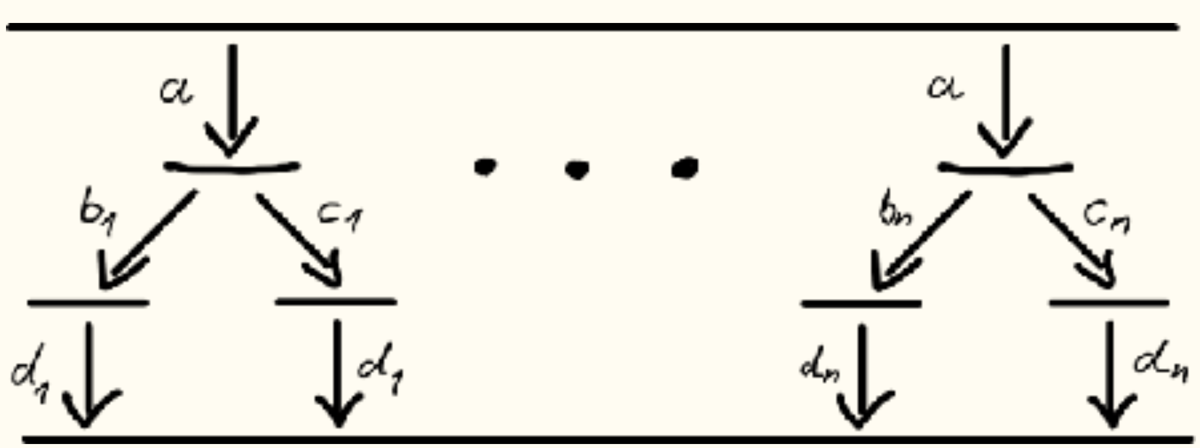
Learning sound deterministic negotiations

Igor Walukiewicz

joint work with Anca Muscholl

Motivation

Learning a finite distributed system may be more efficient than learning a finite automaton representing all the interleavings of the system.



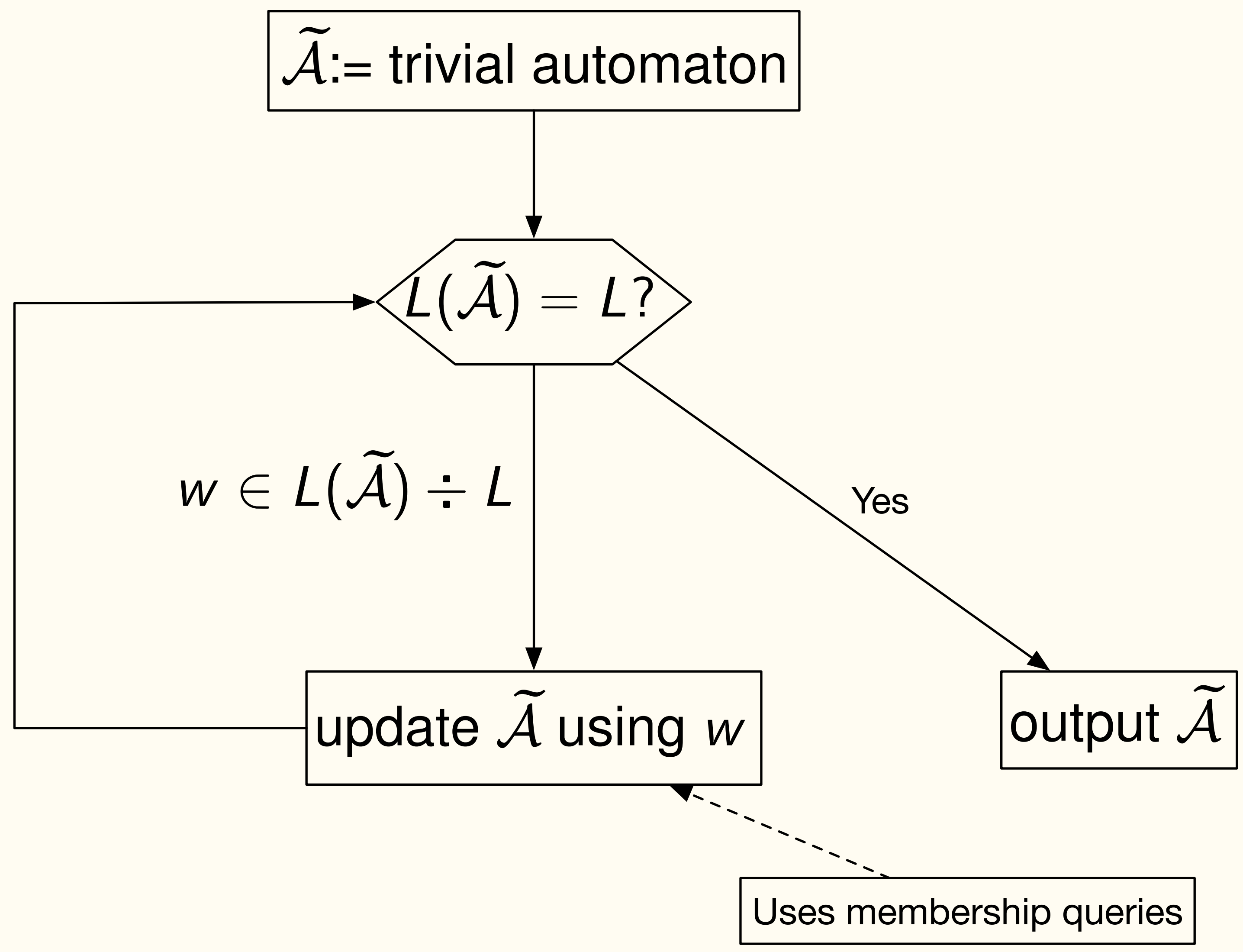
Active learning finite automata [Angluin'87]

Teacher knows a regular language L. Learner wants to construct a finite automaton for L.

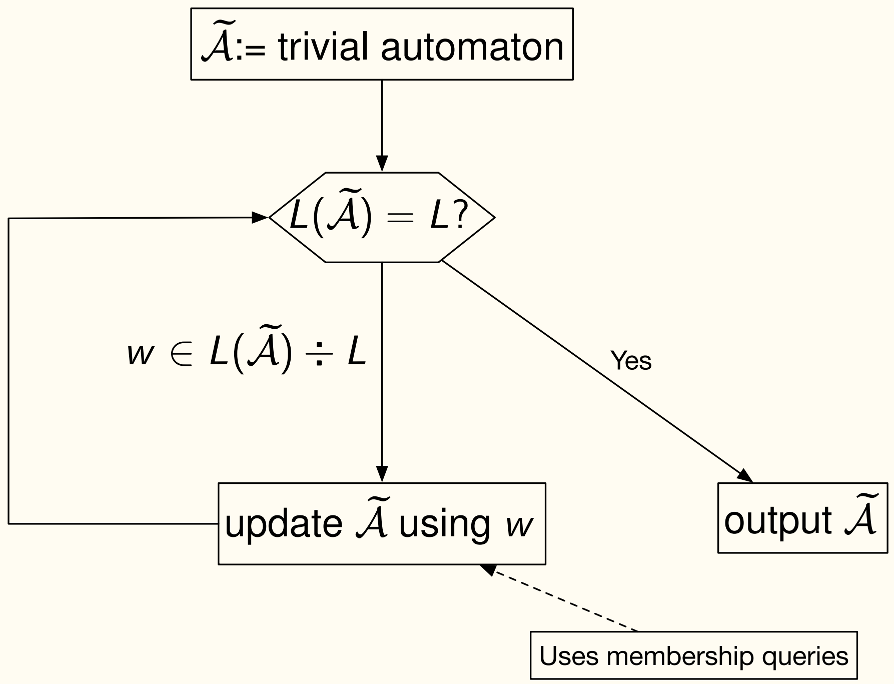
Learner can ask

membership queries: $w \in L?$

equivalence queries: $L(\widetilde{\mathcal{A}}) = L$



Active learning finite automata [Angluin'87]

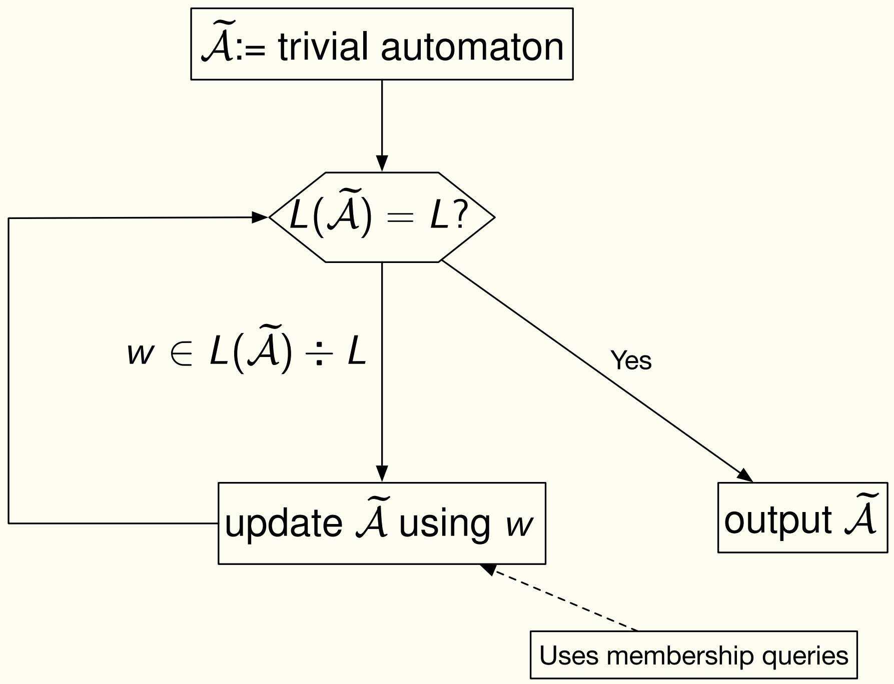


O(s(s+log(m)) membership queries

s equivalence queries

(s: the size of the minimal det. automaton for L)

Active learning finite automata [Angluin'87]



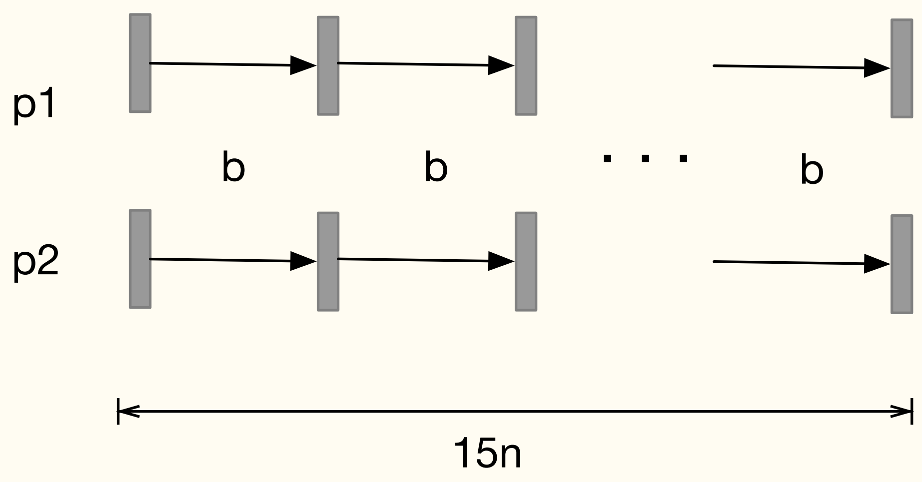
O(s(s+log(m)) membership queries

s equivalence queries

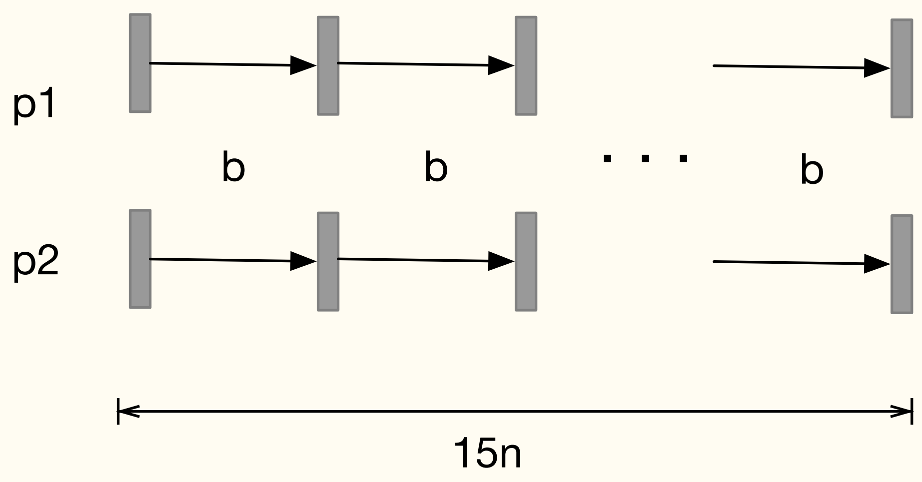
(s: the size of the minimal det. automaton for L)

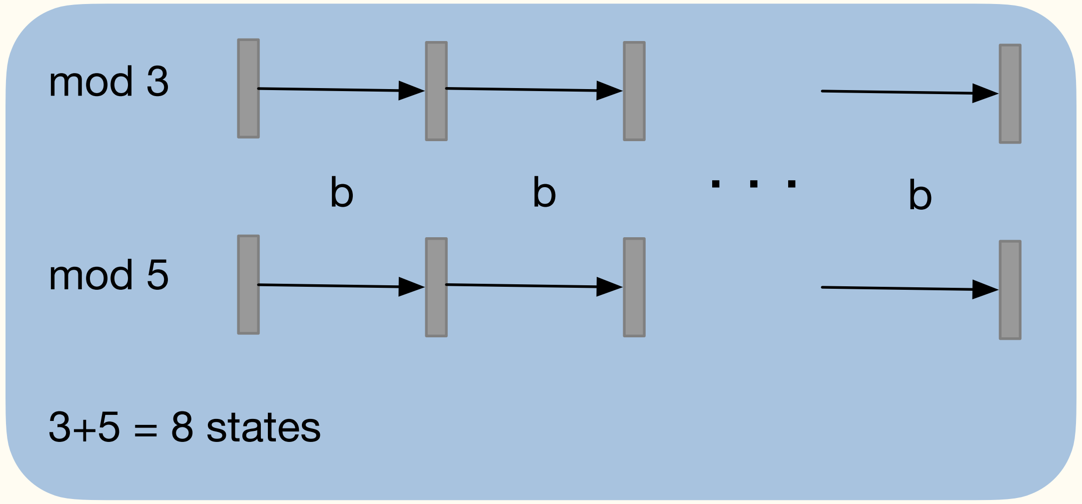
OBS: Learning produces a canonical automaton.

Why learning distributed systems is hard

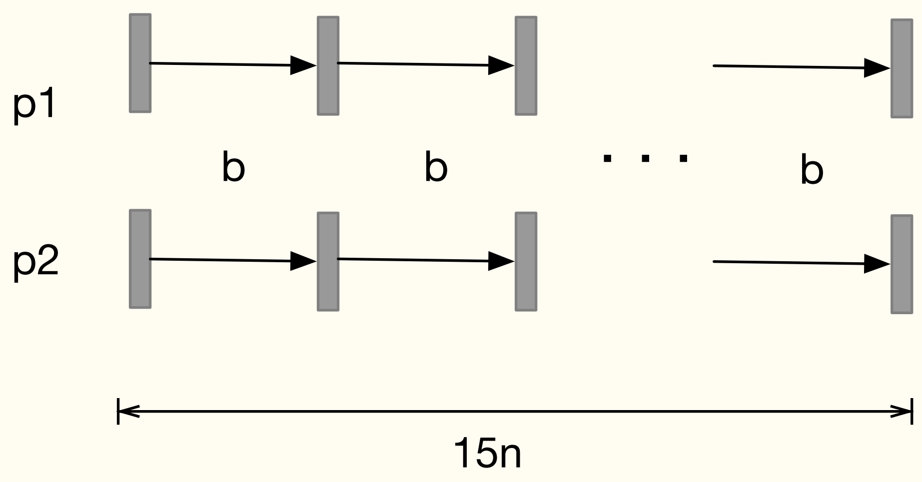


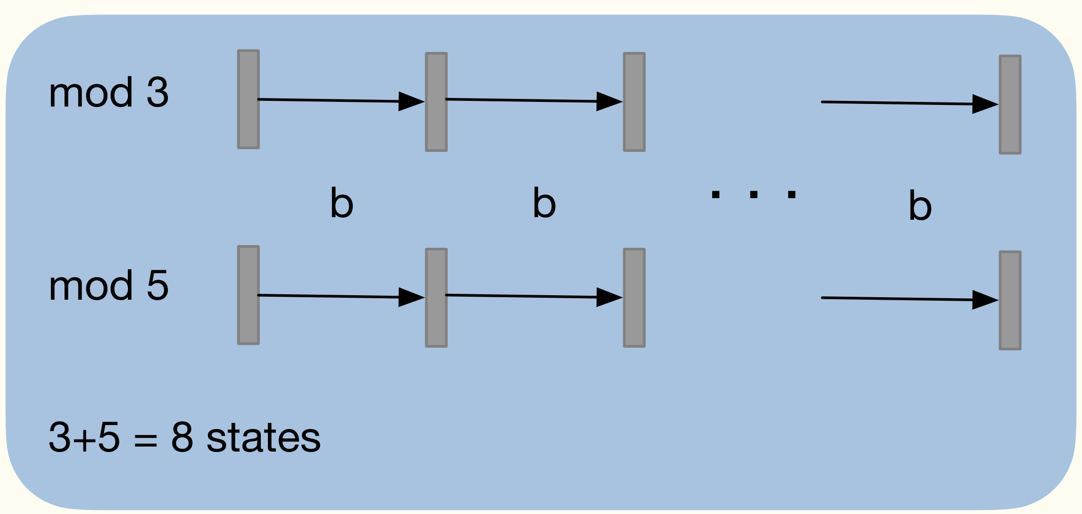
Why learning distributed systems is hard

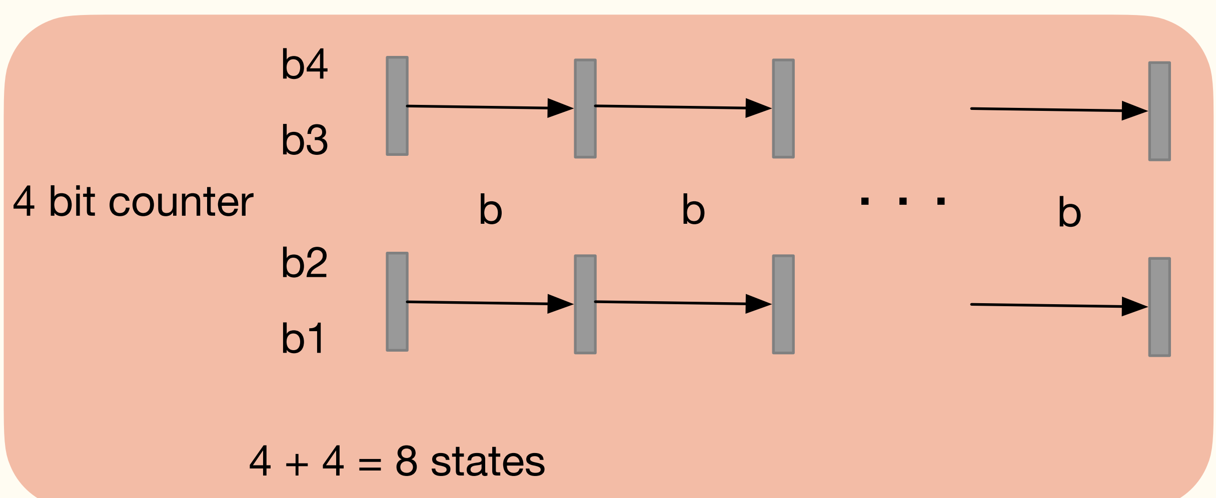




Why learning distributed systems is hard

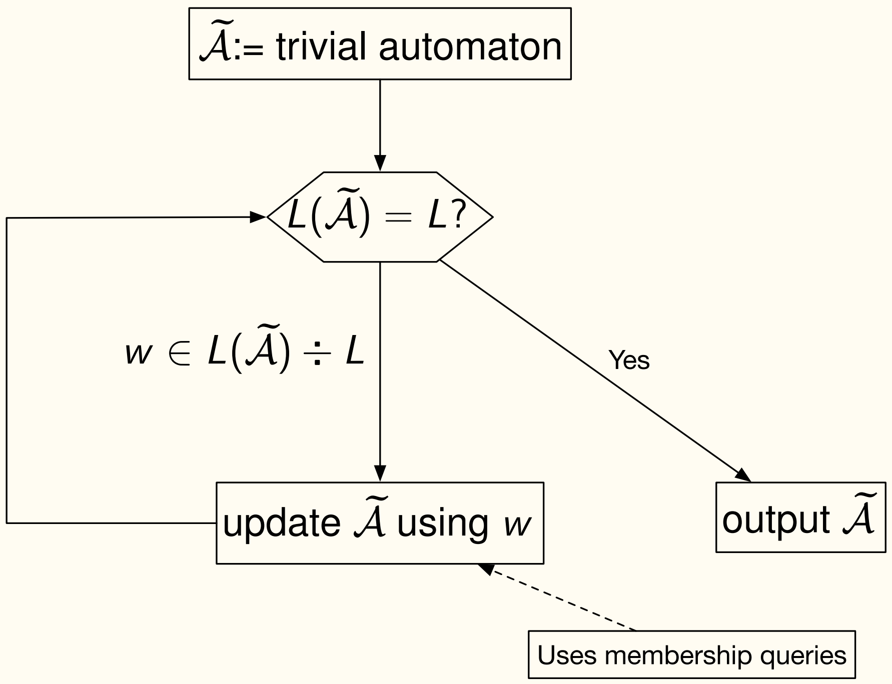






Which of the two is canonical?

Active learning finite automata [Angluin'87]



Tree languages [Drewes and Högberg 2007]

Weighted automata [Balle and Mohri 2015]

Omega-regular languages [Angluin and Fisman 2016]

Nominal automata [Moerman, Sammartino, Silva, Klin, Szynwelski. 2017]

Learning Communicating Automata from MSCs [Bollig, Katoen, Kern, Leucker 2010]

Learning Pomset Automata [Heerdt, Kappé, Rot, Silva 2021]

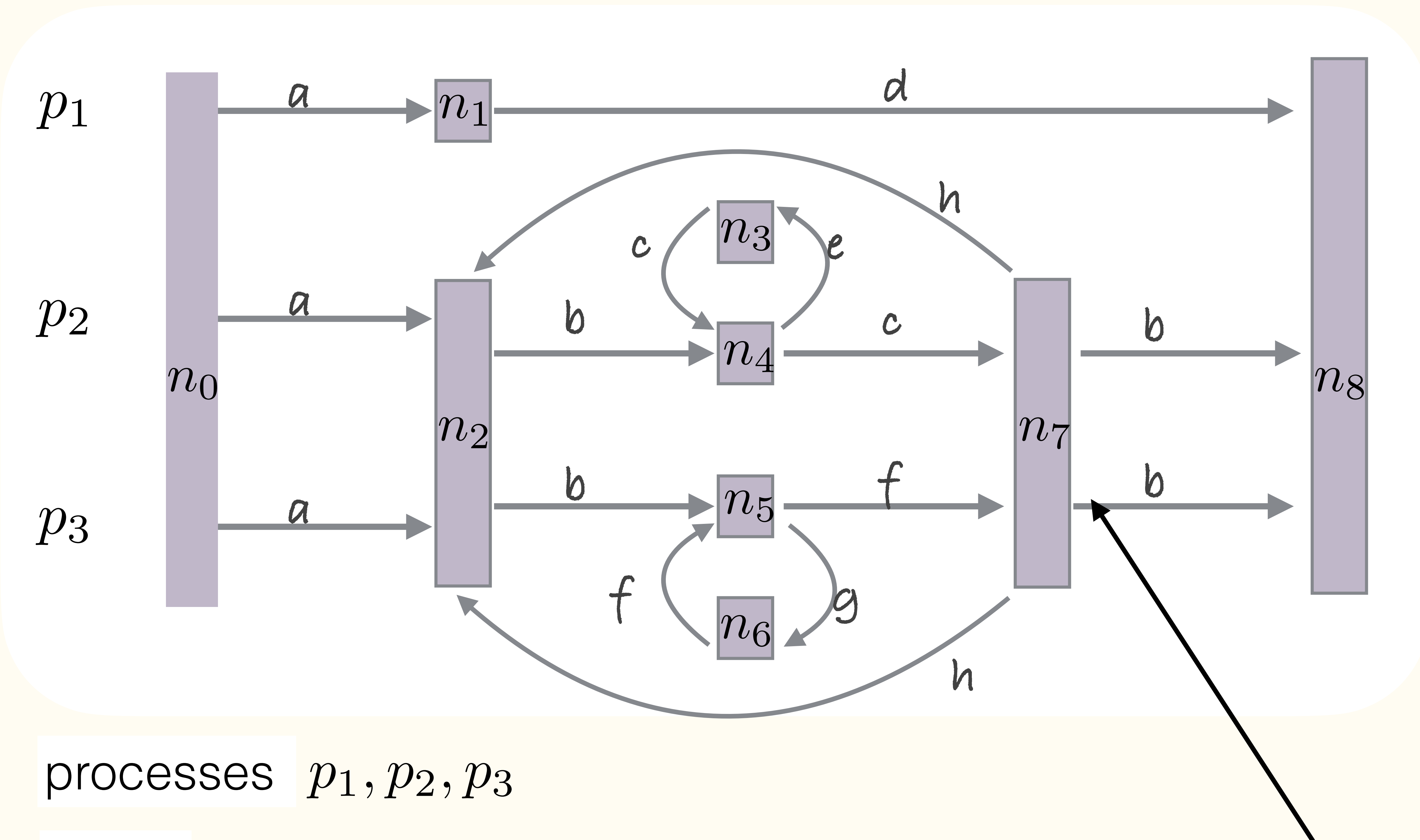
Algebraical/Categorical frameworks:

[Heerdt, Sammartino, Silva 2017]
[Urbat and Lutz Schröder. 2020]
[Colcombet, Petrisan, Stabile. 2021]

Case studies:

[Vaandrager. Model learning. Commun. ACM 2017]
[Neider, Smetsers, Vaandrager, Kuppens LNCS11200, 2019]

Negotiations [Desel & Esparza'13]



processes p_1, p_2, p_3

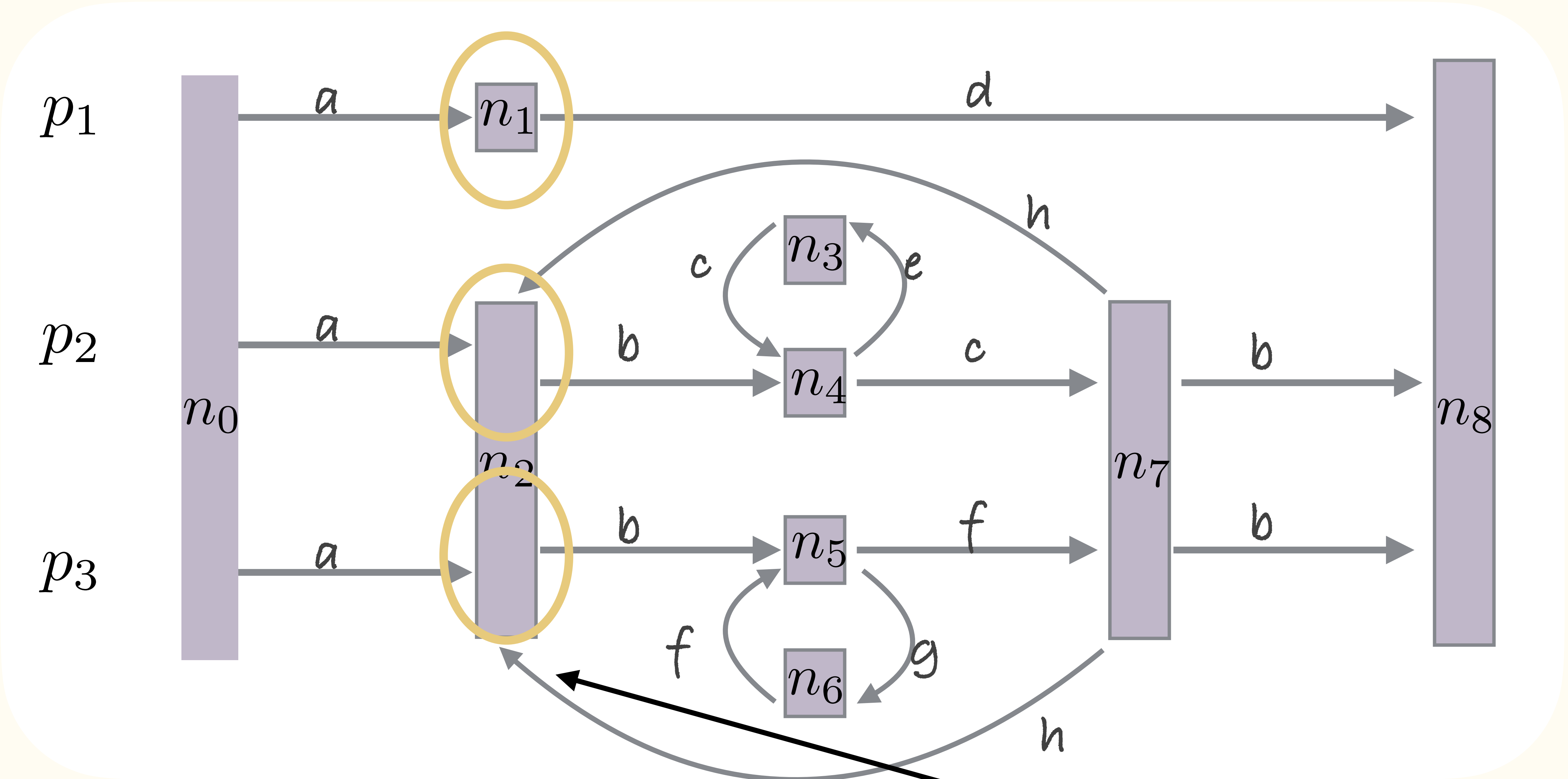
nodes $n_0, n_1, \ldots$

actions a, b, c,...

domain

$\{p_2, p_3\}$

Negotiations [Desel & Esparza'13]



processes p_1, p_2, p_3

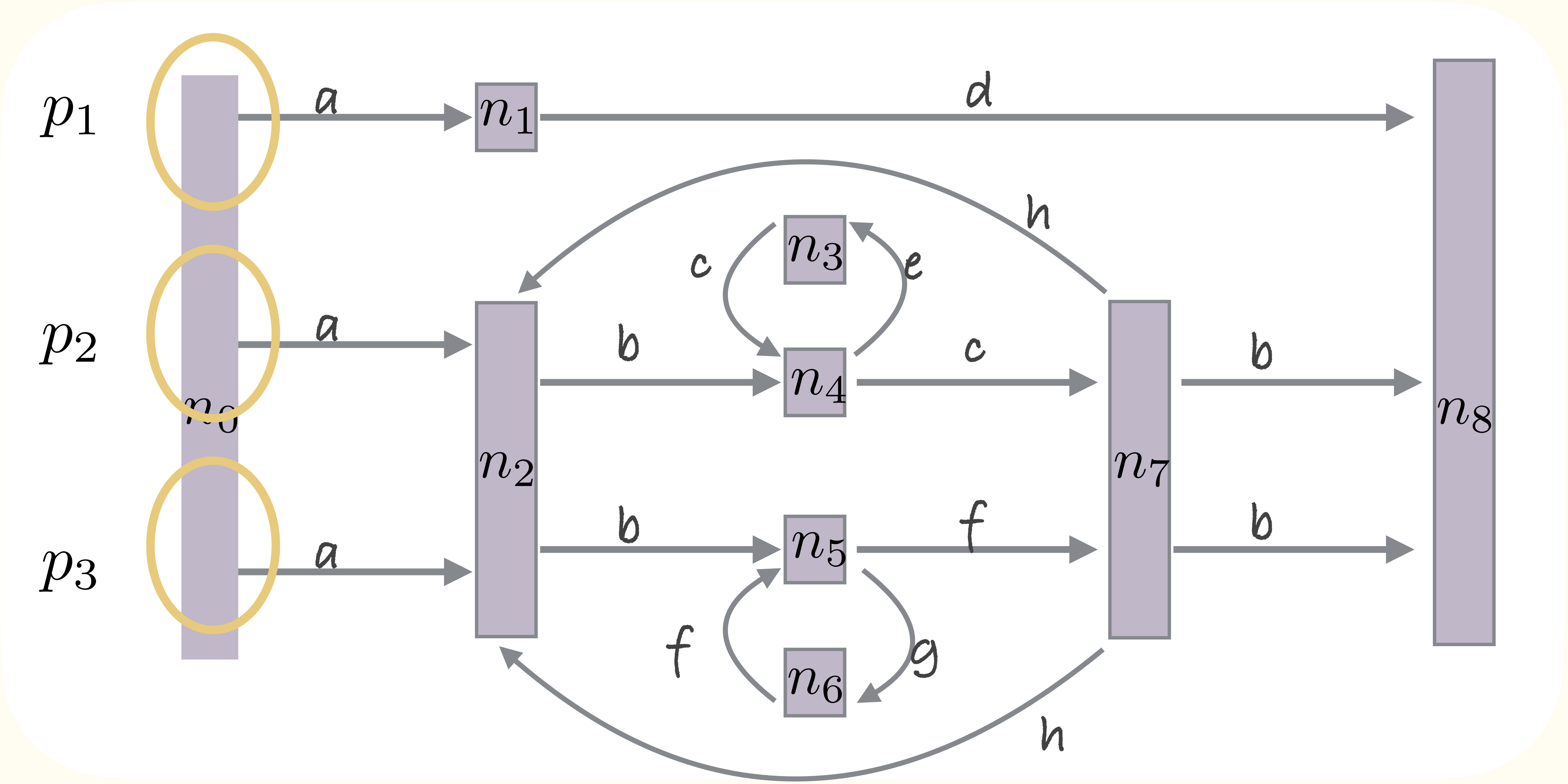
nodes $n_0, n_1, \ldots$

actions a, b, c,...

configuration

$\{n_1, n_2\}$

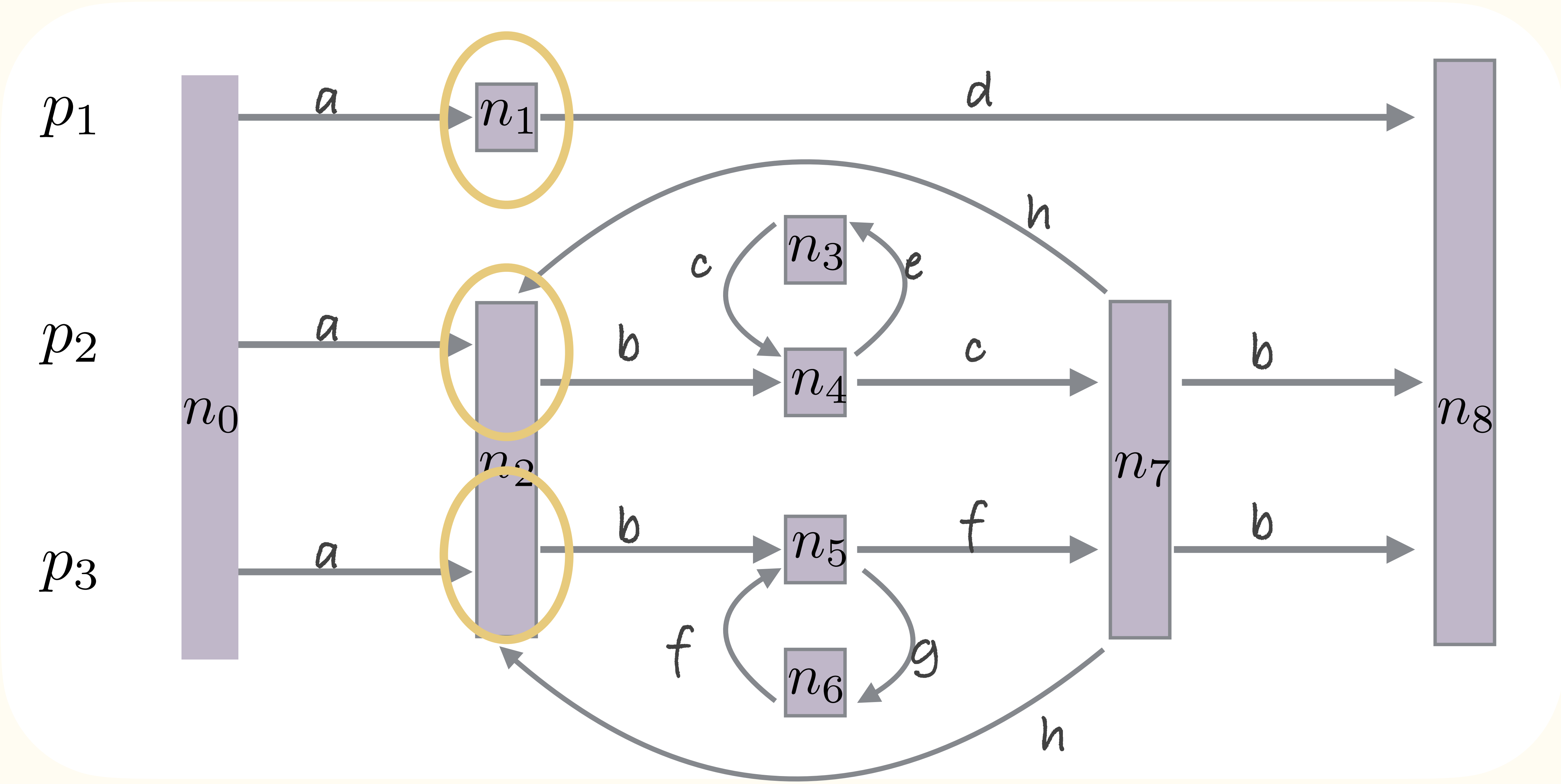
Negotiations [Desel & Esparza'13]



A run is a sequence of configurations

$$\{n_0\} \xrightarrow{a} \{n_1, n_2\} \xrightarrow{b} \{n_1, n_4, n_5\} \xrightarrow{c} \{n_1, n_7, n_5\} \xrightarrow{g} \{n_1, n_7, n_6\}$$

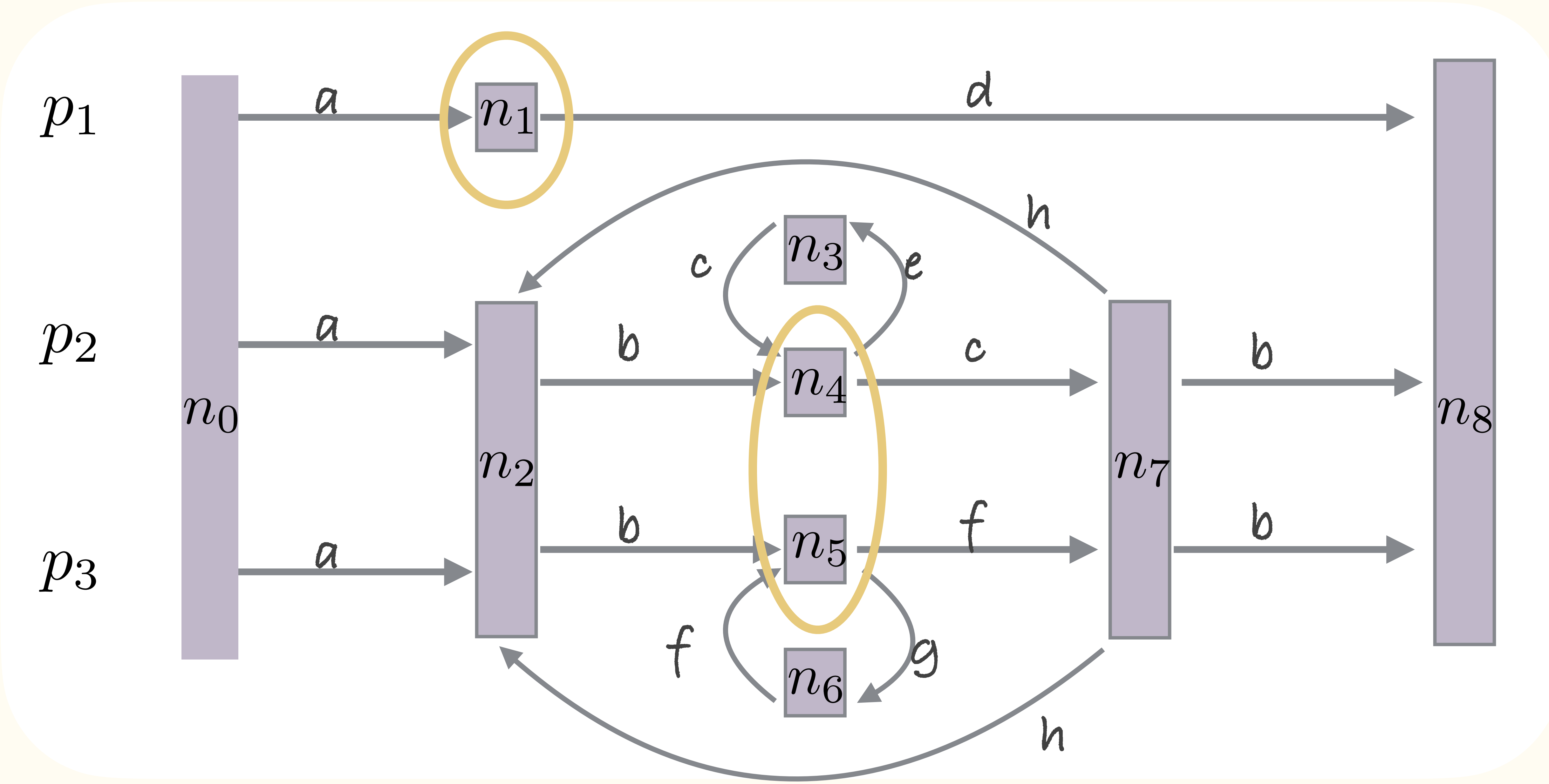
Negotiations [Desel & Esparza'13]



A run is a sequence of configurations

$$\{n_0\} \xrightarrow{a} \{n_1, n_2\} \xrightarrow{b} \{n_1, n_4, n_5\} \xrightarrow{c} \{n_1, n_7, n_5\} \xrightarrow{g} \{n_1, n_7, n_6\}$$

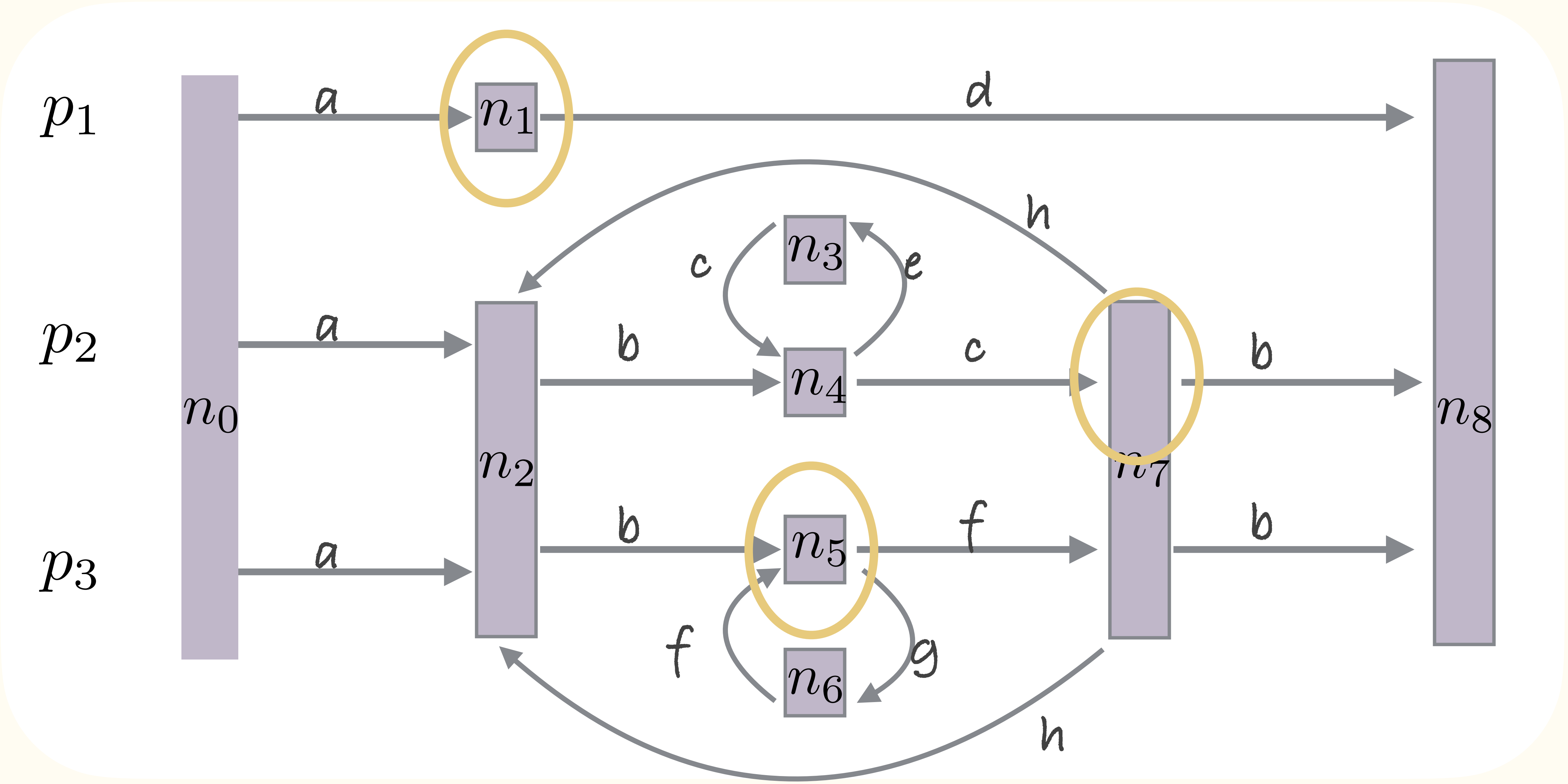
Negotiations [Desel & Esparza'13]



A run is a sequence of configurations

$$\{n_0\} \xrightarrow{a} \{n_1, n_2\} \xrightarrow{b} \{n_1, n_4, n_5\} \xrightarrow{c} \{n_1, n_7, n_5\} \xrightarrow{g} \{n_1, n_7, n_6\}$$

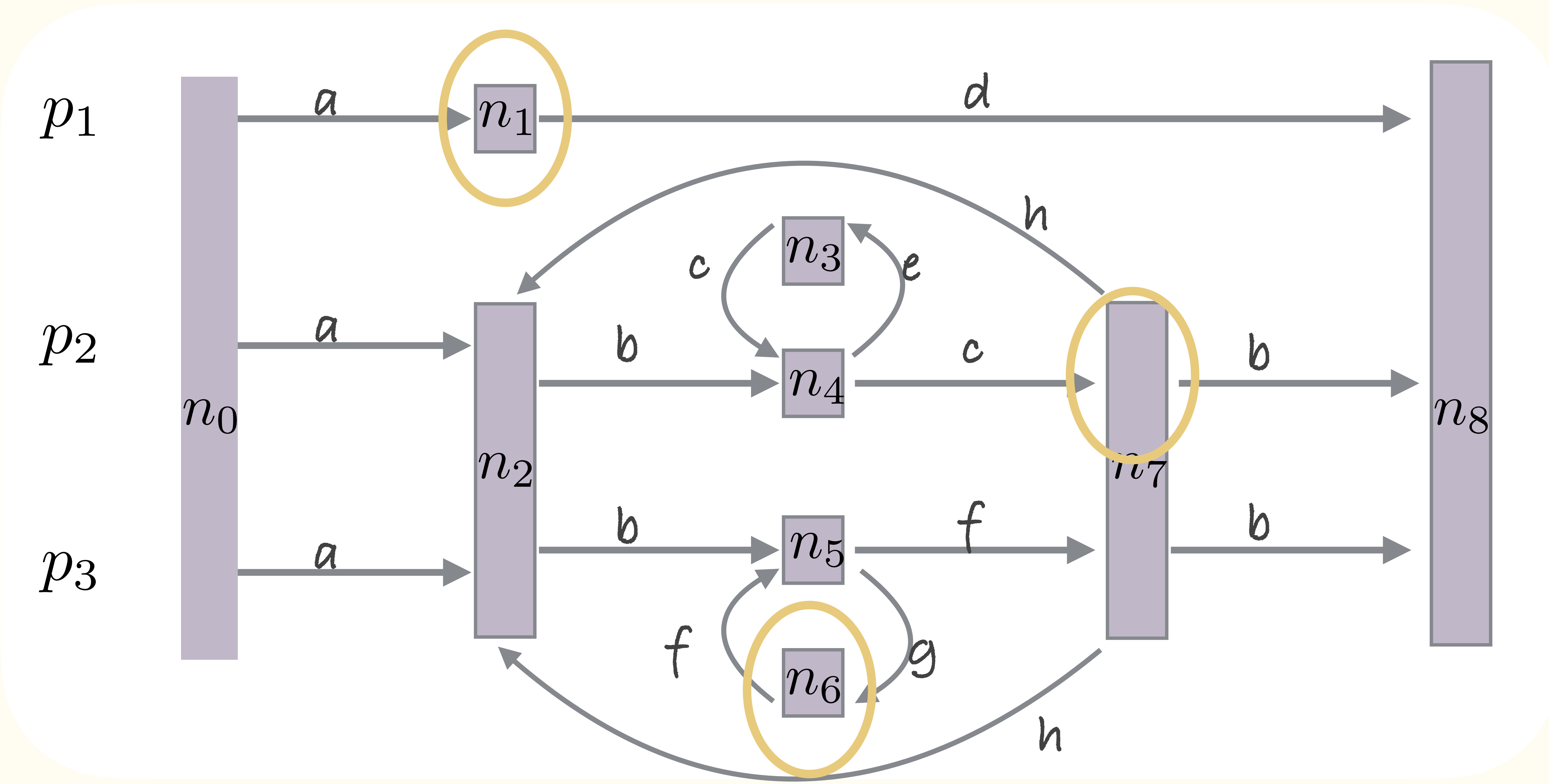
Negotiations [Desel & Esparza'13]



A run is a sequence of configurations

$$\{n_0\} \xrightarrow{a} \{n_1, n_2\} \xrightarrow{b} \{n_1, n_4, n_5\} \xrightarrow{c} \{n_1, n_7, n_5\} \xrightarrow{g} \{n_1, n_7, n_6\}$$

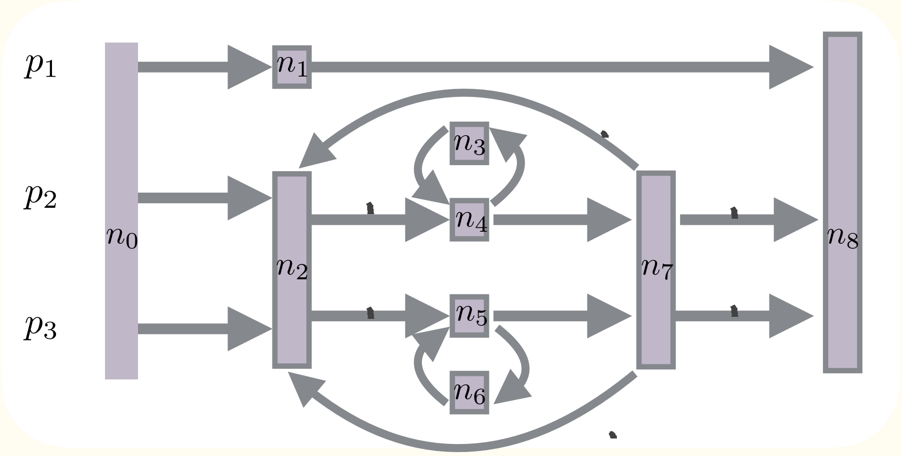
Negotiations [Desel & Esparza'13]



A run is a sequence of configurations

$$\{n_0\} \xrightarrow{a} \{n_1, n_2\} \xrightarrow{b} \{n_1, n_4, n_5\} \xrightarrow{c} \{n_1, n_7, n_5\} \xrightarrow{g} \{n_1, n_7, n_6\}$$

Sound deterministic negotiations



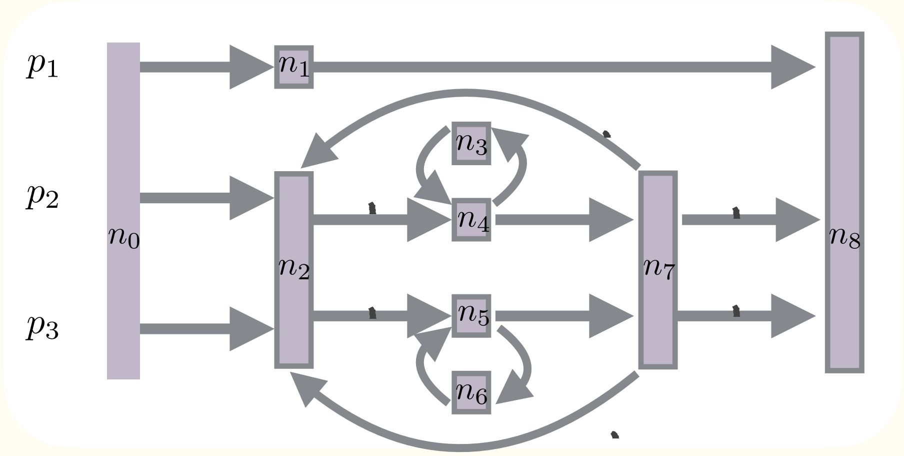
A negotiation is deterministic if its transition relation is a function

$$\delta : N \times \Sigma \times Proc \to N$$

A negotiation is sound if there is a final node n_{fin} such that

every partial run $\{n_{init}\} \xrightarrow{u} C$ can be completed $C \xrightarrow{v} \{n_{\text{fin}}\}$.

Sound deterministic negotiations

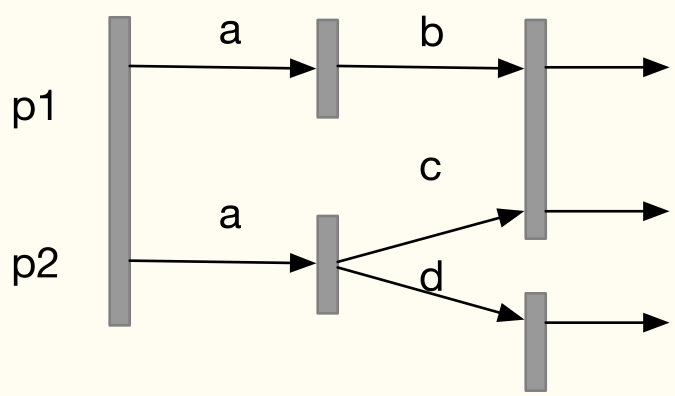


A negotiation is deterministic if its transition relation is a function

$$\delta : N \times \Sigma \times Proc \to N$$

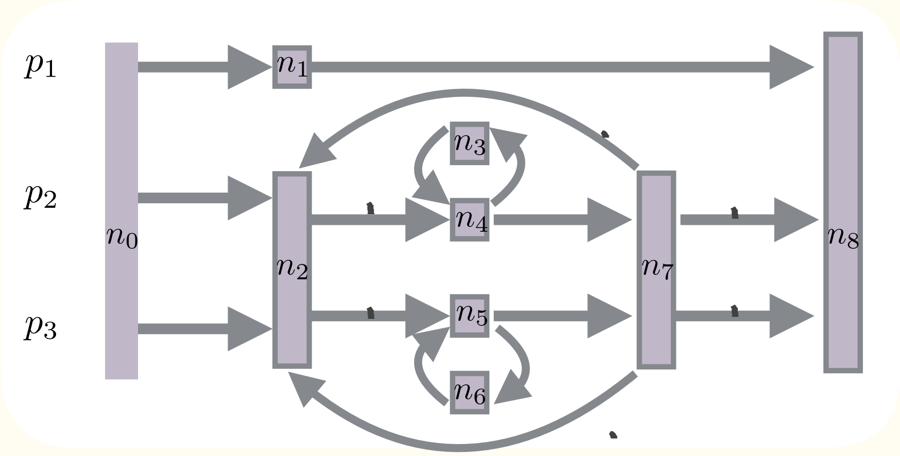
A negotiation is sound if there is a final node n_{fin} such that

every partial run $\{n_{init}\} \xrightarrow{u} C$ can be completed $C \xrightarrow{v} \{n_{\mathrm{fin}}\}$.



← **Not sound**

Sound deterministic negotiations



A negotiation is deterministic if its transition relation is a function

$$\delta : N \times \Sigma \times Proc \to N$$

A negotiation is sound if there is a final node n_{fin} such that

every partial run $\{n_{init}\} \xrightarrow{u} C$ can be completed $C \xrightarrow{v} \{n_{\text{fin}}\}$.

Thm[Desel & Esparza'15]

Sound deterministic negotiations $\equiv$ sound, free-choice Petri nets with initial and final markings.

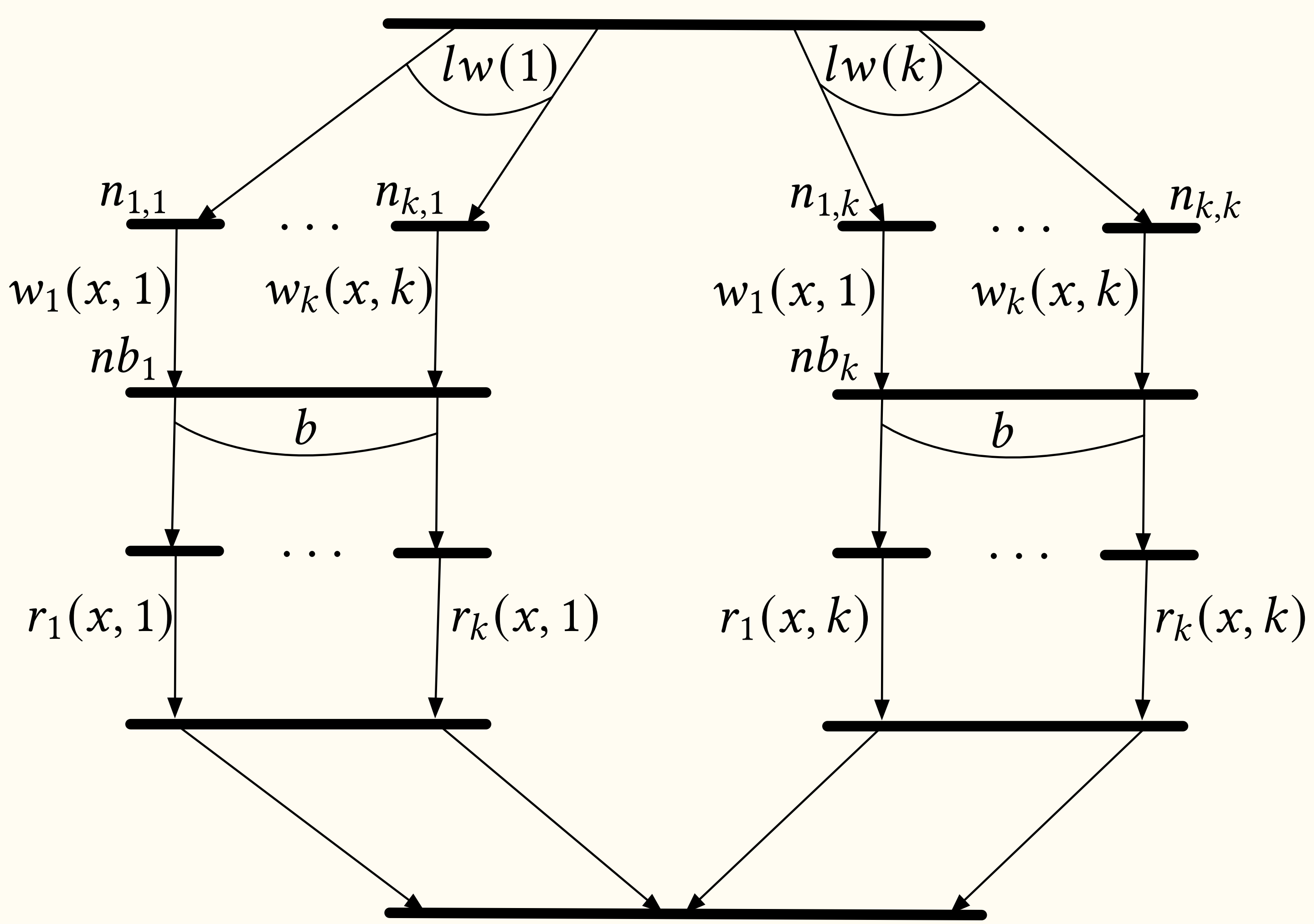
Sound deterministic negotiations

A negotiation is deterministic if its transition relation is a function

$$\delta : N \times \Sigma \times Proc \to N$$

A negotiation is sound if there is a final node n_{fin} such that

every partial run $\{n_{init}\} \xrightarrow{u} C$ can be completed $C \xrightarrow{v} \{n_{\text{fin}}\}$.



Sound deterministic negotiations

A negotiation is deterministic if its transition relation is a function

$$\delta : N \times \Sigma \times Proc \to N$$

A negotiation is sound if there is a final node n_{fin} such that

every partial run $\{n_{init}\} \xrightarrow{u} C$ can be completed $C \xrightarrow{v} \{n_{\text{fin}}\}$.

Thm[Desel & Esparza & Hoffmann'17]

Checking soundness of a deterministic negotiation can be done in PTIME.

Thm[Esparza, Kuperberg, Muscholl, W.'18]

Checking soundness of a deterministic negotiation is NLogSpace-complete.

Soundness is characterized by 3 forbidden patterns.

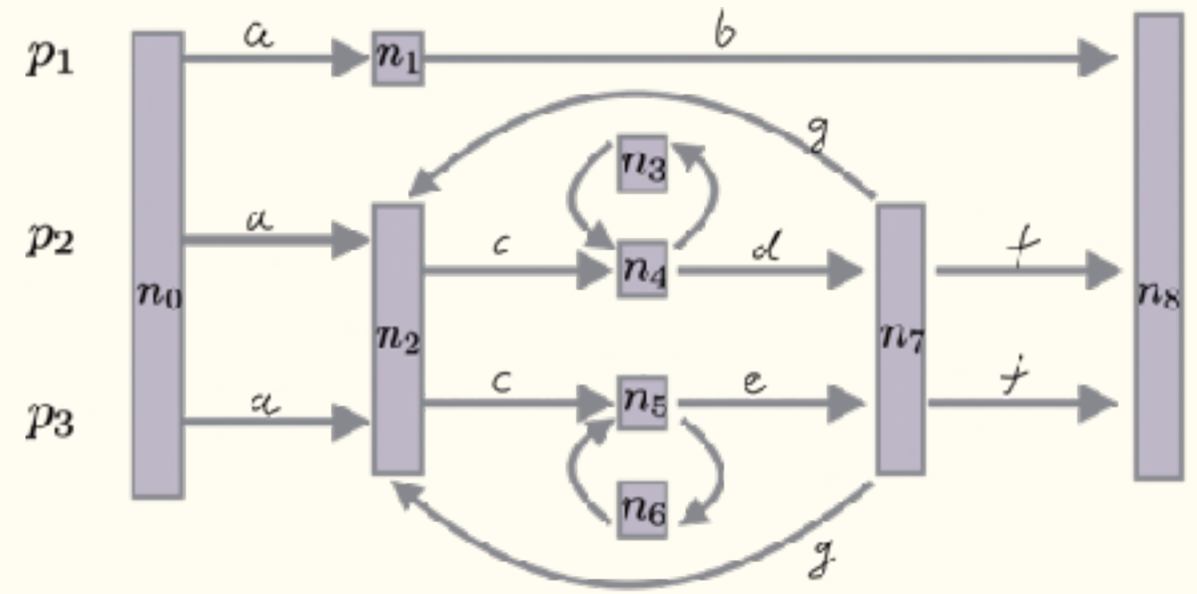
Sound deterministic negotiations are:

- A syntactic restriction of Peri nets.
- A non-trivial extension of finite automata.

- There is an inductive definition of this class.
- Several verification problems are easy for this class.

Sound deterministic negotiations vs. finite automata

Sound deterministic negotiations vs. finite automata



Paths($\mathcal{N}$) $\subseteq (\Sigma \times Proc)^*$ local paths in $\mathcal{N}$.

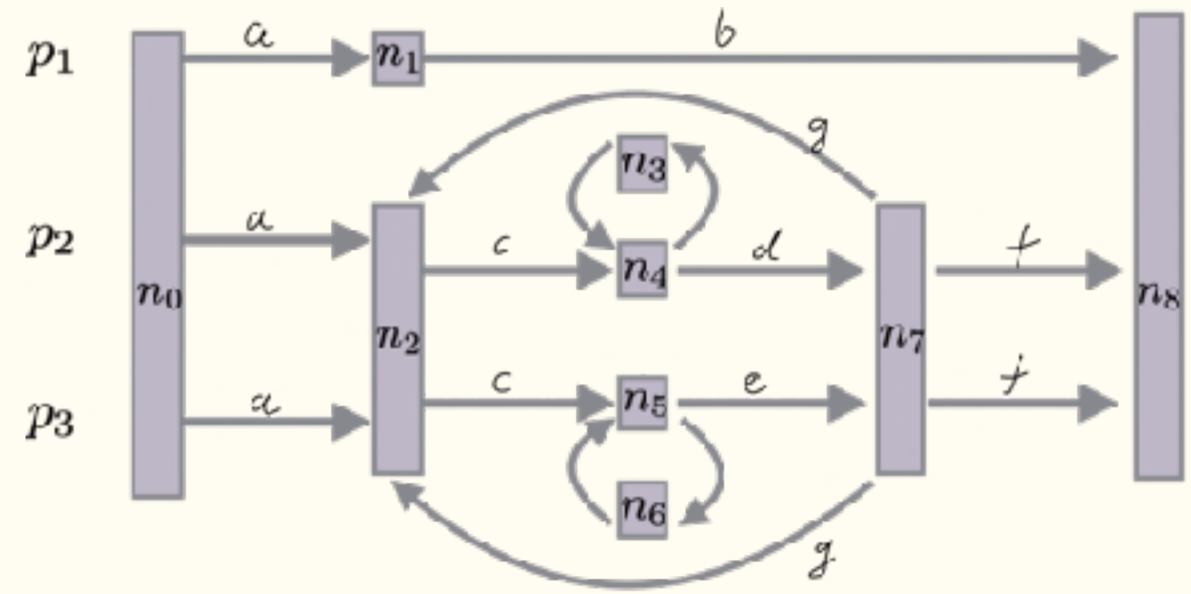
Paths($\mathcal{N}$) is a regular language.

$(a, 1)(b, 1)$

$(a, 2)(c, 3)(e, 3)(f, 2)$

Sound deterministic negotiations vs finite automata

$Paths(\mathcal{N}) \subseteq (\Sigma \times Proc)^*$ local paths in $\mathcal{N}$.



Consider $\mathcal{A}_{\mathcal{N}}$, the minimal deterministic automaton for $Paths(\mathcal{N})$.

We define $\overline{\mathcal{N}}$ from $\mathcal{A}_{\mathcal{N}}$.

$\mathcal{A}_{\mathcal{N}} = \langle S, \Sigma \times Proc, s^0, \delta_{\mathcal{A}} : S \times \Sigma \to S \rangle$:

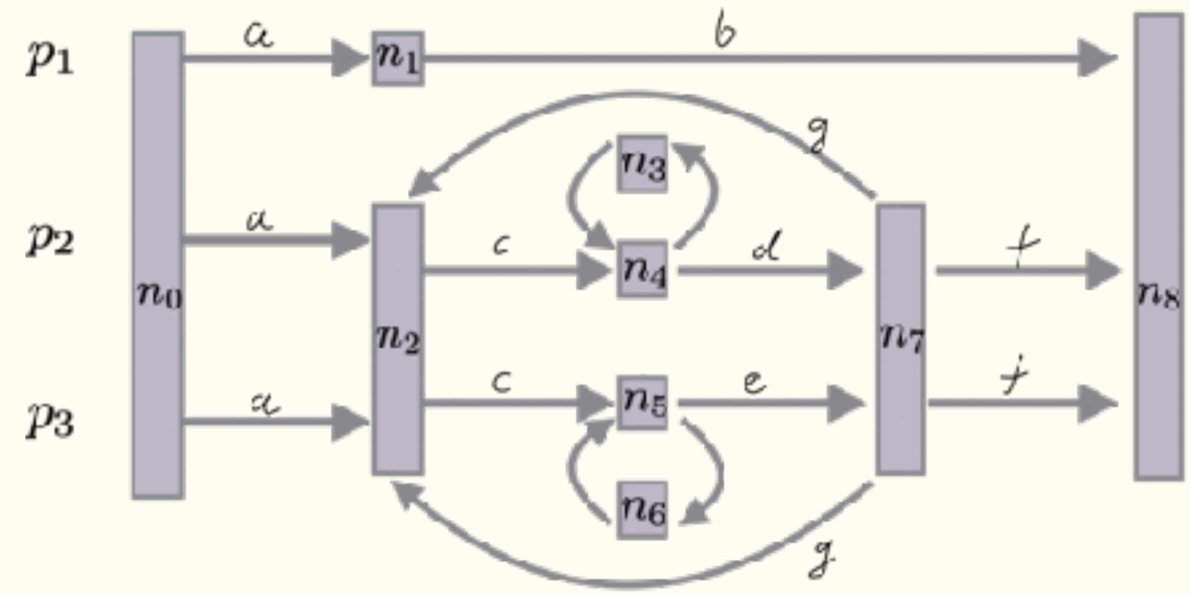
Nodes $N = S - \{\bot\}$, initial node s^0.

$\delta(s, a, p) = n'$ if $\delta_{\mathcal{A}}(s, (a, p)) = n'$.

$dom(s) = \{p : \exists a \in \Sigma.\ \delta(s, a, p) \neq \bot\}$.

Sound deterministic negotiations vs finite automata

$Paths(\mathcal{N}) \subseteq (\Sigma \times Proc)^*$ local paths in $\mathcal{N}$.



Consider $\mathcal{A}_{\mathcal{N}}$, the minimal deterministic automaton for $Paths(\mathcal{N})$.

We define $\overline{\mathcal{N}}$ from $\mathcal{A}_{\mathcal{N}}$.

$\mathcal{A}_{\mathcal{N}} = \langle S, \Sigma \times Proc, s^0, \delta_{\mathcal{A}} : S \times \Sigma \to S \rangle$:

Nodes $N = S - \{\perp\}$, initial node s^0.

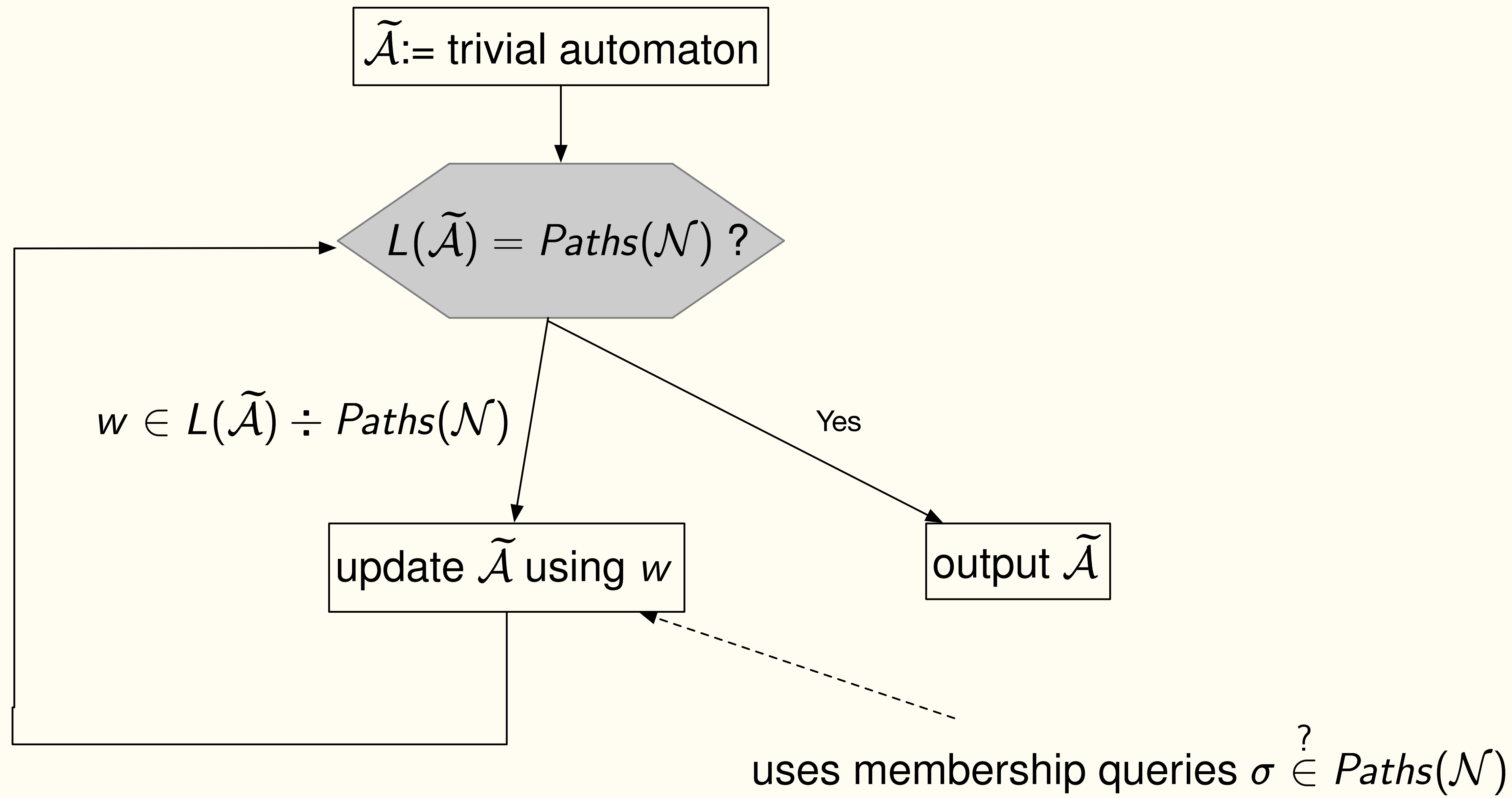
$\delta(s, a, p) = n'$ if $\delta_{\mathcal{A}}(s, (a, p)) = n'$.

$dom(s) = \{p : \exists a \in \Sigma.\ \delta(s, a, p) \neq \perp\}$.

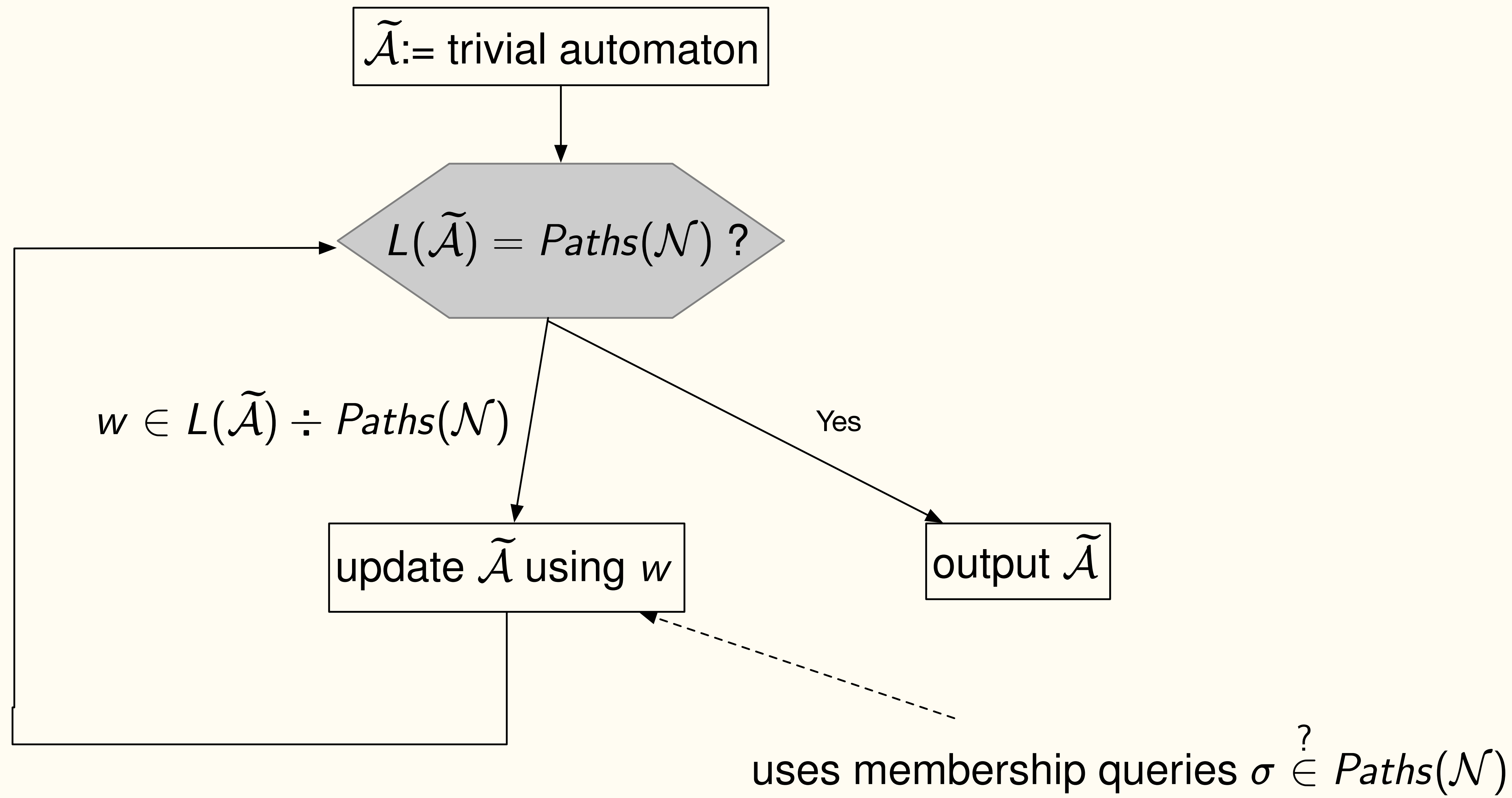
Fact: $\overline{\mathcal{N}}$ is a negotiation and there is a homomorphism $h : \mathcal{N} \to \overline{\mathcal{N}}$.

So we can just learn $Paths(\mathcal{N})$ and then construct $\overline{\mathcal{N}}$.

Using finite automat learning directly



Using finite automat learning directly



1. Automaton $\widetilde{\mathcal{A}}$ may not resemble a negotiation.
2. Answering $\sigma \overset{?}{\in} Paths(\mathcal{N})$ requires to know internals of $\mathcal{N}$

Learning sound deterministic negotiations

We fix a set of processes, $Proc$, and a distributed alphabet $(\Sigma, dom : \Sigma \to Proc)$.

Teacher knows the language L of a sound deterministic negotiation.

1

We want to construct the minimal negotiation of L using two kinds of queries:

membership queries: $\sigma \stackrel{?}{\in} Paths(\mathcal{N})$

equivalence queries: $L(\widetilde{N}) \stackrel{?}{=} L$

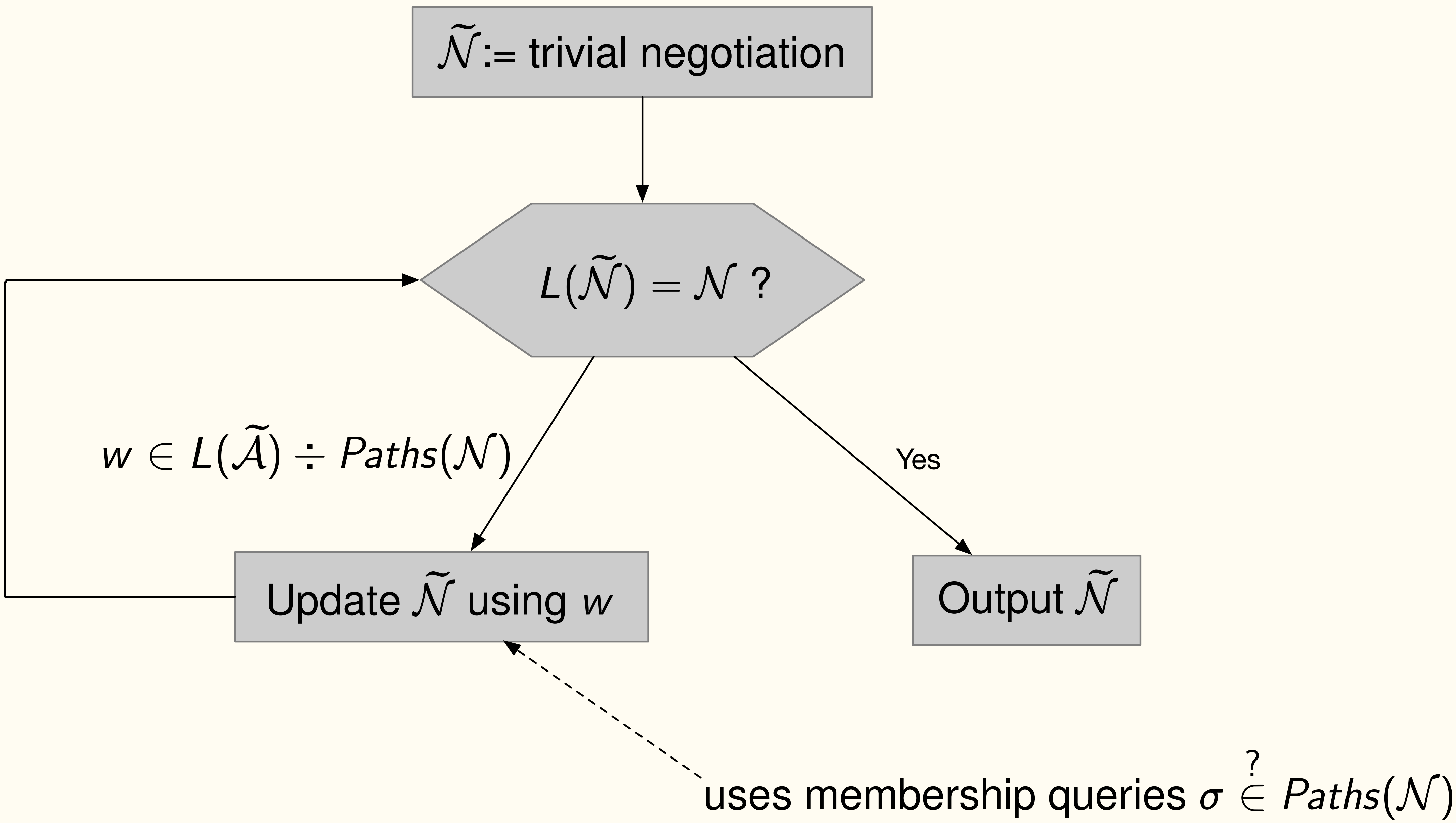
2

We want to construct the minimal negotiation of L using two kinds of queries:

membership queries: $u \stackrel{?}{\in} L(\mathcal{N})$

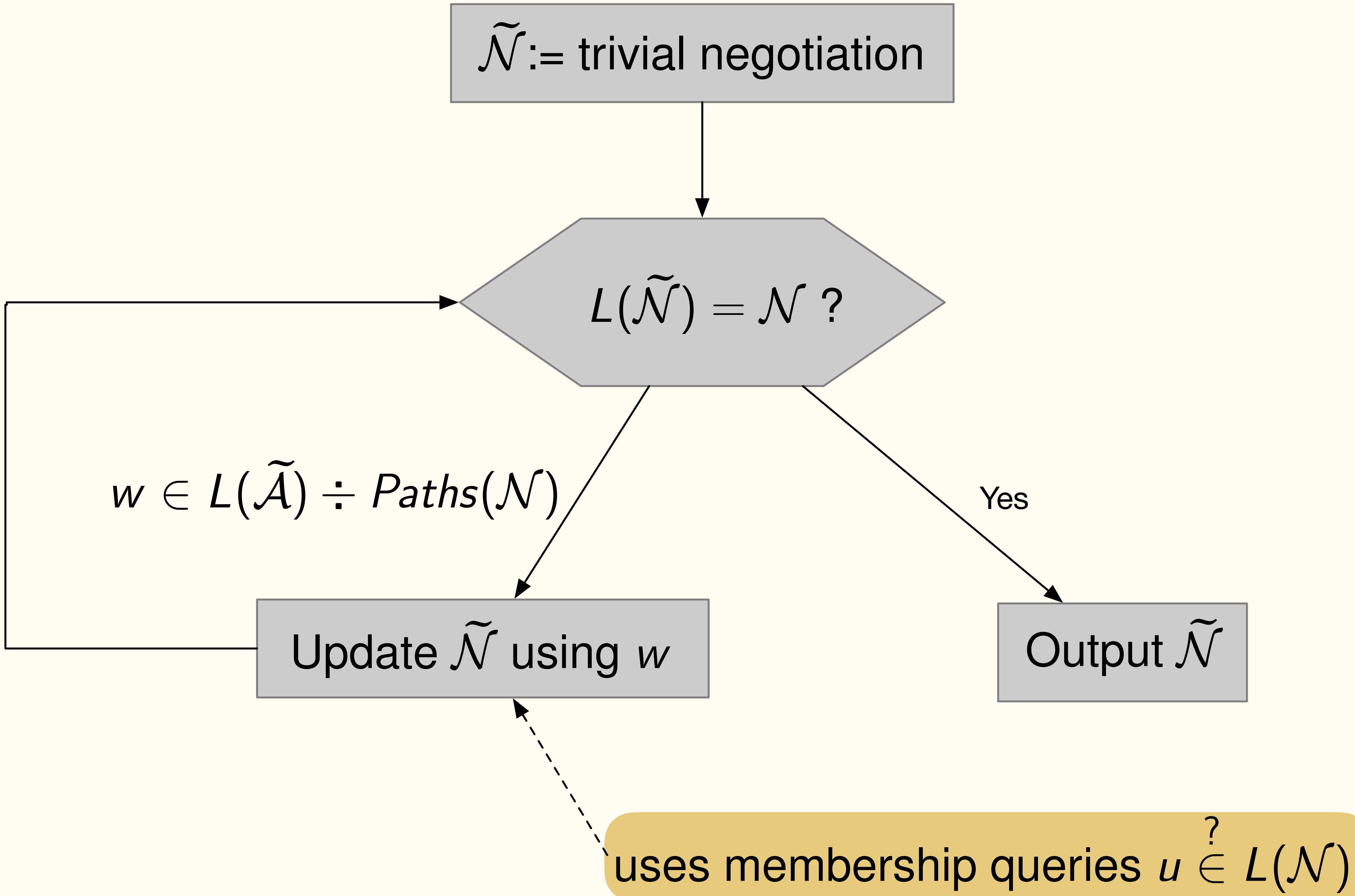
equivalence queries: $L(\widetilde{N}) \stackrel{?}{=} L$

Local paths in membership queries



1. ~~Automaton $\widetilde{\mathcal{A}}$ may not resemble a negotiation.~~
2. Answering $\sigma \overset{?}{\in} Paths(\mathcal{N})$ requires to know internals of $\mathcal{N}$

Executions in membership queries



1. ~~Automaton $\widetilde{\mathcal{A}}$ may not resemble a negotiation.~~
2. ~~Answering $\sigma \overset{?}{\in} Paths(\mathcal{N})$ requires to know internals of $\mathcal{N}$~~

Learning sound deterministic negotiations

We fix a set of processes, $Proc$, and a distributed alphabet $(\Sigma, dom : \Sigma \to Proc)$.

Teacher knows the language L of a sound deterministic negotiation.

1

We want to construct the minimal negotiation of L using two kinds of queries:

membership queries: $\sigma \overset{?}{\in} Paths(\mathcal{N})$

equivalence queries: $L(\widetilde{N}) \overset{?}{=} L$

THM: s(s+|Proc|+log(m)) membership queries, s equivalence queries

2

We want to construct the minimal negotiation of L using two kinds of queries:

membership queries: $u \overset{?}{\in} L(\mathcal{N})$

equivalence queries: $L(\widetilde{N}) \overset{?}{=} L$

THM: s(s+log(m)) membership queries, s equivalence queries

Summary

- Sound deterministic negotiations are a syntactic subclass of Petri nets (as well as Zielonka automata).
- They have a lot of structure:
  finite automaton for the path language (decomposition results)
- Thanks to this structure some analysis problems are PTIME.
- It is also possible to minimize them and get an active learning algorithm.

Further work

- Black box learning.
  [Leemans, Fahland, Aalst: Scalable process discovery and conformance checking, 2016]
  [Ehrenfeucht, Rozenberg: Region theory for Petri Nets, 1990]
- Approximating Zielonka automata by sound deterministic negotiations.

Summary

- Sound deterministic negotiations are a syntactic subclass of Petri nets (as well as Zielonka automata).
- They have a lot of structure:
  finite automaton for the path language (decomposition results)
- Thanks to this structure some analysis problems are PTIME.
- It is also possible to minimize them and get an active learning algorithm.

Further work

- Black box learning.
  [Leemans, Fahland, Aalst: Scalable process discovery and conformance checking, 2016]
  [Ehrenfeucht, Rozenberg: Region theory for Petri Nets, 1990]
- Approximating Zielonka automata by sound deterministic negotiations.

Thank you!