Soundness problems for workflow nets

Filip Mazowiecki

University of Warsaw

Seminarium JAiO 2023

Plan

1. Petri nets

2. Workflow nets and soundness

3. Some proofs

Almost Petri nets

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

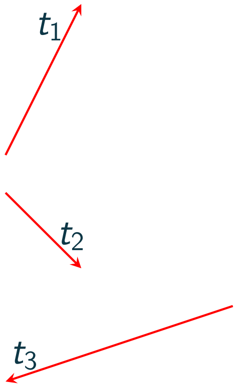
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

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

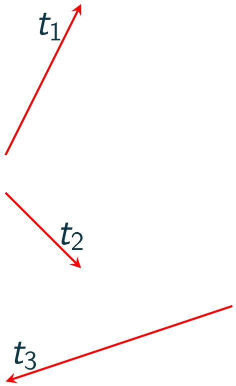
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

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

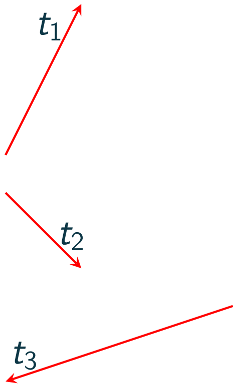
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

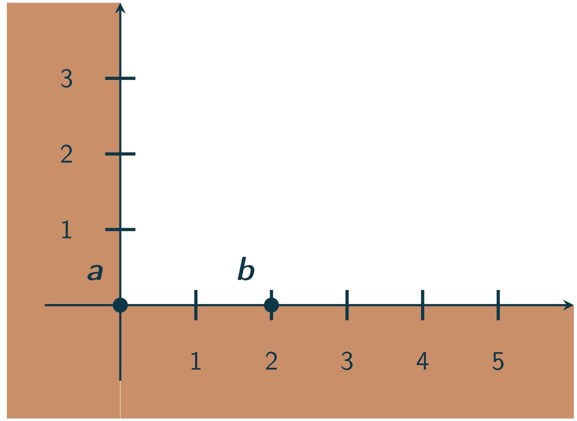
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

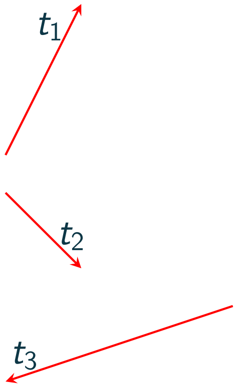
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

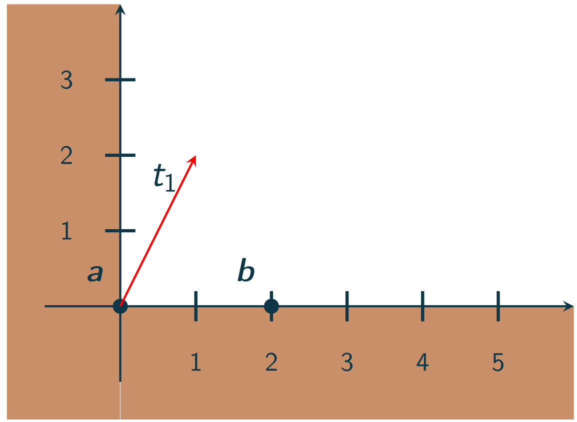
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

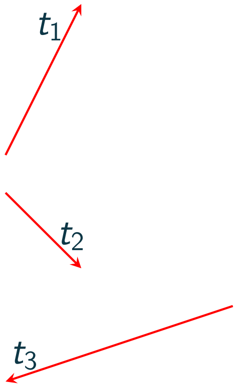
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

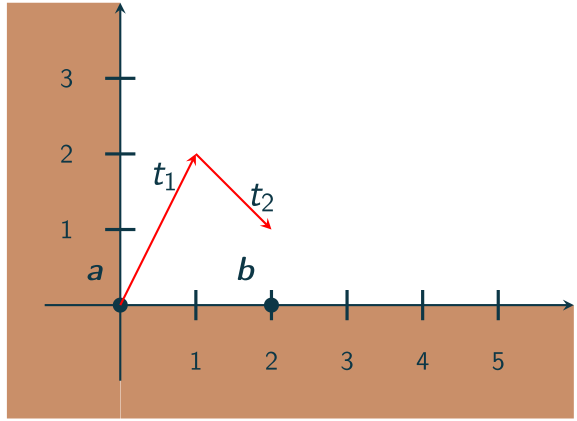
Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

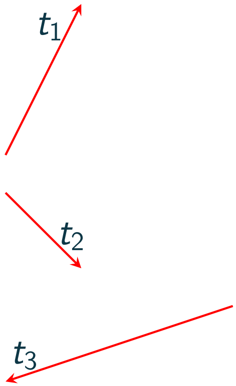
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

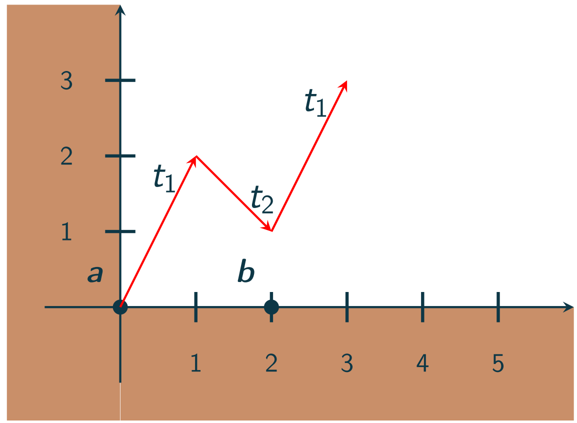
Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

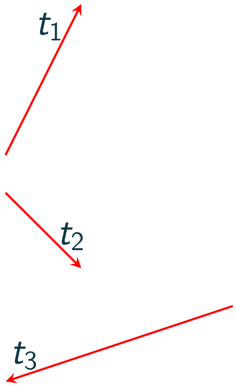
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

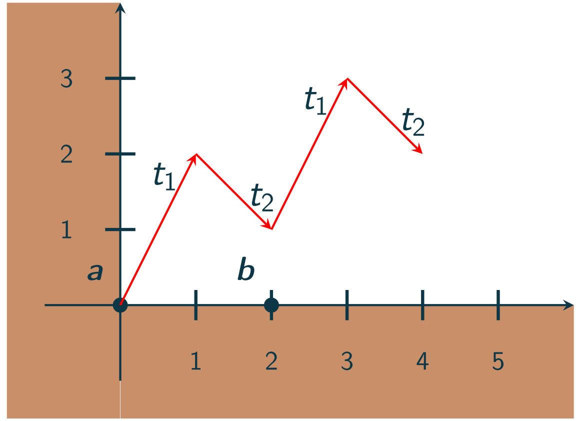
Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

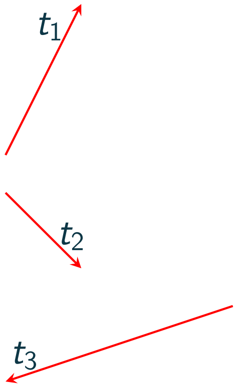
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

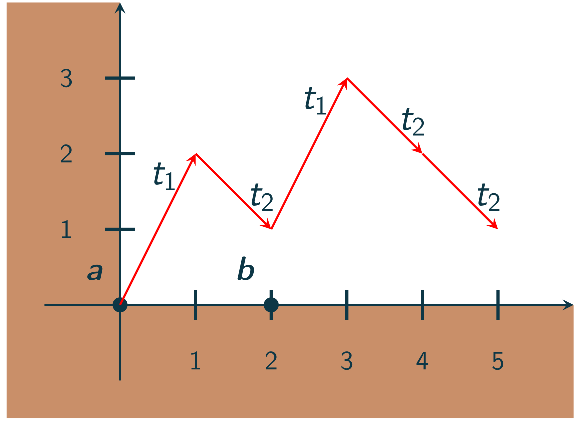
Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

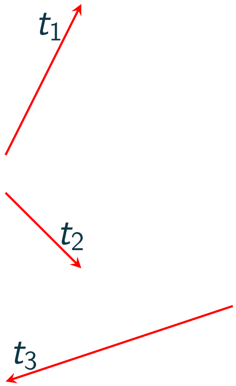
Almost Petri net (d, T): d – dimension, $T \subseteq \mathbb{Z}^d$ (finite)

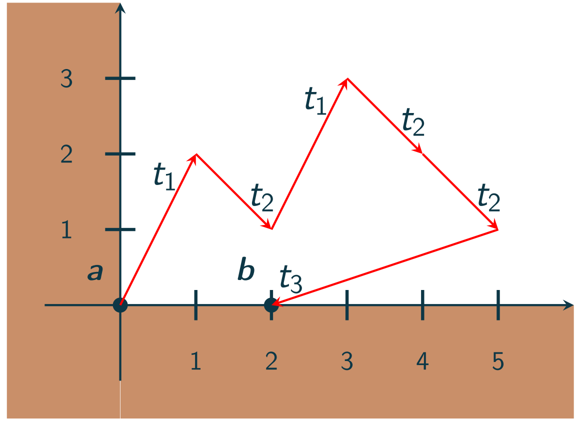
Example: $d = 2$, $T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

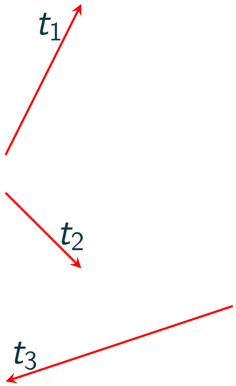
Almost Petri net (d, T): $\quad d$ – dimension, $\quad T \subseteq \mathbb{Z}^d$ (finite)

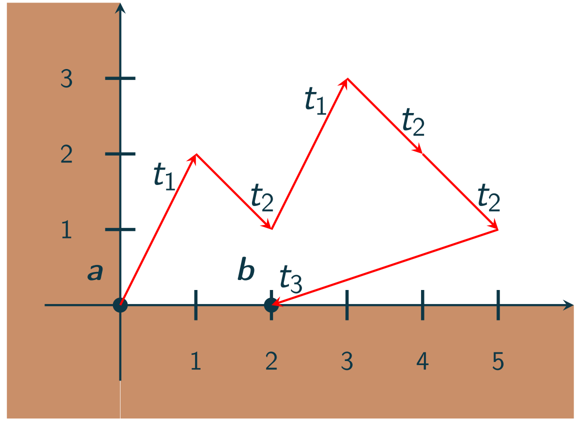
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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 $\mathbb{N}^d$?

(without t_3 it's not possible)

Almost Petri nets

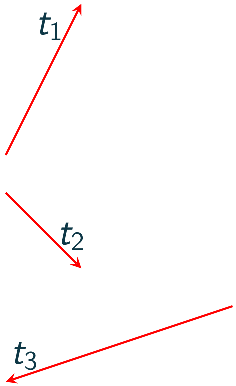
Almost Petri net (d, T): $\quad d$ – dimension, $\quad T \subseteq \mathbb{Z}^d$ (finite)

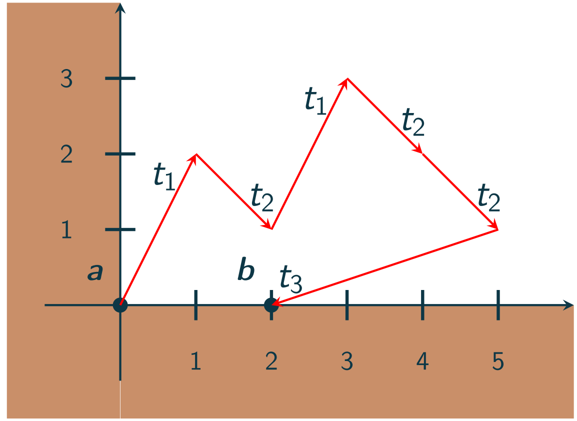
Example: $d = 2,\ T = \{t_1, t_2, t_3\}$

$t_1 = (1, 2)$

$t_2 = (1, -1)$

$t_3 = (-3, -1)$





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

Petri nets

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

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

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

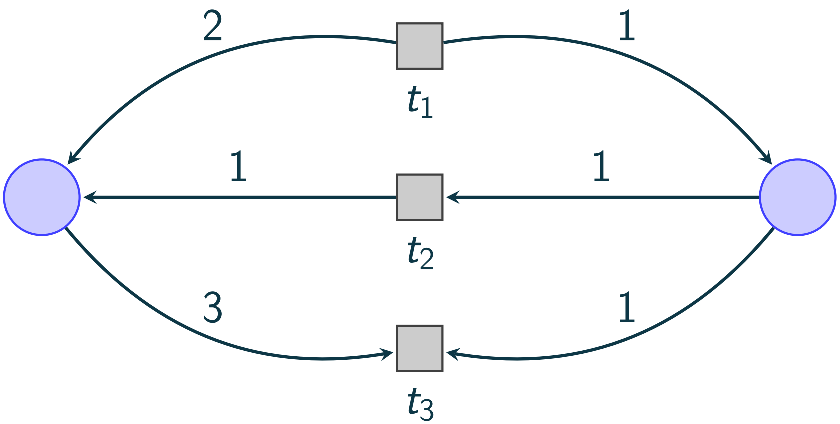
(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

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



Petri nets

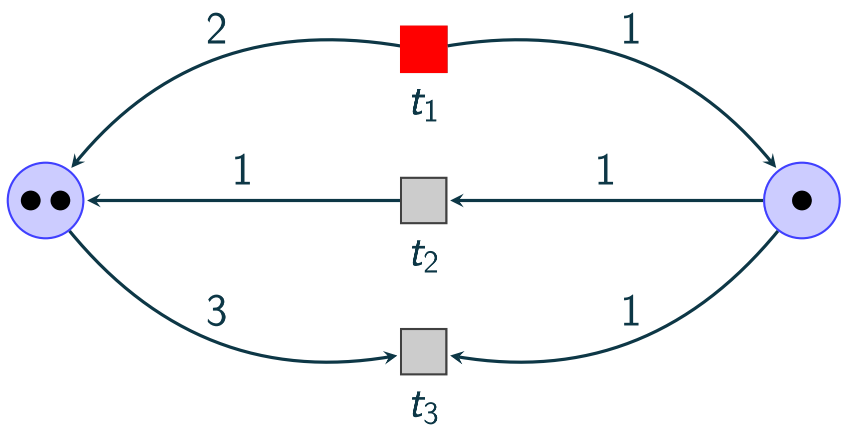
(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

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

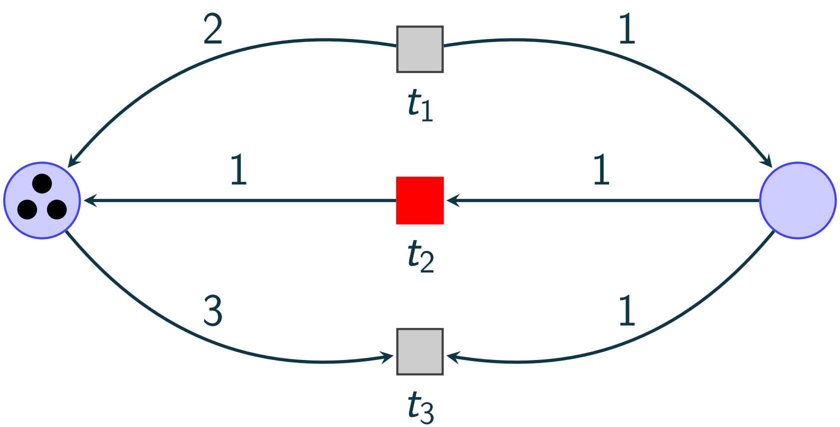
(d, T): d – dimension, $T \subseteq$ $\cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

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

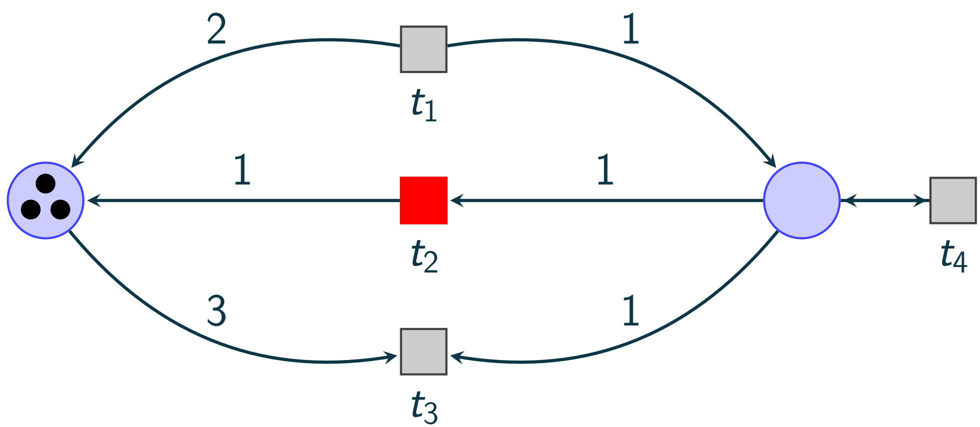
(d, T): d – dimension, $T \subseteq \cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

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



Petri nets

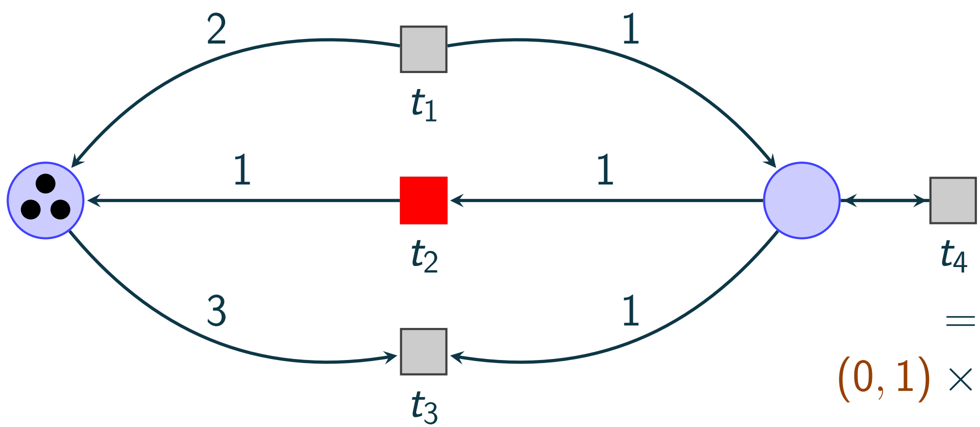
(d, T): d – dimension, $T \subseteq \cancel{\mathbb{Z}^d}$ $\mathbb{N}^d \times \mathbb{N}^d$ (finite)

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 $t_2 = (1, -1)$ $t'_2 = (0, 1) \times (1, 0)$ $(,) = \text{-}(,) + (,)$

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



$=$

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

Plan

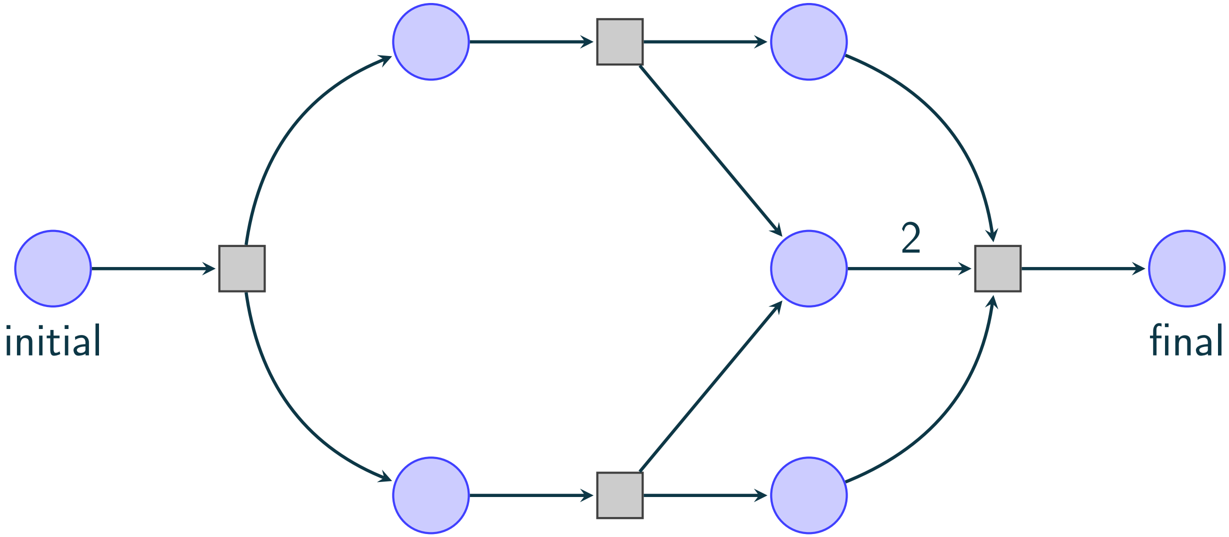
1. Petri nets

2. Workflow nets and soundness

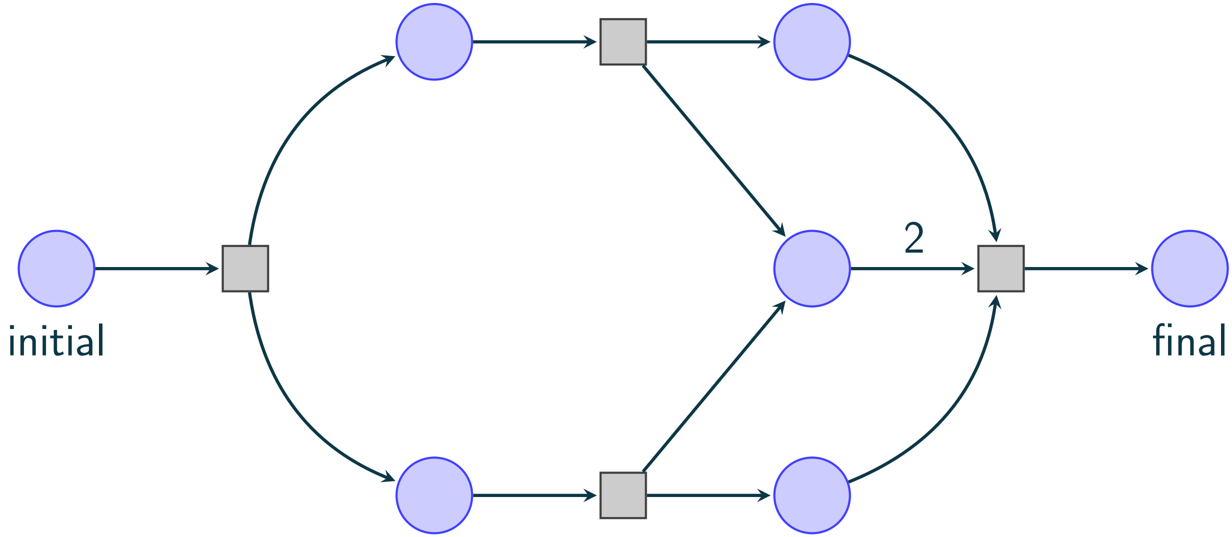
3. Some proofs

Workflow nets

Workflow nets



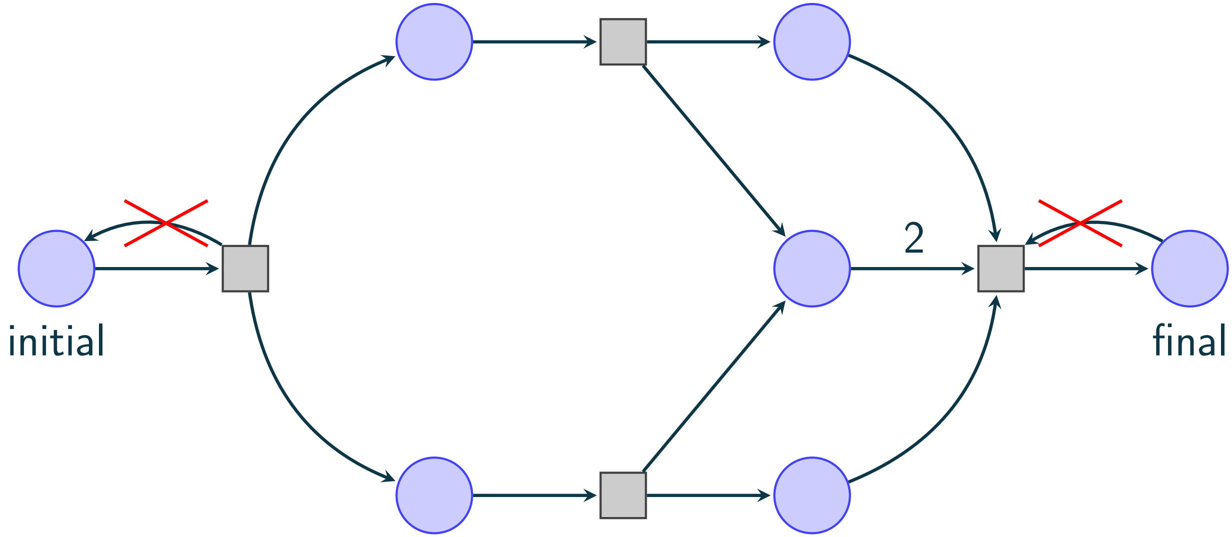
Workflow nets



Workflow nets are Petri nets such that:

- Two places are distinguished: initial and final

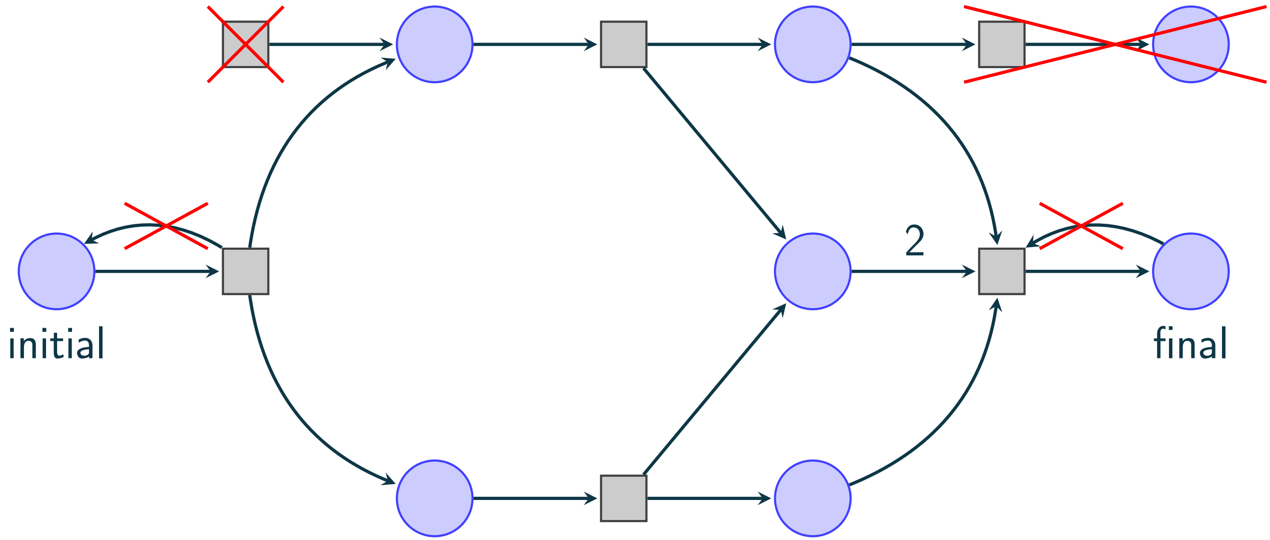
Workflow nets



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

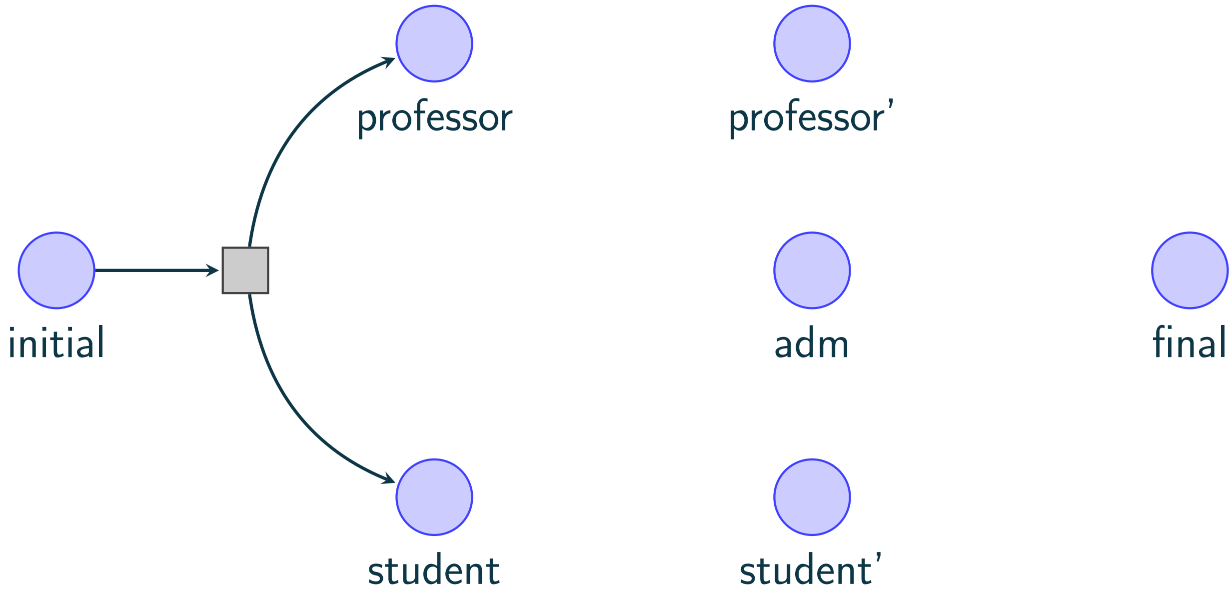
Suppose a professor wants to hire a student

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

Business processes

Suppose a professor wants to hire a student

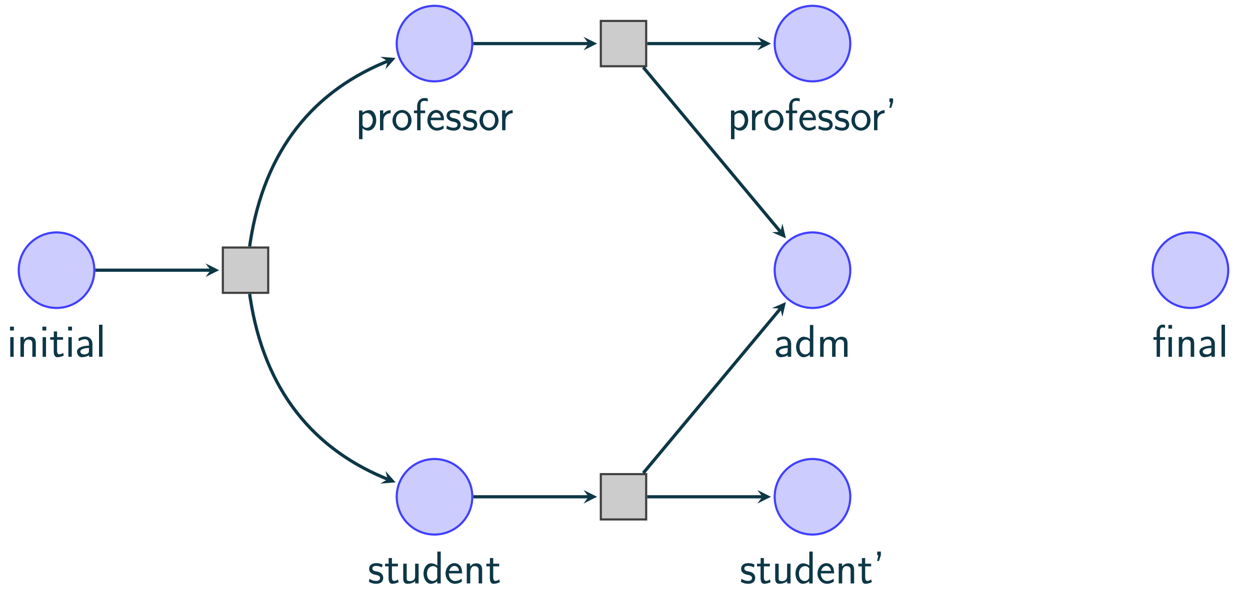
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

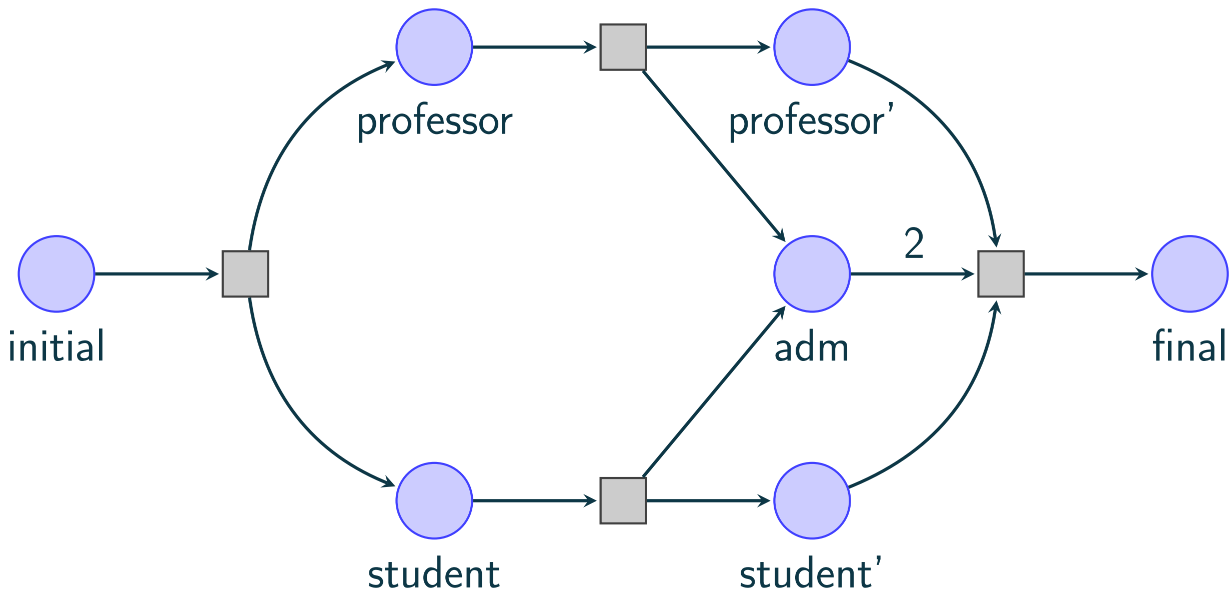
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

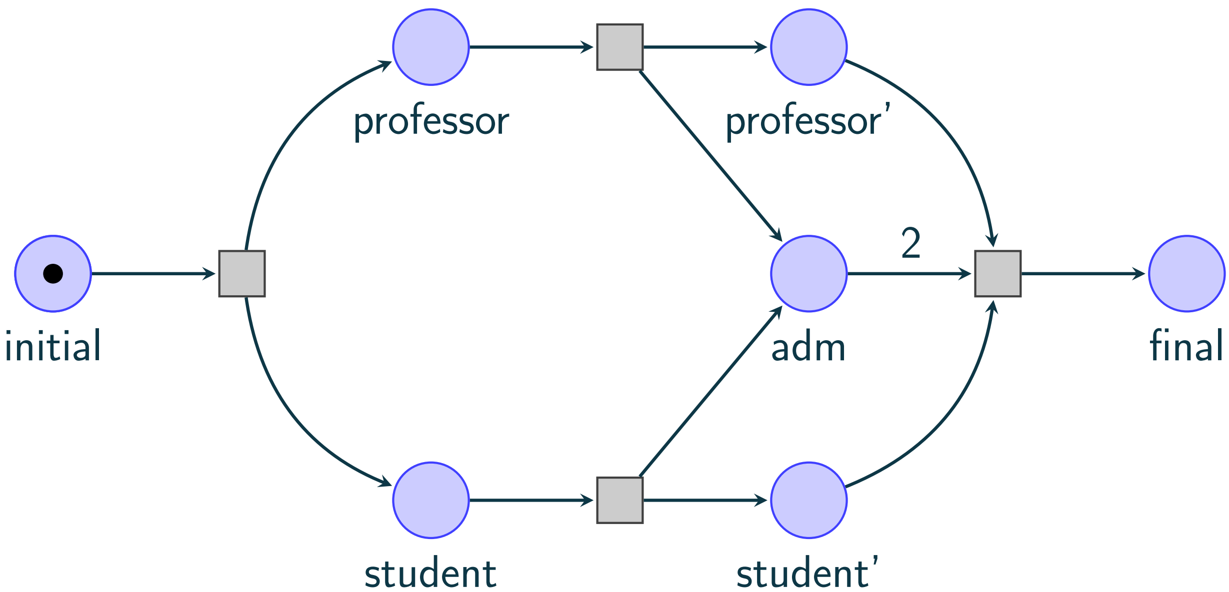
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

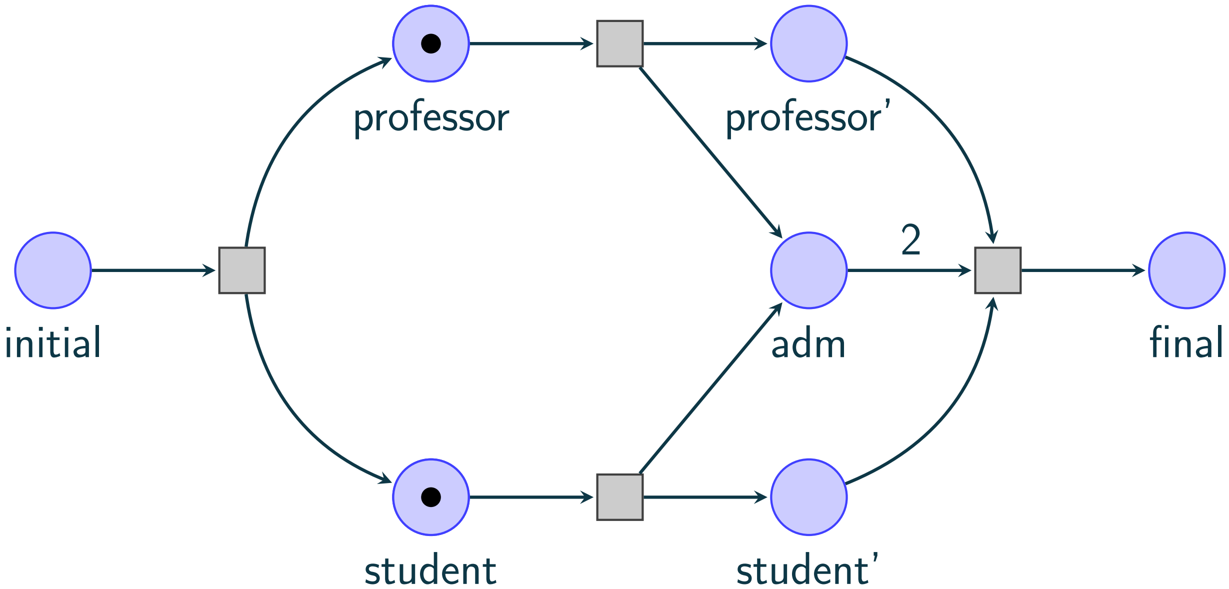
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

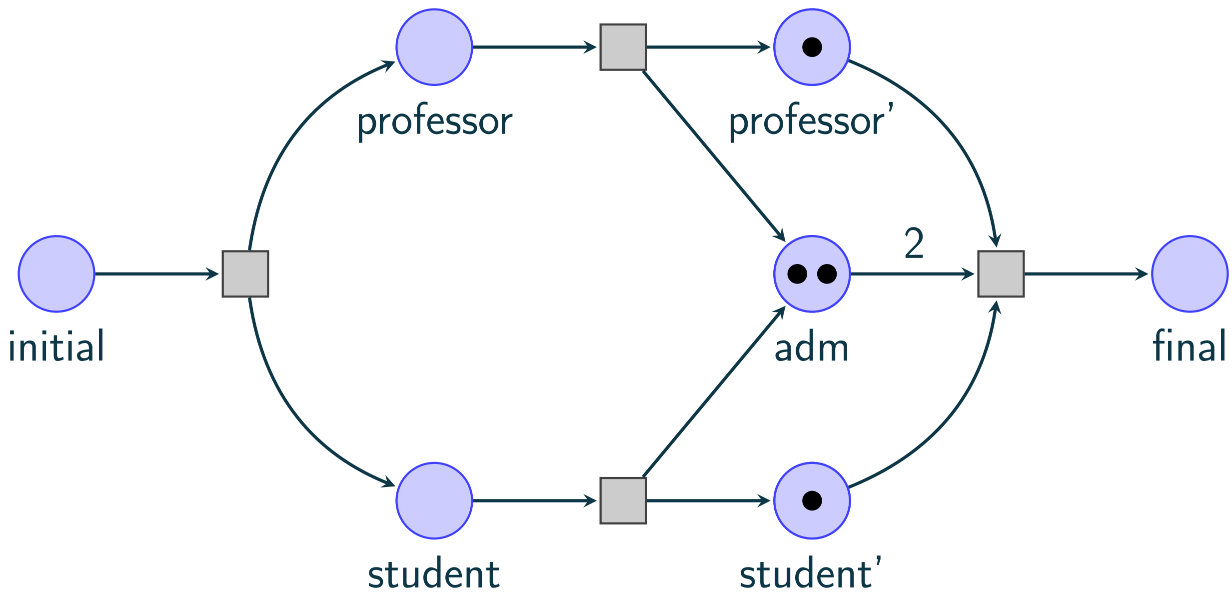
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

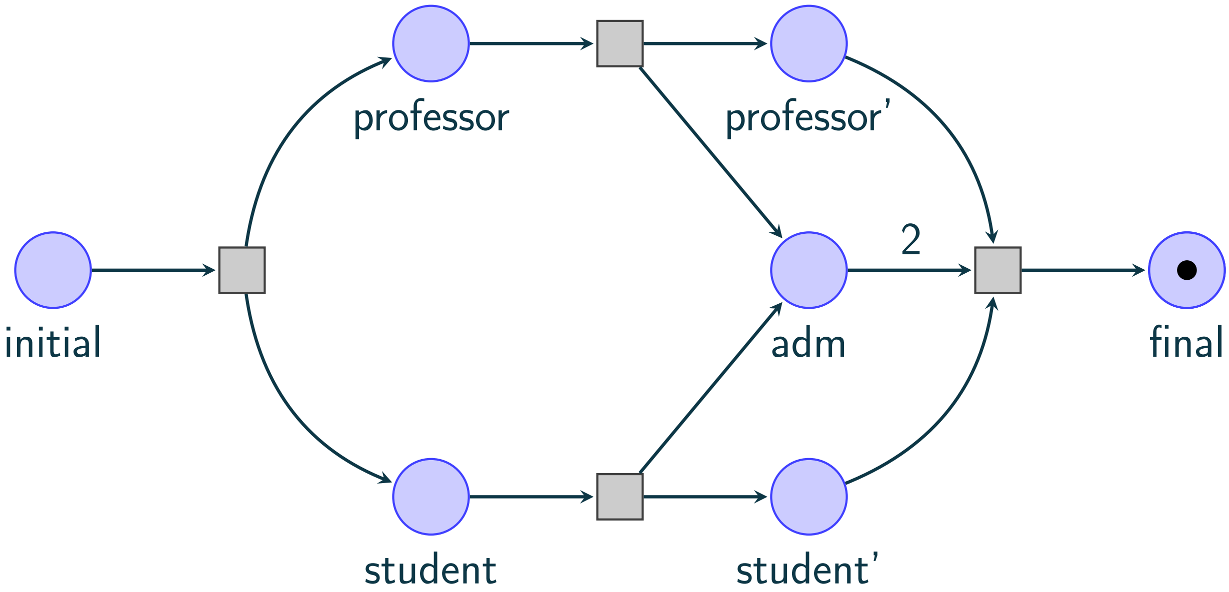
- The student and the professor need to deal with the administration



Business processes

Suppose a professor wants to hire a student

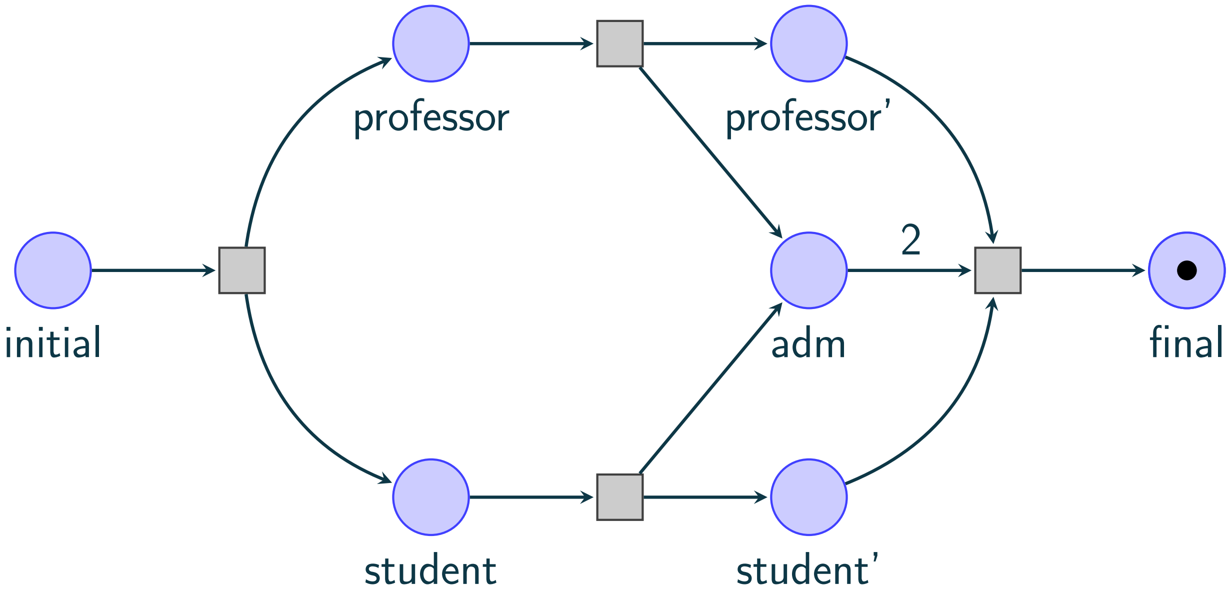
- The student and the professor need to deal with the administration



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)

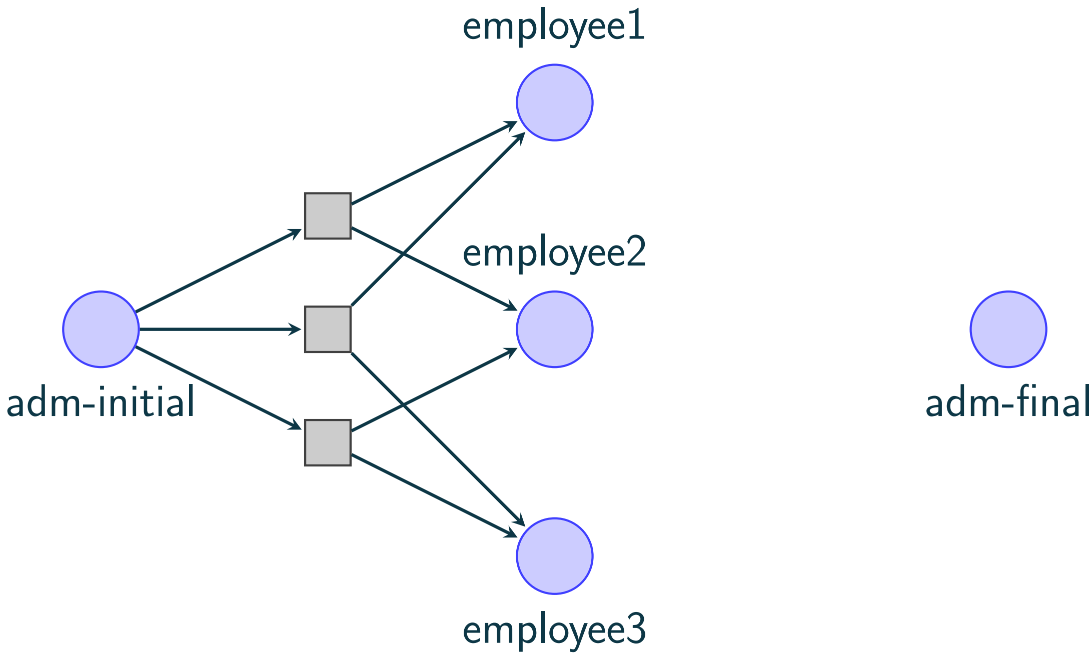
The previous example is sound and even k-sound for every $k > 0$

A more difficult example

Suppose the administration has it's own processes

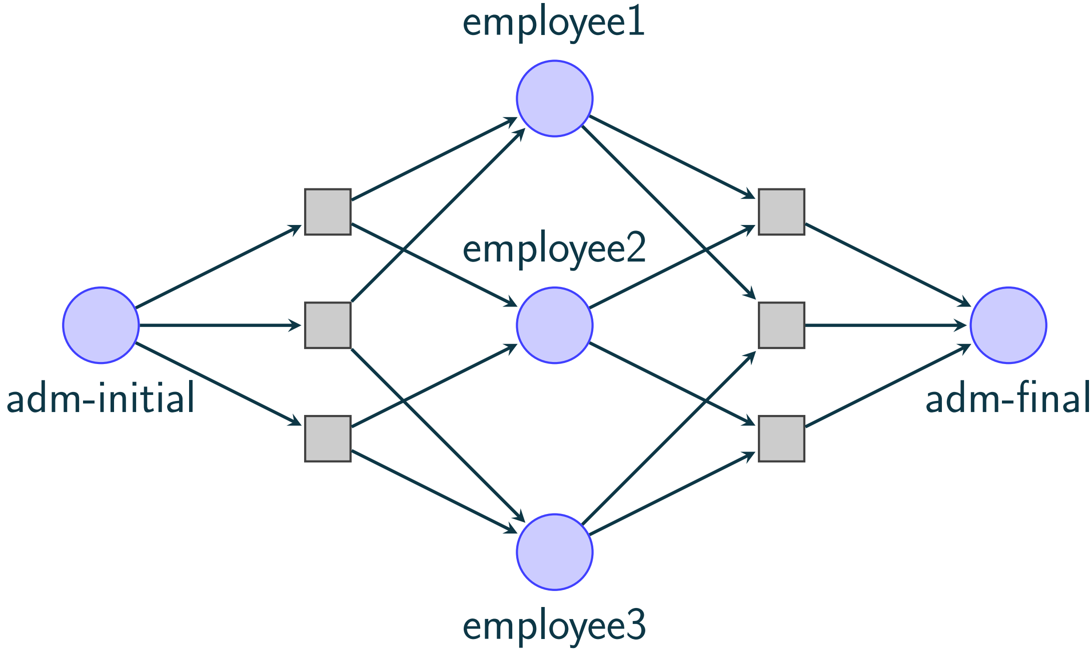
A more difficult example

Suppose the administration has it's own processes



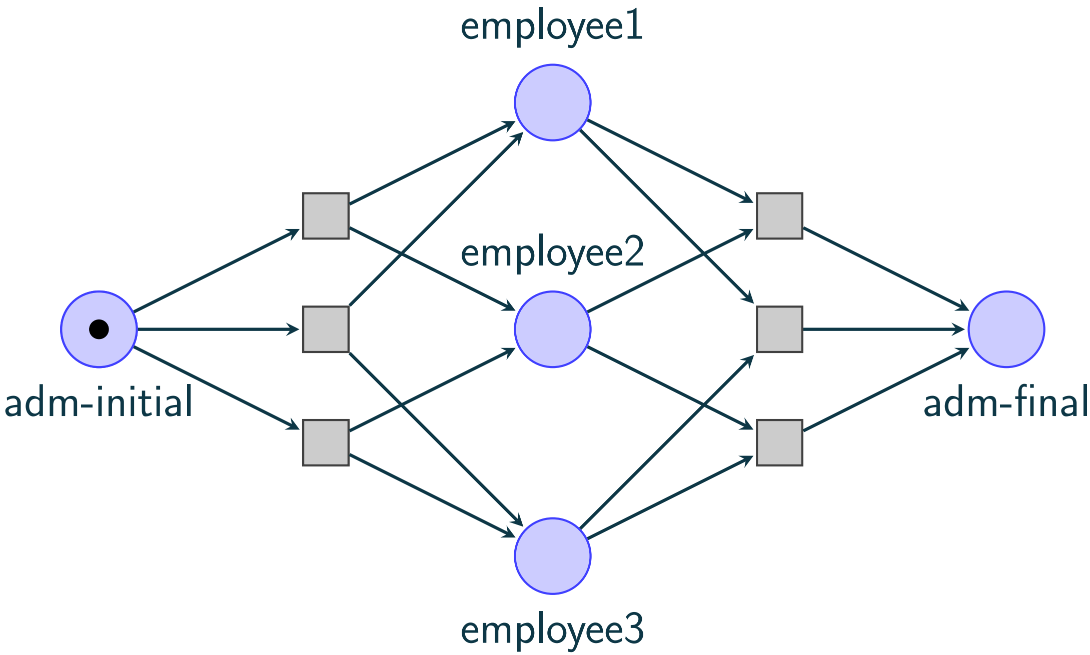
A more difficult example

Suppose the administration has it's own processes



A more difficult example

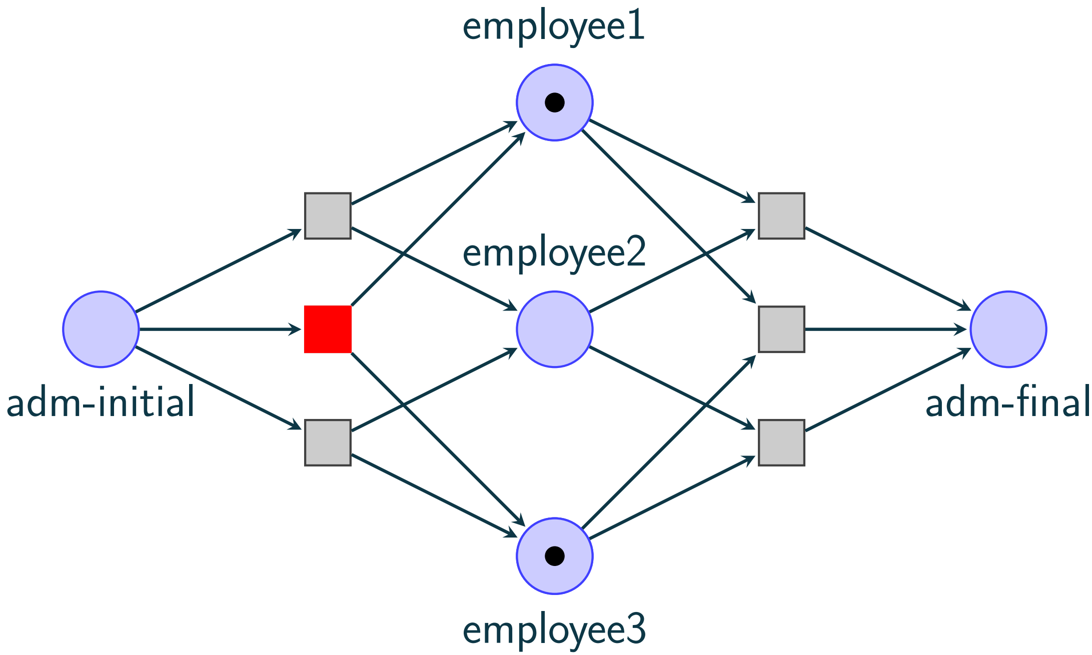
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

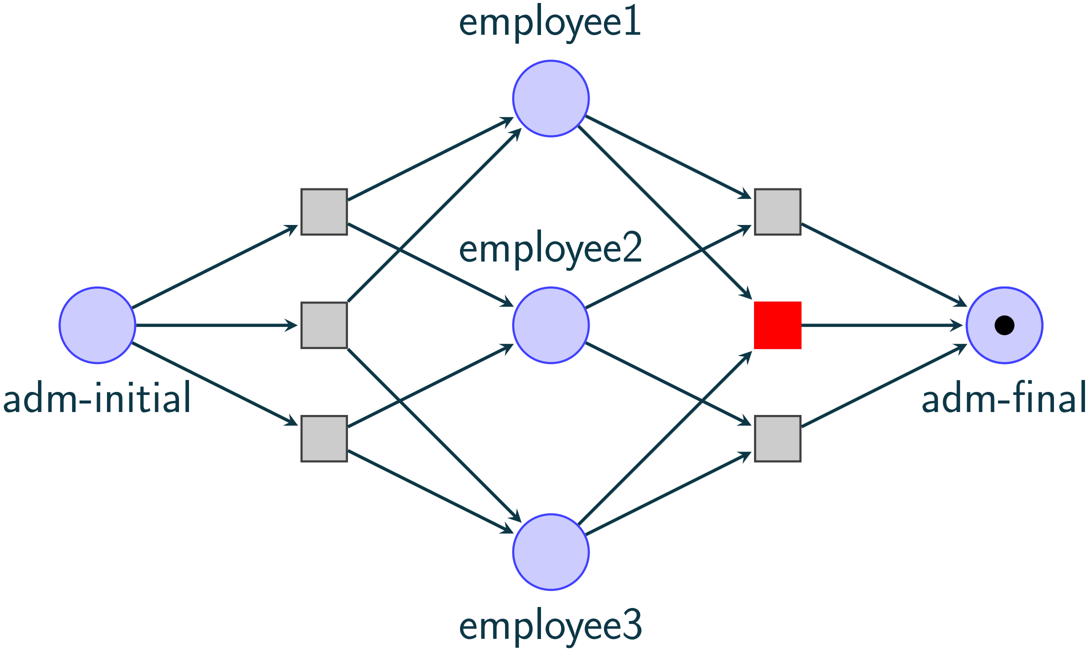
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

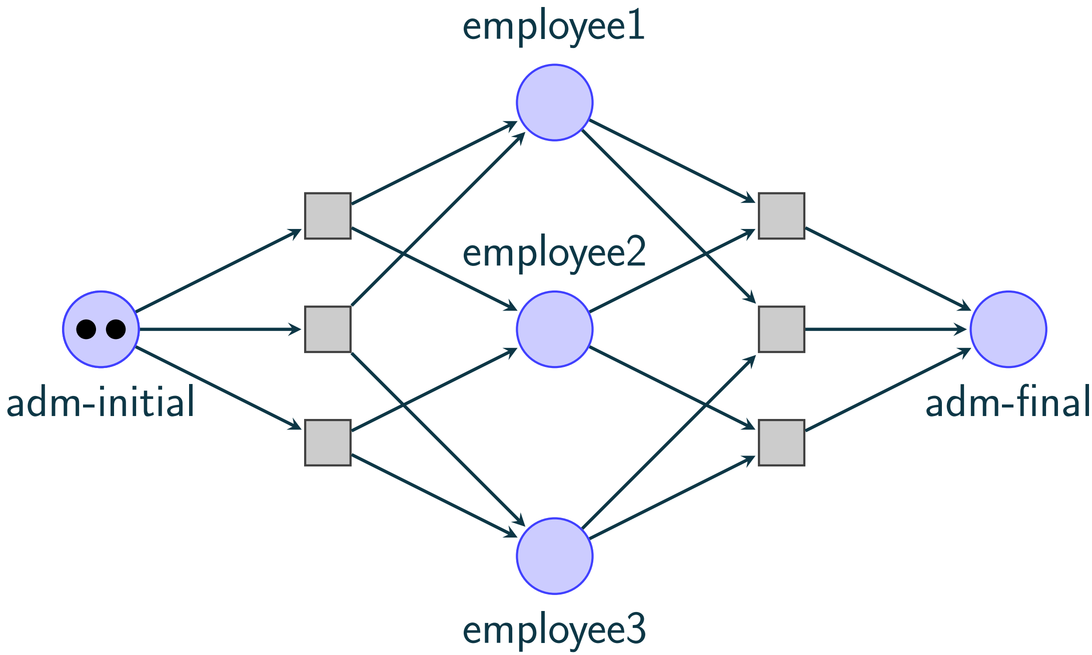
Suppose the administration has it's own processes



- One can verify it's sound

A more difficult example

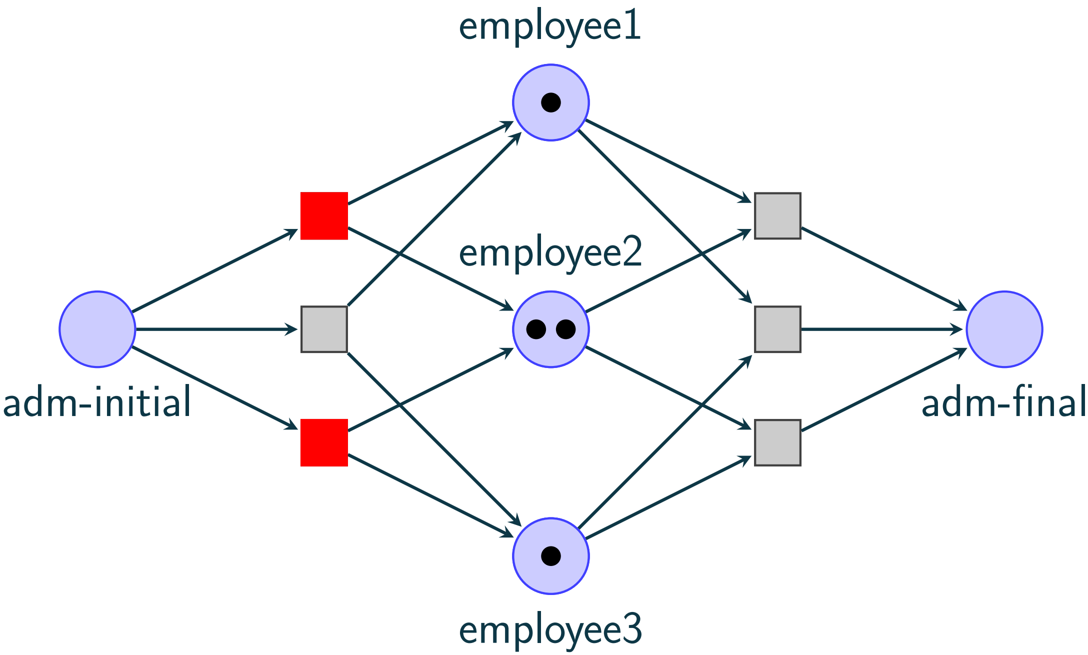
Suppose the administration has it's own processes



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

A more difficult example

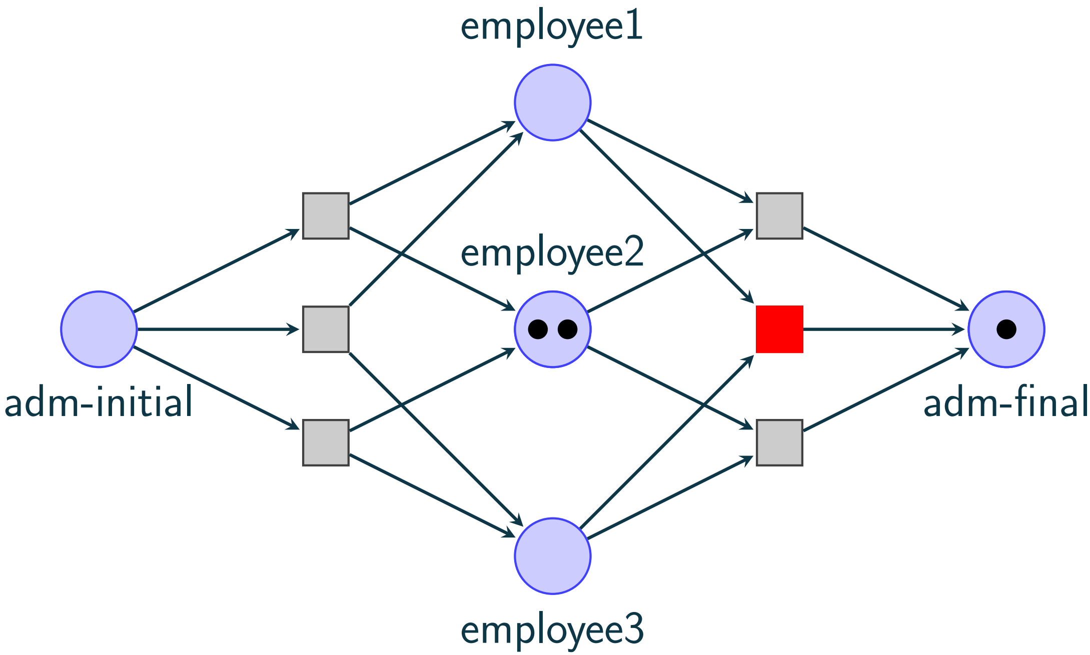
Suppose the administration has it's own processes



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

A more difficult example

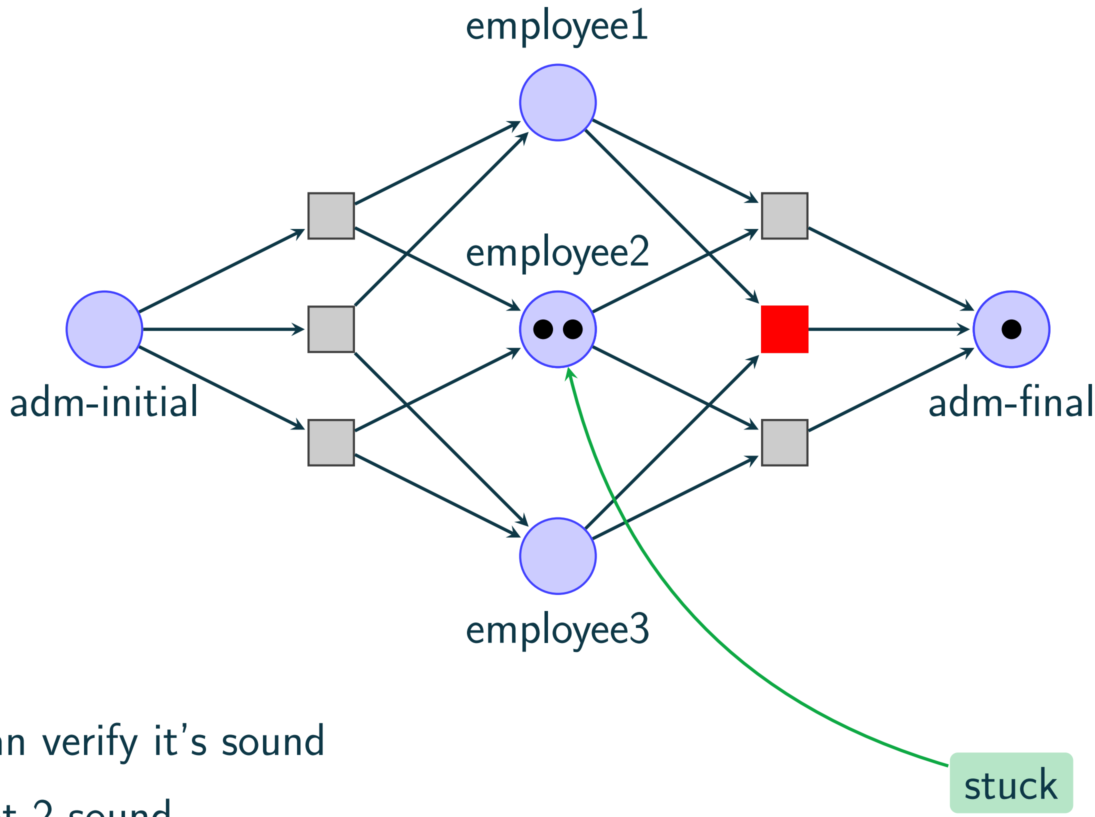
Suppose the administration has it's own processes



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

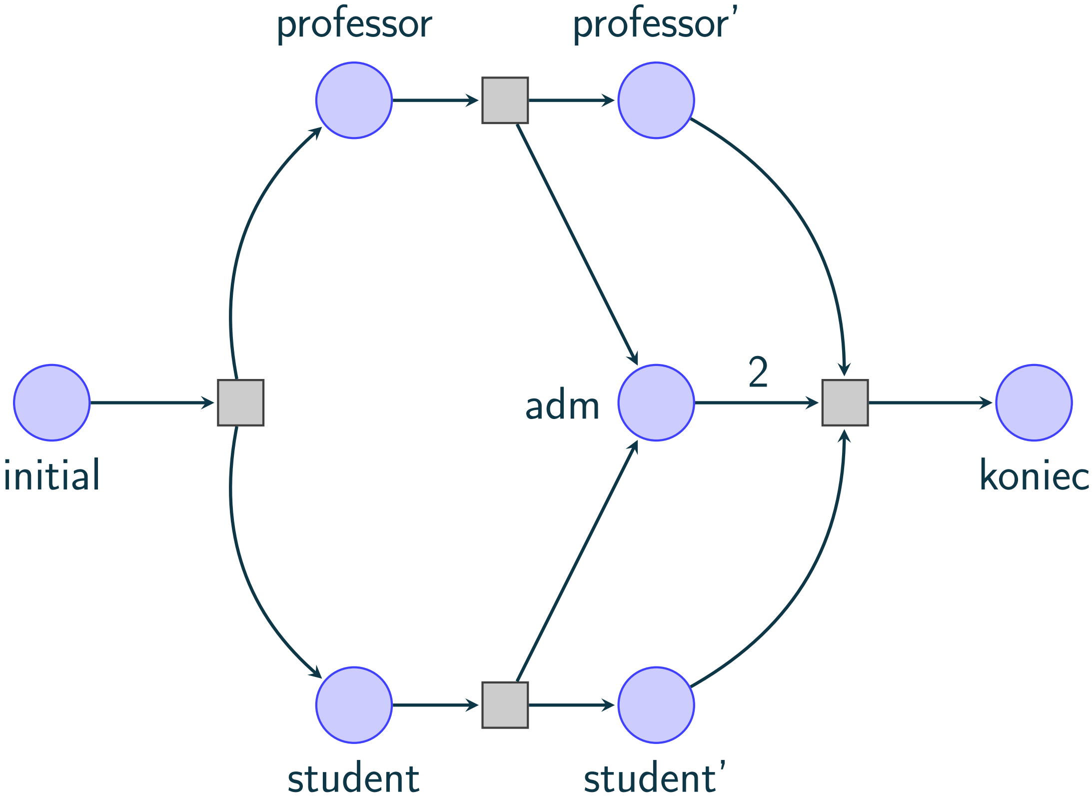
A more difficult example

Suppose the administration has it's own processes

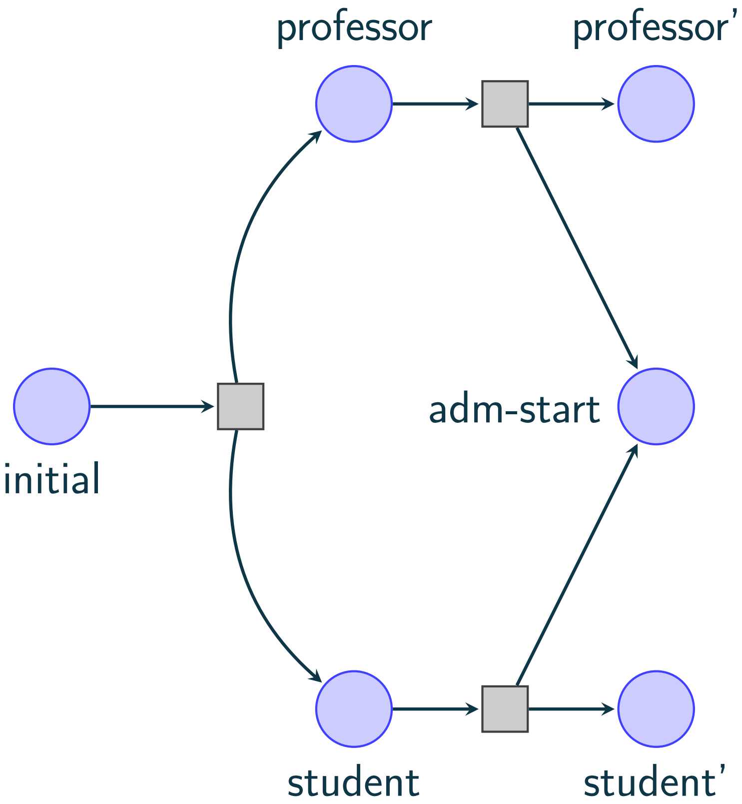


- One can verify it's sound
- But not 2-sound

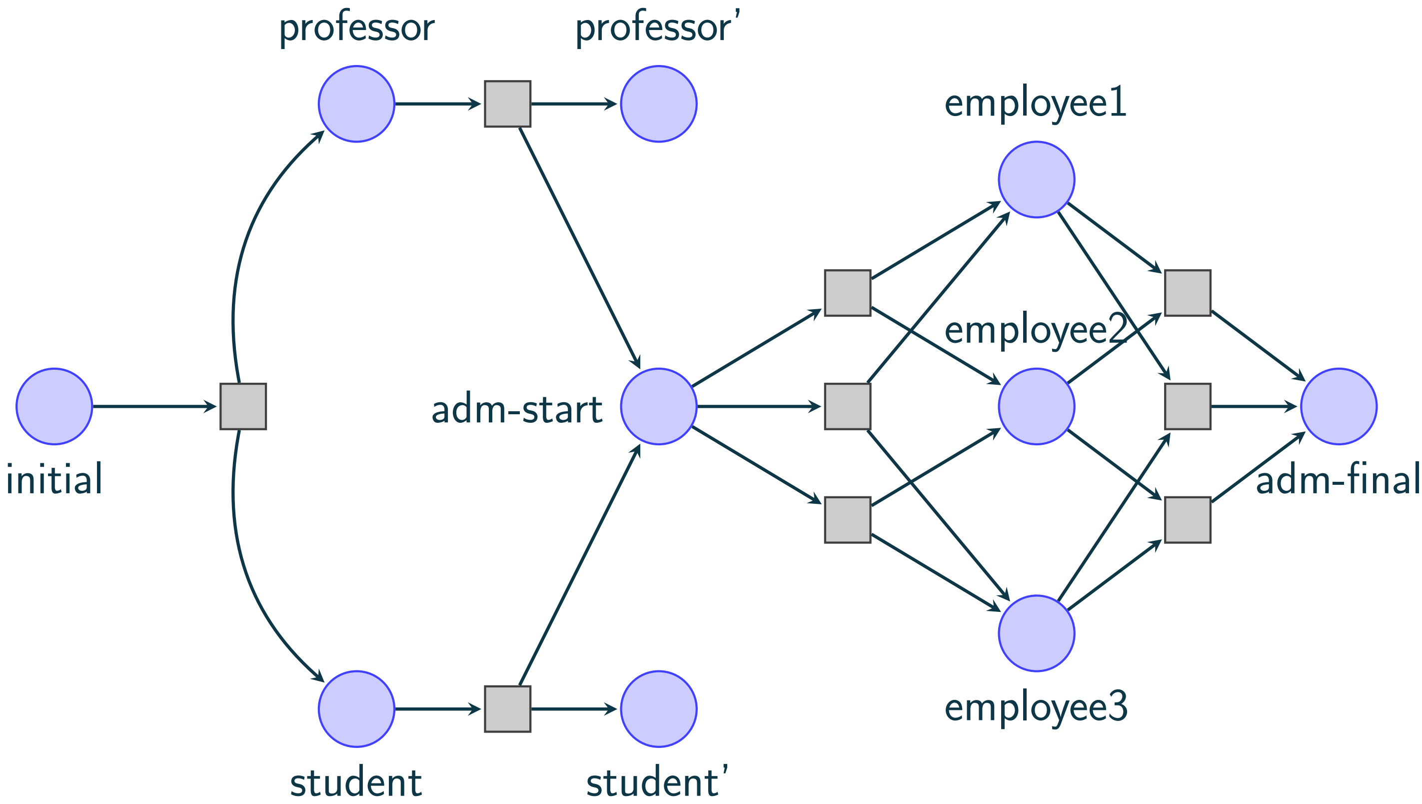
Combining the examples



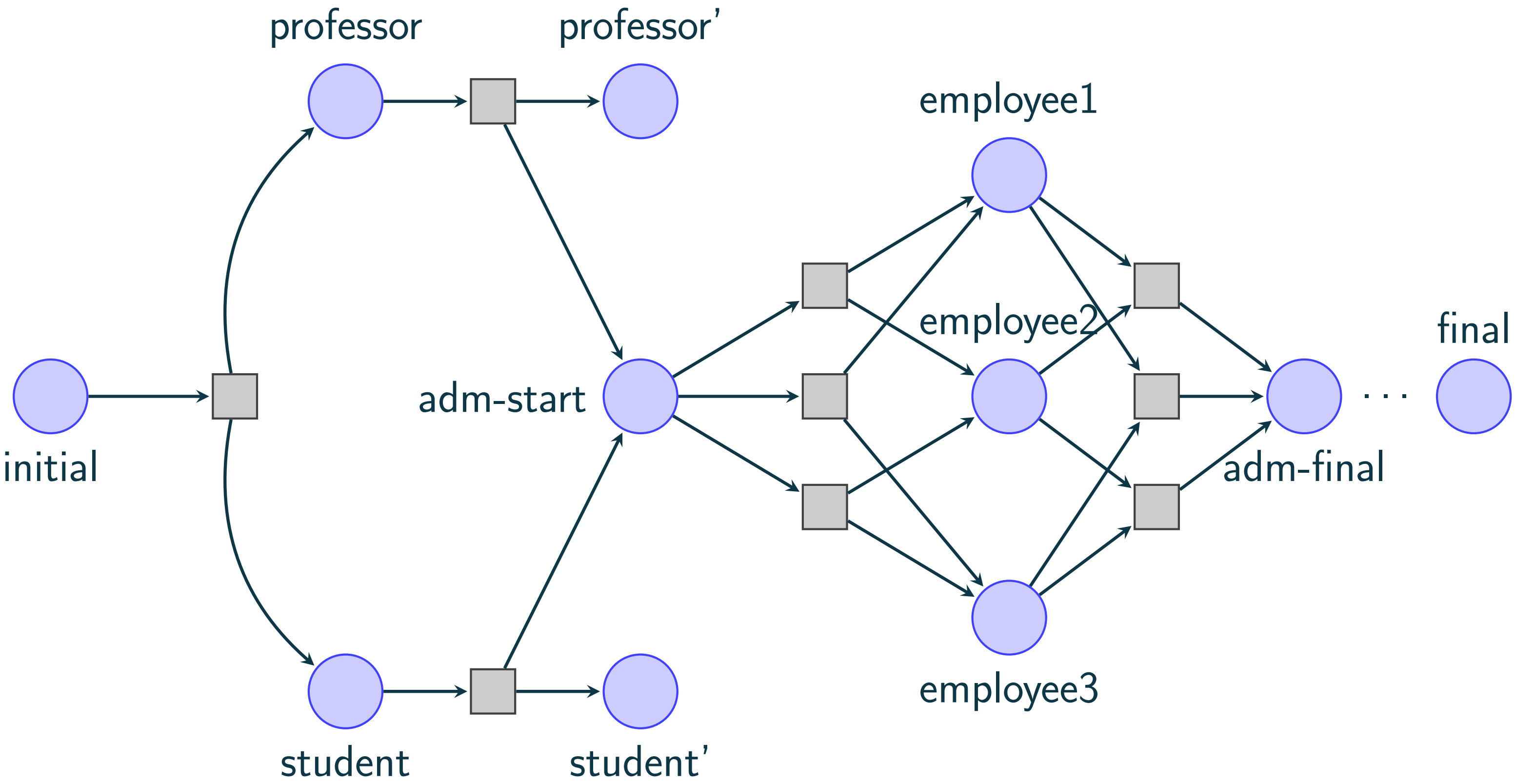
Combining the examples



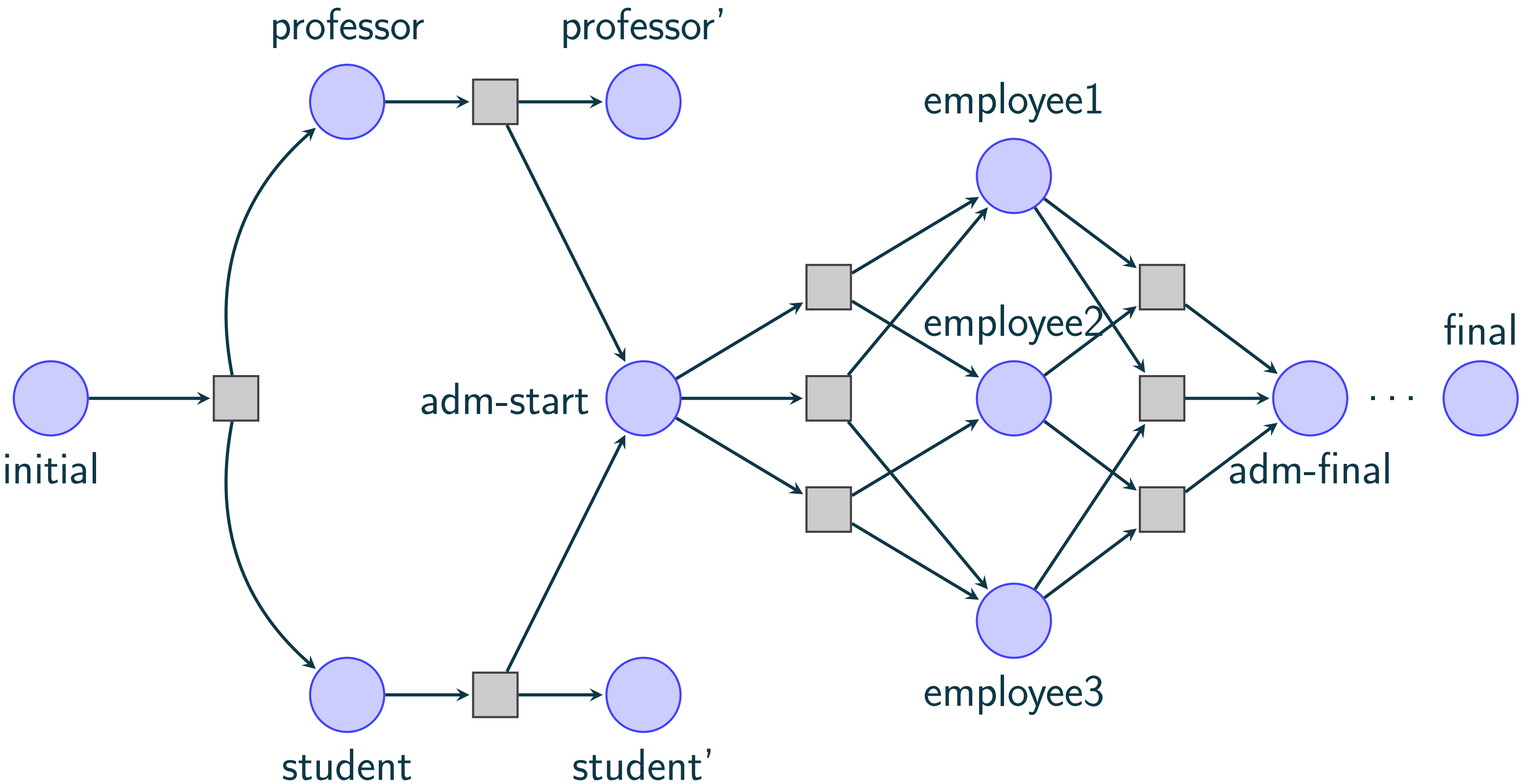
Combining the examples



Combining the examples

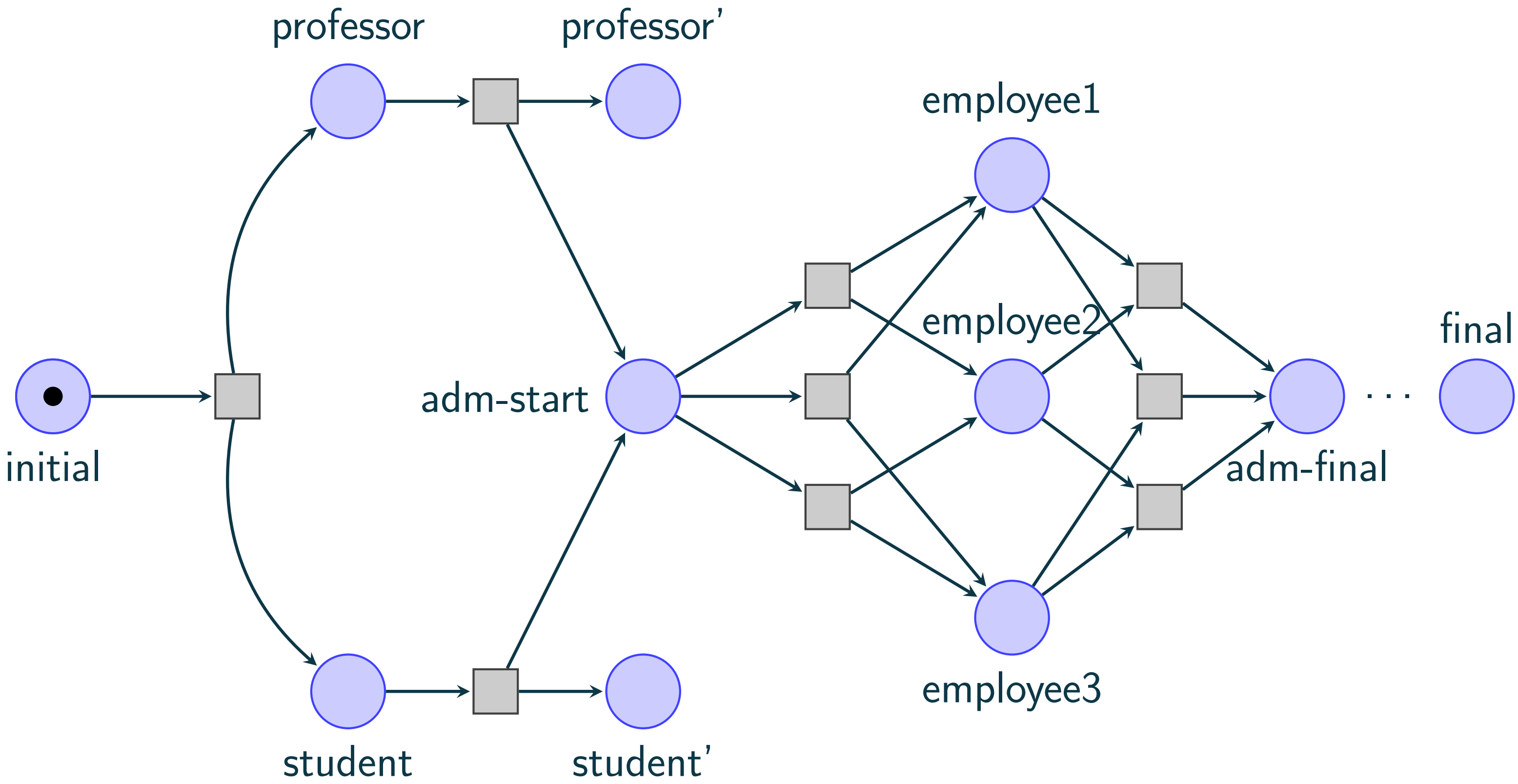


Combining the examples



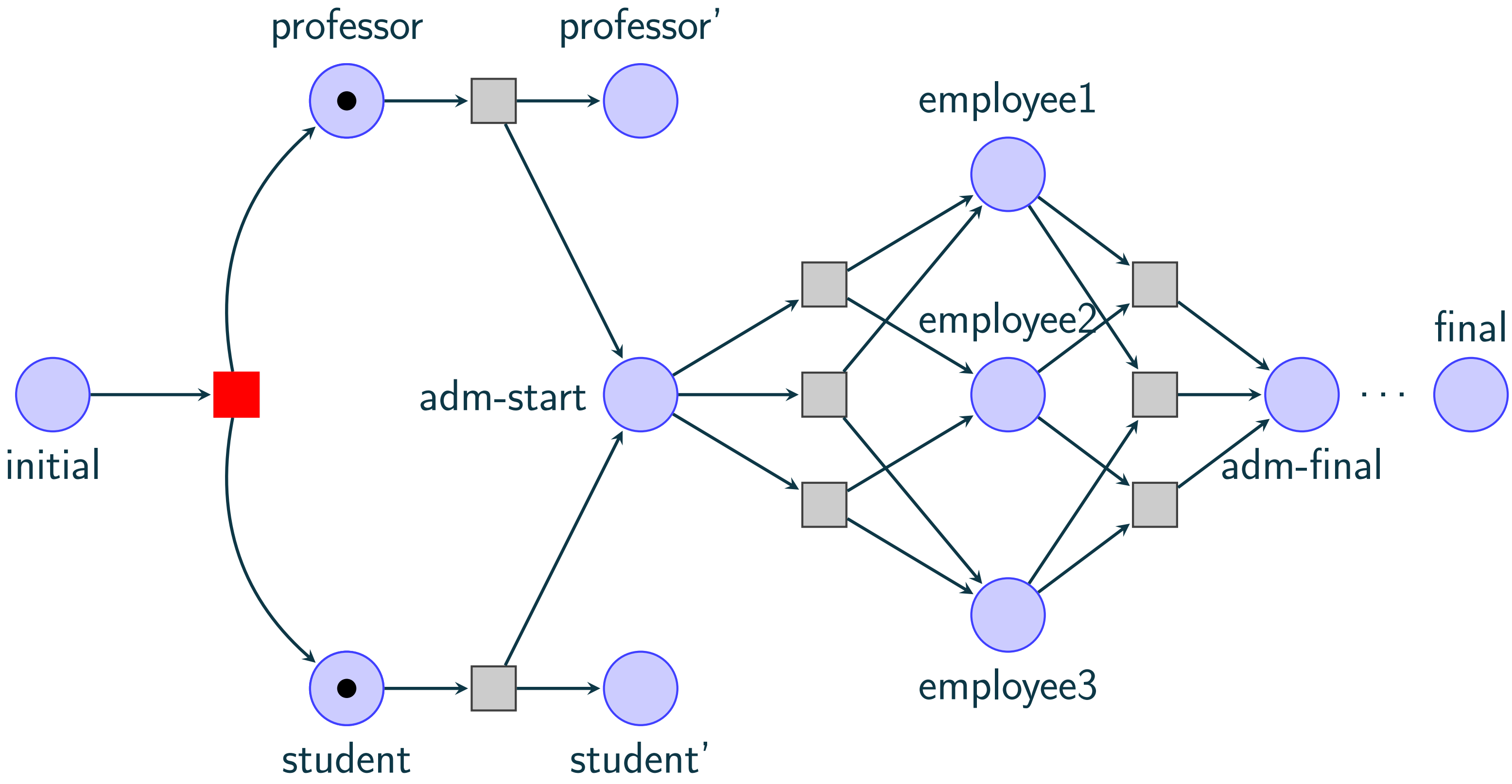
- Recall: both examples were sound

Combining the examples



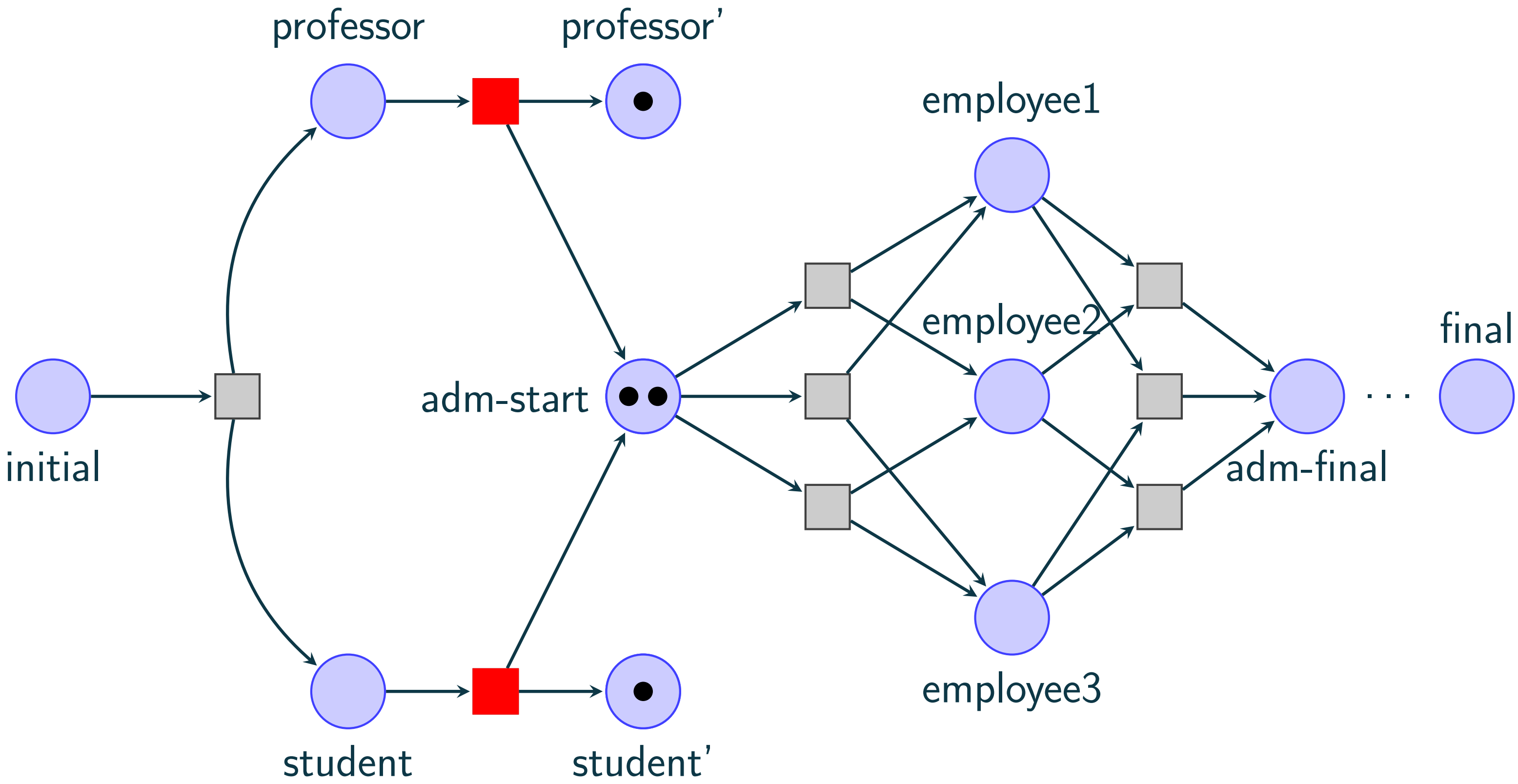
- Recall: both examples were sound

Combining the examples



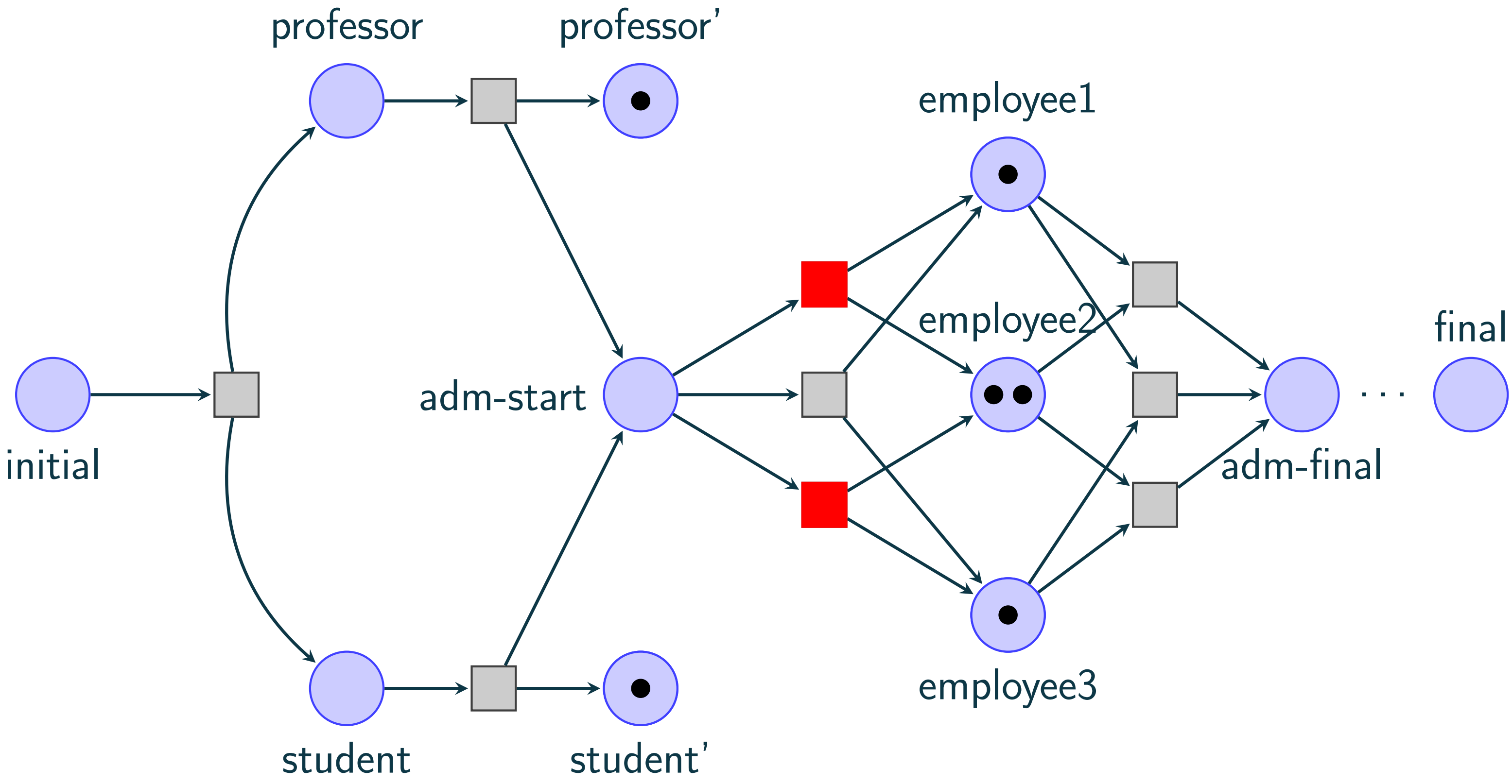
- Recall: both examples were sound

Combining the examples



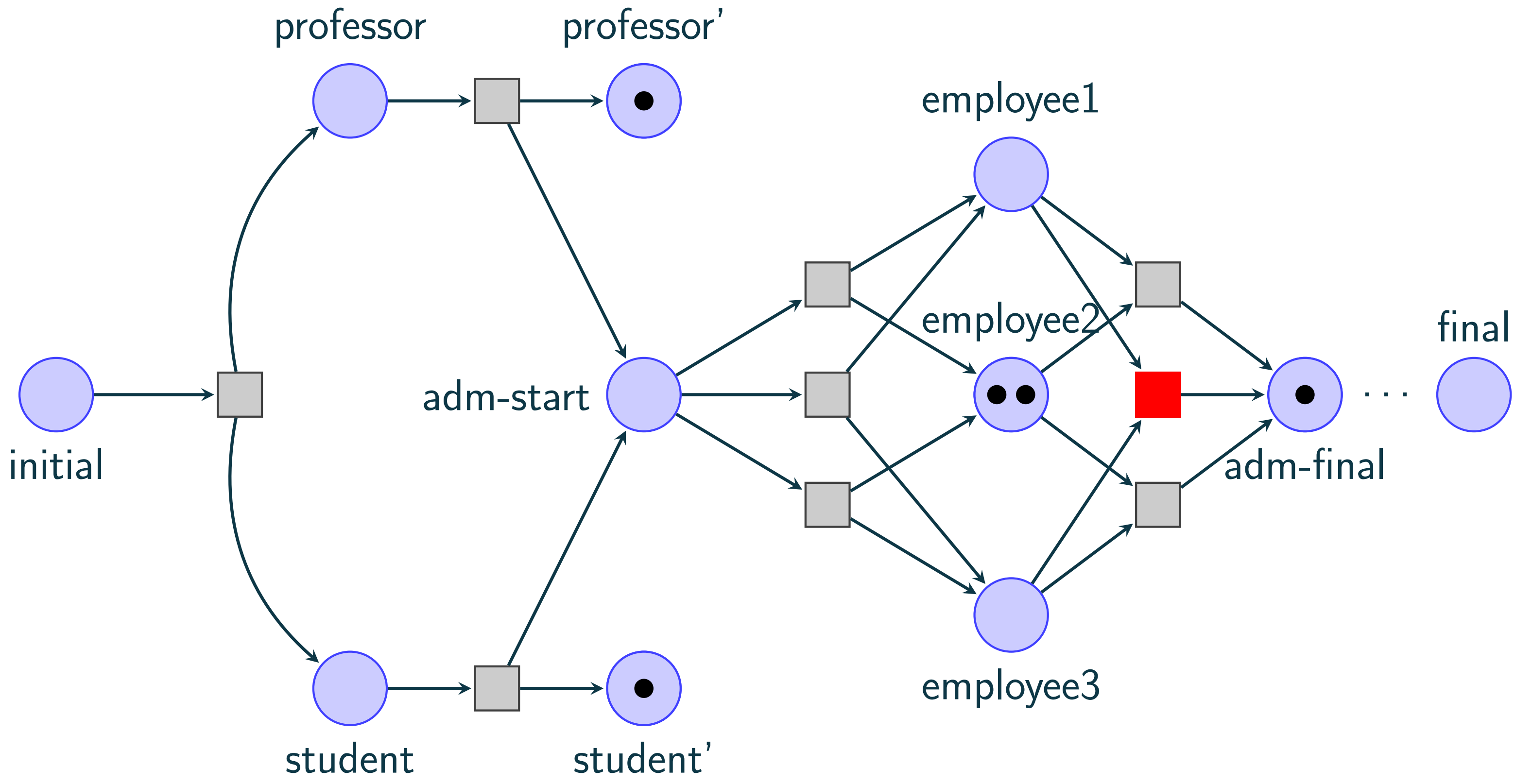
- Recall: both examples were sound

Combining the examples



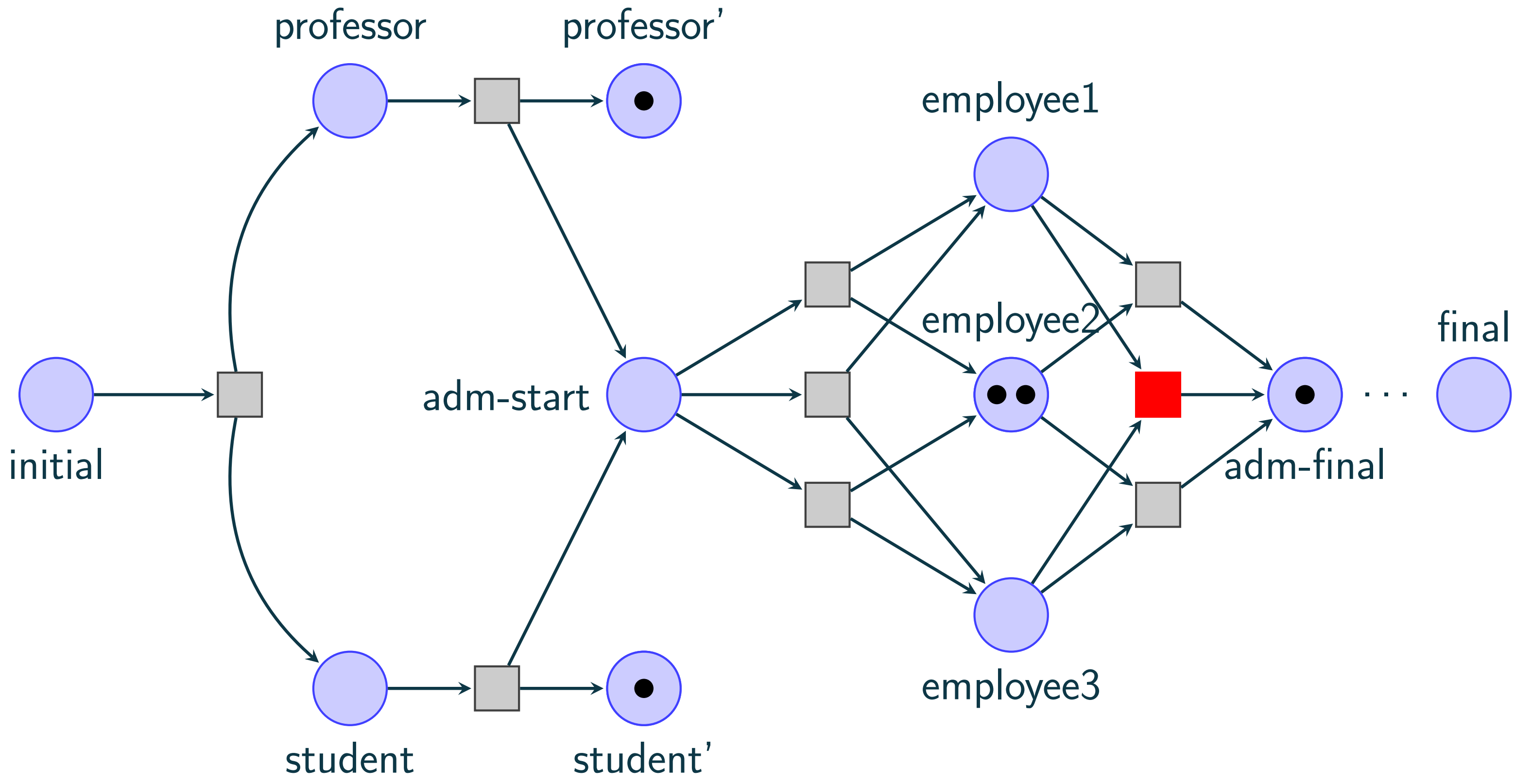
- Recall: both examples were sound

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

Soundness state of art

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

- 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

- 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

- 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

2. Workflow nets and soundness

3. Some proofs

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: $\{i : k\} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ 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

Proof intuition for generalised soundness

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

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

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

- 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 $\boldsymbol{m}' - \boldsymbol{m} > \boldsymbol{0}$

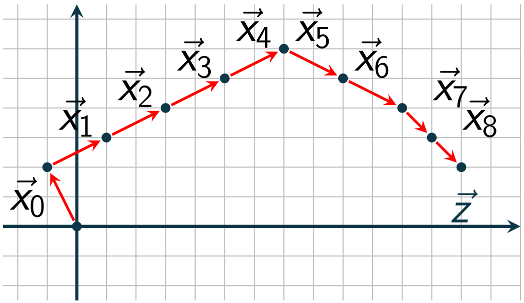
  If generalised sound then $\boldsymbol{m}' \rightarrow^*_{\mathbb{Z}} \{f : k\} + \boldsymbol{m}' - \boldsymbol{m}$

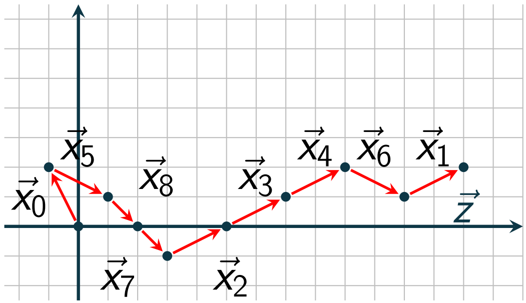
- Suppose we conclude that $\{i : k\} \rightarrow^*_{\mathbb{Z}} \boldsymbol{m}$ then $\boldsymbol{m}$ is small

  Then we can verify generalised soundness in PSPACE

  (go through all reachable configurations)

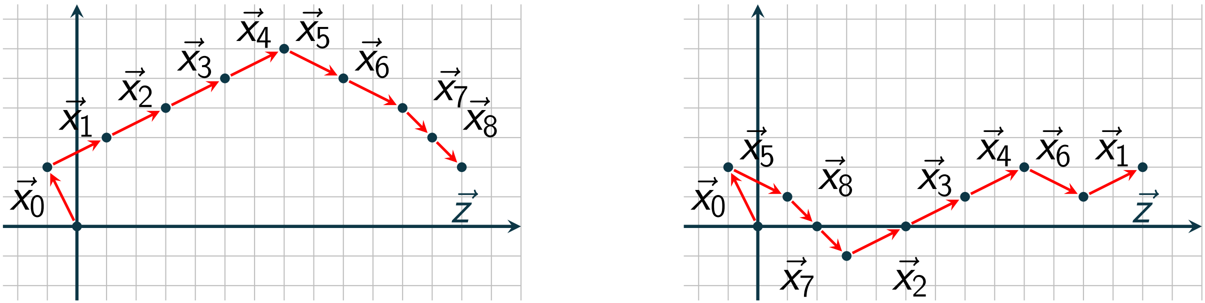
Reachable m are small





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

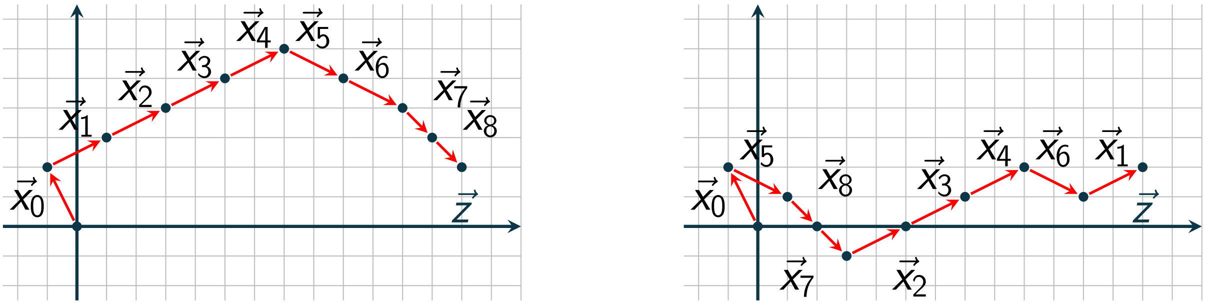
Reachable m are small



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"

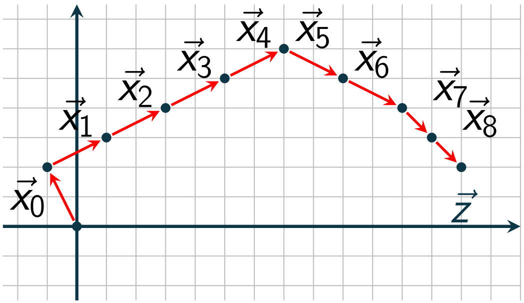
Reachable m are small

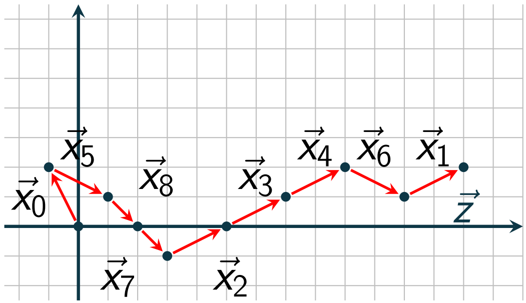


Steinitz Lemma: if $\boldsymbol{m} \rightarrow_{\mathbb{Z}}^{*} \boldsymbol{m}'$ then one can reorder vectors to be "close" to the line $\boldsymbol{m}' - \boldsymbol{m}$

- 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 $\boldsymbol{n} < \boldsymbol{n}'$

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

Conclusion

- Soundness can be seen as a containment problem

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

Conclusion

- Soundness can be seen as a containment problem

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

  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]

- 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]

- For soundness many open problems

  Data nets, reset nets

- The talk is based on recent 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.