languages of infinite traces and deterministic asynchronous automata

Namit Chaturvedi

RWTH AACHEN UNIVERSITY

November 06, 2013

traces and trace languages

Partially ordered models of executions of distributed systems.

- Concurrent execution of independent actions
- Equivalence class of partially commutative sequences

traces and trace languages

Partially ordered models of executions of distributed systems.

- Concurrent execution of independent actions
- Equivalence class of partially commutative sequences
- A language of traces is recognizable if and only if the corresponding "trace closed" word language is recognizable

traces and trace languages

Partially ordered models of executions of distributed systems.

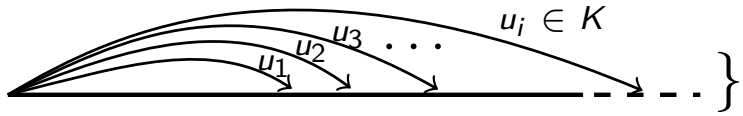
- Concurrent execution of independent actions
- Equivalence class of partially commutative sequences
- A language of traces is recognizable if and only if the corresponding "trace closed" word language is recognizable

Automata recognizing languages of traces

- Finite asynchronous automata (FAA) for languages of finite traces
- Deterministic asynchronous Büchi automata (DABA) and deterministic asynchronous Muller automata (DAMA) for languages of infinite traces

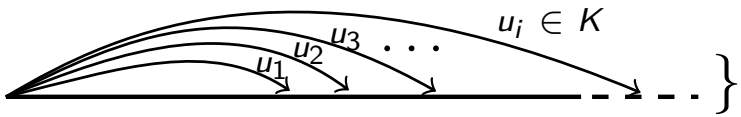
regular to ω-regular (word) languages

regular to ω-regular (word) languages

- $\lim(K) := \{\alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha =$ 

$\}$

regular to ω-regular (word) languages

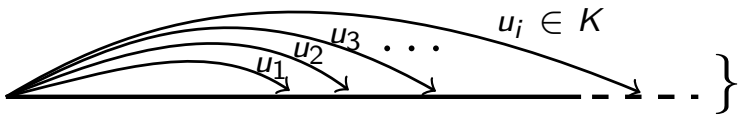
- $\lim(K) := \{\alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha =$ 

$\}$

Theorem (J.R. Büchi)

If a DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes the language $K \subseteq \Sigma^*$ then the DBA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes $\lim(K)$.

regular to ω-regular (word) languages

- $\lim(K) := \{\alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha =$ 

$\}$

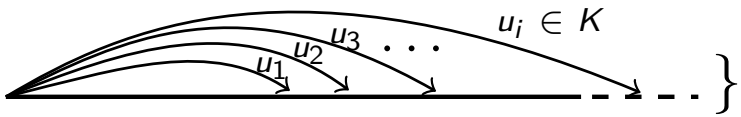
Theorem (J.R. Büchi)

If a DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes the language $K \subseteq \Sigma^*$ then the DBA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes $\lim(K)$.

Theorem (R. McNaughton)

For any DMA $\mathfrak{A}$ recognizing a language L, there exists a DBA recognizing L iff the acceptance component of $\mathfrak{A}$ is closed under (realizable) supersets.

regular to ω-regular (word) languages

- $\lim(K) := \{\alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha =$ 

$\}$

Theorem (J.R. Büchi)

If a DFA $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes the language $K \subseteq \Sigma^*$ then the DBA $\mathfrak{A} = (Q, \Sigma, q_0, \delta, F)$ recognizes $\lim(K)$.

Theorem (R. McNaughton)

For any DMA $\mathfrak{A}$ recognizing a language L, there exists a DBA recognizing L iff the acceptance component of $\mathfrak{A}$ is closed under (realizable) supersets.

Theorem (L. Landweber)

For any language $L \subseteq \Sigma^\omega$, L is recognized by a DMA (L is ω-regular) iff L can be expressed as a finite Boolean combination of DBA recognizable languages.

fundamental questions

- FAA and DABA vs. Büchi's theorem
- DABA and DAMA vs. McNaughton's theorem
- ω-regular trace languages and DABA recognizable languages vs. Landweber's theorem

preliminaries: finite and infinite traces

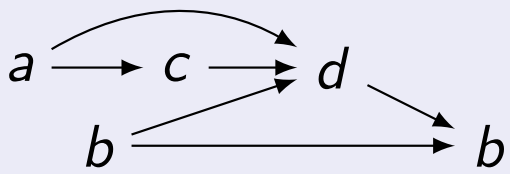
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$

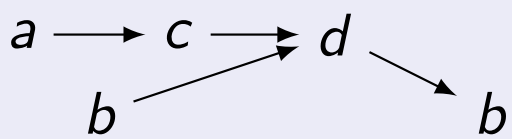


preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$

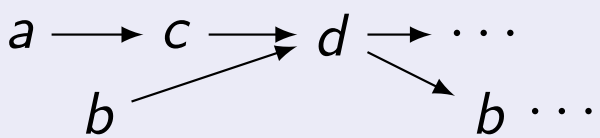


preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$

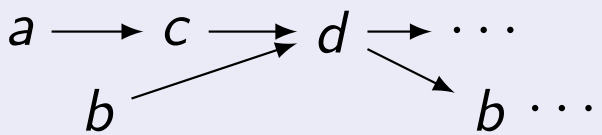


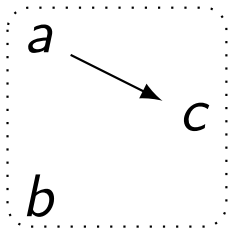
preliminaries: finite and infinite traces

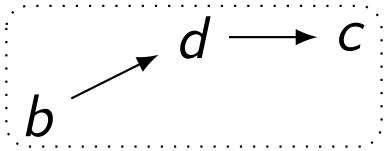
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$





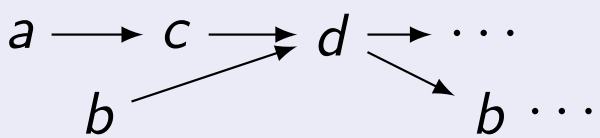


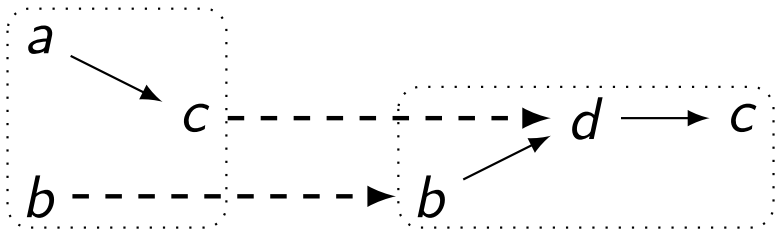
preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$





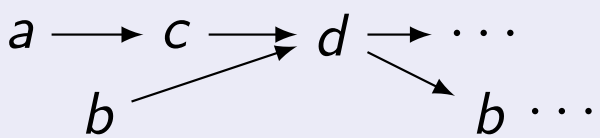
$t_1 \odot t_2$ similarly $t \odot \theta$,
and thus $t \sqsubseteq t \odot \theta$

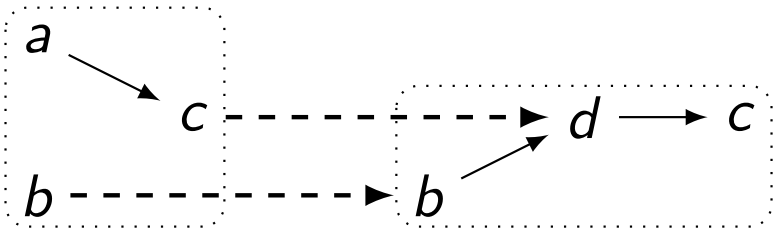
preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

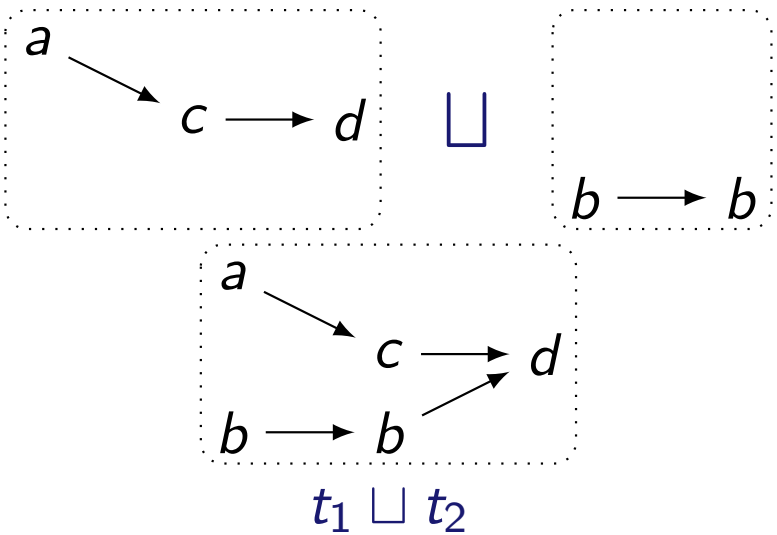
Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$





$t_1 \odot t_2$ similarly $t \odot \theta$,
and thus $t \sqsubseteq t \odot \theta$

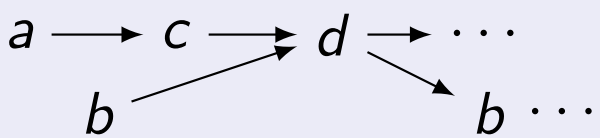


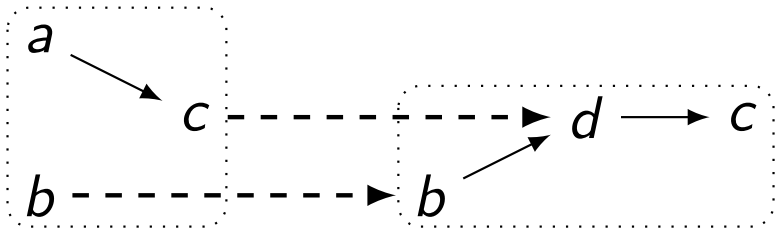
preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

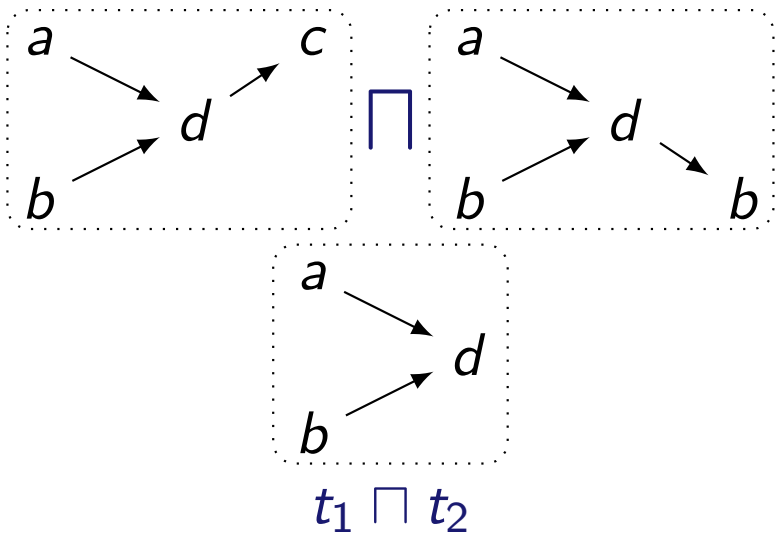
Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$





$t_1 \odot t_2$ similarly $t \odot \theta$,
and thus $t \sqsubseteq t \odot \theta$

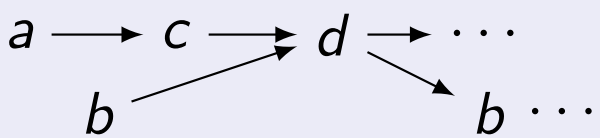


preliminaries: finite and infinite traces

Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **independence alphabet**.

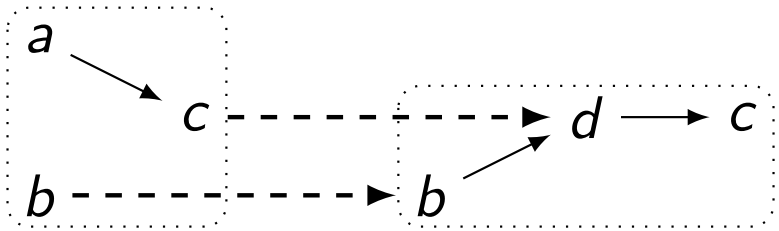
Definition (Finite and infinite traces)

A **trace** $t = [V, \lessdot, \lambda]$ or $\theta = [V, \lessdot, \lambda]$

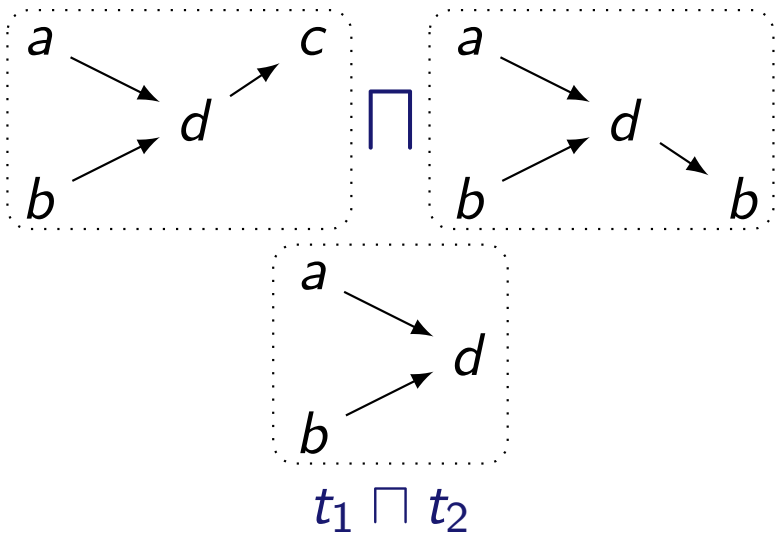


The set of all finite traces: $\mathbb{M}(\Sigma, I)$

The set of all infinite traces: $\mathbb{R}(\Sigma, I)$



$t_1 \odot t_2$ similarly $t \odot \theta$,
and thus $t \sqsubseteq t \odot \theta$



preliminaries: asynchronous transitions systems

Over (Σ, I), an asynchronous transition system (an ATS) comprises of

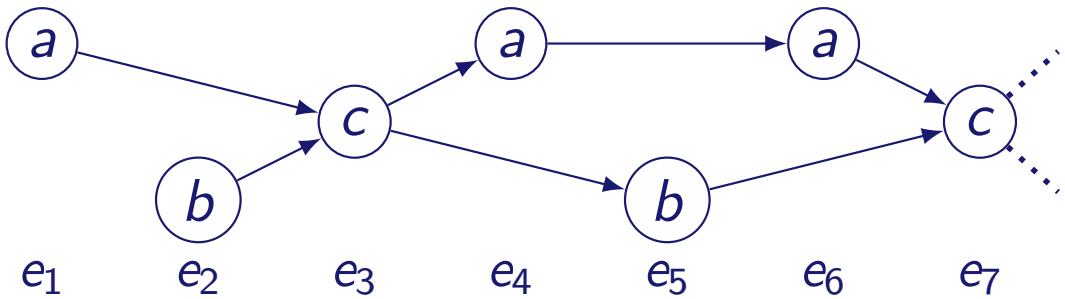
preliminaries: asynchronous transitions systems

Over (Σ, I), an asynchronous transition system (an ATS) comprises of

- a set $\mathcal{P}$ of processes
- a mapping $\mathrm{dom}\colon \Sigma \to 2^{\mathcal{P}}$, with $\bigcup_{a\in\Sigma} \mathrm{dom}(a) = \mathcal{P}$ and $a \; I \; b \Leftrightarrow \mathrm{dom}(a) \cap \mathrm{dom}(b) = \emptyset$
- sets $X_p, p \in \mathcal{P}$ of local p-states (also refer to partial P-states X_P and global states $X_{\mathcal{P}}$)
- transition functions $\delta_a\colon X_{\mathrm{dom}(a)} \to X_{\mathrm{dom}(a)}$ define how processes jointly perform state transitions letters $a \in \Sigma$

preliminaries: asynchronous transitions systems

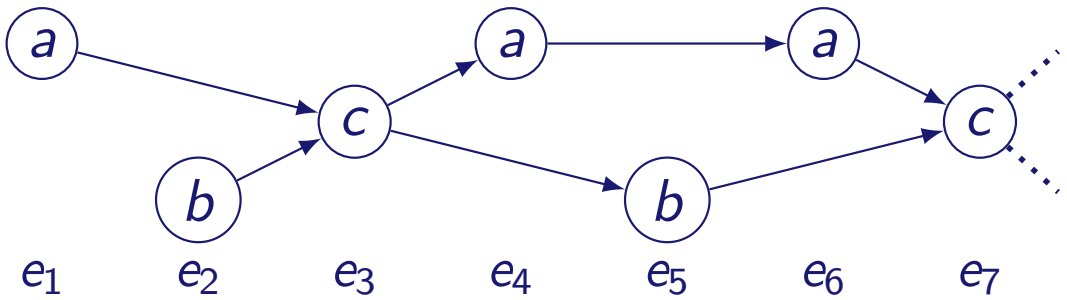
Over (Σ, I), an asynchronous transition system (an ATS) comprises of



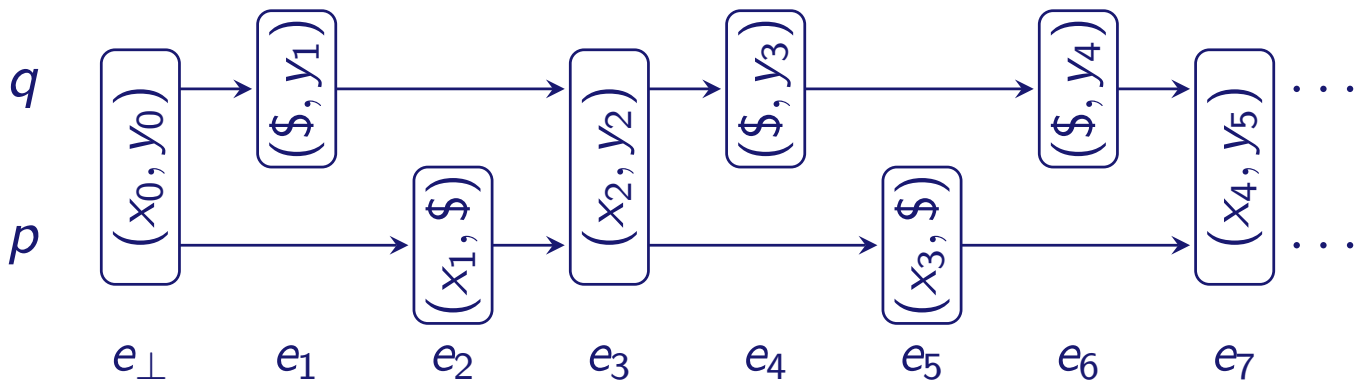
$t/\theta = [V, \lessdot, \lambda]$

preliminaries: asynchronous transitions systems

Over (Σ, I), an asynchronous transition system (an ATS) comprises of



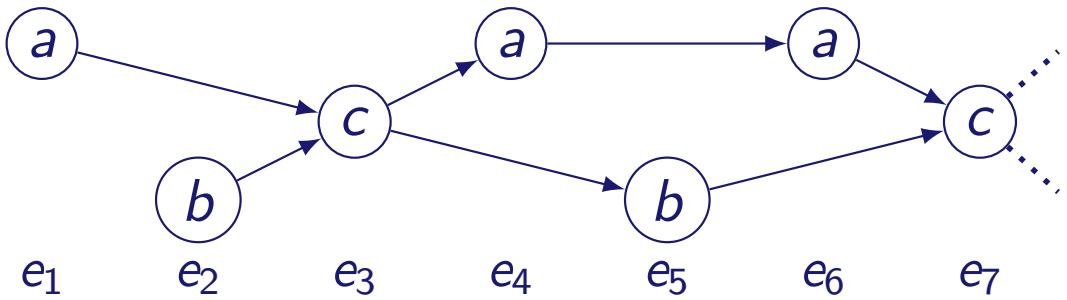
$t/\theta = [V, \lessdot, \lambda]$



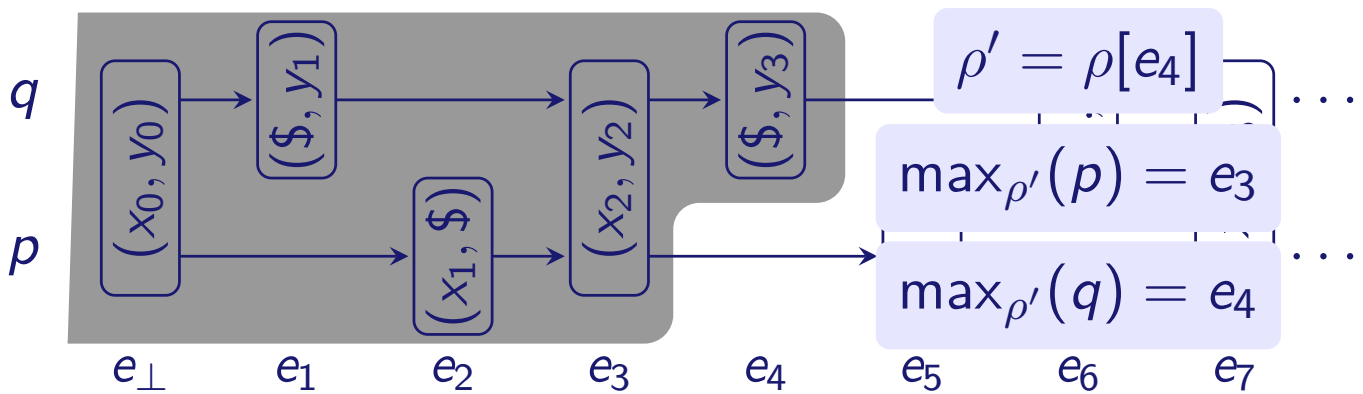
$\rho = [V', \lessdot', \lambda', \Lambda]$

preliminaries: asynchronous transitions systems

Over (Σ, I), an asynchronous transition system (an ATS) comprises of



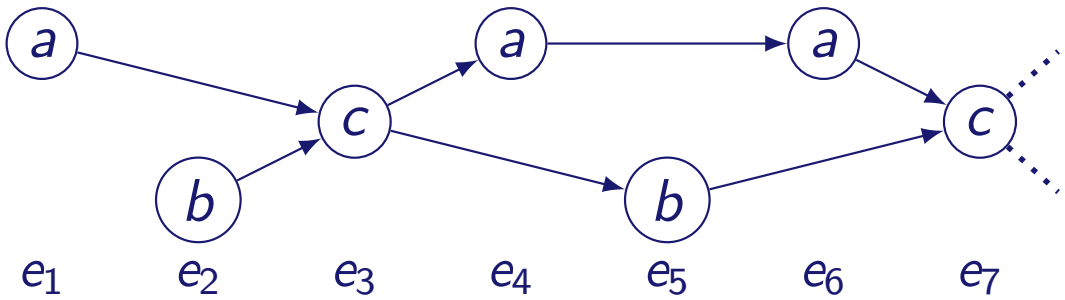
$t/\theta = [V, \lessdot, \lambda]$



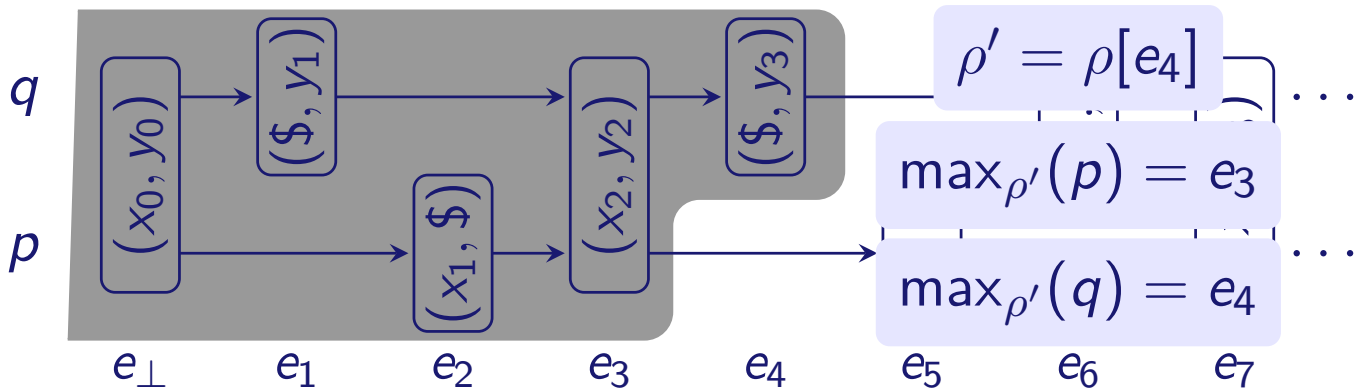
$\rho = [V', \lessdot', \lambda', \Lambda]$

preliminaries: asynchronous transitions systems

Over (Σ, I), an asynchronous transition system (an ATS) comprises of



$t/\theta = [V, \lessdot, \lambda]$



$\rho = [V', \lessdot', \lambda', \Lambda]$

Formally, an ATS is a tuple $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$

preliminaries: classical asynchronous automata

- Local infinity sets

$$\mathrm{Inf}_p(\rho) := \begin{cases} \left\{ x \in X_p \mid \exists^\infty e \in \rho : \Lambda(e)_{|p} = x \right\} & \text{if } p \in \mathrm{dom}(\mathrm{alphinf}(\rho)) \\ \left\{ \begin{array}{l|l} x \in X_p & \exists e \in \rho : e = \max_p(\rho) \\ & \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise} \end{cases}$$

preliminaries: classical asynchronous automata

- Local infinity sets

$$\mathrm{Inf}_p(\rho) := \begin{cases} \left\{ x \in X_p \mid \exists^\infty e \in \rho : \Lambda(e)_{|p} = x \right\} & \text{if } p \in \mathrm{dom}(\mathrm{alphinf}(\rho)) \\ \left\{ \begin{array}{l|l} x \in X_p & \exists e \in \rho : e = \max_p(\rho) \\ & \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise} \end{cases}$$

- Acceptance condition $\mathcal{F} = \{F_1, \dots\}$ with $F_i = (F_p^i)_{p \in \mathcal{P}}$

	p	q	$\cdots$
F_1	F_p^1	F_q^1	$\cdots$
F_2	F_p^2	F_q^2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\cdots$

preliminaries: classical asynchronous automata

- Local infinity sets

$$\mathrm{Inf}_p(\rho) := \begin{cases} \left\{ x \in X_p \mid \exists^\infty e \in \rho : \Lambda(e)_{|p} = x \right\} & \text{if } p \in \mathrm{dom}(\mathrm{alphinf}(\rho)) \\ \left\{ \begin{array}{l|l} x \in X_p & \exists e \in \rho : e = \max_p(\rho) \\ & \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise} \end{cases}$$

- Acceptance condition $\mathcal{F} = \{F_1, \dots\}$ with $F_i = (F_p^i)_{p \in \mathcal{P}}$
- For DABA $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, ρ is an accepting run if $\exists F_i \in \mathcal{F},\ \forall p \in \mathcal{P} : F_p^i \subseteq \mathrm{Inf}_p(\rho)$

	p	q	$\cdots$
F_1	F_p^1	F_q^1	$\cdots$
F_2	F_p^2	F_q^2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\cdots$

preliminaries: classical asynchronous automata

- Local infinity sets

$$\mathrm{Inf}_p(\rho) := \begin{cases} \left\{ x \in X_p \mid \exists^\infty e \in \rho : \Lambda(e)_{|p} = x \right\} & \text{if } p \in \mathrm{dom}(\mathrm{alphinf}(\rho)) \\ \left\{ \begin{array}{l|l} x \in X_p & \exists e \in \rho : e = \max_p(\rho) \\ & \text{and } \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise} \end{cases}$$

- Acceptance condition $\mathcal{F} = \{F_1, \dots\}$ with $F_i = (F_p^i)_{p \in \mathcal{P}}$
- For DABA $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, ρ is an accepting run if $\exists F_i \in \mathcal{F},\ \forall p \in \mathcal{P} : F_p^i \subseteq \mathrm{Inf}_p(\rho)$
- For DAMA $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$, ρ is an accepting run if $\exists F_i \in \mathcal{F},\ \forall p \in \mathcal{P} : F_p^i = \mathrm{Inf}_p(\rho)$

	p	q	$\cdots$
F_1	F_p^1	F_q^1	$\cdots$
F_2	F_p^2	F_q^2	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\cdots$

preliminaries: ω-regular trace languages

preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

preliminaries: ω-regular trace languages

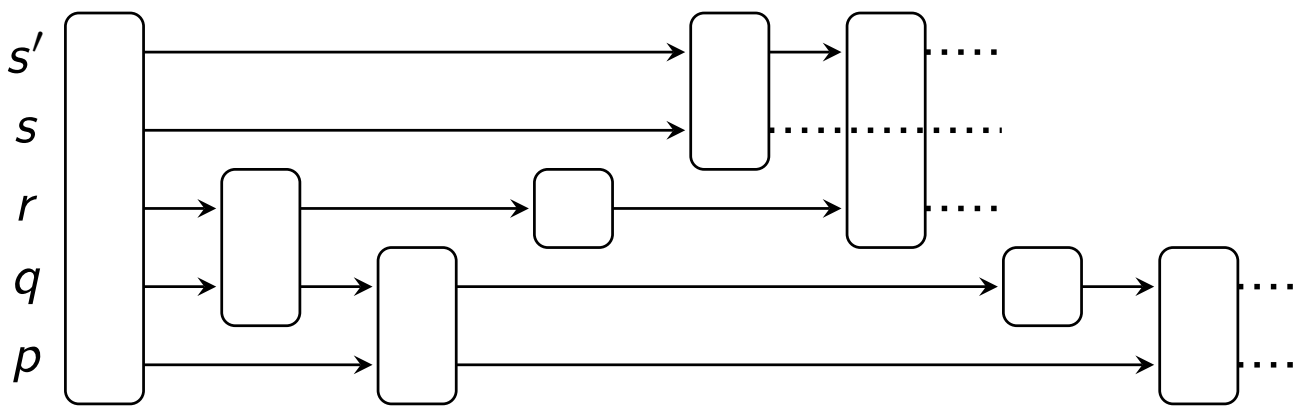
A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.

preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

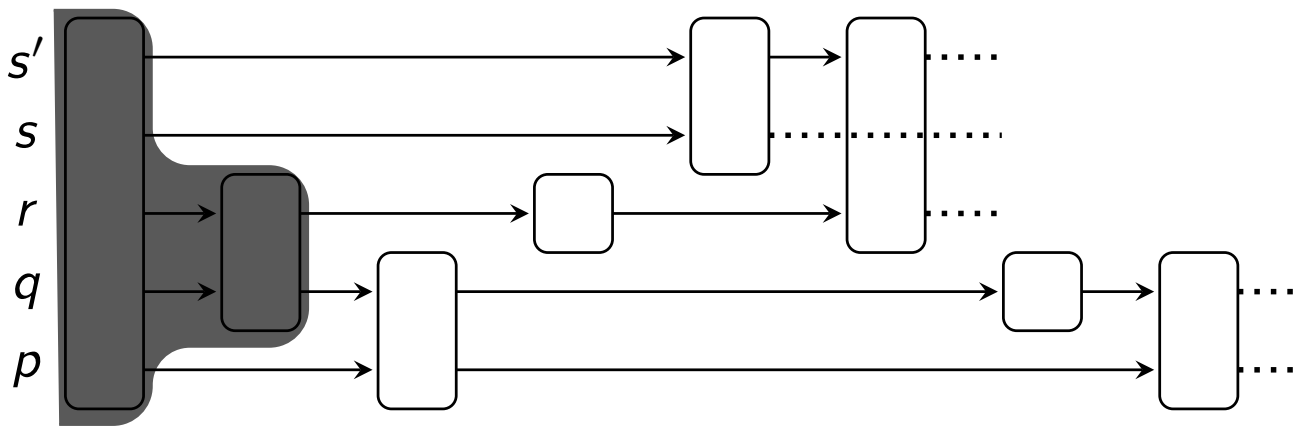
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i\in\mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i\in\mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

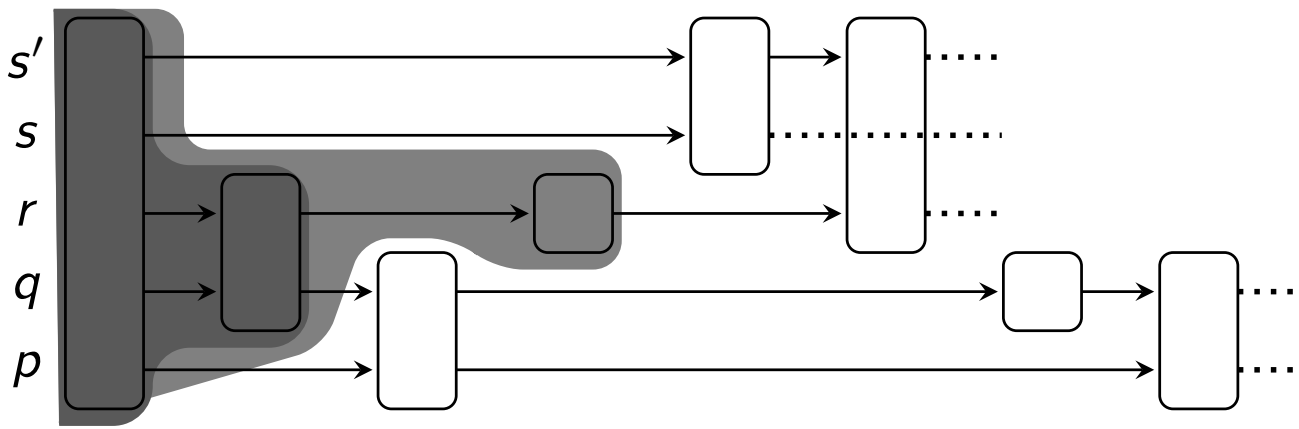
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

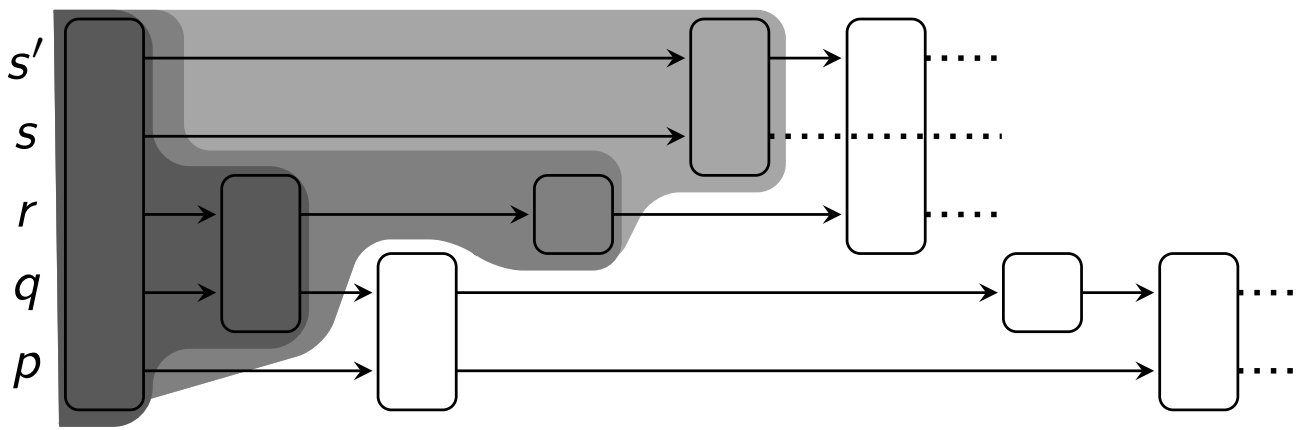
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i\in\mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i\in\mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

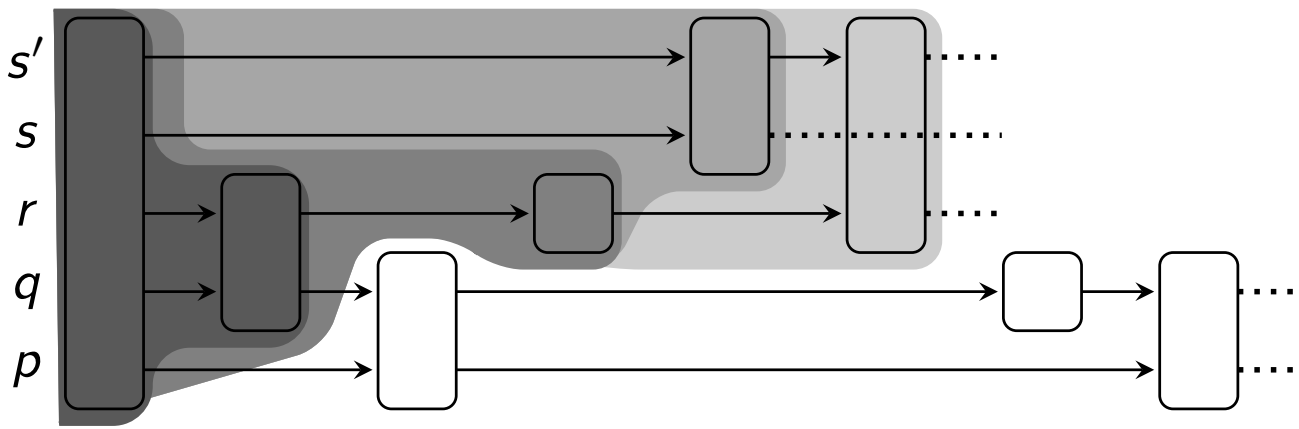
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

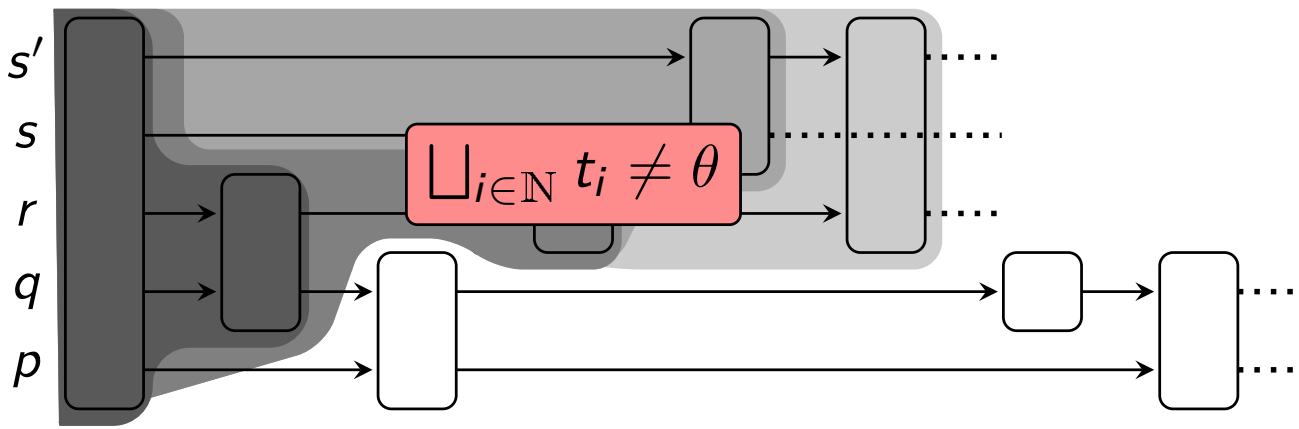
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

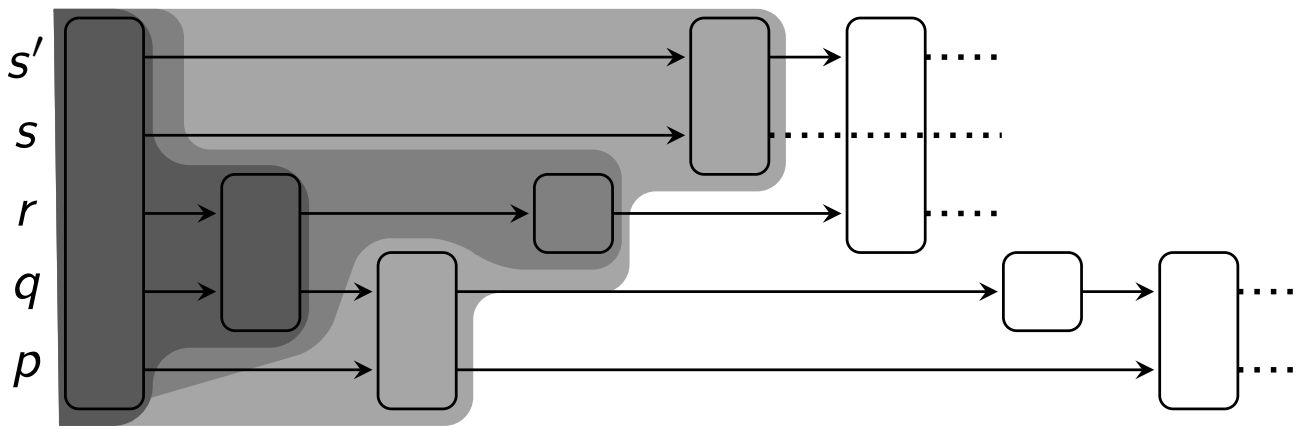
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i\in\mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i\in\mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

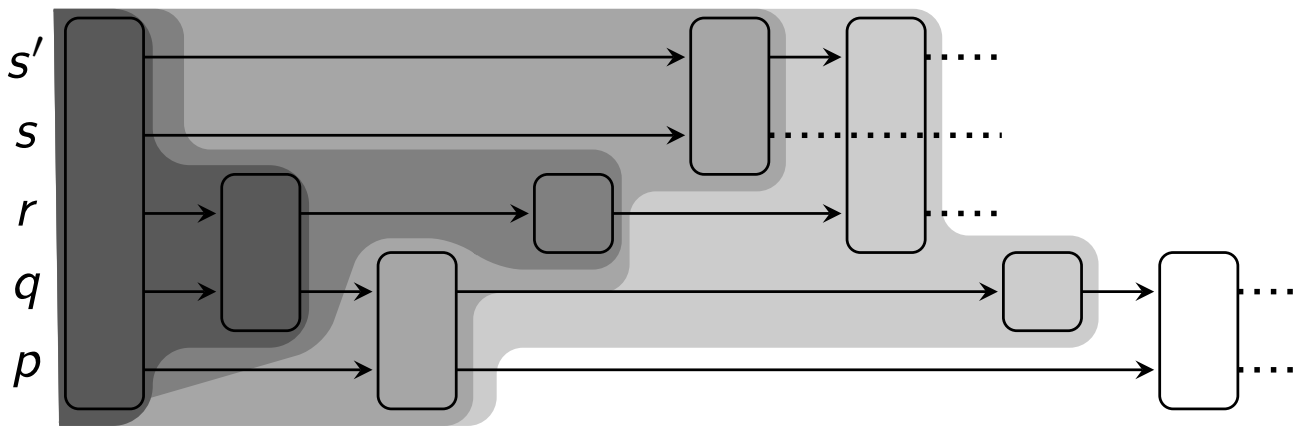
For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.



preliminaries: ω-regular trace languages

A language $\Theta \in \mathbb{R}(\Sigma, I)$ is called an **ω-regular trace language** if there exists a DAMA recognizing it.

For $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$, its **infinitary limit** $\lim(T)$ is the ω-trace language containing all $\theta \in \mathbb{R}(\Sigma, I)$ such that there exists a sequence $(t_i)_{i \in \mathbb{N}}$, $t_i \in T$ satisfying $t_i \sqsubset t_{i+1}$ and $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$.



state of fundamental questions

state of fundamental questions

- FAA and DABA vs. Büchi's theorem: **negative**

state of fundamental questions

- FAA and DABA vs. Büchi's theorem: **negative**
  - There exists $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$ such that no DABA recognizes $\lim(T)$
  - In particular, FAA cannot be exploited as DABA

state of fundamental questions

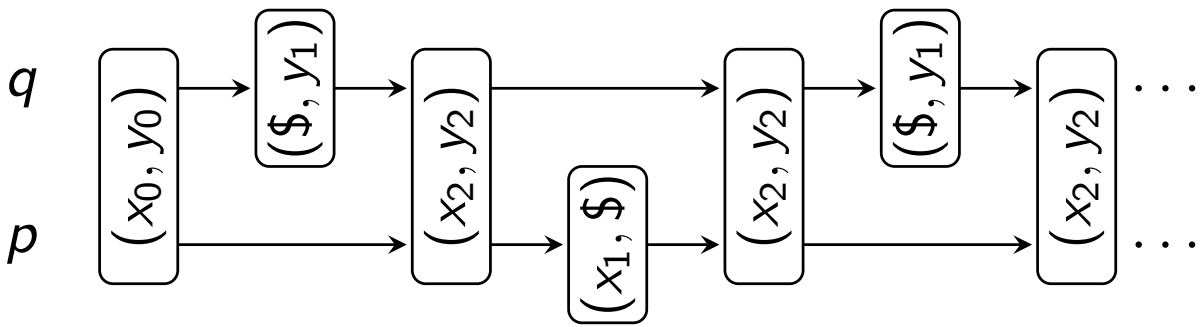
- FAA and DABA vs. Büchi's theorem: **negative**
  - There exists $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$ such that no DABA recognizes $\lim(T)$
  - In particular, FAA cannot be exploited as DABA

$$F = \{(x_1, y_1)\}$$
$$\mathcal{F} = \Big(\big(\{x_1\}, \{y_1\}\big)\Big)$$

state of fundamental questions

- FAA and DABA vs. Büchi's theorem: **negative**
  - There exists $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$ such that no DABA recognizes $\lim(T)$
  - In particular, FAA cannot be exploited as DABA

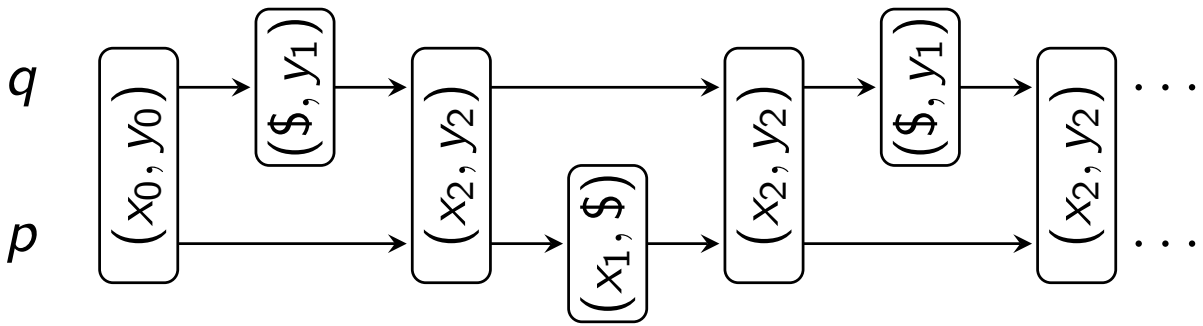
$$F = \{(x_1, y_1)\}$$
$$\mathcal{F} = \Big(\big(\{x_1\}, \{y_1\}\big)\Big)$$



state of fundamental questions

- FAA and DABA vs. Büchi's theorem: **negative**
  - There exists $T \in \mathrm{Rec}(\mathbb{M}(\Sigma, I))$ such that no DABA recognizes $\lim(T)$
  - In particular, FAA cannot be exploited as DABA

$$F = \{(x_1, y_1)\}$$
$$\mathcal{F} = \Big(\big(\{x_1\}, \{y_1\}\big)\Big)$$

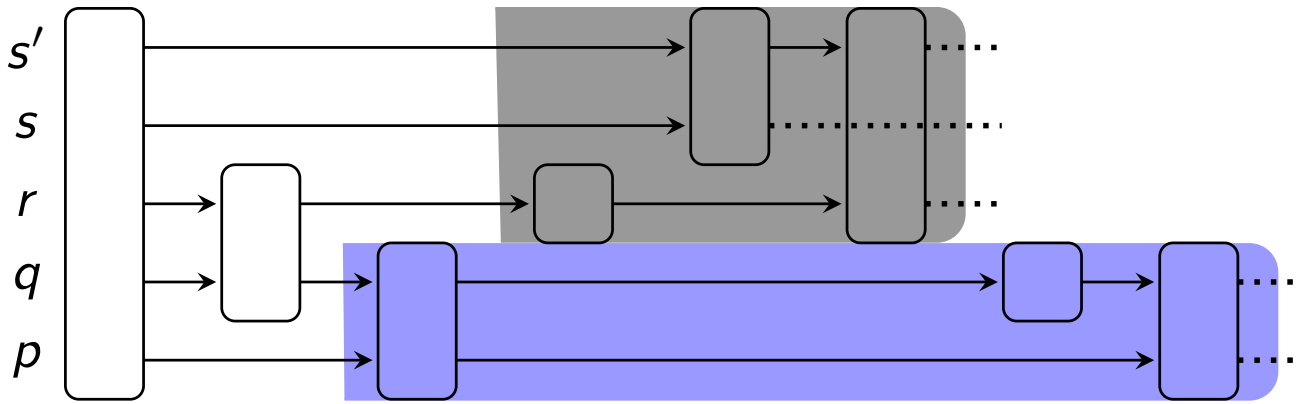


- DABA and DAMA vs. McNaughton's theorem: **open**
- DABA recognizable ω-regular trace languages vs. Landweber's theorem: **open**

goal: awareness of infinitary behavior

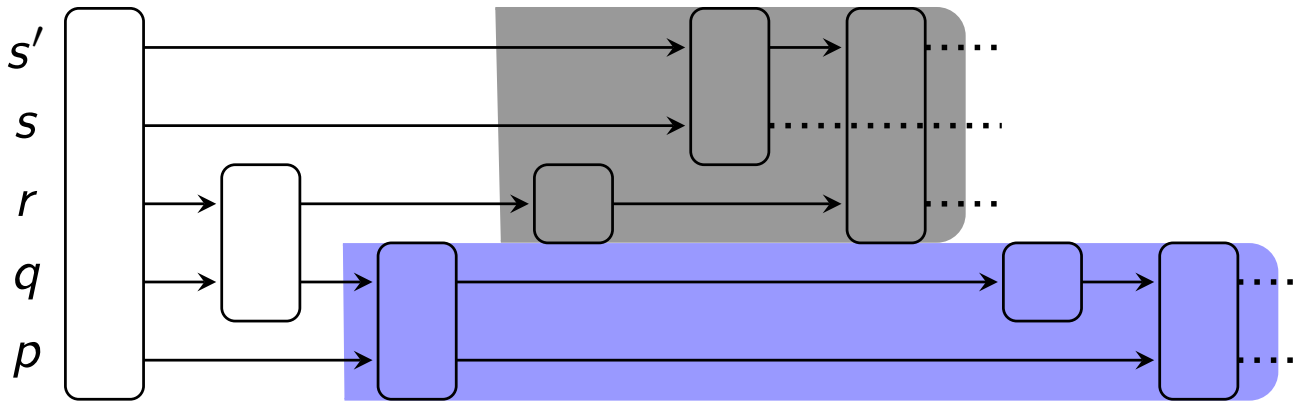
goal: awareness of infinitary behavior

Any infinite run ρ yields a partition $\Psi = \{P_1, \ldots, P_n\}$ of set $\mathcal{P}$ of processes



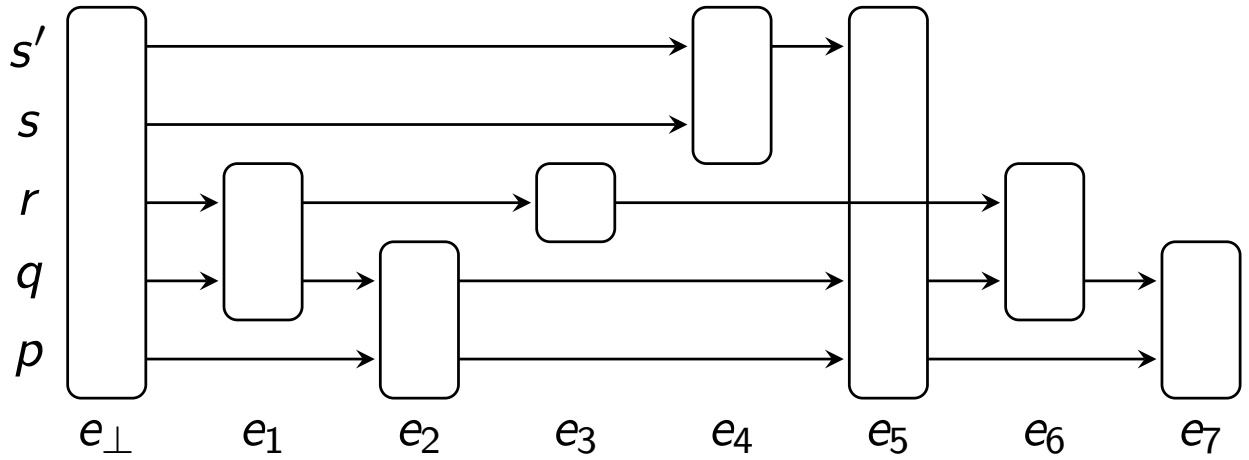
goal: awareness of infinitary behavior

Any infinite run ρ yields a partition $\Psi = \{P_1, \ldots, P_n\}$ of set $\mathcal{P}$ of processes

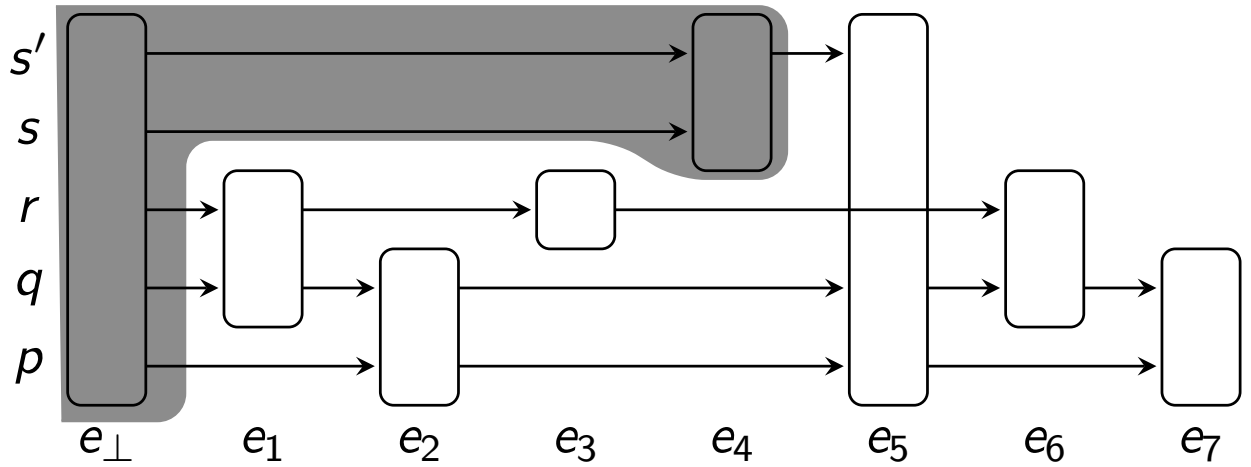


How can each process $p \in \mathcal{P}$ infer the maximal part $P_i \in \Psi$ with whose processes it interacts infinitely often?

interactions and gossip



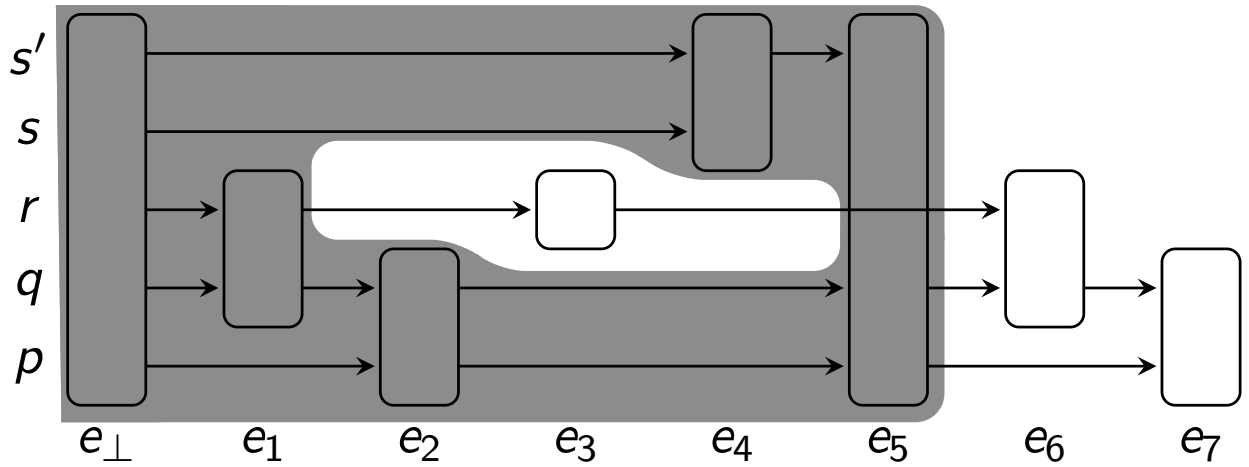
interactions and gossip



$$\rho' = \rho[e_4]$$

$$\mathit{latest}_{s' \to r}(\rho') = e_\perp$$

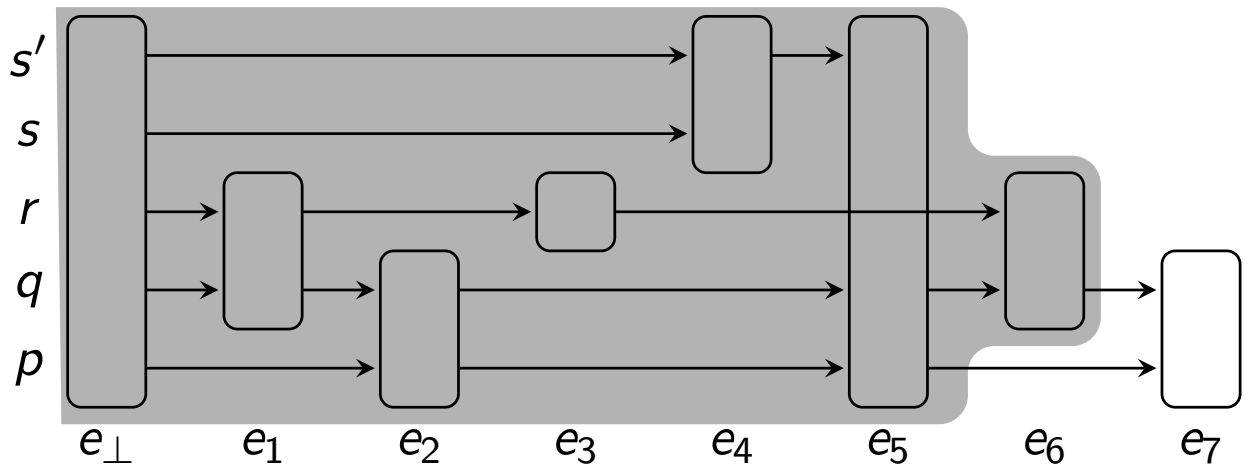
interactions and gossip



$$\rho' = \rho[e_5]$$

$$latest_{s' \to r}(\rho') = e_1$$

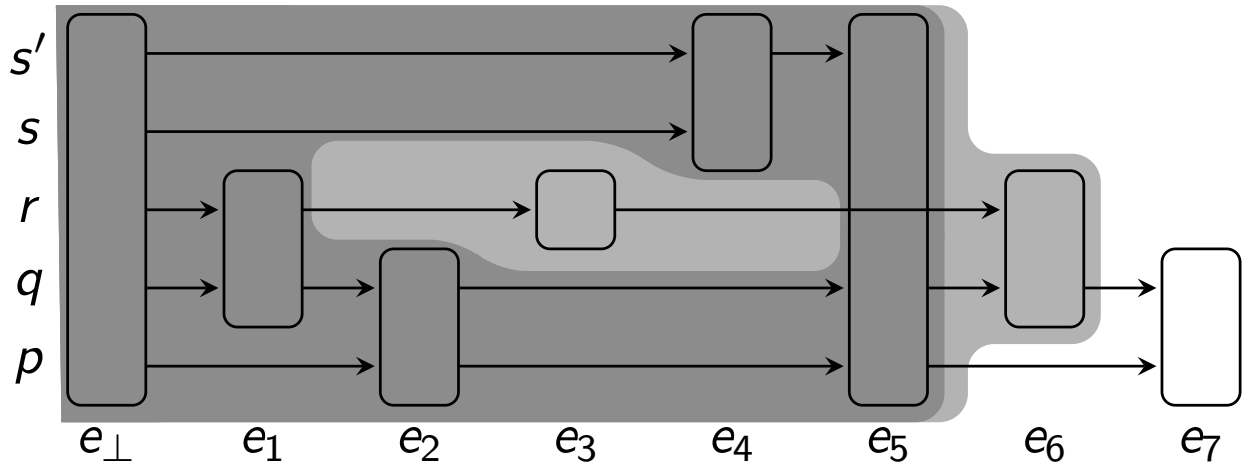
interactions and gossip



$$\rho' = \rho[e_6]$$

$$latest_{r \to s}(\rho') = e_4$$

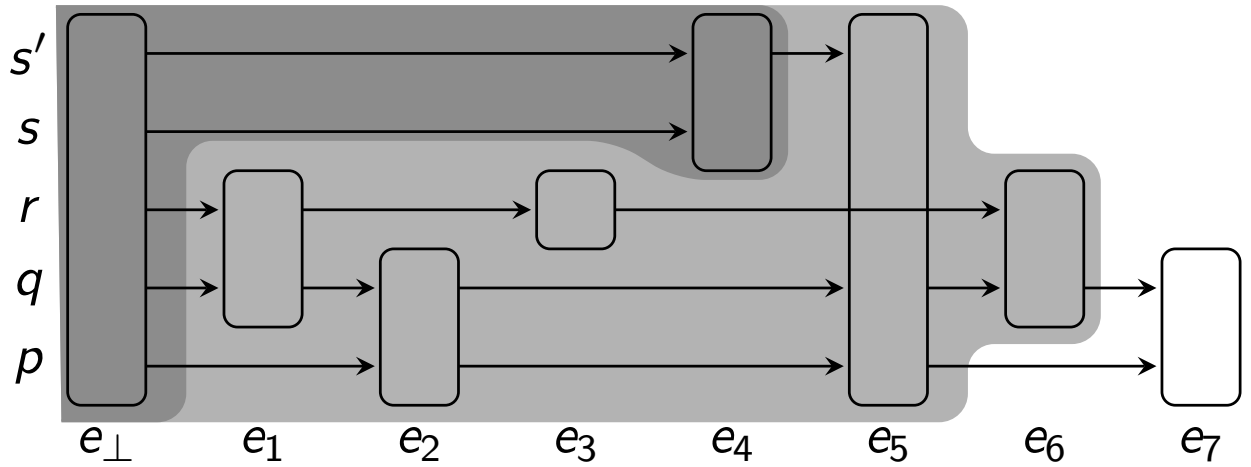
interactions and gossip



$$\rho' = \rho[e_6]$$

$$\mathit{latest}_{r \to s' \to s}(\rho') = e_4$$

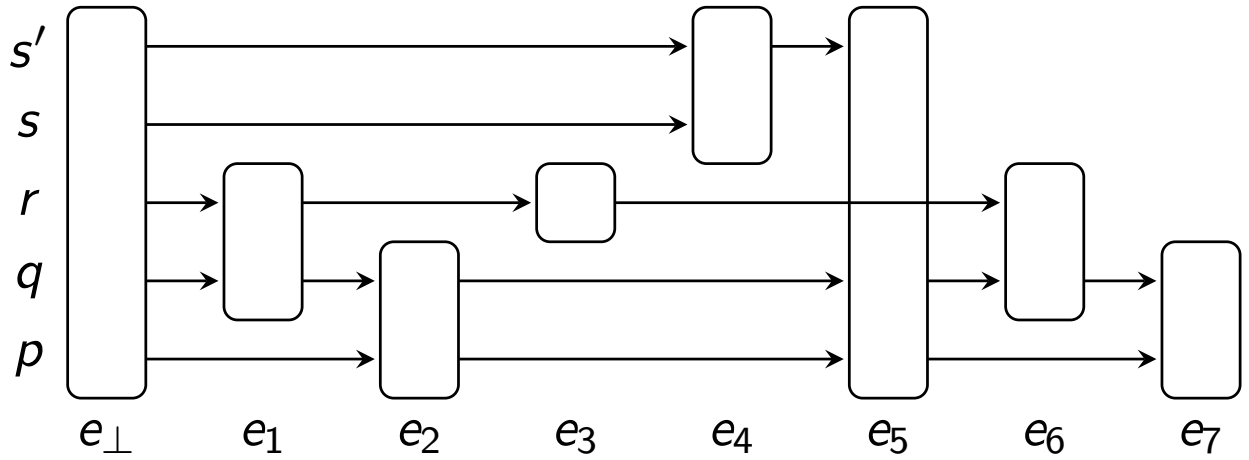
interactions and gossip



$$\rho' = \rho[e_6]$$

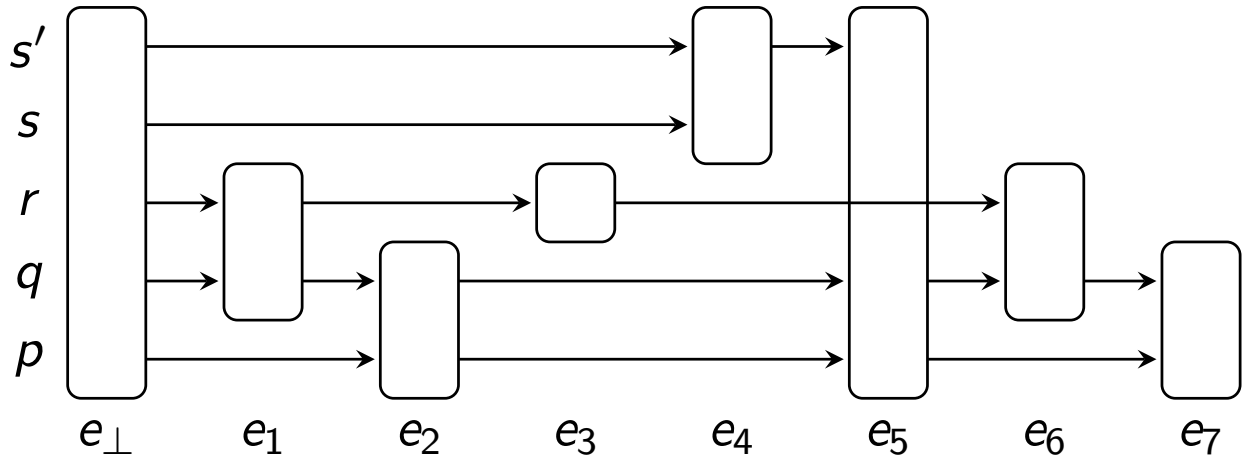
$$latest_{r \to s \to s'}(\rho') = e_4$$

interactions and gossip

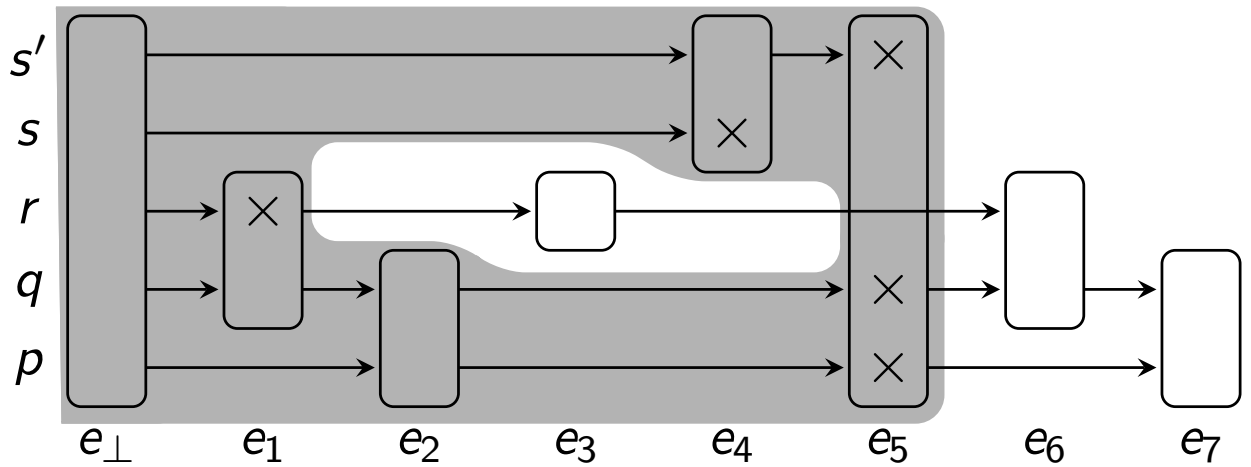


The **primary information** at $e \in \rho$ is the ordered set
$\mathrm{Pri}(e) = \{\mathrm{latest}_{p \to q}(\rho[e]) \mid p \in \mathrm{dom}(e), q \in \mathcal{P}\}$.
The **secondary information** at $e \in \rho$ is the ordered set
$\mathrm{Sec}(e) = \{\mathrm{latest}_{p \to q \to r}(\rho[e]) \mid p \in \mathrm{dom}(e), q, r \in \mathcal{P}\}$.

frontiers and states



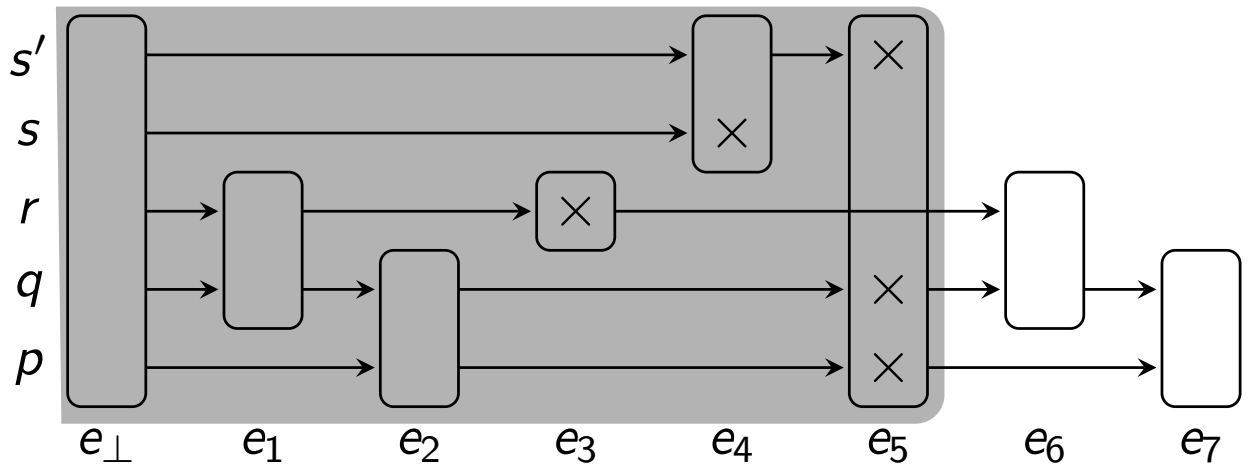
frontiers and states



$\rho' = \rho[e_5]$

frontier $F_{\rho'} \subseteq \rho'$, state $\Lambda(F_{\rho'})$

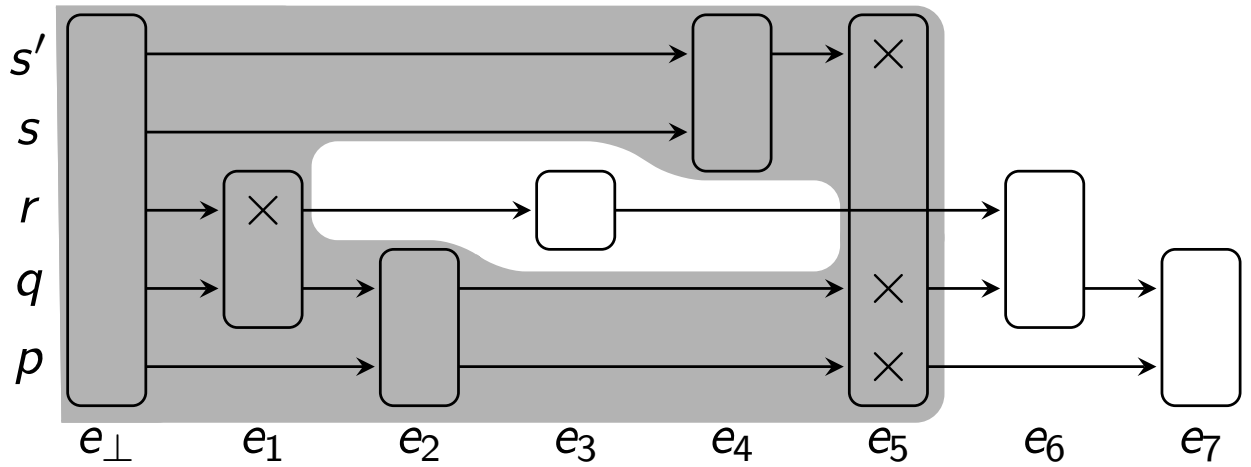
frontiers and states



$\rho' = \rho[\{e_3, e_5\}]$

frontier $F_{\rho'} \subseteq \rho'$, state $\Lambda(F_{\rho'})$

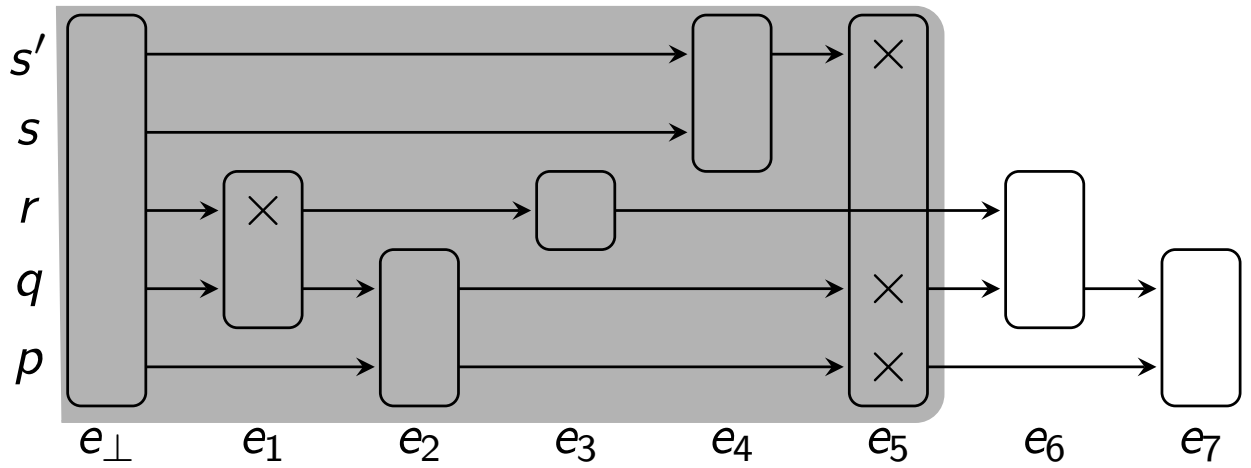
frontiers and states



$\rho' = \rho[e_5]$

partial frontier $F \subseteq F'_\rho$, partial state $\Lambda(F)$

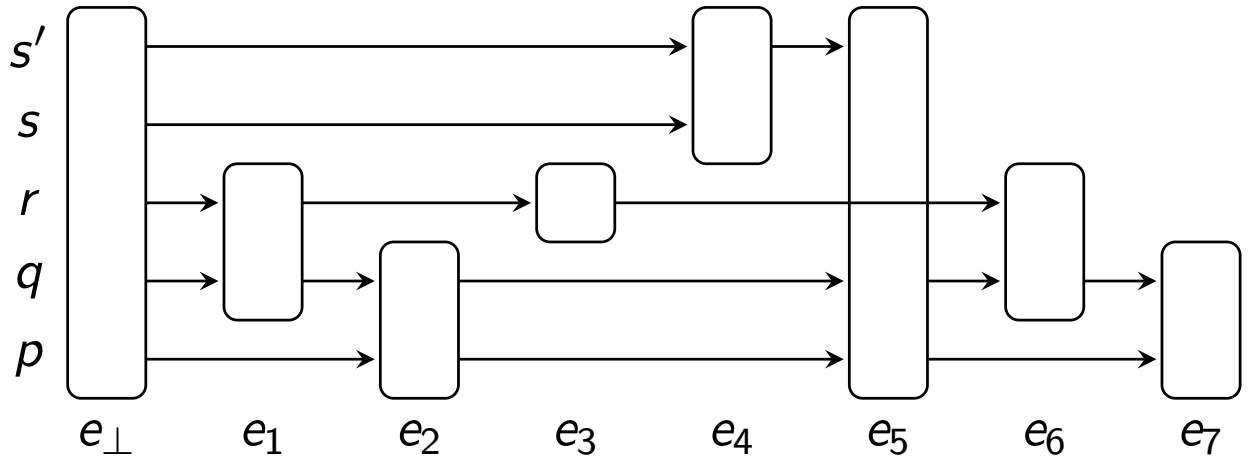
frontiers and states



$\rho' = \rho[\{e_3, e_5\}]$

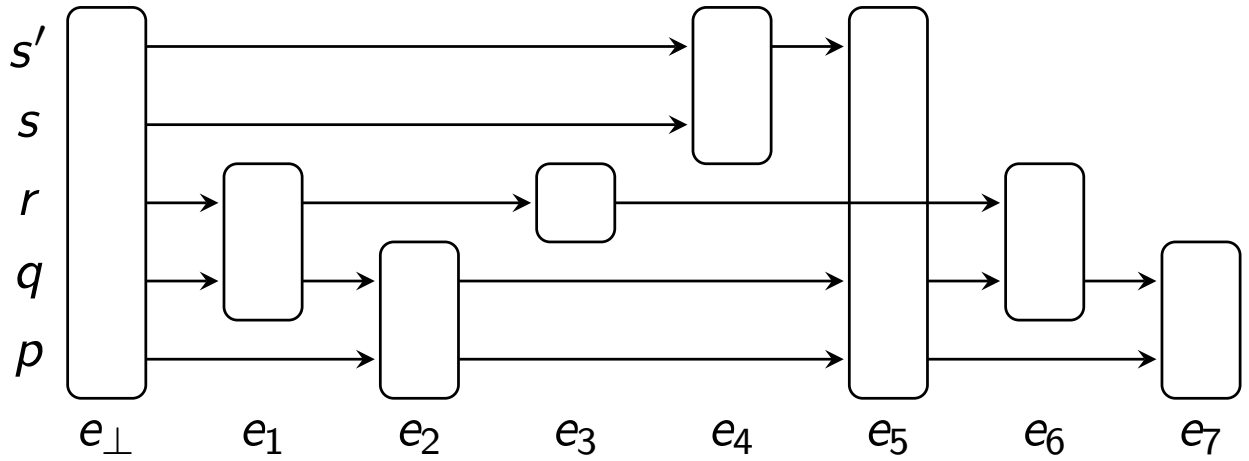
not a partial frontier

frontiers and states



The **frontier** of ρ is $F_\rho := \{e \in \rho \mid \exists p \in \mathcal{P}, e = \max_p(\rho)\}$; and any upward closed subset $F \subseteq F_\rho$ is a **partial frontier** of ρ.

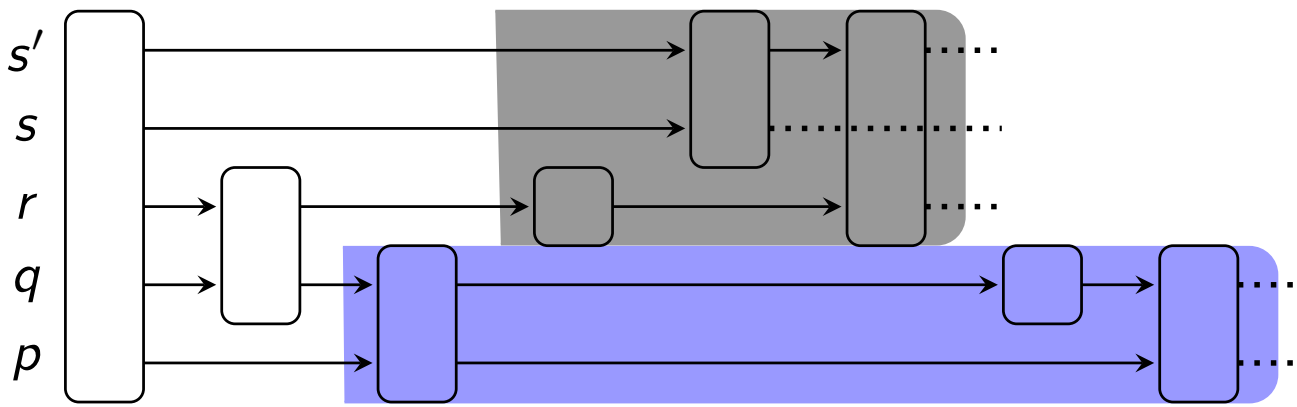
frontiers and states



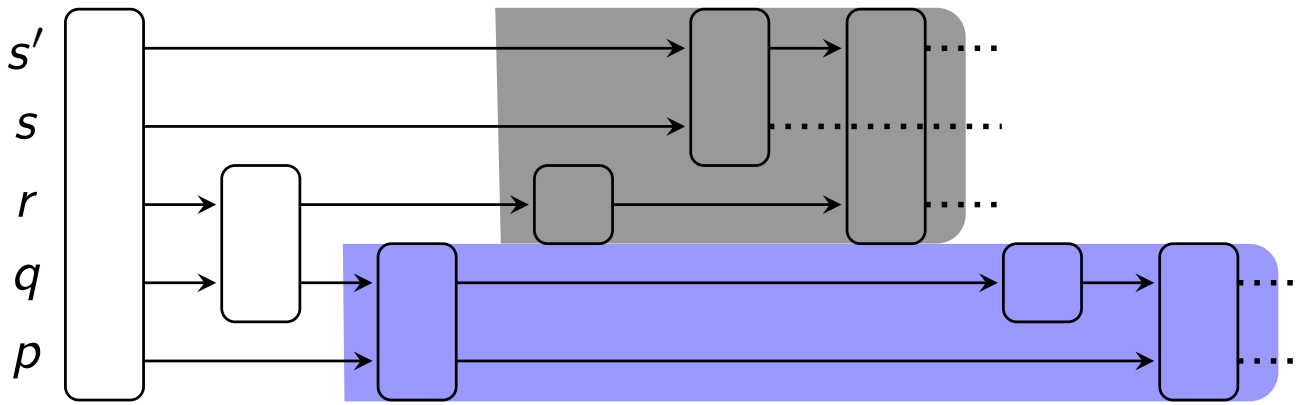
The **frontier** of ρ is $F_\rho := \{e \in \rho \mid \exists p \in \mathcal{P}, e = \max_p(\rho)\}$; and any upward closed subset $F \subseteq F_\rho$ is a **partial frontier** of ρ.

The **top** of e in ρ, $\top_e(\rho) := \{f \in \rho \mid f \in F_\rho \wedge e \leq f\}$.

infinitary behavior vs. partial frontiers



infinitary behavior vs. partial frontiers



Awareness of infinitary behavior

Can local states retrospectively account for relevant partial states?

degrees of synchronization: definition

degrees of synchronization: definition

The **secondary update at** $e \in \rho$ is the set $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q, r \in \mathcal{P}, \exists f_p \lessdot_p e : g = \mathsf{latest}_{p \to q \to r}(f_p) \neq \mathsf{latest}_{p \to q \to r}(e)\}$.

degrees of synchronization: definition

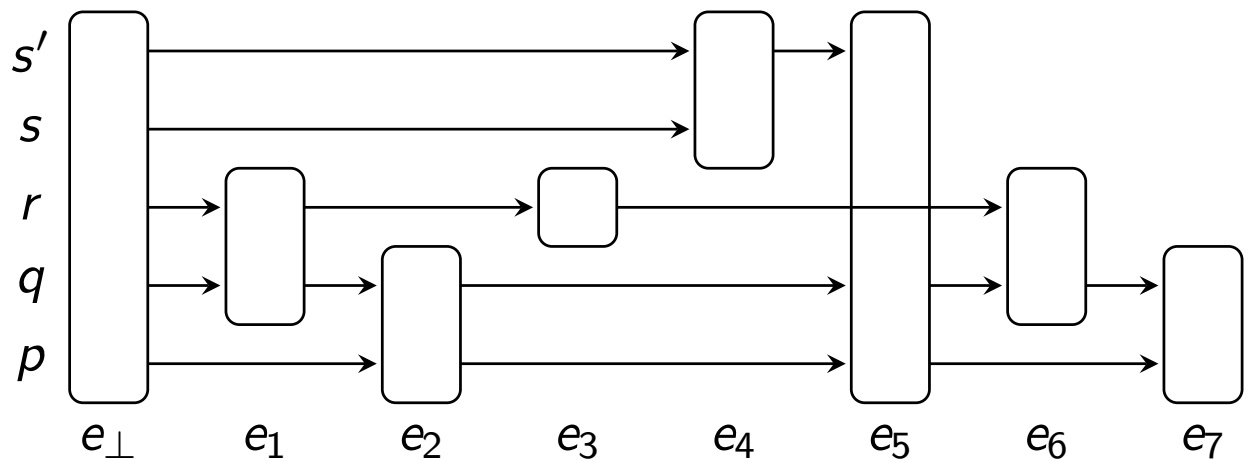
The **secondary update at** $e \in \rho$ is the set $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q, r \in \mathcal{P}, \exists f_p \lessdot_p e : g = \mathsf{latest}_{p \to q \to r}(f_p) \neq \mathsf{latest}_{p \to q \to r}(e)\}$.

The **degree of synchronization at** $e \in \rho$ is $\mathsf{ds}(e) := \bigcup_{g \in \mathcal{U}_e} \mathsf{dom}(\top_{\rho[e]}(g))$. By default, $\mathsf{ds}(e_\perp) := \mathcal{P}$.

degrees of synchronization: definition

The **secondary update at** $e \in \rho$ is the set $\mathcal{U}_e := \{g \in \rho[e] \mid \exists p, q, r \in \mathcal{P}, \exists f_p \lessdot_p e : g = \mathsf{latest}_{p \to q \to r}(f_p) \neq \mathsf{latest}_{p \to q \to r}(e)\}$.

The **degree of synchronization at** $e \in \rho$ is $\mathsf{ds}(e) := \bigcup_{g \in \mathcal{U}_e} \mathsf{dom}(\top_{\rho[e]}(g))$. By default, $\mathsf{ds}(e_\perp) := \mathcal{P}$.



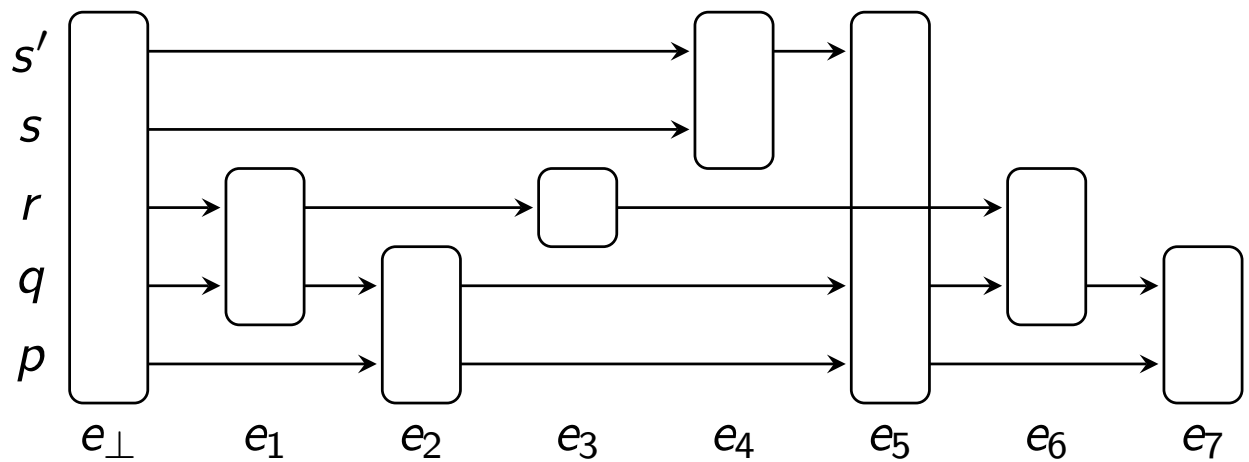
Example on board: compute $\mathsf{ds}(e_7) = \mathcal{P}$

degrees of synchronization: utility (i)

Lemma (Partial fronts and secondary updates)

For $e \in \rho$, $e > e_\perp$, let $\rho_\sqcap := \bigsqcap_{f_p \lessdot_p e} \rho[f_p]$ be the greatest lower bound of all its p-prefixes. then, for every prefix $\rho' \sqsubseteq \rho[e]$ with $\rho' \not\sqsupseteq \rho_\sqcap$, there exist $F \subseteq \rho'$ and $U \subseteq \mathcal{U}_e$ such that

1. *F is a partial frontier in ρ' with $\mathrm{dom}(F) = \mathrm{ds}(e)$; and*
2. *$\bigcup_{g \in U} \top_{\rho'}(g) = F$.*



Example on board: illustrate the lemma for e_7

degrees of synchronization: utility (ii)

Lemma (infinitary behavior)

For an infinite run ρ and $p \in \mathcal{P}$, if $p \in \mathrm{dom}(\mathrm{alphinf}(\rho))$ then there exists a unique maximal $P \subseteq \mathcal{P}$ such that $\exists^{\infty} e \in \rho : p \in \mathrm{dom}(e) \wedge \mathrm{ds}(e) = P$.

degrees of synchronization: utility (ii)

Lemma (infinitary behavior)

For an infinite run ρ and $p \in \mathcal{P}$, if $p \in \mathrm{dom}(\mathrm{alphinf}(\rho))$ then there exists a unique maximal $P \subseteq \mathcal{P}$ such that $\exists^{\infty} e \in \rho : p \in \mathrm{dom}(e) \wedge \mathrm{ds}(e) = P$.

Call the set P as mentioned above the **max-degree of p-synchronizations in ρ**, denoted by $\lceil \mathrm{ds}_p(\rho) \rceil$.

degrees of synchronization: utility (ii)

Lemma (infinitary behavior)

For an infinite run ρ and $p \in \mathcal{P}$, if $p \in \mathrm{dom}(\mathrm{alphinf}(\rho))$ then there exists a unique maximal $P \subseteq \mathcal{P}$ such that $\exists^{\infty} e \in \rho : p \in \mathrm{dom}(e) \wedge \mathrm{ds}(e) = P$.

Call the set P as mentioned above the **max-degree of p-synchronizations in ρ**, denoted by $\lceil \mathrm{ds}_p(\rho) \rceil$.

If a run ρ induces a partition Ψ of the set of states, then for each part $P_i \in \Psi$: $q \in P_i \Leftrightarrow \lceil \mathrm{ds}_q(\rho) \rceil = P_i$.

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

- $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$ is an ATS

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

- $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$ is an ATS
- $\mathcal{D} = (\mathcal{D}_p)_{p\in\mathcal{P}}$ is a collection of mappings $\mathcal{D}_p \colon X_p \to 2^{\mathcal{P}}$ such that

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

- $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$ is an ATS
- $\mathcal{D} = (\mathcal{D}_p)_{p\in\mathcal{P}}$ is a collection of mappings $\mathcal{D}_p \colon X_p \to 2^{\mathcal{P}}$ such that
  - for each $p \in \mathcal{P}$, $\mathcal{D}_p(\pi_{0|p}) = \mathcal{P}$, and

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

- $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$ is an ATS
- $\mathcal{D} = (\mathcal{D}_p)_{p\in\mathcal{P}}$ is a collection of mappings $\mathcal{D}_p \colon X_p \to 2^{\mathcal{P}}$ such that
  - for each $p \in \mathcal{P}$, $\mathcal{D}_p(\pi_{0|p}) = \mathcal{P}$, and
  - for every run ρ of $\mathfrak{T}$ and every event $e \in \rho$, if $\Lambda(e) = \pi$ and $p \in \mathrm{dom}(e)$ then $\mathrm{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi_{|p}) = P$

synchronization-aware transition systems

An **SATS** is a couple $(\mathfrak{T}, \mathcal{D})$ where

- $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$ is an ATS
- $\mathcal{D} = (\mathcal{D}_p)_{p\in\mathcal{P}}$ is a collection of mappings $\mathcal{D}_p \colon X_p \to 2^{\mathcal{P}}$ such that
  - for each $p \in \mathcal{P}$, $\mathcal{D}_p(\pi_{0|p}) = \mathcal{P}$, and
  - for every run ρ of $\mathfrak{T}$ and every event $e \in \rho$, if $\Lambda(e) = \pi$ and $p \in \mathrm{dom}(e)$ then $\mathrm{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi_{|p}) = P$

For an infinite run ρ, processes p refer the **maximal local infinity sets**

$$\lceil \mathrm{Inf}_p(\rho) \rceil = \begin{cases} \left\{ x \in X_p \;\middle|\; \begin{array}{l} \mathcal{D}_p(x) = \lceil \mathrm{ds}_p(\rho) \rceil \wedge \\ \exists^{\infty} e \in \rho : \Lambda(e)_{|p} = x \end{array} \right\} & p \in \mathrm{dom}(\mathrm{alphinf}(\theta)) \\ \left\{ x \in X_p \;\middle|\; \begin{array}{l} \exists e \in \rho : e = \max_p(\rho) \\ \wedge\ \Lambda(e)_{|p} = x \end{array} \right\} & \text{otherwise.} \end{cases}$$

deterministic, synchronization-aware Büchi automata

A **D-SABA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

deterministic, synchronization-aware Büchi automata

A **D-SABA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

A D-SABA $\mathfrak{A}$ **accepts a trace** $\theta \in \mathbb{R}(\Sigma, I)$ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process $p \in \mathcal{P}$, $F_i^p \cap \lceil \mathsf{Inf}_p(\rho) \rceil \neq \emptyset$.

deterministic, synchronization-aware Büchi automata

A **D-SABA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

A D-SABA $\mathfrak{A}$ **accepts a trace** $\theta \in \mathbb{R}(\Sigma, I)$ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process $p \in \mathcal{P}$, $F_i^p \cap \lceil \mathsf{Inf}_p(\rho) \rceil \neq \emptyset$.

Theorem (à la Büchi)

A language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ is recognized by a D-SABA iff $\Theta = \lim(T)$ for a regular trace language $T \subseteq \mathbb{M}(\Sigma, I)$.

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

Idea: During a run $\overline{\rho}$ over any trace θ, $\overline{\mathfrak{T}}$ must:

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

Idea: During a run $\overline{\rho}$ over any trace θ, $\overline{\mathfrak{T}}$ must:

1. Infer the partition $\Psi = \{P_1, \ldots, P_n\}$ of $\mathcal{P}$ induced by θ,

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

Idea: During a run $\overline{\rho}$ over any trace θ, $\overline{\mathfrak{T}}$ must:

1. Infer the partition $\Psi = \{P_1, \ldots, P_n\}$ of $\mathcal{P}$ induced by θ,
2. Mimic the run ρ of $\mathfrak{T}$ over θ, and

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

Idea: During a run $\overline{\rho}$ over any trace θ, $\overline{\mathfrak{T}}$ must:

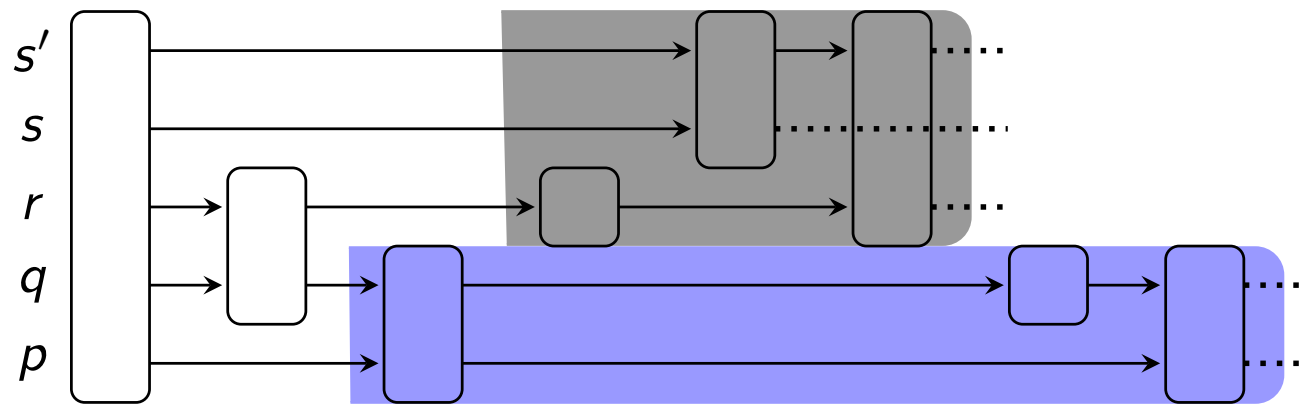
1. Infer the partition $\Psi = \{P_1, \ldots, P_n\}$ of $\mathcal{P}$ induced by θ,
2. Mimic the run ρ of $\mathfrak{T}$ over θ, and
3. Collect appropriate information in order compute (in hindsight) the partial P_i-states of $\mathfrak{T}$ occurring infinitely often

one half of the proof

Given: An FAA $\mathfrak{A} = (\mathfrak{T}, F)$ recognizing T
Goal: Construct a D-SABA $\overline{\mathfrak{A}} = (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ recognizing $\lim(T)$

Idea: During a run $\overline{\rho}$ over any trace θ, $\overline{\mathfrak{T}}$ must:

1. Infer the partition $\Psi = \{P_1, \ldots, P_n\}$ of $\mathcal{P}$ induced by θ,
2. Mimic the run ρ of $\mathfrak{T}$ over θ, and
3. Collect appropriate information in order compute (in hindsight) the partial P_i-states of $\mathfrak{T}$ occurring infinitely often



Example on board: rough illustration of item 3

combining partial states

Let $\pi_1 = (x_1, x_2, \ldots x_N)$ and $\pi_2 = (y_1, y_2, \ldots y_N)$ be two partial states.
The **projection of π_2 on π_1** is defined as $\pi_1 \lhd \pi_2 = (z_1, z_2, \ldots z_N)$ where the local p_i-states $z_i := \begin{cases} y_i & \text{if } y_i \neq \$ \\ x_i & \text{otherwise} \end{cases}$, where $N := |\mathcal{P}|$.

combining partial states

Let $\pi_1 = (x_1, x_2, \ldots x_N)$ and $\pi_2 = (y_1, y_2, \ldots y_N)$ be two partial states. The **projection of π_2 on π_1** is defined as $\pi_1 \lhd \pi_2 = (z_1, z_2, \ldots z_N)$ where the local p_i-states $z_i := \begin{cases} y_i & \text{if } y_i \neq \$ \\ x_i & \text{otherwise} \end{cases}$, where $N := |\mathcal{P}|$.

For π_1 and π_2 two "compatible states", then their **join** is defined as $\pi_1 \otimes \pi_2 := \pi_1 \lhd \pi_2 = \pi_2 \lhd \pi_1$. The **join** of two "incompatible states" is not defined.

combining partial states

Let $\pi_1 = (x_1, x_2, \ldots x_N)$ and $\pi_2 = (y_1, y_2, \ldots y_N)$ be two partial states. The **projection of π_2 on π_1** is defined as $\pi_1 \lhd \pi_2 = (z_1, z_2, \ldots z_N)$ where the local p_i-states $z_i := \begin{cases} y_i & \text{if } y_i \neq \$ \\ x_i & \text{otherwise} \end{cases}$, where $N := |\mathcal{P}|$.

For π_1 and π_2 two "compatible states", then their **join** is defined as $\pi_1 \otimes \pi_2 := \pi_1 \lhd \pi_2 = \pi_2 \lhd \pi_1$. The **join** of two "incompatible states" is not defined.

For a partial frontier $F \subseteq \rho$ and an event $e \in \rho$ s.t. $F \cap \top_\rho(e) \neq \emptyset$,

combining partial states

Let $\pi_1 = (x_1, x_2, \ldots x_N)$ and $\pi_2 = (y_1, y_2, \ldots y_N)$ be two partial states. The **projection of π_2 on π_1** is defined as $\pi_1 \lhd \pi_2 = (z_1, z_2, \ldots z_N)$ where the local p_i-states $z_i := \begin{cases} y_i & \text{if } y_i \neq \$ \\ x_i & \text{otherwise} \end{cases}$, where $N := |\mathcal{P}|$.

For π_1 and π_2 two "compatible states", then their **join** is defined as $\pi_1 \otimes \pi_2 := \pi_1 \lhd \pi_2 = \pi_2 \lhd \pi_1$. The **join** of two "incompatible states" is not defined.

For a partial frontier $F \subseteq \rho$ and an event $e \in \rho$ s.t. $F \cap \top_\rho(e) \neq \emptyset$, the **projection of F on e in ρ** is defined as a state $\rho[e \lhd F] := (x_p)_{p \in \mathcal{P}}$, where the local p-states

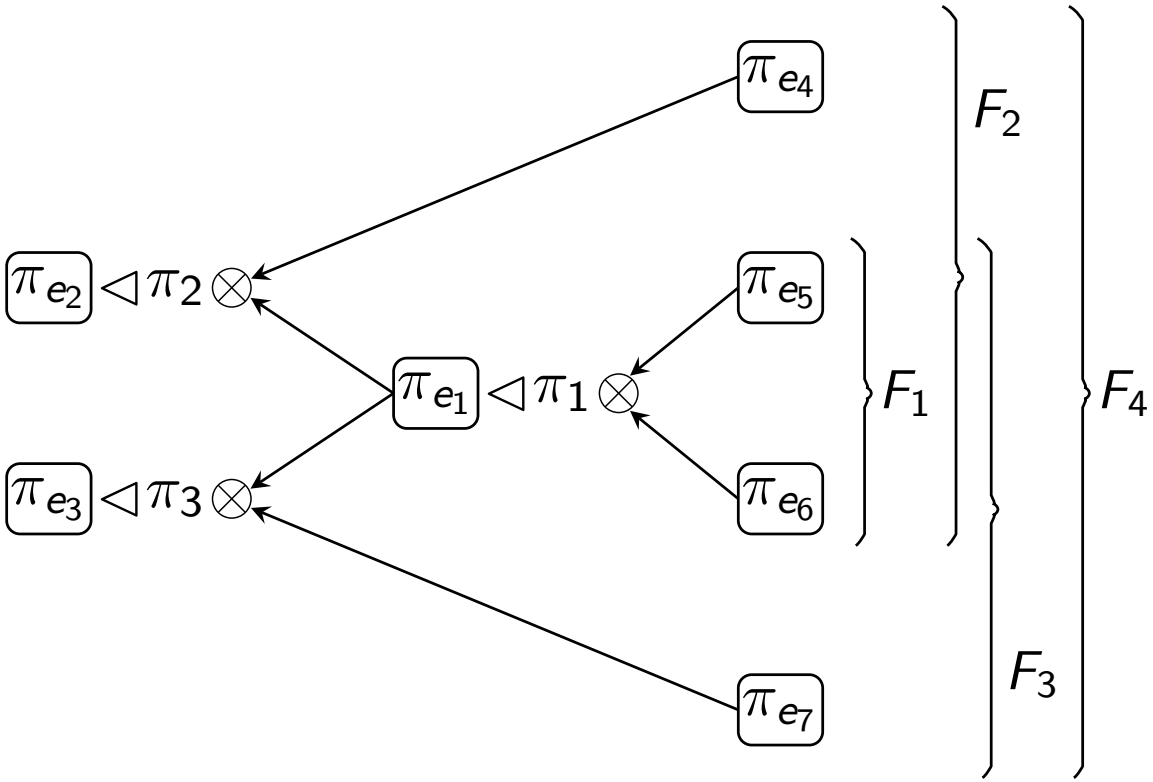
$$x_p := \begin{cases} \Lambda(e_p)_{|p} & \text{if there exists } e_p = \max_p(\rho[F]), e < e_p, \\ \$ & \text{otherwise.} \end{cases}$$

constructing partial states in hindsight

Note: If $F = \top_\rho(e)$ then $\Lambda(F) = \Lambda(e) \lhd \rho[e \lhd F]$

constructing partial states in hindsight

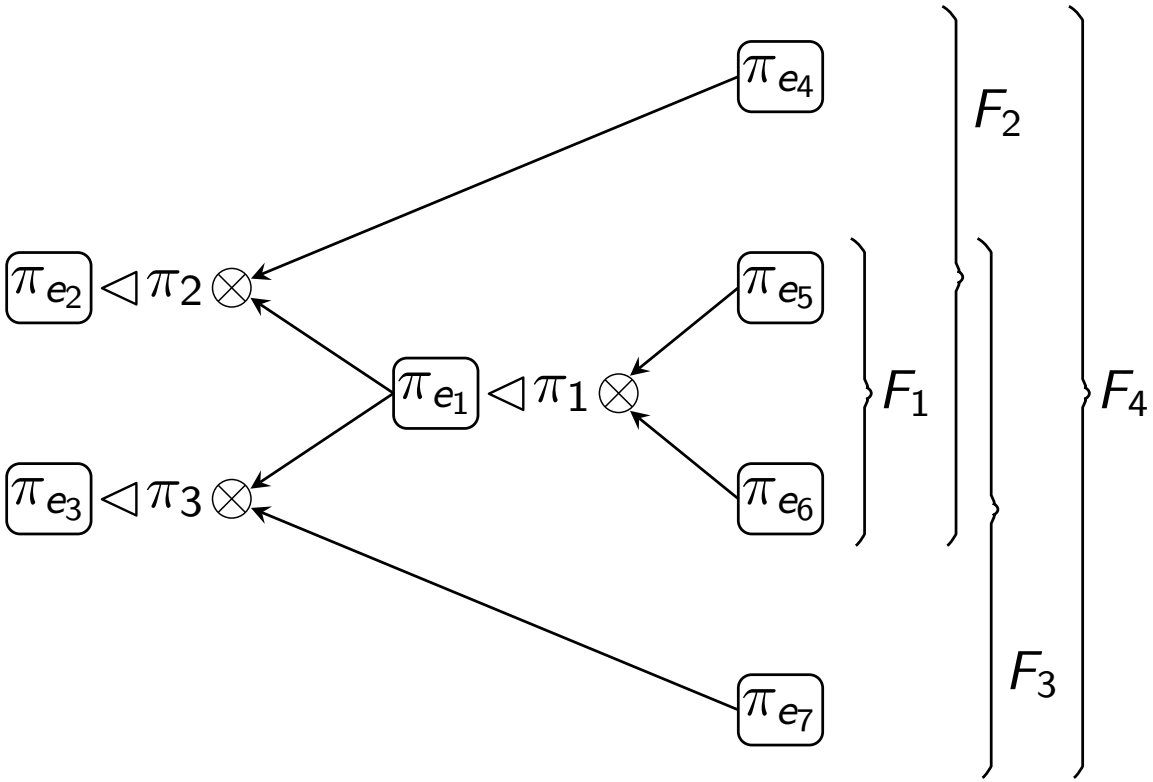
Note: If $F = \top_\rho(e)$ then $\Lambda(F) = \Lambda(e) \lhd \rho[e \lhd F]$



For mutually concurrent events e_i, if $F = \bigcup_{i=1}^{n} \top_\rho(e_i)$, then the partial $\mathrm{dom}(F)$-state in ρ can be computed as $\Lambda(F) =$

constructing partial states in hindsight

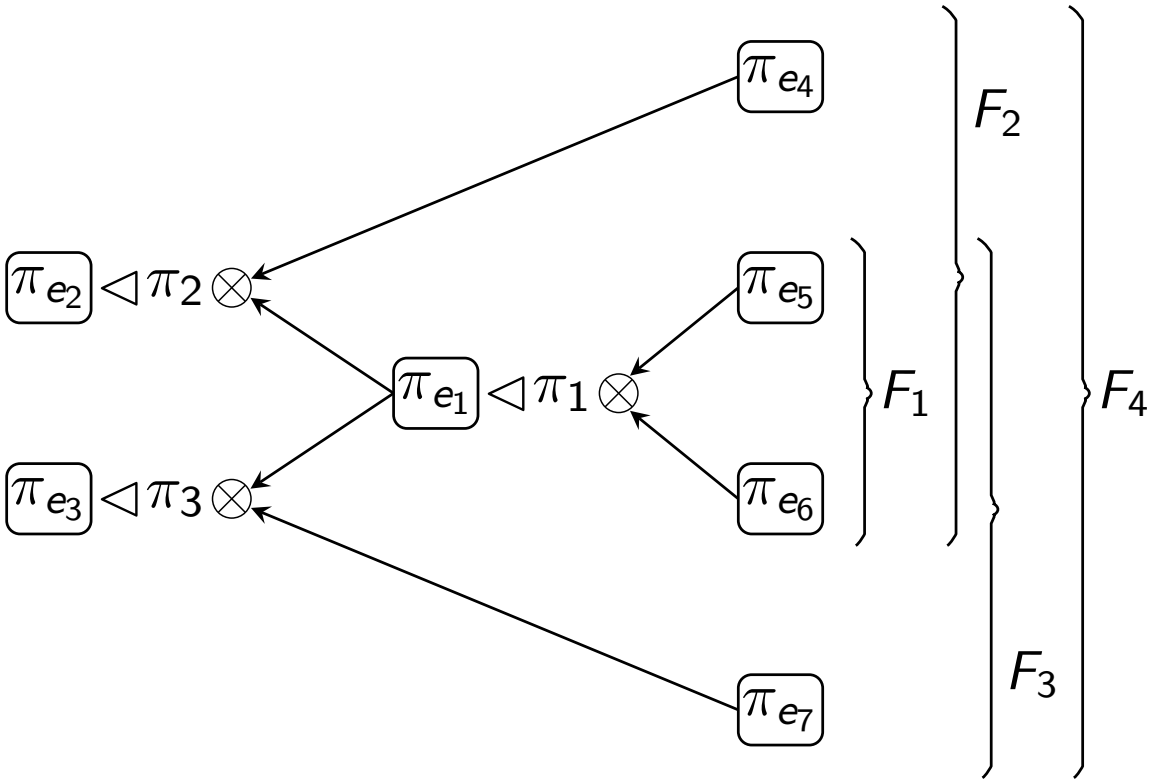
Note: If $F = \top_\rho(e)$ then $\Lambda(F) = \Lambda(e) \lhd \rho[e \lhd F]$



For mutually concurrent events e_i, if $F = \bigcup_{i=1}^{n} \top_\rho(e_i)$, then the partial $\mathrm{dom}(F)$-state in ρ can be computed as $\Lambda(F) = \qquad\qquad \rho[e_i \lhd F]$

constructing partial states in hindsight

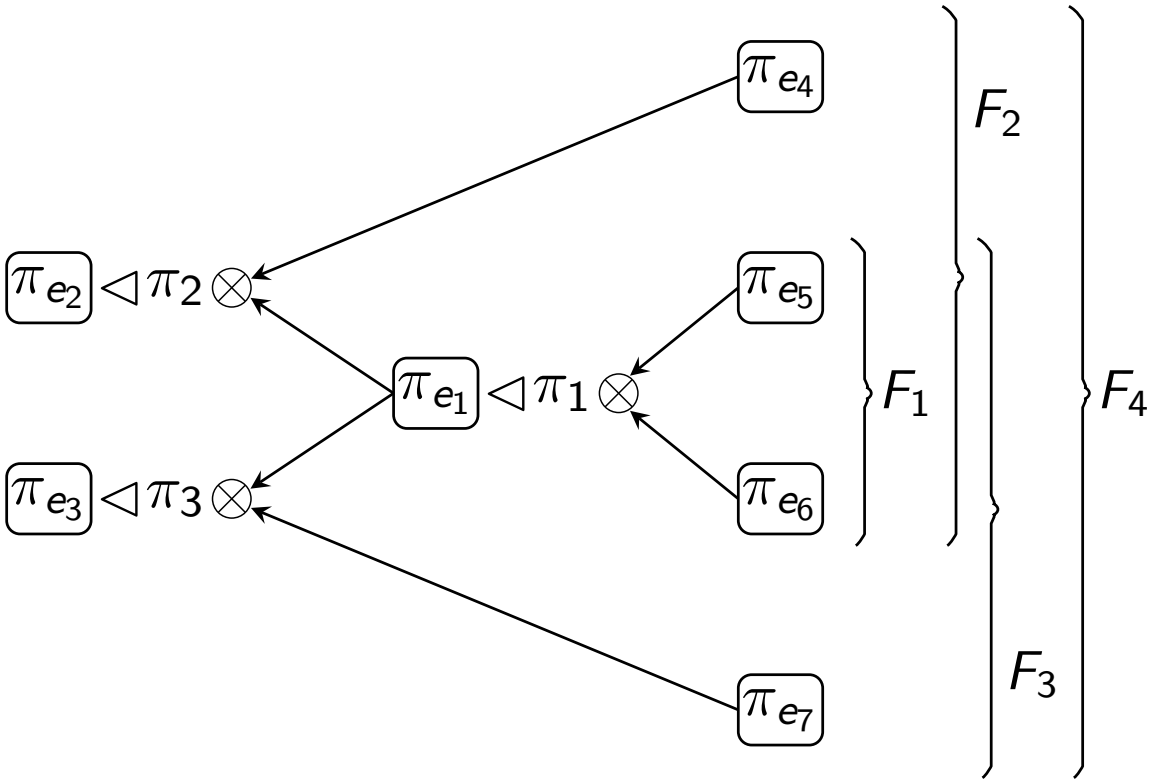
Note: If $F = \top_\rho(e)$ then $\Lambda(F) = \Lambda(e) \lhd \rho[e \lhd F]$



For mutually concurrent events e_i, if $F = \bigcup_{i=1}^{n} \top_\rho(e_i)$, then the partial dom$(F)$-state in ρ can be computed as $\Lambda(F) = \qquad \Lambda(e_i) \lhd \rho[e_i \lhd F]$

constructing partial states in hindsight

Note: If $F = \top_\rho(e)$ then $\Lambda(F) = \Lambda(e) \lhd \rho[e \lhd F]$



For mutually concurrent events e_i, if $F = \bigcup_{i=1}^{n} \top_\rho(e_i)$, then the partial $\mathrm{dom}(F)$-state in ρ can be computed as $\Lambda(F) = \bigotimes_{i=1}^{n} \Big(\Lambda(e_i) \lhd \rho[e_i \lhd F] \Big)$

effective construction: finite memory

effective construction: finite memory

At any event $e \in \rho$, the **augmented secondary information** is defined as $\overline{\mathrm{Sec}}(e) := \{(f, \Pi, \pi) \mid f \in \mathrm{Sec}(e), \Pi \subseteq X_{2^{\mathcal{P}}}, \pi = \Lambda(f)\}$. Moreover, for each secondary event $f \in \mathrm{Sec}(e)$ there exists exactly one **augmented event** $\overline{f} = (f, \Pi, \pi) \in \overline{\mathrm{Sec}}(e)$.

effective construction: finite memory

At any event $e \in \rho$, the **augmented secondary information** is defined as $\overline{\mathrm{Sec}}(e) := \{(f, \Pi, \pi) \mid f \in \mathrm{Sec}(e), \Pi \subseteq X_{2^{\mathcal{P}}}, \pi = \Lambda(f)\}$. Moreover, for each secondary event $f \in \mathrm{Sec}(e)$ there exists exactly one **augmented event** $\overline{f} = (f, \Pi, \pi) \in \overline{\mathrm{Sec}}(e)$.

Invariant: Modularity of ASI

For each event $\overline{f} = (f, \Pi, \pi) \in \overline{\mathrm{Sec}}(e)$, there exists a partial state $\pi' \in \Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier F in ρ' such that

1. $F = \top_{\rho'}(f)$;
2. $\pi' = \rho'[f \lhd F]$; and
3. if $E \subseteq \mathrm{Sec}(e)$ is a set s.t. $g \in E \Rightarrow g > f$ then $F \notin \bigcup_{g \in E} \top_{\rho'}(g)$.

effective construction: finite memory

At any event $e \in \rho$, the **augmented secondary information** is defined as $\overline{\mathrm{Sec}}(e) := \{(f, \Pi, \pi) \mid f \in \mathrm{Sec}(e), \Pi \subseteq X_{2^{\mathcal{P}}}, \pi = \Lambda(f)\}$. Moreover, for each secondary event $f \in \mathrm{Sec}(e)$ there exists exactly one **augmented event** $\overline{f} = (f, \Pi, \pi) \in \overline{\mathrm{Sec}}(e)$.

Invariant: Modularity of ASI

For each event $\overline{f} = (f, \Pi, \pi) \in \overline{\mathrm{Sec}}(e)$, there exists a partial state $\pi' \in \Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier F in ρ' such that

1. $F = \top_{\rho'}(f)$;
2. $\pi' = \rho'[f \lhd F]$; and
3. if $E \subseteq \mathrm{Sec}(e)$ is a set s.t. $g \in E \Rightarrow g > f$ then $F \notin \bigcup_{g \in E} \top_{\rho'}(g)$.

Example on board: Illustrate $\overline{\mathrm{Sec}}(e_4)$ and $\overline{\mathrm{Sec}}(e_7)$.

effective construction: combining ASIs

PartialStates(e)

effective construction: combining ASIs

PartialStates(e)

1. "Combine" all $\overline{\mathrm{Sec}}(f_p), f_p \lessdot_p e$

effective construction: combining ASIs

PartialStates(e)

1. "Combine" all $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$
2. `return` ι-$\overline{\mathsf{Sec}}(e)$

effective construction: combining ASIs

PartialStates(e)

1. "Combine" all $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$
2. `return` ι-$\overline{\mathsf{Sec}}(e)$

Invariant: Generality of ι-$\overline{\mathsf{Sec}}(e)$

For each event $\iota\text{-}\overline{f} = (f, \iota\text{-}\Pi, \pi) \in \iota\text{-}\overline{\mathsf{Sec}}(e)$, there exists a partial state $\pi' \in \iota\text{-}\Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier F in ρ' such that

1. $F = \top_{\rho'}(f)$; and
2. $\pi' = \rho'[f \lhd F]$.

effective construction: combining ASIs

PartialStates(e)

1. "Combine" all $\overline{\mathsf{Sec}}(f_p), f_p \lessdot_p e$
2. `return` ι-$\overline{\mathsf{Sec}}(e)$

Invariant: Generality of ι-$\overline{\mathsf{Sec}}(e)$

For each event $\iota\text{-}\overline{f} = (f, \iota\text{-}\Pi, \pi) \in \iota\text{-}\overline{\mathsf{Sec}}(e)$, there exists a partial state $\pi' \in \iota\text{-}\Pi$ iff there exists a prefix $\rho' \sqsubseteq \rho[e]$ and a partial frontier F in ρ' such that

1. $F = \top_{\rho'}(f)$; and
2. $\pi' = \rho'[f \lhd F]$.

Recall Lemma 5, and for minimal events $\{\iota\text{-}\overline{e}_1, \ldots, \iota\text{-}\overline{e}_\ell\}$, compute the set

$$Y_e := \{\pi \in \bigotimes_{i=1}^{\ell} (\pi_i \lhd \iota\text{-}\Pi_i) \mid \mathsf{dom}(\pi) = \mathsf{ds}(e)\}.$$

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}_p} \times X_Q$, where sets Q are such that $p \in Q$;

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}_p} \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}_p} \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}_p}, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}_e}, Y_e)$, where

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}_p} \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}_p}, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}_e}, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathrm{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}_p} \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}_p}, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}_e}, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathrm{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;
  - $\overline{\mathrm{Sec}_e}$ is obtained from $\textsc{SecondaryUpdate}(e)$; and

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}}_p \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}}_p, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}}_e, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathrm{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;
  - $\overline{\mathrm{Sec}}_e$ is obtained from $\textsc{SecondaryUpdate}(e)$; and
  - Y_e is obtained as a consequence of $\textsc{PartialStates}(e)$ procedure.

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}}_p \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}}_p, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}}_e, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathrm{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;
  - $\overline{\mathrm{Sec}}_e$ is obtained from $\textsc{SecondaryUpdate}(e)$; and
  - Y_e is obtained as a consequence of $\textsc{PartialStates}(e)$ procedure.
- For each local p-state $\overline{x} = (y, \overline{\mathrm{Sec}}, Y) \in \overline{X}_p$, if for any $\pi \in Y$, $\mathrm{dom}(\pi) = Q$, then assign $\mathcal{D}_p(\overline{x}) = Q$.

equivalent SATS

Starting from an ATS $\mathfrak{T} = ((X_p)_{p\in\mathcal{P}}, (\delta_a)_{a\in\Sigma}, \pi_0)$, construct an ATS $\overline{\mathfrak{T}} = ((\overline{X}_p)_{p\in\mathcal{P}}, (\overline{\delta}_a)_{a\in\Sigma}, \overline{\pi}_0)$ where:

- $\overline{X}_p \subseteq \bigcup_{Q\subseteq\mathcal{P}} X_p \times \overline{\mathrm{SEC}}_p \times X_Q$, where sets Q are such that $p \in Q$;
- $\overline{\pi}_{0|p} = (x_0, \{(e_\perp, \{\pi_0\}, \pi_0)\}, \{\pi_0\})$ where $x_0 = \pi_{0|p}$ is the initial p-state of $\mathfrak{T}$;
- for $a \in \Sigma$, $\overline{\pi} = (x_p, \overline{\mathrm{Sec}}_p, Y_p)_{p\in\mathrm{dom}(a)} \in \overline{X}_{\mathrm{dom}(a)}$, and $q \in \mathrm{dom}(a)$, define $\overline{\delta}_a(\overline{\pi})_{|q} := (y_q, \overline{\mathrm{Sec}}_e, Y_e)$, where
  - $y_q = \delta_a((x_p)_{p\in\mathrm{dom}(a)})_{|q}$ is obtained from $\mathfrak{T}$;
  - $\overline{\mathrm{Sec}}_e$ is obtained from $\textsc{SecondaryUpdate}(e)$; and
  - Y_e is obtained as a consequence of $\textsc{PartialStates}(e)$ procedure.
- For each local p-state $\overline{x} = (y, \overline{\mathrm{Sec}}, Y) \in \overline{X}_p$, if for any $\pi \in Y$, $\mathrm{dom}(\pi) = Q$, then assign $\mathcal{D}_p(\overline{x}) = Q$.

$(\overline{\mathfrak{T}}, \mathcal{D})$ is the SATS we need.

finally, the D-SABA

`for each` partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ `and each` global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

`done`

finally, the D-SABA

`for each` partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ `and each` global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

`for each` part $P_{i,j},\ 1 \leq j \leq n_i$

`done`

`done`

finally, the D-SABA

for each partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ and each global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

    for each part $P_{i,j},\ 1 \leq j \leq n_i$

        for each processes $q \in P_{i,j}$

        done

    done

done

finally, the D-SABA

for each partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ and each global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

for each part $P_{i,j},\ 1 \leq j \leq n_i$

for each processes $q \in P_{i,j}$

- ★ construct the set $\overline{F}^q_{i,k} = \{(x, \overline{\mathrm{Sec}}, Y) \in \overline{X}_q \mid q \in P_{i,j} \text{ and } \pi_{k|P_{i,j}} \in Y\}$

done

done

done

finally, the D-SABA

for each partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ and each global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

for each part $P_{i,j},\ 1 \leq j \leq n_i$

for each processes $q \in P_{i,j}$

- ⋆ construct the set $\overline{F}^q_{i,k} = \{(x, \overline{\mathsf{Sec}}, Y) \in \overline{X}_q \mid q \in P_{i,j} \text{ and } \pi_{k|P_{i,j}} \in Y\}$

done

done

done

$$\text{Define } \mathcal{F} := \bigcup_{i,k} \{(\overline{F}^q_{i,k})_{q \in \mathcal{P}}\}$$

finally, the D-SABA

for each partition $\Psi_i = \{P_{i,1}, \ldots P_{i,n_i}\}$ of $\mathcal{P}$ and each global accepting state $\pi_k \in F$ of the FAA $\mathfrak{A}$

for each part $P_{i,j},\ 1 \leq j \leq n_i$

for each processes $q \in P_{i,j}$

- ⋆ construct the set $\overline{F}^q_{i,k} = \{(x, \overline{\mathrm{Sec}}, Y) \in \overline{X}_q \mid q \in P_{i,j} \text{ and } \pi_{k|P_{i,j}} \in Y\}$

done

done

done

$$\text{Define } \mathcal{F} := \bigcup_{i,k} \{(\overline{F}^q_{i,k})_{q \in \mathcal{P}}\}$$

Claim

$\overline{\mathfrak{A}} := (\overline{\mathfrak{T}}, \mathcal{D}, \mathcal{F})$ is a deterministic, synchronization-aware Büchi automaton that recognizes the language $\Theta = \lim(T)$.

are we there yet?

The reverse direction of the theorem.

are we there yet?

The reverse direction of the theorem.

Claim

Let $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$ be a D-SABA recognizing the language $\Theta \subseteq \mathbb{R}(\Sigma, I)$, we can construct an NFAA $\overline{\mathfrak{A}} = ((\overline{X}_p)_{p \in \mathcal{P}}, (\Delta_a)_{a \in \Sigma}, \overline{\pi}_0, F)$.

deterministic, synchronization-aware Muller automata

A **D-SAMA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

deterministic, synchronization-aware Muller automata

A **D-SAMA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

A D-SAMA $\mathfrak{A}$ **accepts a trace** θ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process $p \in \mathcal{P}$, $F_i^p = \lceil \mathrm{Inf}_p(\rho) \rceil$.

deterministic, synchronization-aware Muller automata

A **D-SAMA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

A D-SAMA $\mathfrak{A}$ **accepts a trace** θ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process $p \in \mathcal{P}$, $F_i^p = \lceil \mathsf{Inf}_p(\rho) \rceil$.

Theorem (ω-regular trace languages)

Any language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces is recognized by a D-SAMA if and only if Θ is recognized by a DAMA (as defined in the literature).

deterministic, synchronization-aware Muller automata

A **D-SAMA** is a tuple $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$, where $(\mathfrak{T}, \mathcal{D})$ is an SATS, and the acceptance condition is given as a table $\mathcal{F} = \{F_1, F_2, \dots\}$ such that

- each $F_i = (F_i^p)_{p \in \mathcal{P}}$ is a tuple of subsets of local states of the processes; and
- for all i, for all $x, y \in F_i^p$, $\mathcal{D}_p(x) = \mathcal{D}_p(y)$.

A D-SAMA $\mathfrak{A}$ **accepts a trace** θ if, for the run ρ of $\mathfrak{A}$ on θ, there exists a tuple $F_i \in \mathcal{F}$ such that for each process $p \in \mathcal{P}$, $F_i^p = \lceil \mathsf{Inf}_p(\rho) \rceil$.

Theorem (ω-regular trace languages)

Any language $\Theta \subseteq \mathbb{R}(\Sigma, I)$ of infinite traces is recognized by a D-SAMA if and only if Θ is recognized by a DAMA (as defined in the literature).

Lemma (Closure properties)

The family of deterministic, synchronization-aware Muller automata is closed under finite Boolean operations.

conclusion

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$
  - Direct, automata based construction can be applied to word automata, i.e. DFA to D-SABA, instead of DFA to FAA to D-SABA

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$
  - Direct, automata based construction can be applied to word automata, i.e. DFA to D-SABA, instead of DFA to FAA to D-SABA
  - Augmenting secondary information with different memory structures is a useful generalization

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$
  - Direct, automata based construction can be applied to word automata, i.e. DFA to D-SABA, instead of DFA to FAA to D-SABA
  - Augmenting secondary information with different memory structures is a useful generalization
- Equivalents of Landweber's and McNaughton's theorems still open

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$
  - Direct, automata based construction can be applied to word automata, i.e. DFA to D-SABA, instead of DFA to FAA to D-SABA
  - Augmenting secondary information with different memory structures is a useful generalization
- Equivalents of Landweber's and McNaughton's theorems still open
- Utility in studying families of "weak" asynchronous automata and corresponding families of languages?

conclusion

- Introduced the family of synchronization-aware asynchronous transition system, with several nice properties.
  - First model for languages of the form $\lim(T)$
  - Direct, automata based construction can be applied to word automata, i.e. DFA to D-SABA, instead of DFA to FAA to D-SABA
  - Augmenting secondary information with different memory structures is a useful generalization
- Equivalents of Landweber's and McNaughton's theorems still open
- Utility in studying families of "weak" asynchronous automata and corresponding families of languages?
- Utility in distributed synthesis from ω-regular trace specifications?