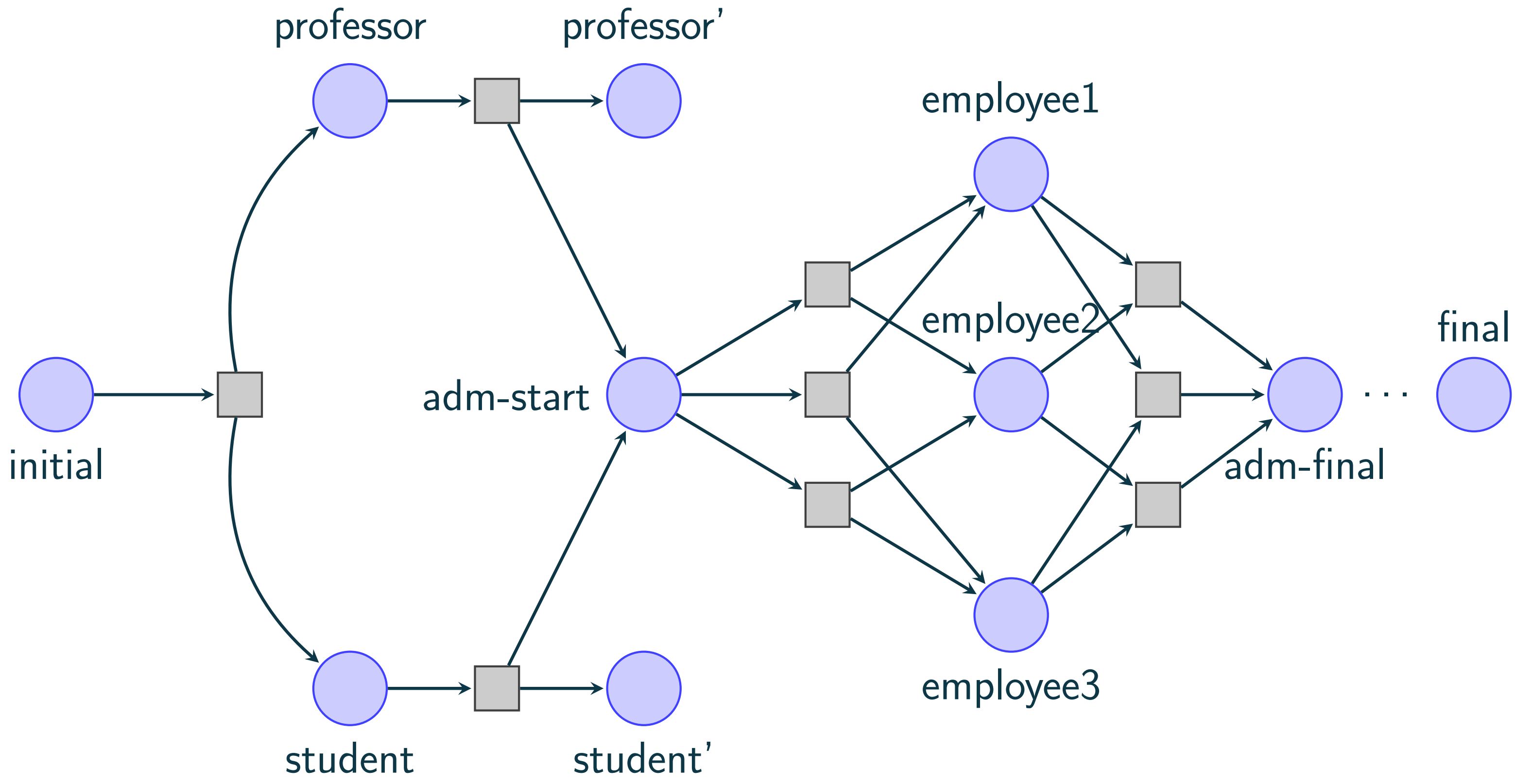
Combining the examples



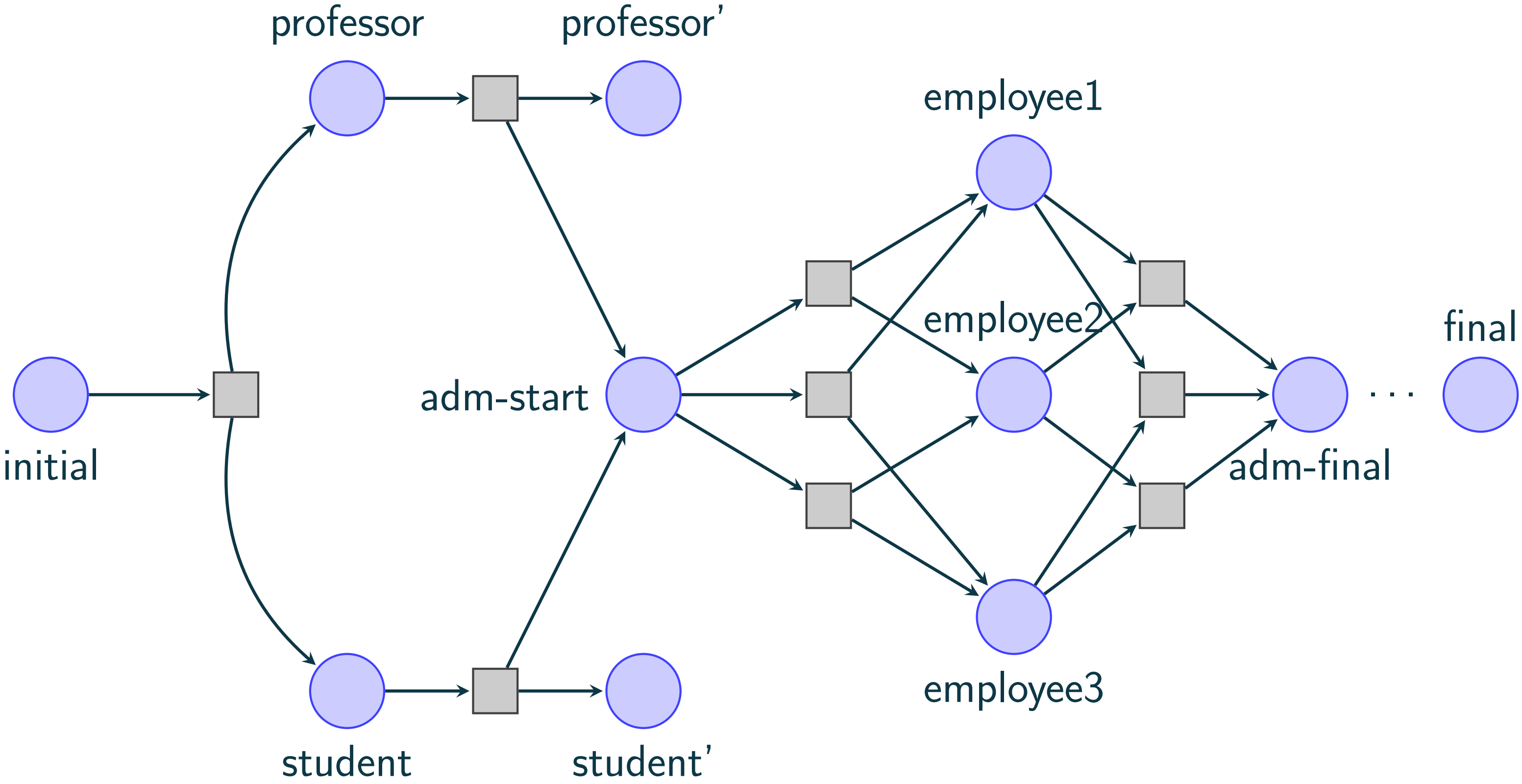
Combining the examples



Combining the examples

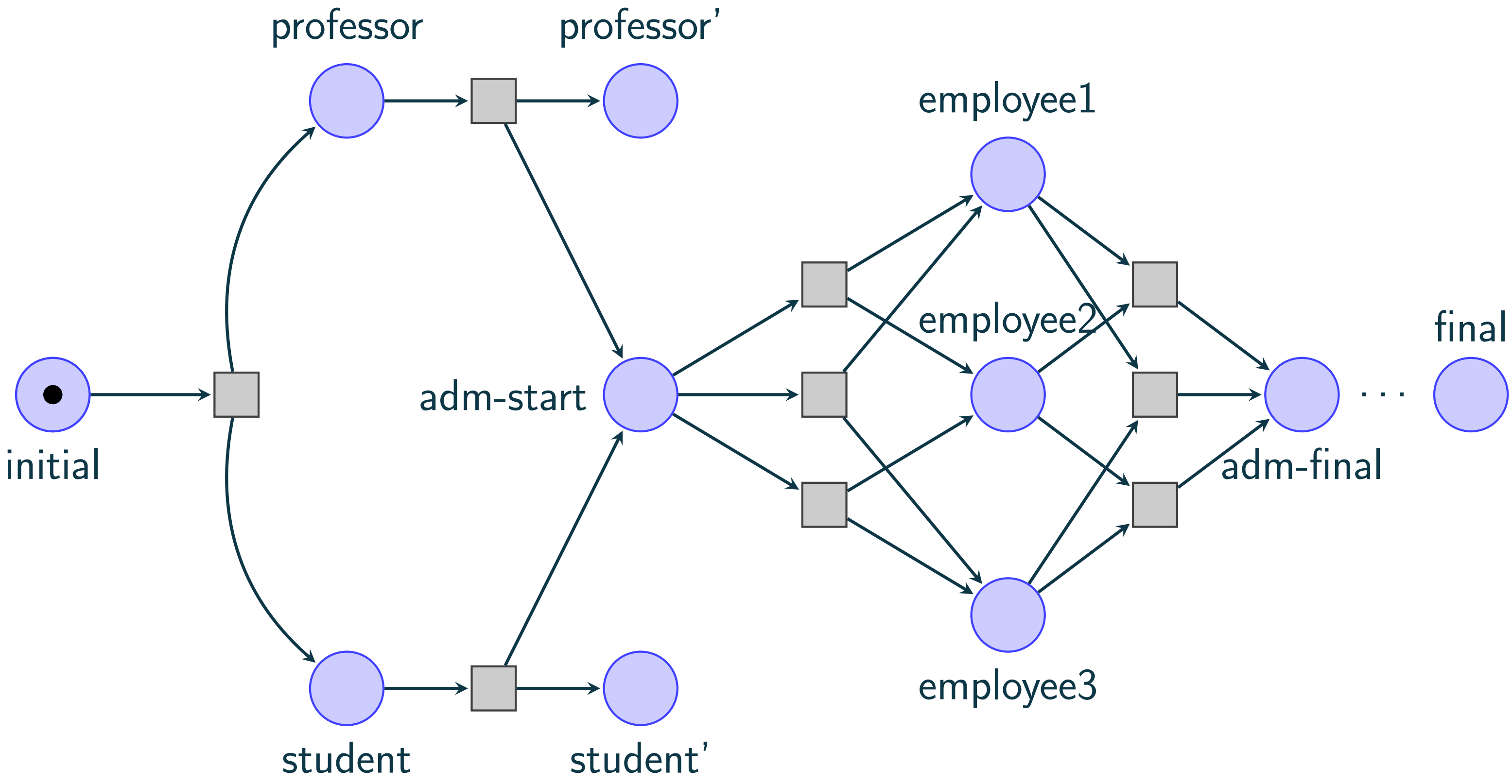


Combining the examples



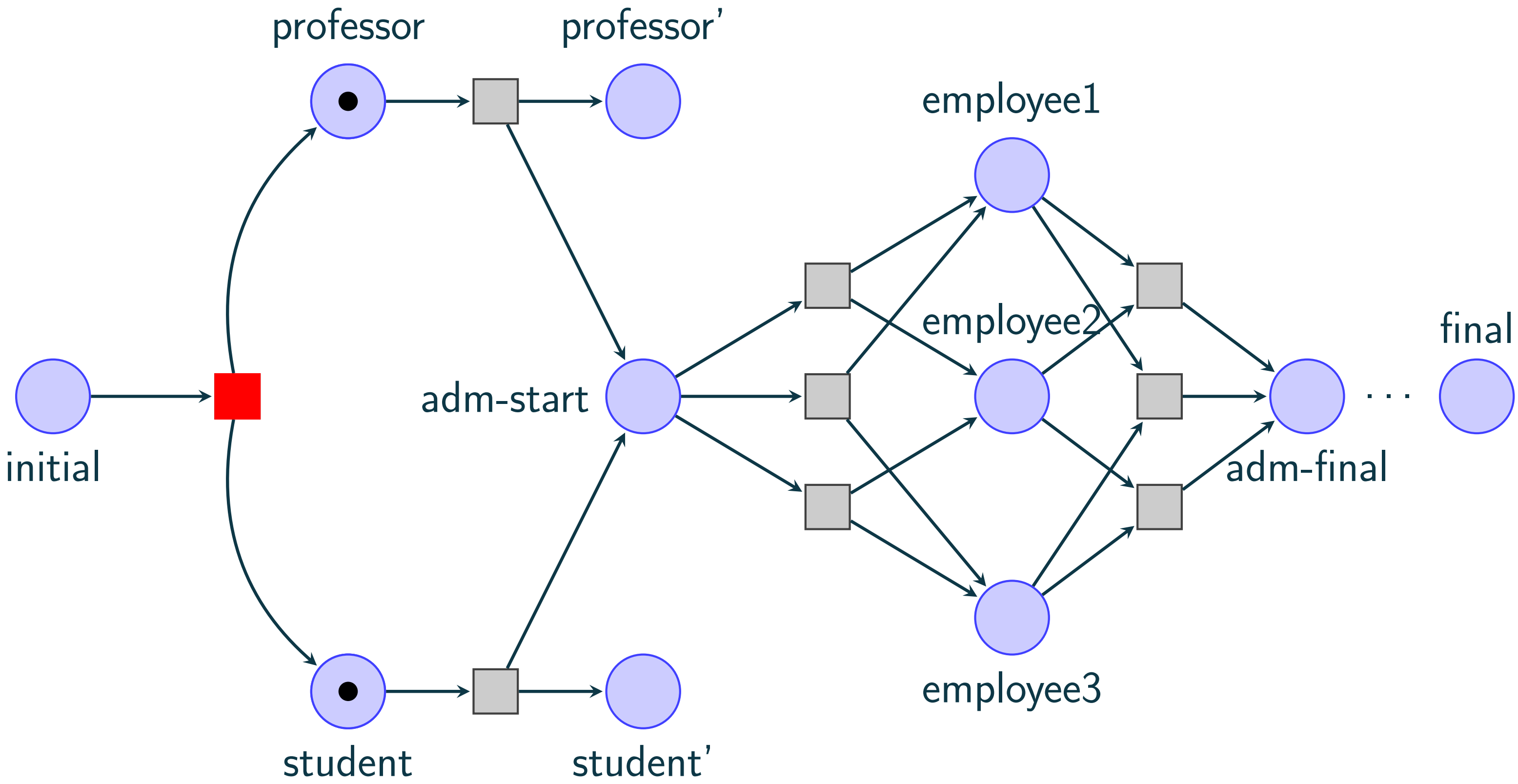
- Recall: both examples were sound

Combining the examples



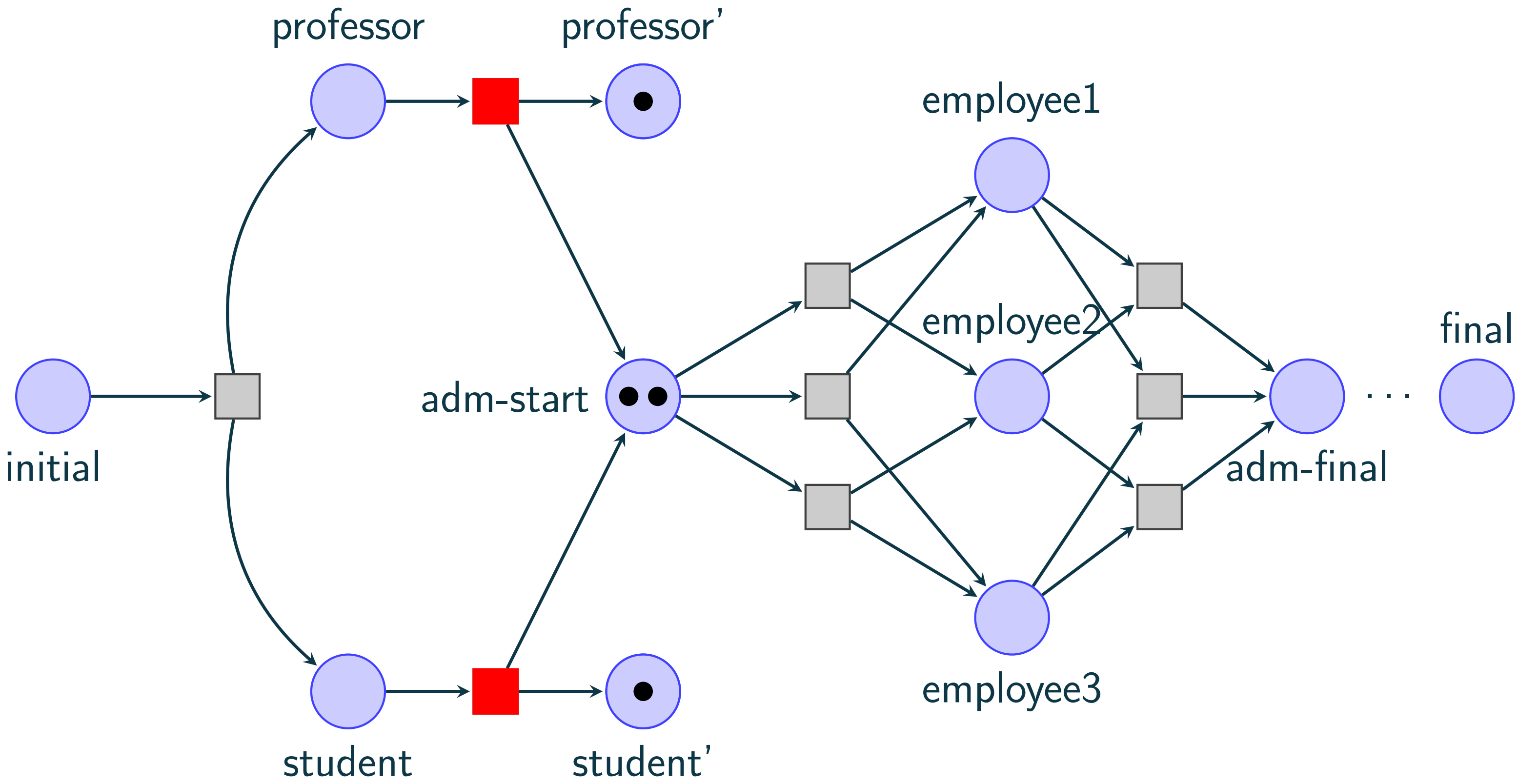
- Recall: both examples were sound

Combining the examples



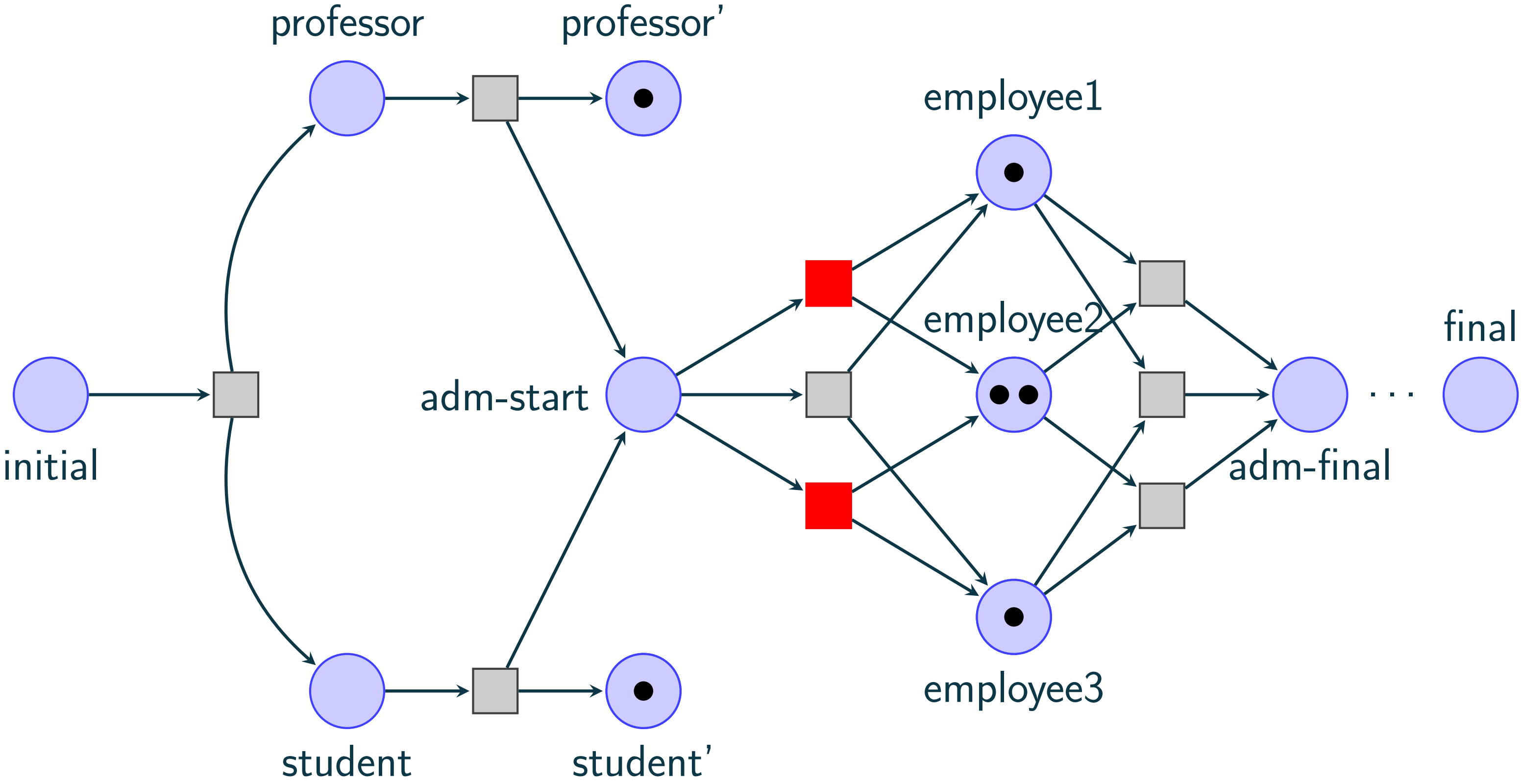
- Recall: both examples were sound

Combining the examples



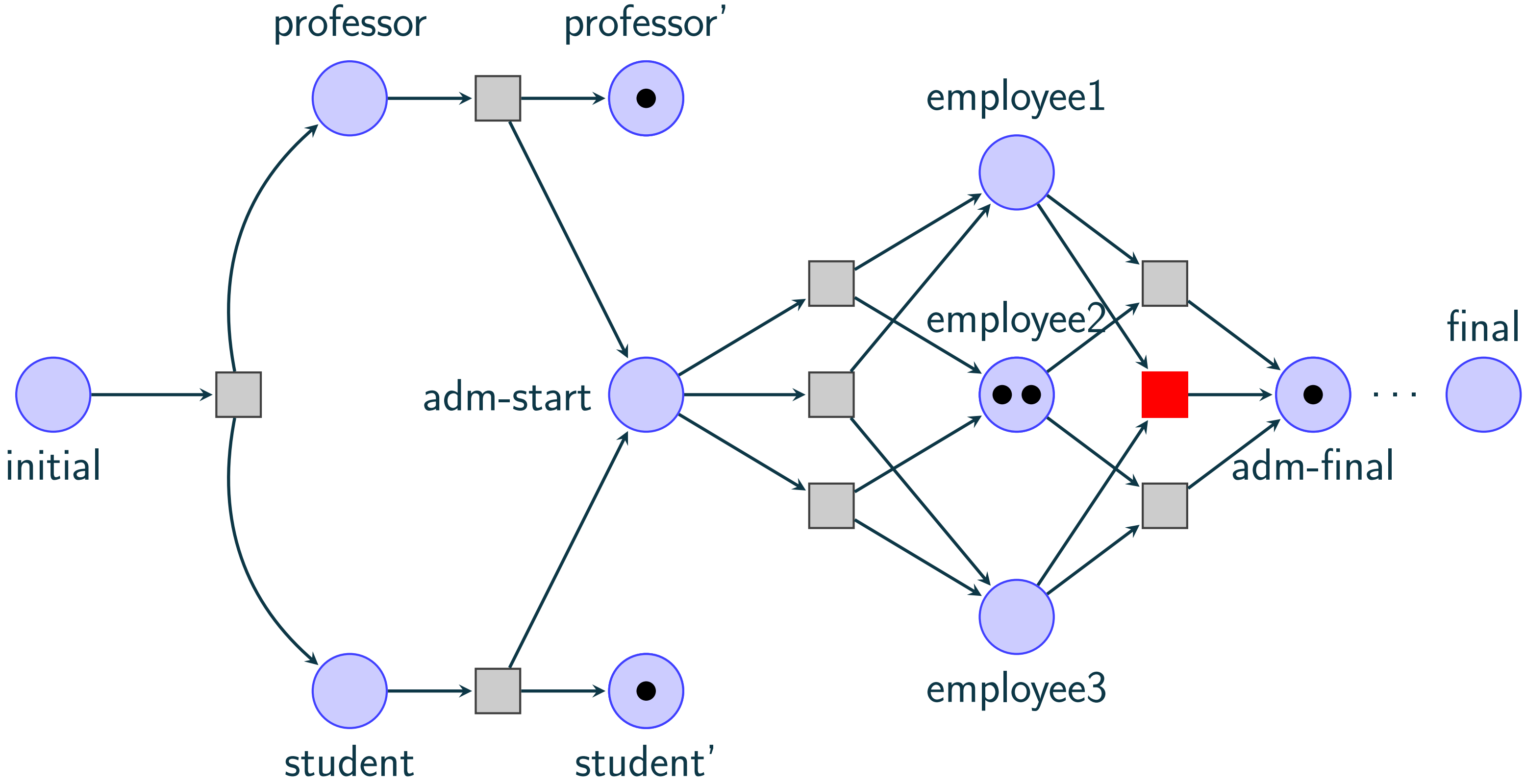
- Recall: both examples were sound

Combining the examples



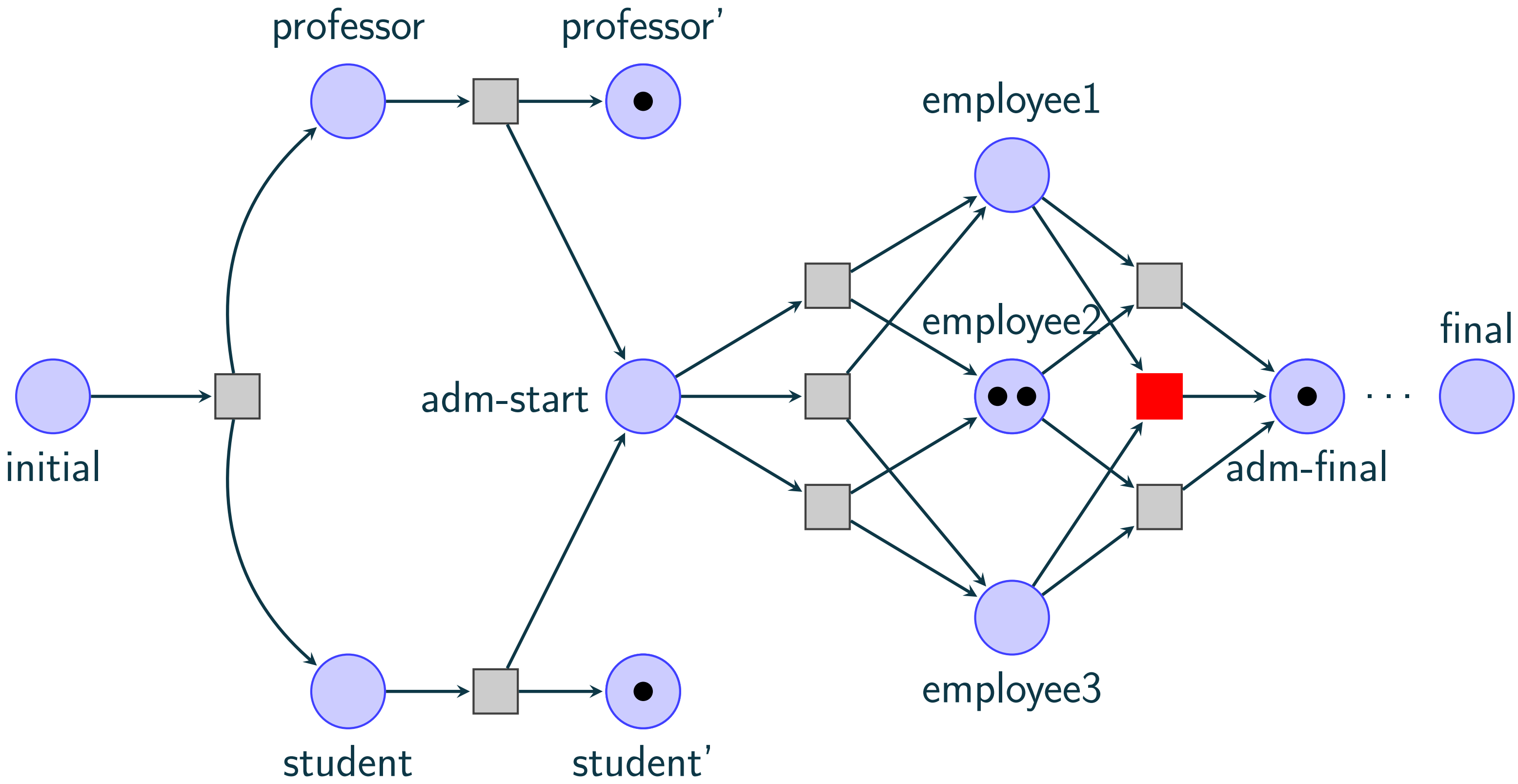
- Recall: both examples were sound

Combining the examples



- Recall: both examples were sound

Combining the examples



- Recall: both examples were sound
- But now it's not sound

Decision problems

Given a workflow net (d, T)

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound + quasi-live (can every transition be fired)?

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound $+$ quasi-live (can every transition be fired)?

2. Generalised soundness:

Determine if it is k-sound for all $k > 0$?

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound + quasi-live (can every transition be fired)?

2. Generalised soundness:

Determine if it is k-sound for all $k > 0$?

3. Structural soundness:

Determine if it is k-sound for some $k > 0$?

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound + quasi-live (can every transition be fired)?

2. Generalised soundness:

Determine if it is k-sound for all $k > 0$?

3. Structural soundness:

Determine if it is k-sound for some $k > 0$?

- Classical soundness (1) is the most common

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound + quasi-live (can every transition be fired)?

2. Generalised soundness:

Determine if it is k-sound for all $k > 0$?

3. Structural soundness:

Determine if it is k-sound for some $k > 0$?

- Classical soundness (1) is the most common
- Generalised soundness (2) is preserved under nice properties (e.g. composition)

Decision problems

Given a workflow net (d, T)

1. Classical soundness:

Determine if it is 1-sound $+$ quasi-live (can every transition be fired)?

2. Generalised soundness:

Determine if it is k-sound for all $k > 0$?

3. Structural soundness:

Determine if it is k-sound for some $k > 0$?

- Classical soundness (1) is the most common
- Generalised soundness (2) is preserved under nice properties (e.g. composition)
- Structural soundness (3) $\approx$ computing k s.t. the net is k-sound

Soundness state of art

- Classical soundness is decidable [Aalst, 1997], probably Ackermann upper bound
  Some papers vaguely claim it's EXPSPACE-hard (see later)

Soundness state of art

- Classical soundness is decidable [Aalst, 1997], probably Ackermann upper bound
  Some papers vaguely claim it's EXPSPACE-hard (see later)
- Generalised soundness is decidable [Kees van Hee et al. 2004]

Soundness state of art

- Classical soundness is decidable [Aalst, 1997], probably Ackermann upper bound
  Some papers vaguely claim it's EXPSPACE-hard (see later)

- Generalised soundness is decidable [Kees van Hee et al. 2004]

- Structural soundness is decidable [Ţiplea and Marinescu, 2005]

Soundness state of art

- Classical soundness is decidable [Aalst, 1997], probably Ackermann upper bound

  Some papers vaguely claim it's EXPSPACE-hard (see later)

- Generalised soundness is decidable [Kees van Hee et al. 2004]

- Structural soundness is decidable [Ţiplea and Marinescu, 2005]

Our results

Theorem (Blondin, M., Offtermatt 2022)

1. Classical soundness is EXPSPACE-complete
2. Generalised soundness is PSPACE-complete
3. Structural soundness is EXPSPACE-complete

Plan

1. Petri nets and reachability

2. Workflow nets and soundness

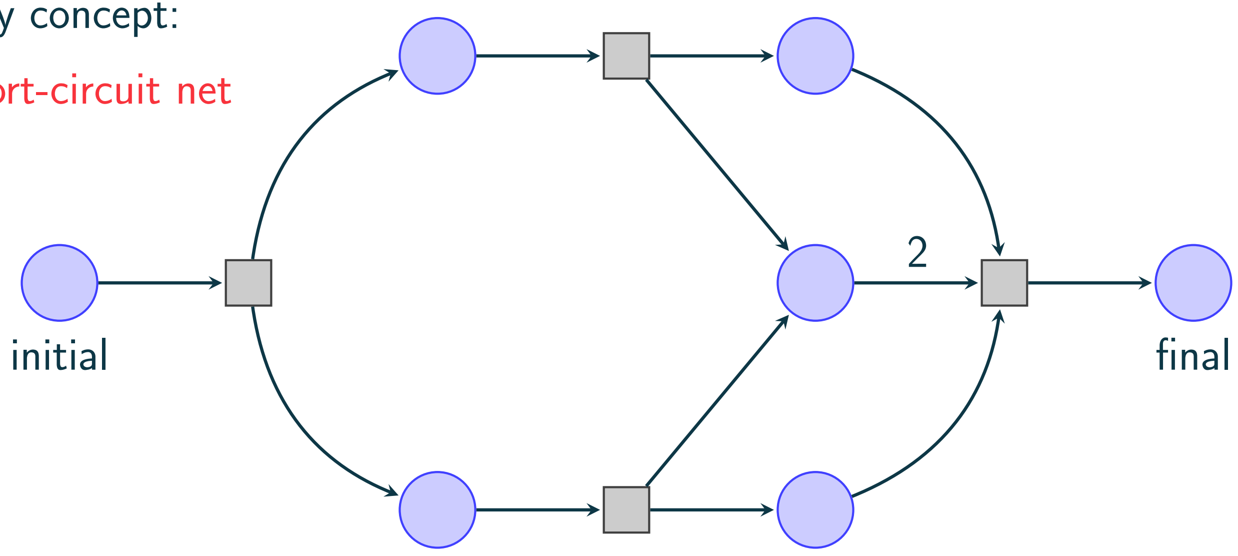
3. Some proofs

4. Implementation

Soundness decidability

Key concept:

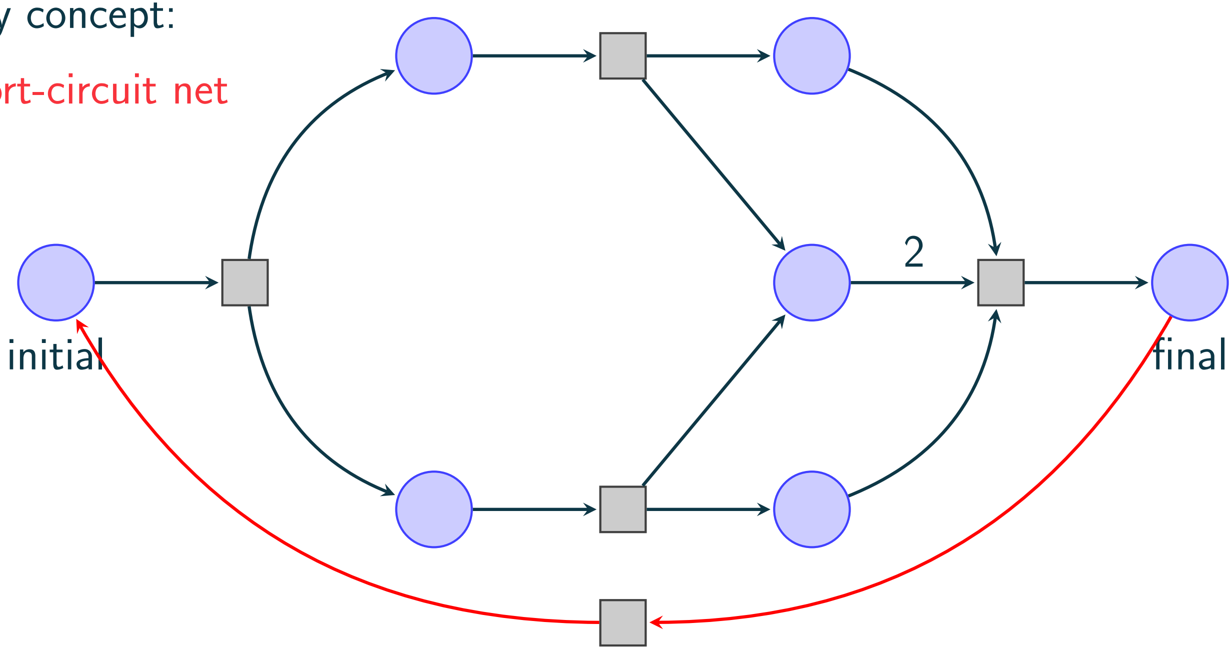
short-circuit net



Soundness decidability

Key concept:

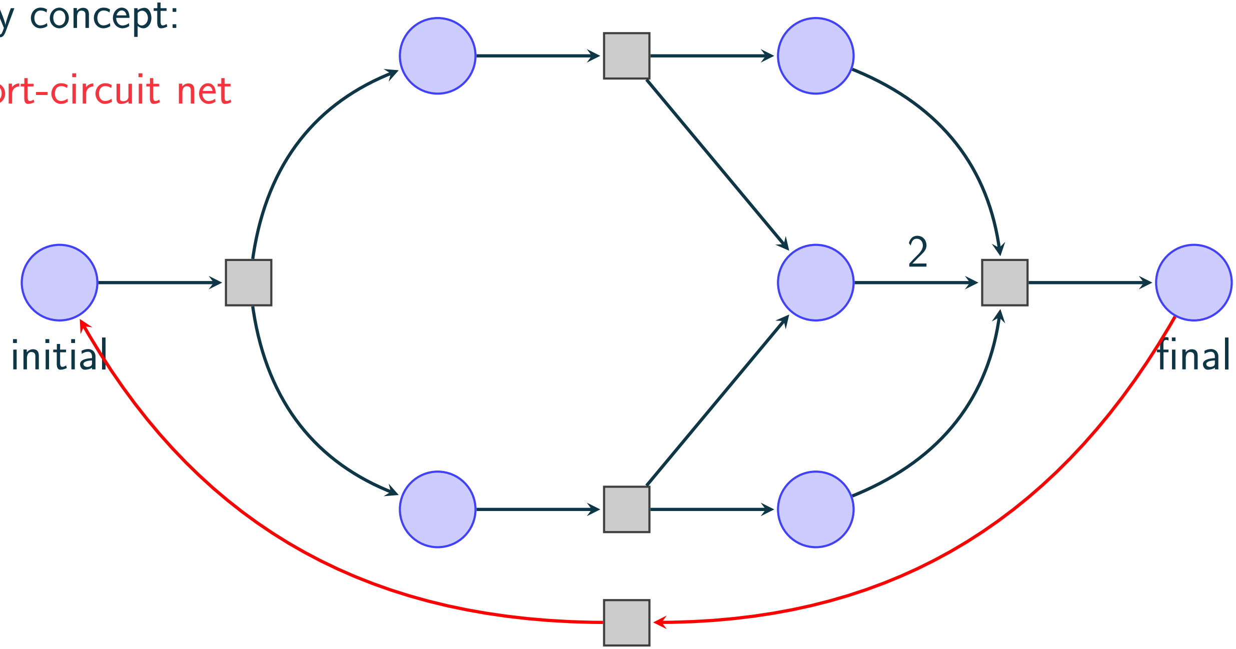
short-circuit net



Soundness decidability

Key concept:

short-circuit net



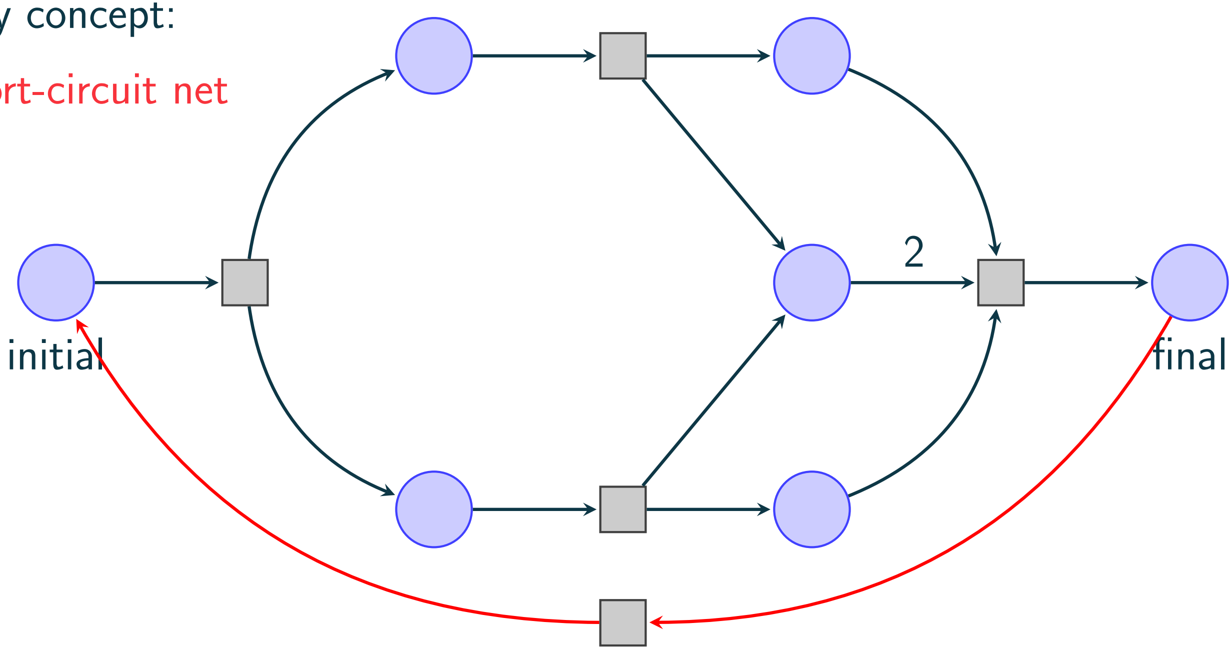
Lemma (Aalst 1997)

A workflow net is classical sound iff it's short circuit net is bounded and live

Soundness decidability

Key concept:

short-circuit net



Lemma (Aalst 1997)

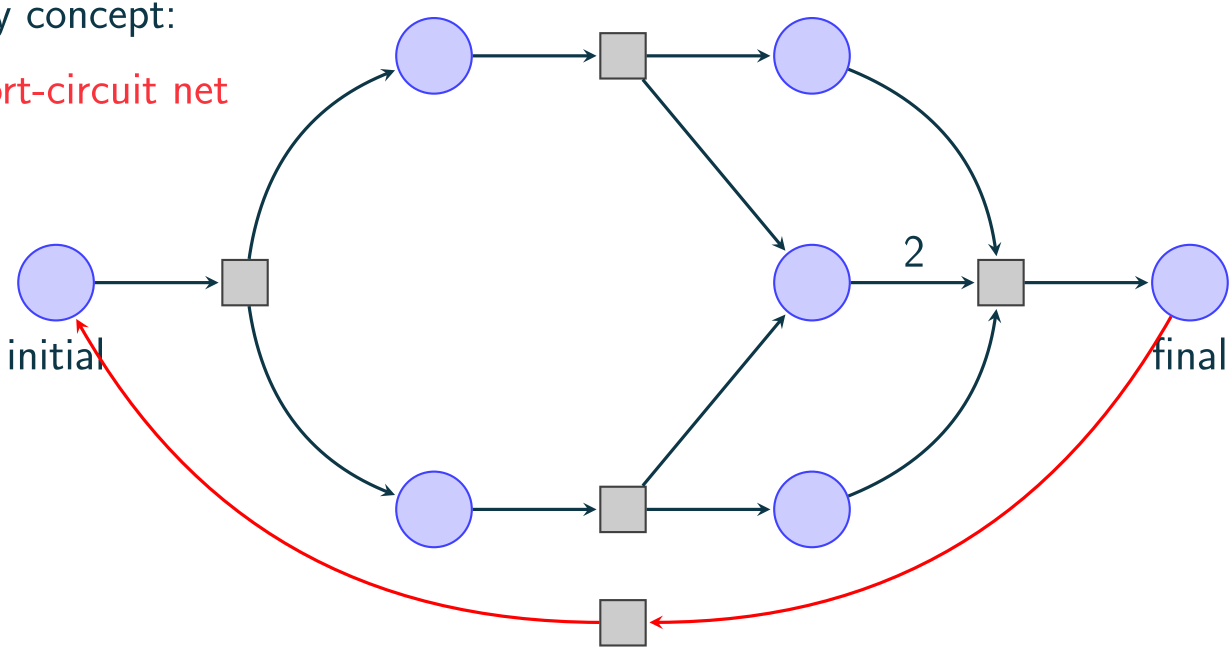
A workflow net is classical sound iff it's short circuit net is bounded and live

Unbounded means: $\{i : 1\} \rightarrow^* \boldsymbol{m} \rightarrow^* \boldsymbol{m}'$ and $\boldsymbol{m} < \boldsymbol{m}'$

Soundness decidability

Key concept:

short-circuit net



Lemma (Aalst 1997)

A workflow net is classical sound iff it's short circuit net is bounded and live

Unbounded means: $\{i : 1\} \rightarrow^* \boldsymbol{m} \rightarrow^* \boldsymbol{m}'$ and $\boldsymbol{m} < \boldsymbol{m}'$

If 1-sound then $\boldsymbol{m}' \rightarrow^* \{f : 1\} + \boldsymbol{m}' - \boldsymbol{m}$ ∎

Our improvement to EXPSPACE

Lemma (Aalst 1997) quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Our improvement to EXPSPACE

Lemma (Aalst 1997) quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Essentially the same proof works

Our improvement to EXPSPACE

Lemma (Aalst 1997) quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Essentially the same proof works

- Boundedness and quasi-liveness in EXPSPACE [Rackoff,1978]
- Cyclicity in EXPSPACE [Bouziane and Finkel, 1997]

Our improvement to EXPSPACE

Lemma (Aalst 1997) quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Essentially the same proof works

- Boundedness and quasi-liveness in EXPSPACE [Rackoff,1978]
- Cyclicity in EXPSPACE [Bouziane and Finkel, 1997]
- Liveness is reachability hard (i.e. Ackermann-hard)

Our improvement to EXPSPACE

Lemma (Aalst 1997) quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Essentially the same proof works

- Boundedness and quasi-liveness in EXPSPACE [Rackoff,1978]
- Cyclicity in EXPSPACE [Bouziane and Finkel, 1997]
- Liveness is reachability hard (i.e. Ackermann-hard)

Some papers vaguely conclude that classical soundness is EXPSPACE-hard (because both boundedness and liveness are EXPSPACE-hard)

Our improvement to EXPSPACE

Lemma (Aalst 1997)

quasi-live + cyclic

A workflow net is classical sound iff it's short circuit net is bounded and ~~live~~

Essentially the same proof works

- Boundedness and quasi-liveness in EXPSPACE [Rackoff,1978]
- Cyclicity in EXPSPACE [Bouziane and Finkel, 1997]
- Liveness is reachability hard (i.e. Ackermann-hard)

Some papers vaguely conclude that classical soundness is EXPSPACE-hard
(because both boundedness and liveness are EXPSPACE-hard)

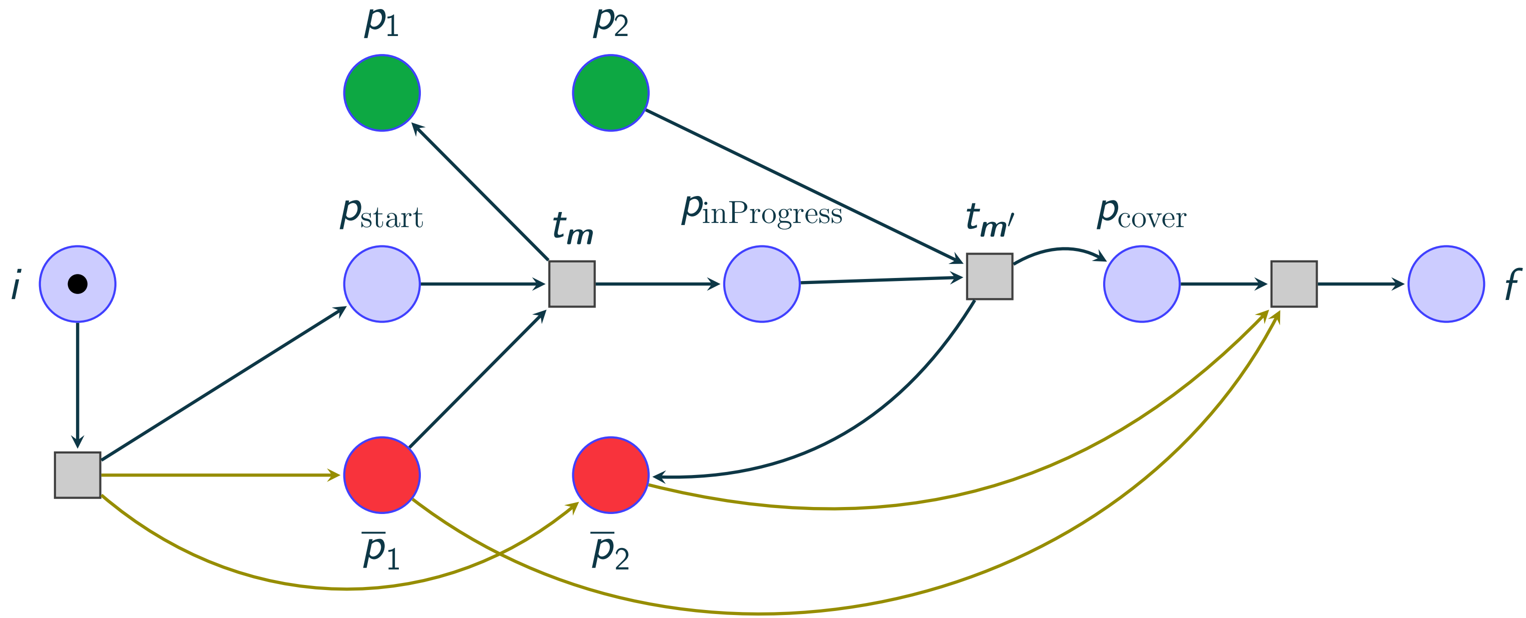
Lemma (Blondin, M., Offtermatt 2022)

Classical soundness is EXPSPACE-hard

via a technical reduction from reachability for reversible Petri nets

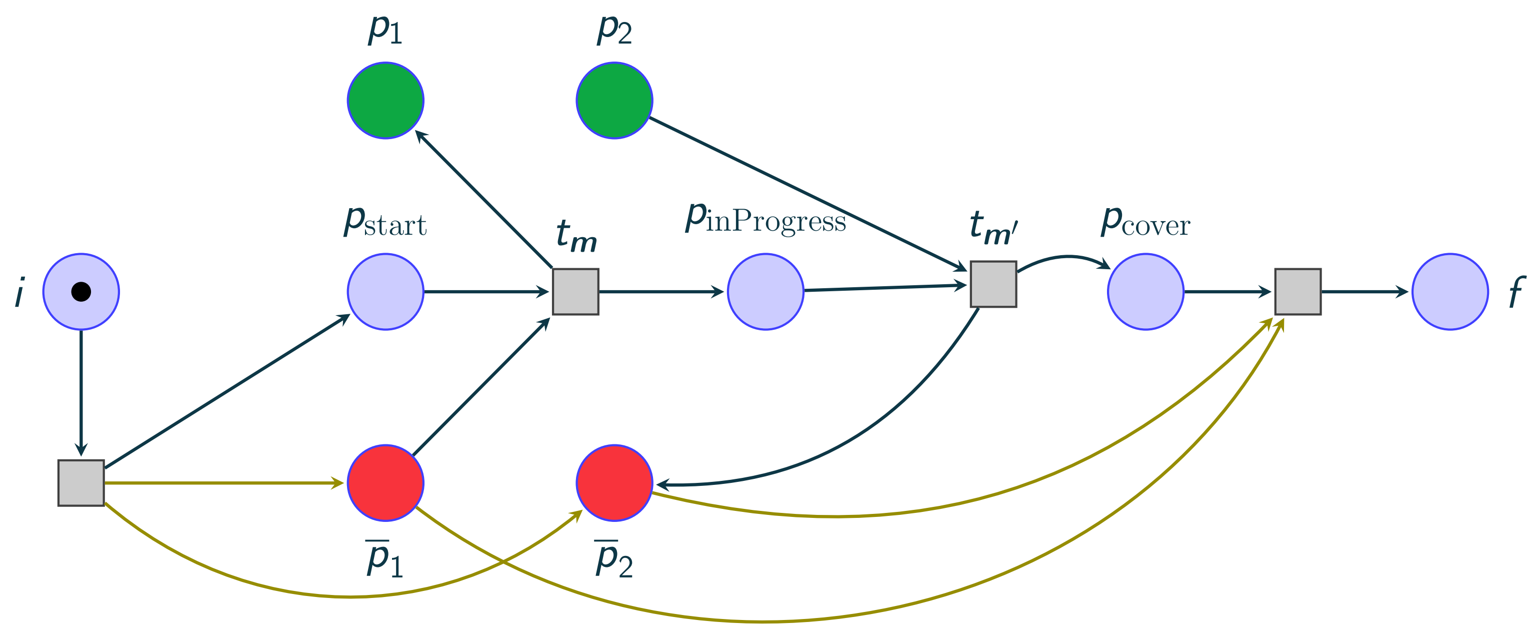
EXPSPACE-hardness

Encoding reachability $m \to^* m'$ in a reversible Petri net



EXPSPACE-hardness

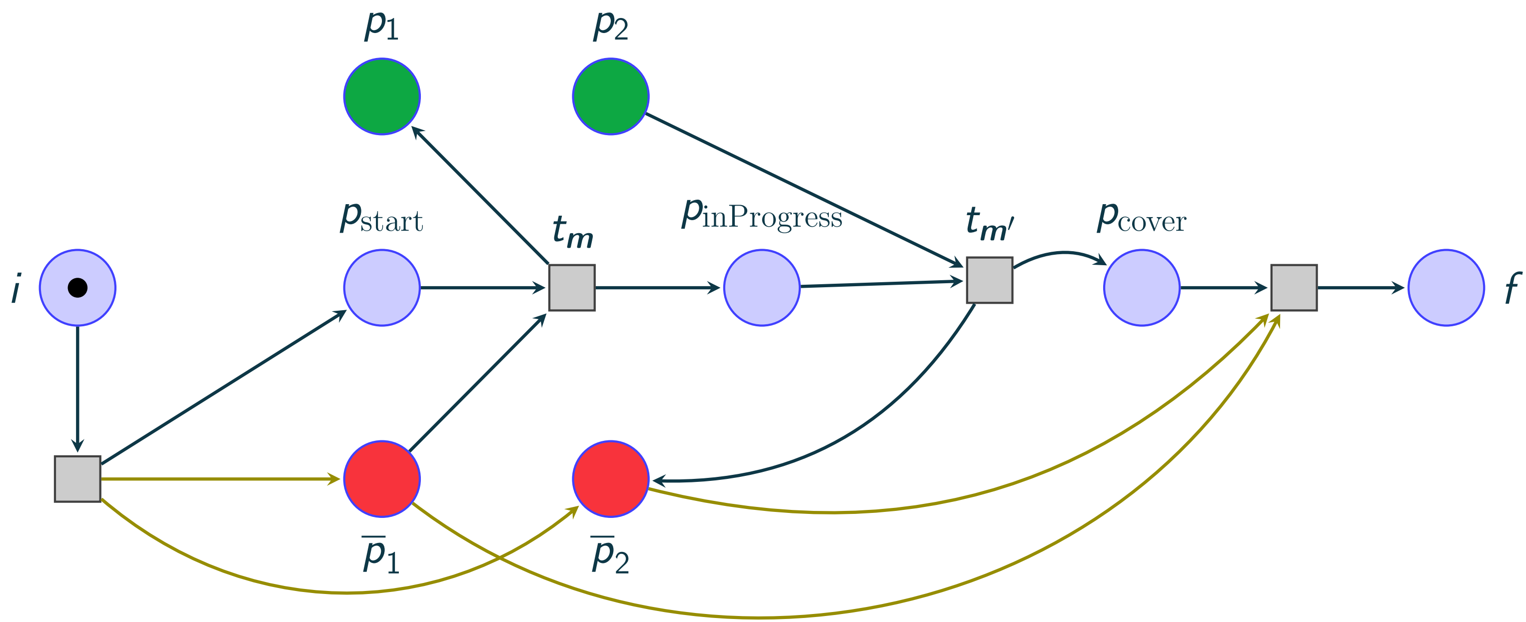
Encoding reachability $m \to^* m'$ in a reversible Petri net



- Original places, and their copies

EXPSPACE-hardness

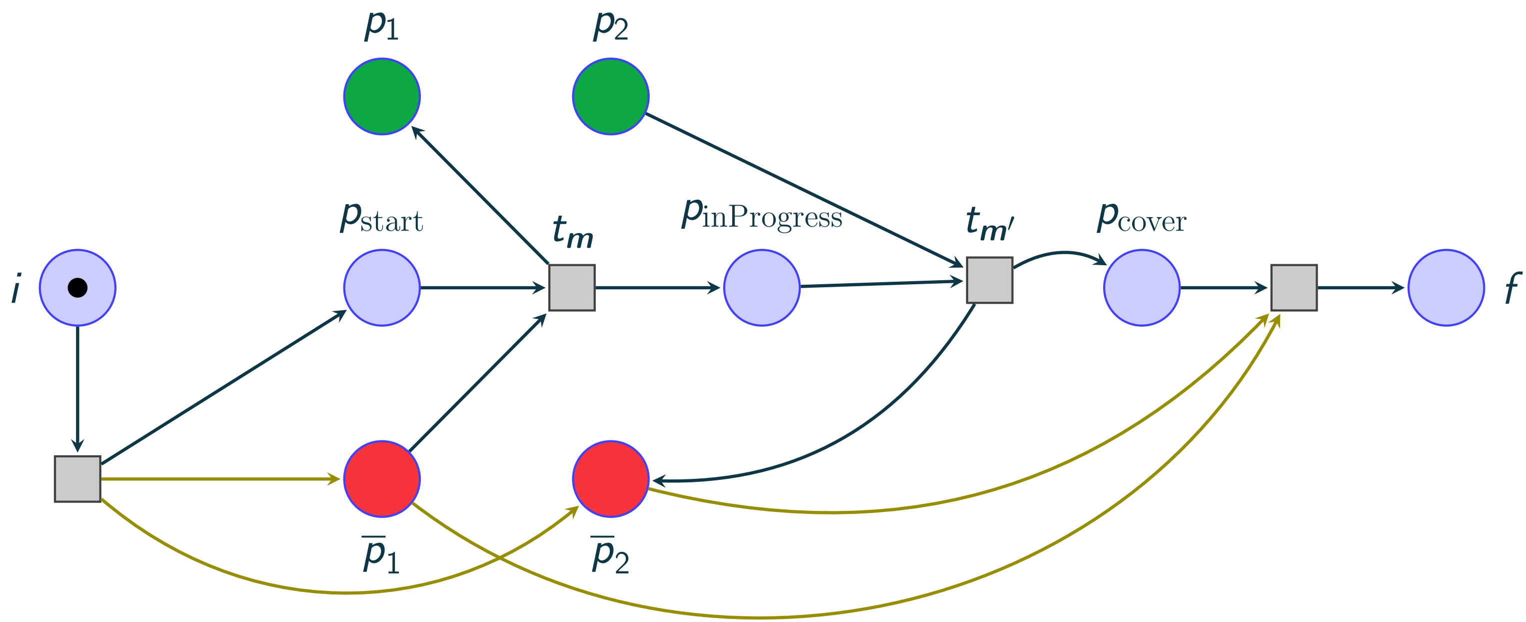
Encoding reachability $m \to^* m'$ in a reversible Petri net



- Original places, and their copies
- We can assume that counters are bounded by 2^{2^n}

EXPSPACE-hardness

Encoding reachability $m \rightarrow^* m'$ in a reversible Petri net



- Original places, and their copies
- We can assume that counters are bounded by 2^{2^n}
- Keep as an invariant that $p + \overline{p} = 2^{2^n}$

Generalised soundness

We write $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Generalised soundness

We write $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Reachability $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ is NP-complete (Integer Linear Programming)

Generalised soundness

We write $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Reachability $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ is NP-complete (Integer Linear Programming)

Recall that generalised soundness is $\forall_k \ \{i : k\} \rightarrow^{*} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^{*} \{f : k\}$

Generalised soundness

We write $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Reachability $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ is NP-complete (Integer Linear Programming)

Recall that generalised soundness is $\forall_k \; \{i : k\} \rightarrow^{*} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^{*} \{f : k\}$

Lemma (Kees van Hee et al. 2004)

Generalised soundness is equivalent to $\forall_k \; \{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^{*} \{f : k\}$

(we call this strong k-soundness)

Generalised soundness

We write $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Reachability $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ is NP-complete (Integer Linear Programming)

Recall that generalised soundness is $\forall_k \ \{i : k\} \rightarrow^{*} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^{*} \{f : k\}$

Lemma (Kees van Hee et al. 2004)

Generalised soundness is equivalent to $\forall_k \ \{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^{*} \{f : k\}$

(we call this strong k-soundness)

Lemma (Blondin, M., Offtermatt 2022)

1. If not generalised sound then not k-sound from "small" k
2. If not k-sound then it suffices to consider "small" $\boldsymbol{m}$

Generalised soundness

We write $\boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ if reachability holds in $\mathbb{Z}^d$ (runs possibly leave $\mathbb{N}^d$)

Reachability $\boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ is NP-complete (Integer Linear Programming)

Recall that generalised soundness is $\forall_k \ \{i : k\} \rightarrow^* \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^* \{f : k\}$

Lemma (Kees van Hee et al. 2004)

Generalised soundness is equivalent to $\forall_k \ \{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^* \{f : k\}$

(we call this strong k-soundness)

Lemma (Blondin, M., Offtermatt 2022)

1. If not generalised sound then not k-sound from "small" k
2. If not k-sound then it suffices to consider "small" $\boldsymbol{m}$

"small" $=$ exponential (for 1-soudness (2) was double exponential)

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

- If a net is $\mathbb{Z}$-unbounded then it's not generalised sound

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

- If a net is $\mathbb{Z}$-unbounded then it's not generalised sound

  Let $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

- If a net is $\mathbb{Z}$-unbounded then it's not generalised sound

  Let $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$

  If generalised sound then $\boldsymbol{m}' \rightarrow^*_{\mathbb{Z}} \{f : k\} + \boldsymbol{m}' - \boldsymbol{m}$

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \to_{\mathbb{Z}}^{*} \boldsymbol{m} \to_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

- If a net is $\mathbb{Z}$-unbounded then it's not generalised sound

  Let $\{i : k\} \to_{\mathbb{Z}}^{*} \boldsymbol{m} \to_{\mathbb{Z}}^{*} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$

  If generalised sound then $\boldsymbol{m}' \to_{\mathbb{Z}}^{*} \{f : k\} + \boldsymbol{m}' - \boldsymbol{m}$

- Suppose we conclude that $\{i : k\} \to_{\mathbb{Z}}^{*} \boldsymbol{m}$ then $\boldsymbol{m}$ is small

Proof intuition for generalised soundness

$\mathbb{Z}$-unboundedness: $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$ (any k)

- Checking $\mathbb{Z}$-boundedness is an Integer Linear Program

  Check if some multiset of transitions (seen as vectors) sums to $> \mathbf{0}$

- If a net is $\mathbb{Z}$-unbounded then it's not generalised sound

  Let $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ and $\boldsymbol{m}' - \boldsymbol{m} > \mathbf{0}$

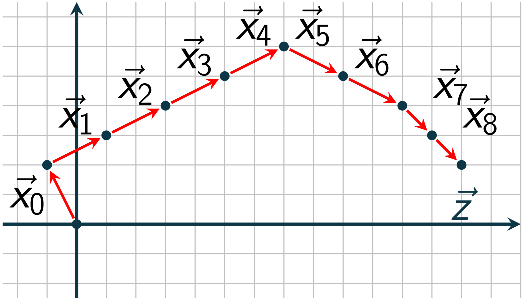
  If generalised sound then $\boldsymbol{m}' \rightarrow^*_{\mathbb{Z}} \{f : k\} + \boldsymbol{m}' - \boldsymbol{m}$

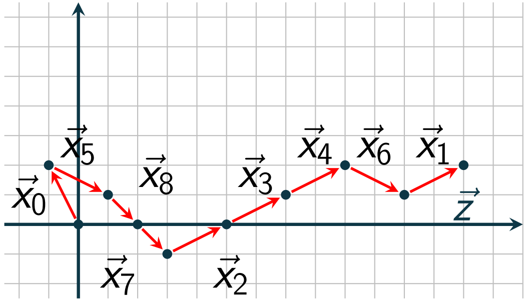
- Suppose we conclude that $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$ then $\boldsymbol{m}$ is small

  Then we can verify generalised soundness in PSPACE

  (go through all reachable configurations)

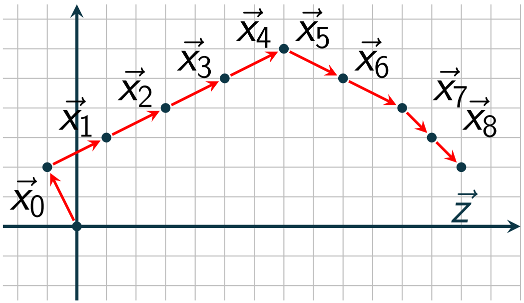
Reachable m are small

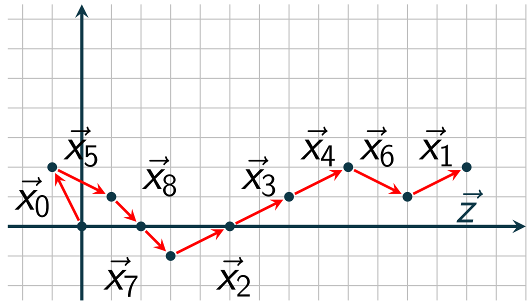




Steinitz Lemma: if $\boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ then one can reorder vectors to be "close" to the line $\boldsymbol{m}' - \boldsymbol{m}$

Reachable m are small

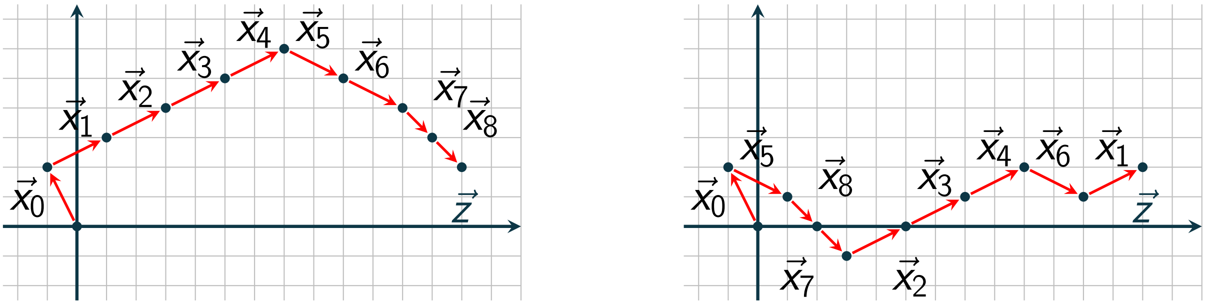




Steinitz Lemma: if $\boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ then one can reorder vectors to be "close" to the line $\boldsymbol{m}' - \boldsymbol{m}$

- Suppose $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$ and $\boldsymbol{m}$ is "big"

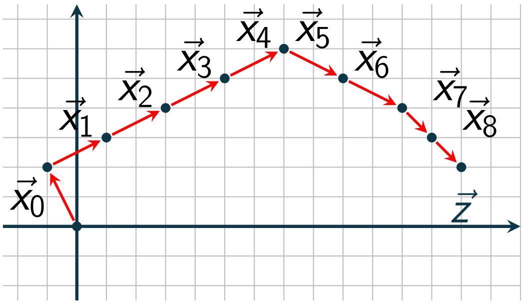
Reachable m are small

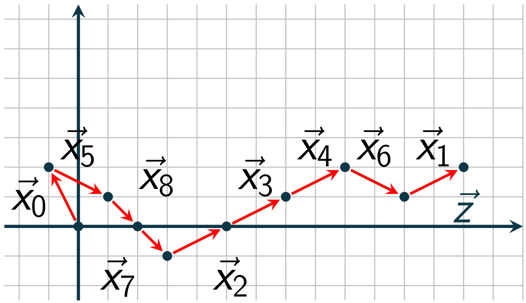


Steinitz Lemma: if $m \rightarrow_{\mathbb{Z}}^{*} m'$ then one can reorder vectors to be "close" to the line $m' - m$

- Suppose $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} m$ and m is "big"
- Since k is "small" by Steinitz Lemma and a simple pumping argument in the run for $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} m$ one can find $n < n'$

Reachable m are small





Steinitz Lemma: if $\boldsymbol{m} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}'$ then one can reorder vectors to be "close" to the line $\boldsymbol{m}' - \boldsymbol{m}$

- Suppose $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$ and $\boldsymbol{m}$ is "big"
- Since k is "small" by Steinitz Lemma and a simple pumping argument in the run for $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$ one can find $\boldsymbol{n} < \boldsymbol{n}'$
- By previous slide $\mathbb{Z}$-unboundedness implies not generalised soundness

Structural soundness

Recall it's: $\exists_k \;\; \{i : k\} \to^* \boldsymbol{m} \implies \boldsymbol{m} \to^* \{f : k\}$?

Structural soundness

Recall it's: $\exists_k \ \{i : k\} \rightarrow^* \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^* \{f : k\}$?

- EXPSPACE-hardness is almost immediate

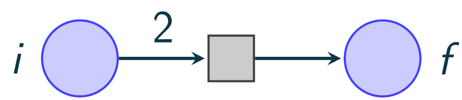
Structural soundness

Recall it's: $\exists_k \ \{i : k\} \to^* \boldsymbol{m} \implies \boldsymbol{m} \to^* \{f : k\}$?

- EXPSPACE-hardness is almost immediate

  We encode 1-soundness by adding one transition

  (it cannot be k-sound for $k > 1$)



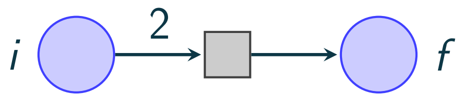
Structural soundness

Recall it's: $\exists_k \ \{i : k\} \to^* \boldsymbol{m} \implies \boldsymbol{m} \to^* \{f : k\}$?

- EXPSPACE-hardness is almost immediate

  We encode 1-soundness by adding one transition

  (it cannot be k-sound for $k > 1$)



- For the EXPSPACE upper bound the set of sound numbers is

  $S = \{i \cdot p \mid 1 \leq i \leq k\}$

  for "small" p, and k either "small" or $+\infty$

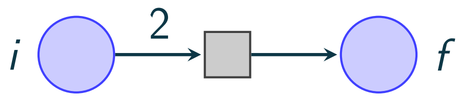
Structural soundness

Recall it's: $\exists_k \ \{i : k\} \rightarrow^* \boldsymbol{m} \implies \boldsymbol{m} \rightarrow^* \{f : k\}$?

- EXPSPACE-hardness is almost immediate

  We encode 1-soundness by adding one transition

  (it cannot be k-sound for $k > 1$)



- For the EXPSPACE upper bound the set of sound numbers is

  $S = \{i \cdot p \mid 1 \leq i \leq k\}$

  for "small" p, and k either "small" or $+\infty$

  Essentially, because S is closed under subtraction

Plan

1. Petri nets and reachability

2. Workflow nets and soundness

3. Some proofs

4. Implementation

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow_{\mathbb{Z}}^{*}$ instead of $\rightarrow^{*}$

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow_{\mathbb{Z}}^{*}$ instead of $\rightarrow^{*}$

  The complexity of reachability drops to NP

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow^*_{\mathbb{Z}}$ instead of $\rightarrow^*$

  The complexity of reachability drops to NP

  if $\not\rightarrow^*_{\mathbb{Z}}$ then $\not\rightarrow^*$

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow^*_{\mathbb{Z}}$ instead of $\rightarrow^*$

  The complexity of reachability drops to NP

  if $\not\rightarrow^*_{\mathbb{Z}}$ then $\not\rightarrow^*$

  This filters some benchmarks, or some cases in the algorithm

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow^*_{\mathbb{Z}}$ instead of $\rightarrow^*$

  The complexity of reachability drops to NP

  if $\not\rightarrow^*_{\mathbb{Z}}$ then $\not\rightarrow^*$

  This filters some benchmarks, or some cases in the algorithm

- Another popular one is continuous reachability $\rightarrow^*_c$

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow_{\mathbb{Z}}^*$ instead of $\rightarrow^*$

  The complexity of reachability drops to NP

  if $\not\rightarrow_{\mathbb{Z}}^*$ then $\not\rightarrow^*$

  This filters some benchmarks, or some cases in the algorithm

- Another popular one is continuous reachability $\rightarrow_c^*$

  Every transition can be scaled by $\delta \in (0, 1]$ before firing

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow_{\mathbb{Z}}^{*}$ instead of $\rightarrow^{*}$

  The complexity of reachability drops to NP

  if $\not\rightarrow_{\mathbb{Z}}^{*}$ then $\not\rightarrow^{*}$

  This filters some benchmarks, or some cases in the algorithm

- Another popular one is continuous reachability $\rightarrow_{c}^{*}$

  Every transition can be scaled by $\delta \in (0, 1]$ before firing

  The complexity of reachability drops to PTime [Fraca and Haddad, 2015]

How to implement EXPSPACE-hard etc problems?

By relaxing problems.

- Recall $\rightarrow^*_{\mathbb{Z}}$ instead of $\rightarrow^*$

  The complexity of reachability drops to NP

  if $\not\rightarrow^*_{\mathbb{Z}}$ then $\not\rightarrow^*$

  This filters some benchmarks, or some cases in the algorithm

- Another popular one is continuous reachability $\rightarrow^*_c$

  Every transition can be scaled by $\delta \in (0, 1]$ before firing

  The complexity of reachability drops to PTime [Fraca and Haddad, 2015]

- Used in many implementations [Esparza et al. 2014], [Blondin et al., 2016], ...

Problems with relaxations for soundness

Relaxations usually don't work for soundness

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

For some problems $\forall\exists$ e.g. $\rightarrow_c^*$ provably doesn't work [Blondin and Haase, 2017]

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

For some problems $\forall\exists$ e.g. $\rightarrow_c^*$ provably doesn't work [Blondin and Haase, 2017]

Lemma (Blondin, M., Offtermatt)

$m \rightarrow_c^* m'$ iff exists $b \in \mathbb{N}$ s.t. $bm \rightarrow^* bm'$.

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs

- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

For some problems $\forall\exists$ e.g. $\rightarrow_c^*$ provably doesn't work [Blondin and Haase, 2017]

Lemma (Blondin, M., Offtermatt)

$\boldsymbol{m} \rightarrow_c^* \boldsymbol{m}'$ iff exists $b \in \mathbb{N}$ s.t. $b\boldsymbol{m} \rightarrow^* b\boldsymbol{m}'$.

Corollary

Continuous ($\rightarrow_c^*$) unsoundness implies generalised unsoundness

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

For some problems $\forall\exists$ e.g. $\rightarrow_c^*$ provably doesn't work [Blondin and Haase, 2017]

Lemma (Blondin, M., Offtermatt)

$\boldsymbol{m} \rightarrow_c^* \boldsymbol{m}'$ iff exists $b \in \mathbb{N}$ s.t. $b\boldsymbol{m} \rightarrow^* b\boldsymbol{m}'$.

Corollary

Continuous ($\rightarrow_c^*$) unsoundness implies generalised unsoundness

Let $\{i : 1\} \rightarrow_c^* \boldsymbol{m} \not\rightarrow_c^* \{f : 1\}$

Problems with relaxations for soundness

Relaxations usually don't work for soundness

- Reachability asks $\exists$ run

  Relaxing $\rightarrow^*$ we get more runs
- Soundness asks $\forall$ runs $\exists$ run

  Relaxing $\rightarrow^*$ gives us ???

For some problems $\forall\exists$ e.g. $\rightarrow_c^*$ provably doesn't work [Blondin and Haase, 2017]

Lemma (Blondin, M., Offtermatt)

$\boldsymbol{m} \rightarrow_c^* \boldsymbol{m}'$ iff exists $b \in \mathbb{N}$ s.t. $b\boldsymbol{m} \rightarrow^* b\boldsymbol{m}'$.

Corollary

Continuous ($\rightarrow_c^*$) unsoundness implies generalised unsoundness

Let $\{i : 1\} \rightarrow_c^* \boldsymbol{m} \not\rightarrow_c^* \{f : 1\}$, by Lemma $\{i : b\} \rightarrow^* b\boldsymbol{m}$ and $b\boldsymbol{m} \not\rightarrow^* \{f : b\}$

Continuous soundness used in the implementation

Theorem (Blondin, M., Offtermatt 2022)

Continuous soudness is coNP-complete

Continuous soundness used in the implementation

Theorem (Blondin, M., Offtermatt 2022)

Continuous soudness is coNP-complete

Theorem (Ping et al., 2004)

For free-choice workflow nets 1-soundness and generalised soundness coincide

Continuous soundness used in the implementation

Theorem (Blondin, M., Offtermatt 2022)

Continuous soudness is coNP-complete

Theorem (Ping et al., 2004)

For free-choice workflow nets 1-soundness and generalised soundness coincide

Theorem (Blondin, M., Offtermatt 2022)

For free-choice workflow nets 1-, generalised, structural and continuous soundness all coincide

Continuous soundness used in the implementation

Theorem (Blondin, M., Offtermatt 2022)

Continuous soudness is coNP-complete

Theorem (Ping et al., 2004)

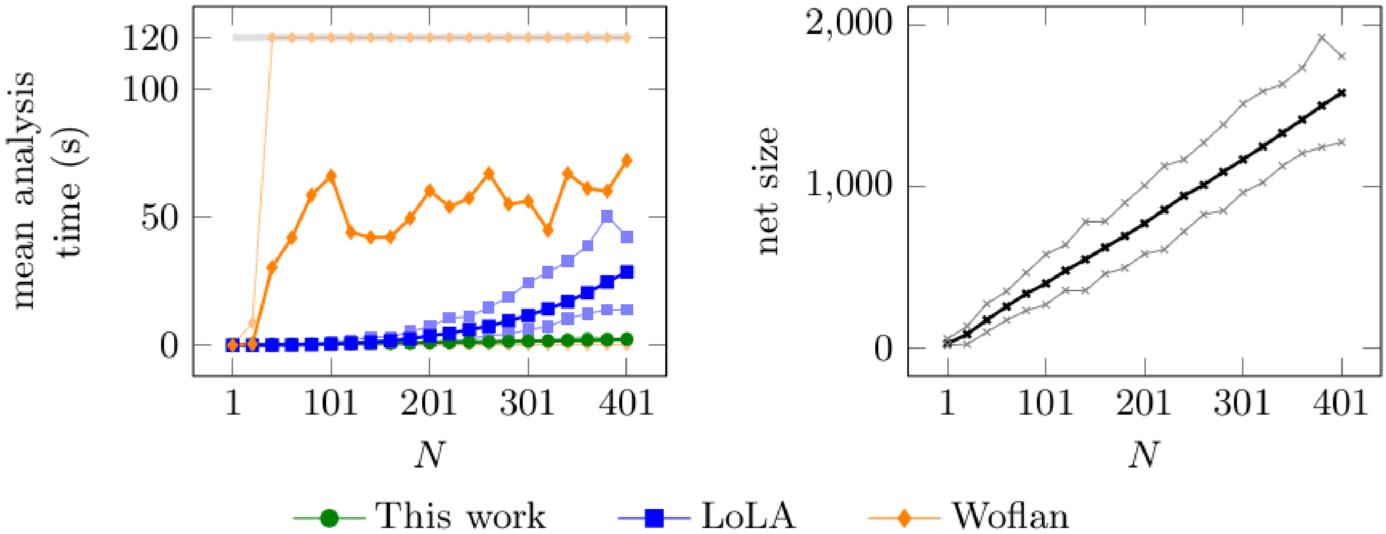
For free-choice workflow nets 1-soundness and generalised soundness coincide

Theorem (Blondin, M., Offtermatt 2022)

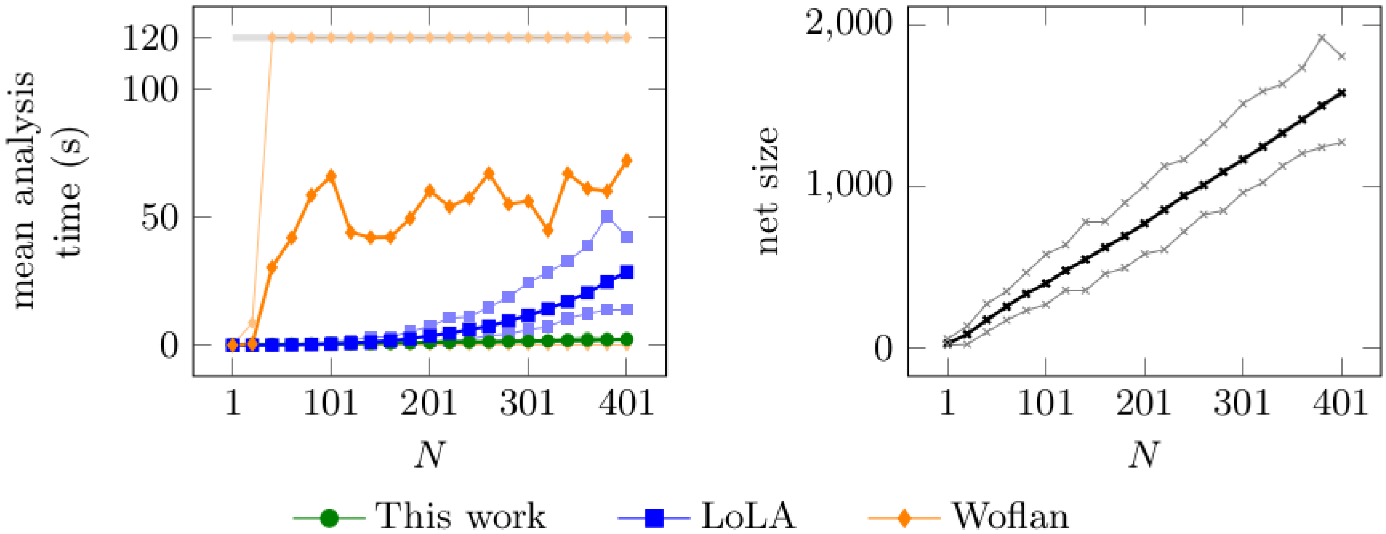
For free-choice workflow nets 1-, generalised, structural and continuous soundness all coincide

This and some other observations give us a nice implementation
(most benchmarks are free-choice nets)

Results

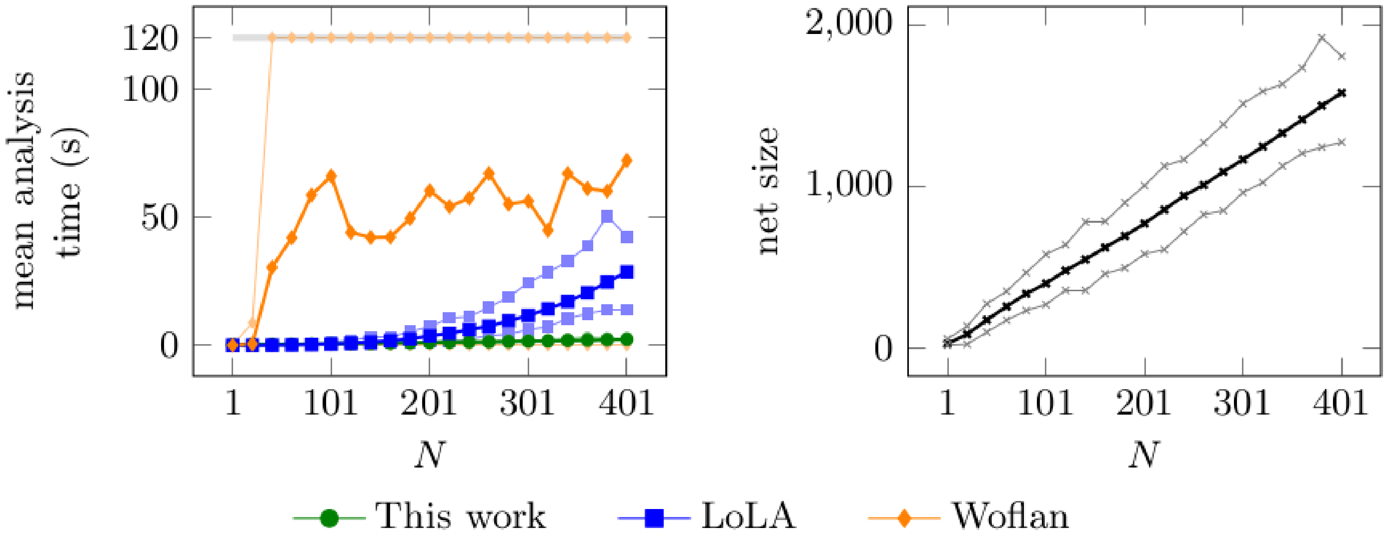

mean analysis time (s)
120
100
50
0
1
101
201
301
401
N
net size
2,000
1,000
0
1
101
201
301
401
N
This work
LoLA
Woflan

Results



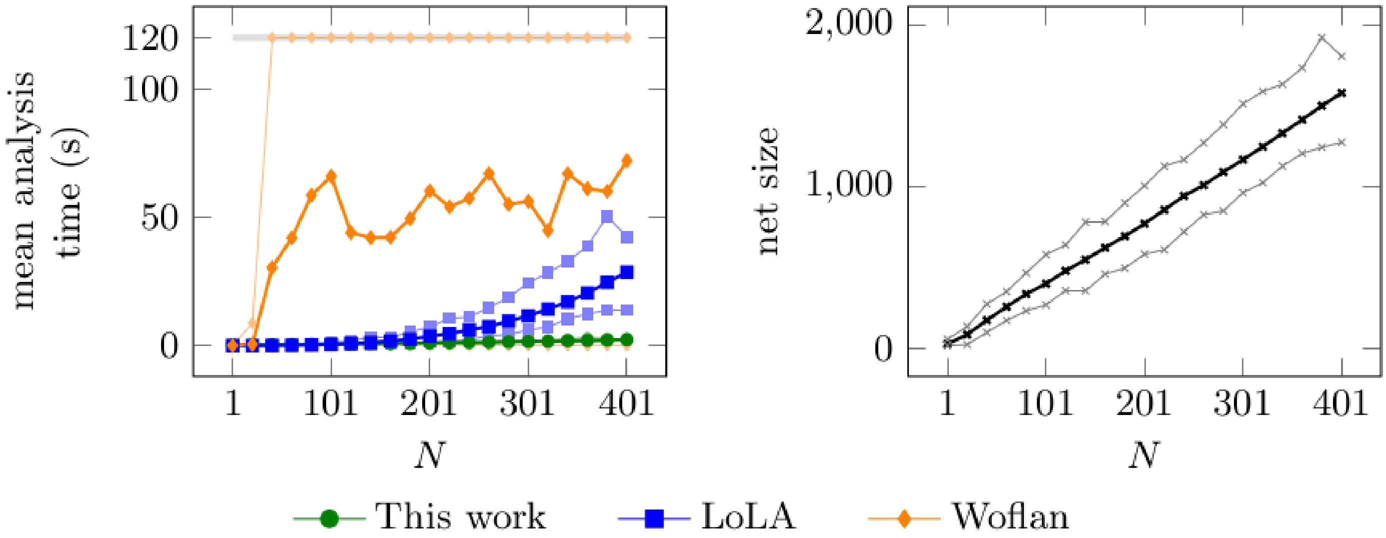
- On "small" benchmarks our tool is slower (but all three are very fast)

Results



- On "small" benchmarks our tool is slower (but all three are very fast)
- Larger benchmarks are obtained by composing them with each other (N times)

Results



- On “small” benchmarks our tool is slower (but all three are very fast)
- Larger benchmarks are obtained by composing them with each other (N times)
- For big N our tool scales much better

Conclusion

- Soundness can be seen as a containment problem

  $\mathsf{Reachable}(\{i : 1\}) \subseteq \mathsf{CoReachable}(\{f : 1\})$?

Conclusion

- Soundness can be seen as a containment problem

  $\mathrm{Reachable}(\{i : 1\}) \subseteq \mathrm{CoReachable}(\{f : 1\})$?

  Containment is undecidable in general [Hack, 1975]

Conclusion

- Soundness can be seen as a containment problem

  Reachable($\{i : 1\}$) $\subseteq$ CoReachable($\{f : 1\}$)?

  Containment is undecidable in general [Hack, 1975]

  Variants still studied [Jančar and Leroux, 2022], [Guttenberg and Raskin, 2022]

Conclusion

- Soundness can be seen as a containment problem

  Reachable($\{i : 1\}$) $\subseteq$ CoReachable($\{f : 1\}$)?

  Containment is undecidable in general [Hack, 1975]

  Variants still studied [Jančar and Leroux, 2022], [Guttenberg and Raskin, 2022]

- For soundness many open problems

  Data nets, reset nets

Conclusion

- Soundness can be seen as a containment problem

  Reachable($\{i : 1\}$) $\subseteq$ CoReachable($\{f : 1\}$)?

  Containment is undecidable in general [Hack, 1975]

  Variants still studied [Jančar and Leroux, 2022], [Guttenberg and Raskin, 2022]

- For soundness many open problems

  Data nets, reset nets

- The talk is based on two papers with Michael Blondin and Philip Offtermatt

  1. "The complexity of soundness in workflow nets". LICS 2022.

  2. "Verifying Generalised and Structural Soundness of Workflow Nets via Relaxations". CAV 2022.