

# Toward a structure theory of $\omega$ -regular trace languages

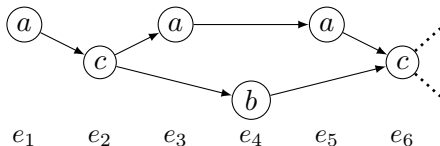
Namit Chaturvedi



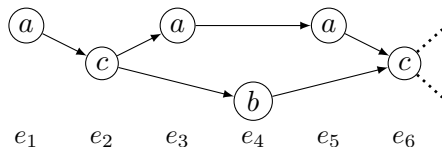
Frontiers of Recognizability

29 Apr. 2014

Models of concurrent executions of distributed systems



Models of concurrent executions of distributed systems

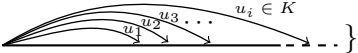


Automata recognizing languages of traces

- 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 $\omega$ -regular (word) languages

# regular to $\omega$ -regular (word) languages

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

## regular to $\omega$ -regular (word) languages

- $\lim(K) := \{ \alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha = \underbrace{\hspace{10em}}_{u_i \in K} \}$

### Theorem (det. Büchi automata)

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 $\omega$ -regular (word) languages

- $\lim(K) := \{ \alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha = \underbrace{\hspace{10em}}_{u_i \in K} \}$

### Theorem (det. Büchi automata)

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 (L. Landweber)

A language  $L \subseteq \Sigma^\omega$  is DBA recognizable iff for each Muller automaton recognizing  $L$  the acceptance component is closed under supersets.

## regular to $\omega$ -regular (word) languages

- $\lim(K) := \{ \alpha \in \Sigma^\omega \mid \exists^\infty i \in \mathbb{N}, \alpha = \underbrace{\hspace{10em}}_{u_i \in K} \}$

### Theorem (det. Büchi automata)

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 (L. Landweber)

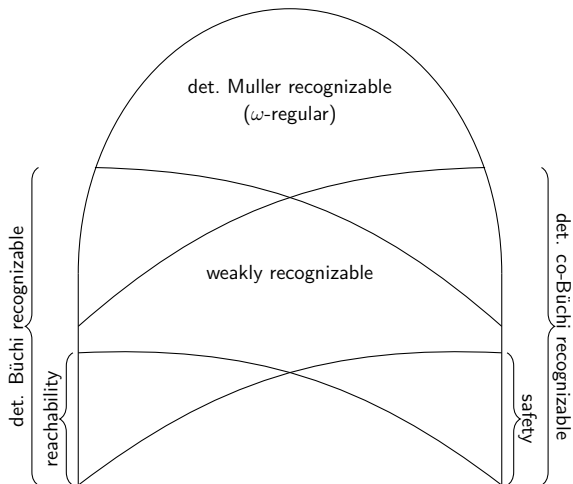
A language  $L \subseteq \Sigma^\omega$  is DBA recognizable iff for each Muller automaton recognizing  $L$  the acceptance component is closed under supersets.

### Theorem (R. Büchi, R. McNaughton)

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



# the Borel classification of $\omega$ -regular languages



# fundamental questions

- FAA recognizing  $T$  to DABA recognizing “ $\lim(T)$ ”?
- Relation between DABA and DAMA as in Landweber’s theorem?
- Is the class of  $\omega$ -regular trace languages same as the class of Boolean combinations of DABA recognizable languages?

## preliminaries: finite and infinite traces

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

## preliminaries: finite and infinite traces

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

### Finite and infinite traces

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

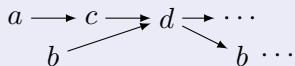


## preliminaries: finite and infinite traces

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

### Finite and infinite traces

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

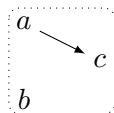
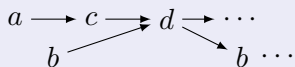


## preliminaries: finite and infinite traces

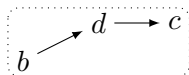
Let  $I \subseteq \Sigma^2$  be a symmetric, irreflexive **independence relation**, then  $(\Sigma, I)$  is called a **independence alphabet**.

### Finite and infinite traces

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



$aIb; cIb$

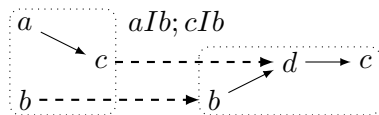
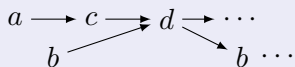


## preliminaries: finite and infinite traces

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

### Finite and infinite traces

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



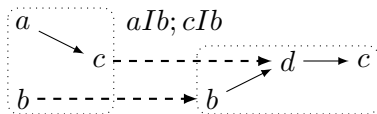
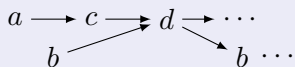
$t_1 \odot t_2$  similarly  $t \odot \theta$ ,  
and thus  $t \sqsubseteq t \odot \theta$

# preliminaries: finite and infinite traces

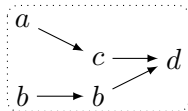
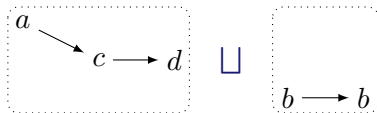
Let  $I \subseteq \Sigma^2$  be a symmetric, irreflexive **independence relation**, then  $(\Sigma, I)$  is called a **independence alphabet**.

## Finite and infinite traces

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



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



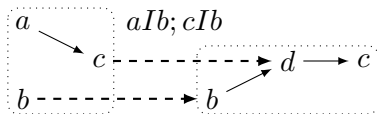
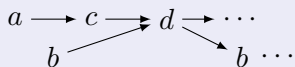


# preliminaries: finite and infinite traces

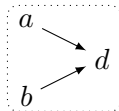
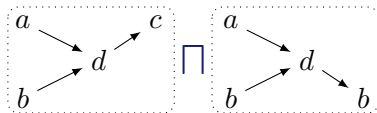
Let  $I \subseteq \Sigma^2$  be a symmetric, irreflexive **independence relation**, then  $(\Sigma, I)$  is called a **independence alphabet**.

## Finite and infinite traces

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



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



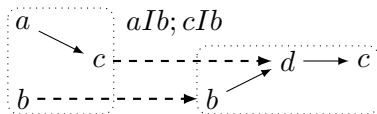
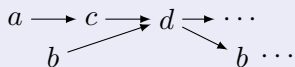
$t_1 \sqcap t_2$

# preliminaries: finite and infinite traces

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

## Finite and infinite traces

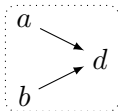
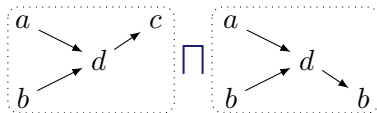
A **trace**  $t = [V, \prec, \lambda]$  or  $\theta = [V, \prec, \lambda]$



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

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

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



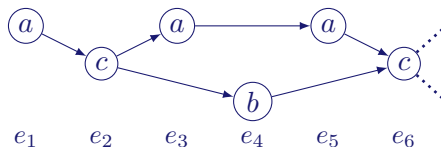
$t_1 \sqcap t_2$

## preliminaries: asynchronous transitions systems

Fix an independence alphabet  $(\Sigma, I)$ .

# preliminaries: asynchronous transitions systems

Fix an independence alphabet  $(\Sigma, I)$ .

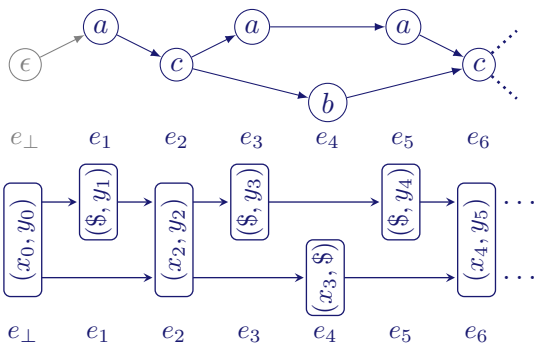


**Trace**

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

# preliminaries: asynchronous transitions systems

Fix an independence alphabet  $(\Sigma, I)$ .



**Trace**

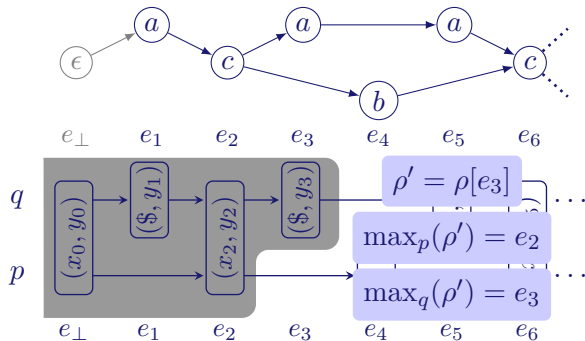
$$t/\theta = [V, \triangleleft, \lambda]$$

**Corresponding\* run**

$$\rho = [V', (\triangleleft_p)_{p \in \mathcal{P}}, \lambda', \Lambda]$$

# preliminaries: asynchronous transitions systems

Fix an independence alphabet  $(\Sigma, I)$ .



**Trace**

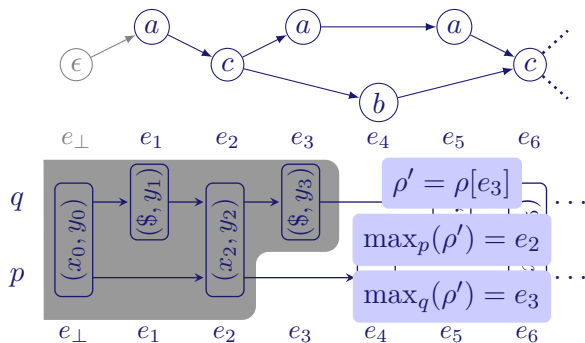
$$t/\theta = [V, \triangleleft, \lambda]$$

**Corresponding\* run**

$$\rho = [V', (\triangleleft_p)_{p \in \mathcal{P}}, \lambda', \Lambda]$$

# preliminaries: asynchronous transitions systems

Fix an independence alphabet  $(\Sigma, I)$ .



**Trace**

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

**Corresponding\* run**

$$\rho = [V', (\triangleleft_p)_{p \in \mathcal{P}}, \lambda', \Lambda]$$

Formally, given  $(\Sigma, I)$  and a mapping  $\text{dom}: \Sigma \rightarrow 2^{\mathcal{P}}$ , an **asynchronous transition system** (an **ATS**) is a tuple  $\mathfrak{T} = ((X_p)_{p \in \mathcal{P}}, (\delta_a)_{a \in \Sigma}, \pi_0)$ .

- Local infinity sets



- Local infinity sets

$$\text{Inf}_p(\rho) := \begin{cases} \{x \in X_p \mid x \text{ occurs inf. often in } \rho\} & \text{if } p \text{ never stops} \\ \{x \in X_p \mid x \text{ is the last } p\text{-state in } \rho\} & \text{otherwise} \end{cases}$$

- Local infinity sets

$$\text{Inf}_p(\rho) := \begin{cases} \{x \in X_p \mid x \text{ occurs inf. often in } \rho\} & \text{if } p \text{ never stops} \\ \{x \in X_p \mid x \text{ is the last } p\text{-state in } \rho\} & \text{otherwise} \end{cases}$$

- Acceptance table  $\mathcal{F} = \{F_1, \dots, F_k\}$  with

$$F_i = (F_i^p)_{p \in \mathcal{P}}$$

	$p$	$q$	$\dots$
$F_1$	$F_1^p$	$F_1^q$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\dots$
$F_k$	$F_k^p$	$F_k^q$	$\dots$

- Local infinity sets

$$\text{Inf}_p(\rho) := \begin{cases} \{x \in X_p \mid x \text{ occurs inf. often in } \rho\} & \text{if } p \text{ never stops} \\ \{x \in X_p \mid x \text{ is the last } p\text{-state in } \rho\} & \text{otherwise} \end{cases}$$

- Acceptance table  $\mathcal{F} = \{F_1, \dots, F_k\}$  with  $F_i = (F_i^p)_{p \in \mathcal{P}}$
- For DABA<sup>‡</sup>  $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$ ,  $\rho$  is an accepting run if  $\exists F_i \in \mathcal{F}, \forall p \in \mathcal{P} : F_i^p \subseteq \text{Inf}_p(\rho)$

	$p$	$q$	$\dots$
$F_1$	$F_1^p$	$F_1^q$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\dots$
$F_k$	$F_k^p$	$F_k^q$	$\dots$

<sup>‡</sup>Due to V. Diekert & A. Muscholl

# preliminaries: classical asynchronous automata

- Local infinity sets

$$\text{Inf}_p(\rho) := \begin{cases} \{x \in X_p \mid x \text{ occurs inf. often in } \rho\} & \text{if } p \text{ never stops} \\ \{x \in X_p \mid x \text{ is the last } p\text{-state in } \rho\} & \text{otherwise} \end{cases}$$

- Acceptance table  $\mathcal{F} = \{F_1, \dots, F_k\}$  with  $F_i = (F_i^p)_{p \in \mathcal{P}}$
- For DABA<sup>‡</sup>  $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$ ,  $\rho$  is an accepting run if  $\exists F_i \in \mathcal{F}, \forall p \in \mathcal{P} : F_i^p \subseteq \text{Inf}_p(\rho)$
- For DAMA<sup>‡</sup>  $\mathfrak{A} = (\mathfrak{T}, \mathcal{F})$ ,  $\rho$  is an accepting run if  $\exists F_i \in \mathcal{F}, \forall p \in \mathcal{P} : F_i^p = \text{Inf}_p(\rho)$

	$p$	$q$	$\dots$
$F_1$	$F_1^p$	$F_1^q$	$\dots$
$\vdots$	$\vdots$	$\vdots$	$\dots$
$F_k$	$F_k^p$	$F_k^q$	$\dots$

<sup>‡</sup>Due to V. Diekert & A. Muscholl

# preliminaries: $\omega$ -regular trace languages

## preliminaries: $\omega$ -regular trace languages

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

## preliminaries: $\omega$ -regular trace languages

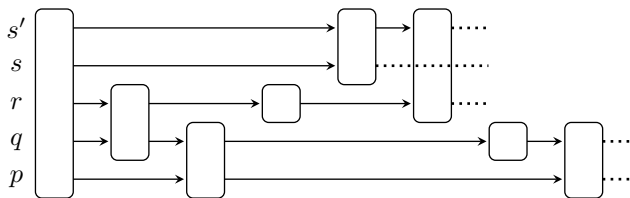
A language  $\Theta \in \mathbb{R}(\Sigma, I)$  is called an  **$\omega$ -regular trace language** if there exists a DAMA recognizing it.

For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -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: $\omega$ -regular trace languages

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

For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -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  $\bigsqcup_{i \in \mathbb{N}} t_i = \theta$ .

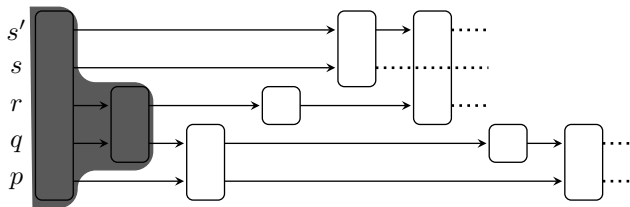




## preliminaries: $\omega$ -regular trace languages

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

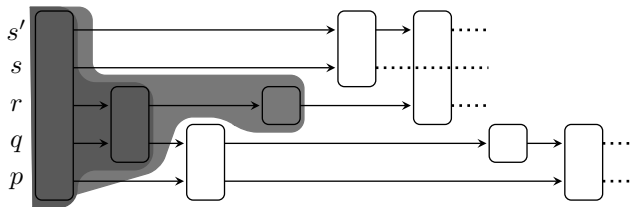
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

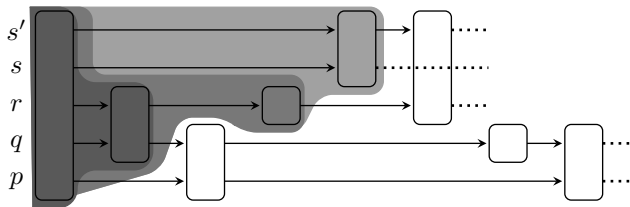
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

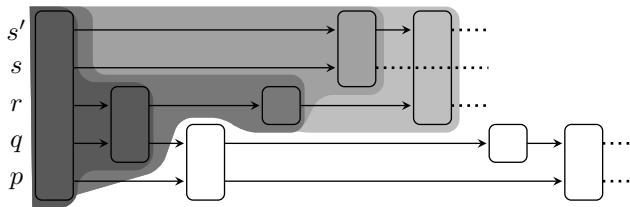
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

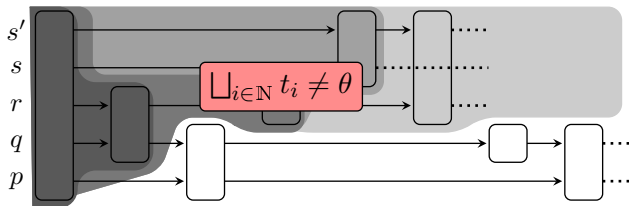
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

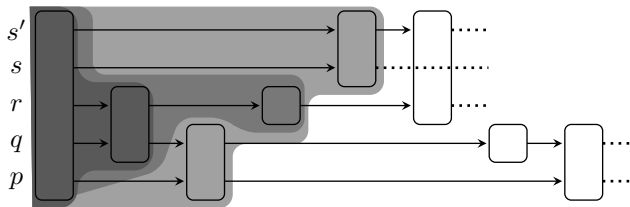
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

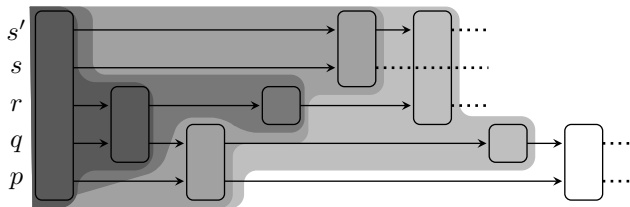
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

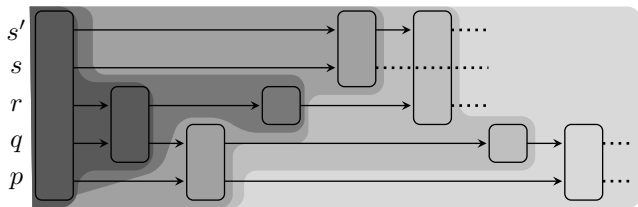
For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -trace language containing all  $\theta \in \mathbb{R}(\Sigma, I)$  such that there exists a sequence  $(t_i)_{i \in \mathbb{N}}, t_i \in T$  satisfying  $t_i \sqsubset t_{i+1}$  and  $\bigsqcup_{i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



## preliminaries: $\omega$ -regular trace languages

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

For  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , its **infinitary limit**  $\lim(T)$  is the  $\omega$ -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: deterministic trace languages

Given  $A \subseteq \Sigma$  and  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , the  **$A$ -infinitary limit of  $T$**  is defined as  $\lim_A(T) := \{\theta \in \text{lim}(T) \mid D(\text{alphinf}(\theta)) = D(A)\}$ .

## preliminaries: deterministic trace languages

Given  $A \subseteq \Sigma$  and  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , the  **$A$ -infinitary limit of  $T$**  is defined as  $\lim_A(T) := \{\theta \in \lim(T) \mid D(\text{alphinf}(\theta)) = D(A)\}$ .

A language  $\Theta \subseteq \mathbb{R}(\Sigma, I)$  is called a **deterministic trace language** if  $\Theta$  can be expressed as a finite union  $\bigcup_i \lim_{A_i}(T_i)$  for regular trace languages  $T_i \subseteq \mathbb{M}(\Sigma, I)$  and  $A_i \subseteq \Sigma$ .

## preliminaries: deterministic trace languages

Given  $A \subseteq \Sigma$  and  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , the  **$A$ -infinitary limit of  $T$**  is defined as  $\lim_A(T) := \{\theta \in \lim(T) \mid D(\text{alphinf}(\theta)) = D(A)\}$ .

A language  $\Theta \subseteq \mathbb{R}(\Sigma, I)$  is called a **deterministic trace language** if  $\Theta$  can be expressed as a finite union  $\bigcup_i \lim_{A_i}(T_i)$  for regular trace languages  $T_i \subseteq \mathbb{M}(\Sigma, I)$  and  $A_i \subseteq \Sigma$ .

### Theorem (V. Diekert & A. Muscholl)

*Every  $\omega$ -regular trace language can be expressed as a finite Boolean combination of deterministic trace languages.*

## preliminaries: deterministic trace languages

Given  $A \subseteq \Sigma$  and  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$ , the  **$A$ -infinitary limit of  $T$**  is defined as  $\lim_A(T) := \{\theta \in \lim(T) \mid D(\text{alphinf}(\theta)) = D(A)\}$ .

A language  $\Theta \subseteq \mathbb{R}(\Sigma, I)$  is called a **deterministic trace language** if  $\Theta$  can be expressed as a finite union  $\bigcup_i \lim_{A_i}(T_i)$  for regular trace languages  $T_i \subseteq \mathbb{M}(\Sigma, I)$  and  $A_i \subseteq \Sigma$ .

### Theorem (V. Diekert & A. Muscholl)

*Every  $\omega$ -regular trace language can be expressed as a finite Boolean combination of deterministic trace languages.*

### The challenge

The class of deterministic trace languages has no equivalent among the known classes of deterministic asynchronous automata.

# motivation

State of fundamental questions

# motivation

State of fundamental questions

- From FAA recognizing  $T$  to DABA recognizing  $\lim(T)$ : **negative**

## State of fundamental questions

- From FAA recognizing  $T$  to DABA recognizing  $\lim(T)$ : **negative**
  - ▶ There exists  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$  such that no DABA recognizes  $\lim(T)$
  - ▶ In particular, FAA cannot be exploited as DABA

## State of fundamental questions

- From FAA recognizing  $T$  to DABA recognizing  $\lim(T)$ : **negative**
  - ▶ There exists  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$  such that no DABA recognizes  $\lim(T)$
  - ▶ In particular, FAA cannot be exploited as DABA

$$F = \{(x_1, y_1)\}$$

$$\mathcal{F} = \left( (\{x_1\}, \{y_1\}) \right)$$

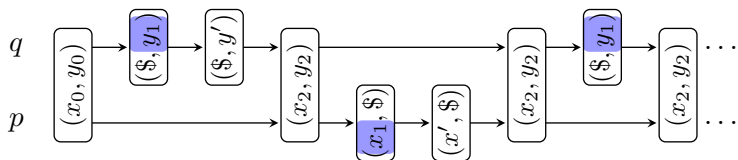


# motivation

## State of fundamental questions

- From FAA recognizing  $T$  to DABA recognizing  $\lim(T)$ : **negative**
  - ▶ There exists  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$  such that no DABA recognizes  $\lim(T)$
  - ▶ In particular, FAA cannot be exploited as DABA

$$F = \{(x_1, y_1)\}$$
$$\mathcal{F} = \left( (\{x_1\}, \{y_1\}) \right)$$

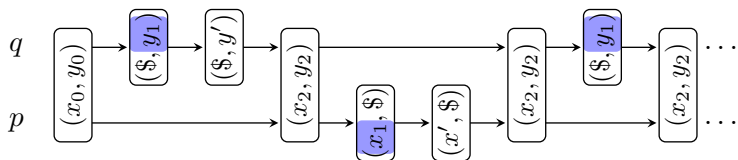


# motivation

## State of fundamental questions

- From FAA recognizing  $T$  to DABA recognizing  $\lim(T)$ : **negative**
  - There exists  $T \in \text{Rec}(\mathbb{M}(\Sigma, I))$  such that no DABA recognizes  $\lim(T)$
  - In particular, FAA cannot be exploited as DABA

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



- DABA and DAMA vs. Landweber's characterization: **open**
- $\omega$ -regular trace languages (or DAMA recognizable languages) vs. Boolean combinations of DABA recognizable languages: **open**

## summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

## summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

- “Synchronization-aware Büchi automata” for det. trace languages.

# summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

- “Synchronization-aware Büchi automata” for det. trace languages.
- “Synchronization-aware Muller automata” for  $\omega$ -regular trace languages.

# summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

- “Synchronization-aware Büchi automata” for det. trace languages.
- “Synchronization-aware Muller automata” for  $\omega$ -regular trace languages.
- Characterization of det. languages in terms of recognition by a special subset of Muller automata (à la Landweber).

# summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

- “Synchronization-aware Büchi automata” for det. trace languages.
- “Synchronization-aware Muller automata” for  $\omega$ -regular trace languages.
- Characterization of det. languages in terms of recognition by a special subset of Muller automata (à la Landweber).
  - ▶ And hence an automata-theoretic proof for:  
 $\text{REG}(\mathbb{R}(\Sigma, I)) = \text{BC}(\text{det. trace languages})$ .

# summary of results

A Borel-like hierarchy of  $\omega$ -regular trace languages:

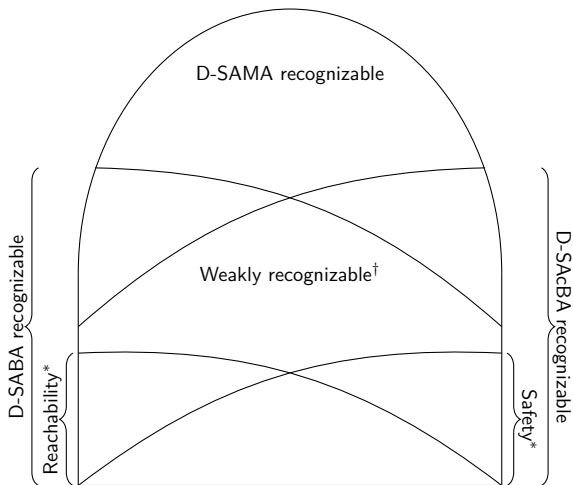
- “Synchronization-aware Büchi automata” for det. trace languages.
- “Synchronization-aware Muller automata” for  $\omega$ -regular trace languages.
- Characterization of det. languages in terms of recognition by a special subset of Muller automata (à la Landweber).
  - ▶ And hence an automata-theoretic proof for:  
 $\text{REG}(\mathbb{R}(\Sigma, I)) = \text{BC}(\text{det. trace languages})$ .

Work in progress:

- “Weak automata” – recognizing languages that are both det. Büchi and det. co-Büchi recognizable.
- Characterization of weak languages in terms of recognition by a special subset of Muller automata.



# summary of results. . . in a picture



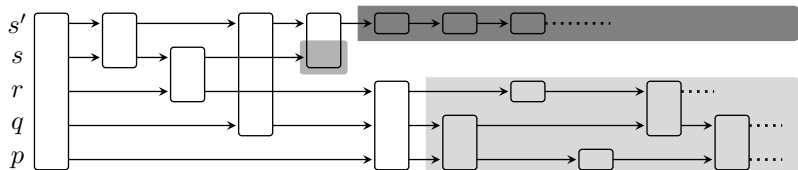
<sup>†</sup> Work in progress

\* Impossible to characterize in terms of asynchronous automata

goal: awareness of infinitary behavior

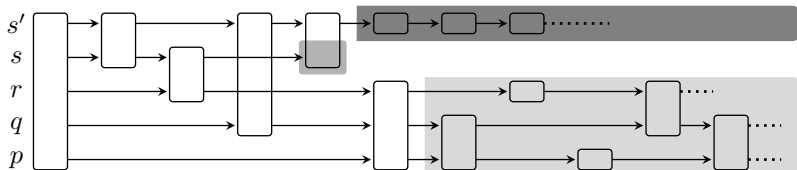
# goal: awareness of infinitary behavior

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



## goal: awareness of infinitary behavior

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

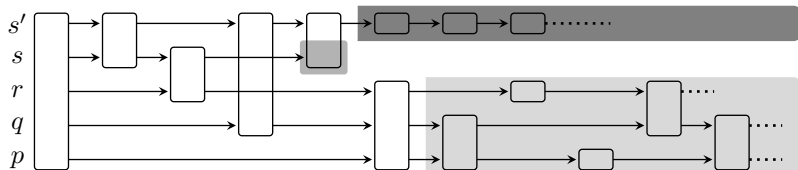


### The “synchronization-awareness” problem

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

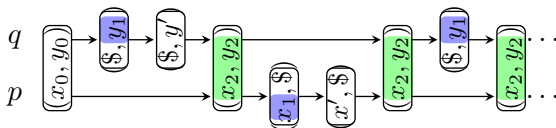
# goal: awareness of infinitary behavior

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

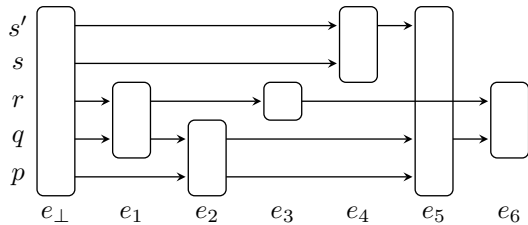


## The “synchronization-awareness” problem

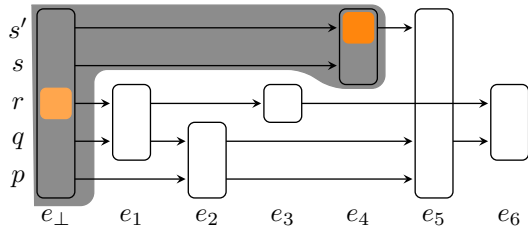
How can each process  $p \in \mathcal{P}$  infer the minimal part  $P_i \in \Psi$  containing all the processes with whom it interacts infinitely often?



# interactions and gossip



# interactions and gossip

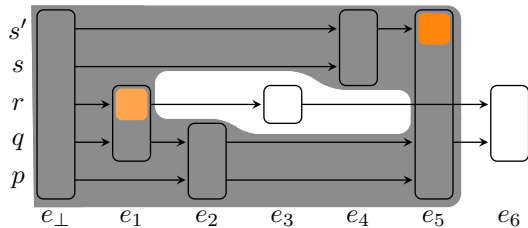


First hand information

$$\text{latest}_{s' \rightarrow r}(\rho') = e_{\perp}$$

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

# interactions and gossip



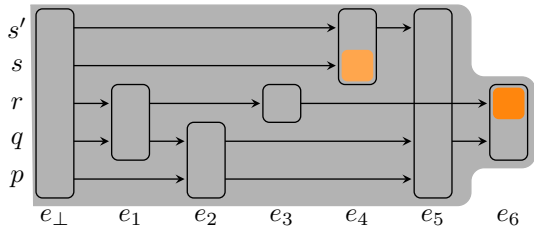
First hand information

$$\text{latest}_{s' \rightarrow r}(\rho') = e_1$$

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



# interactions and gossip

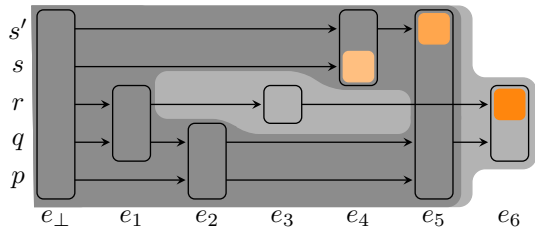


First hand information

$$\text{latest}_{r \rightarrow s}(\rho') = e_4$$

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

# interactions and gossip

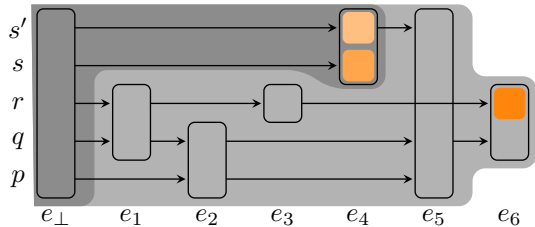


Second hand information

$$\text{latest}_{r \rightarrow s' \rightarrow s}(\rho') = e_4$$

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

# interactions and gossip

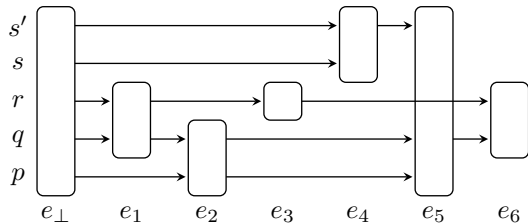


Second hand information

$$\text{latest}_{r \rightarrow s \rightarrow s'}(\rho') = e_4$$

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

# interactions and gossip



The **future** of  $e$  in  $\rho$ ,  $\text{future}_{\rho}(e) := \{f \in \rho \mid f \in F_{\rho} \wedge e \leq f\}$ .

## degrees of synchronization: definition

The update  $\mathcal{U}_e^p$  in the information of process  $p$  at  $e \in \rho$  is given by

$$\mathcal{U}_e^p := \{g \in \rho[e] \mid \text{for } f_p \prec_p e : g = \text{latest}_{p \rightarrow q \rightarrow r}(f_p) \neq \text{latest}_{p \rightarrow q \rightarrow r}(e)\}$$

## degrees of synchronization: definition

The update  $\mathcal{U}_e^p$  in the information of process  $p$  at  $e \in \rho$  is given by  
$$\mathcal{U}_e^p := \{g \in \rho[e] \mid \text{for } f_p \prec_p e : g = \text{latest}_{p \rightarrow q \rightarrow r}(f_p) \neq \text{latest}_{p \rightarrow q \rightarrow r}(e)\}$$

The **information update at**  $e \in \rho$  is the set  $\mathcal{U}_e := \bigcup_{p \in \text{dom}(e)} \mathcal{U}_e^p$ .

## degrees of synchronization: definition

The update  $\mathcal{U}_e^p$  in the information of process  $p$  at  $e \in \rho$  is given by 
$$\mathcal{U}_e^p := \{g \in \rho[e] \mid \text{for } f_p \prec_p e : g = \text{latest}_{p \rightarrow q \rightarrow r}(f_p) \neq \text{latest}_{p \rightarrow q \rightarrow r}(e)\}$$

The **information update at**  $e \in \rho$  is the set  $\mathcal{U}_e := \bigcup_{p \in \text{dom}(e)} \mathcal{U}_e^p$ .

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

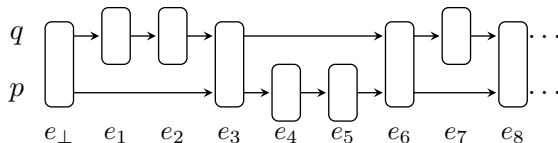
## degrees of synchronization: definition

The update  $\mathcal{U}_e^p$  in the information of process  $p$  at  $e \in \rho$  is given by  $\mathcal{U}_e^p := \{g \in \rho[e] \mid \text{for } f_p \prec_p e : g = \text{latest}_{p \rightarrow q \rightarrow r}(f_p) \neq \text{latest}_{p \rightarrow q \rightarrow r}(e)\}$

The **information update at**  $e \in \rho$  is the set  $\mathcal{U}_e := \bigcup_{p \in \text{dom}(e)} \mathcal{U}_e^p$ .

The **degree of synchronization at**  $e \in \rho$  is

$\text{ds}(e) := \bigcup_{g \in \mathcal{U}_e} \text{dom}(\text{future}_{\rho[e]}(g))$ . By default,  $\text{ds}(e_{\perp}) := \mathcal{P}$ .





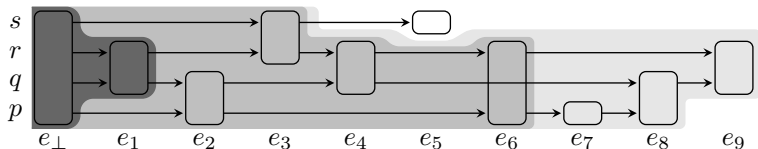
## degrees of synchronization: definition

The update  $\mathcal{U}_e^p$  in the information of process  $p$  at  $e \in \rho$  is given by 
$$\mathcal{U}_e^p := \{g \in \rho[e] \mid \text{for } f_p \prec_p e : g = \text{latest}_{p \rightarrow q \rightarrow r}(f_p) \neq \text{latest}_{p \rightarrow q \rightarrow r}(e)\}$$

The **information update at**  $e \in \rho$  is the set  $\mathcal{U}_e := \bigcup_{p \in \text{dom}(e)} \mathcal{U}_e^p$ .

The **degree of synchronization at**  $e \in \rho$  is

$\text{ds}(e) := \bigcup_{g \in \mathcal{U}_e} \text{dom}(\text{future}_{\rho[e]}(g))$ . By default,  $\text{ds}(e_{\perp}) := \mathcal{P}$ .



Example on board: compute  $\text{ds}(e_8) = \mathcal{P}$  and  $\text{ds}(e_9) = \{p, q, r\}$

### Lemma (infinitary behavior)

*Over an infinite run  $\rho$ , if a process  $p$  remains “live”, then there exists a **unique, maximal** set  $P \subseteq \mathcal{P}$  such that for infinitely many  $p$ -events  $e \in \rho : ds(e) = P$ .*

### Lemma (infinitary behavior)

Over an infinite run  $\rho$ , if a process  $p$  remains “live”, then there exists a **unique, maximal** set  $P \subseteq \mathcal{P}$  such that for infinitely many  $p$ -events  $e \in \rho : ds(e) = P$ .

Call the set  $P$  as mentioned above the **max-degree of  $p$ -synchronizations in  $\rho$** , denoted by  $\lceil ds_p(\rho) \rceil$ .  
And for  $p \notin \text{dom}(\text{alphinf}(\rho))$ , define  $\lceil ds_p(\rho) \rceil := \{p\}$ .

### Lemma (infinitary behavior)

Over an infinite run  $\rho$ , if a process  $p$  remains “live”, then there exists a **unique, maximal** set  $P \subseteq \mathcal{P}$  such that for infinitely many  $p$ -events  $e \in \rho : ds(e) = P$ .

Call the set  $P$  as mentioned above the **max-degree of  $p$ -synchronizations in  $\rho$** , denoted by  $\lceil ds_p(\rho) \rceil$ .  
And for  $p \notin \text{dom}(\text{alphinf}(\rho))$ , define  $\lceil ds_p(\rho) \rceil := \{p\}$ .

If a run  $\rho$  induces a partition  $\Psi$  of the set of states, then for each part  $P_i \in \Psi : q \in P_i \Leftrightarrow \lceil ds_q(\rho) \rceil = P_i$ .

## synchronization-aware transition systems

An **SATS** is a couple  $(\mathcal{T}, \mathcal{D})$  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  $\mathcal{D}_p: X_p \rightarrow 2^{\mathcal{P}}$  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  $\mathcal{D}_p: X_p \rightarrow 2^{\mathcal{P}}$  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  $\mathcal{D}_p: X_p \rightarrow 2^{\mathcal{P}}$  such that
  - ▶ for each  $p \in \mathcal{P}$ ,  $\mathcal{D}_p(\pi_0|_p) = \mathcal{P}$ , and
  - ▶ for every run  $\rho$  of  $\mathfrak{T}$  and every event  $e \in \rho$ , if  $\Lambda(e) = \pi$  and  $p \in \text{dom}(e)$  then  $\text{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi|_p) = P$

# 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  $\mathcal{D}_p: X_p \rightarrow 2^{\mathcal{P}}$  such that
  - ▶ for each  $p \in \mathcal{P}$ ,  $\mathcal{D}_p(\pi_0|_p) = \mathcal{P}$ , and
  - ▶ for every run  $\rho$  of  $\mathfrak{T}$  and every event  $e \in \rho$ , if  $\Lambda(e) = \pi$  and  $p \in \text{dom}(e)$  then  $\text{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi|_p) = P$

A set  $X \subseteq X_p$  of local  $p$ -states is called **homosynchronous** if for all local  $p$ -states  $x, y \in X$ :  $\mathcal{D}_p(x) = \mathcal{D}_p(y)$ .

# 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  $\mathcal{D}_p: X_p \rightarrow 2^{\mathcal{P}}$  such that
  - ▶ for each  $p \in \mathcal{P}$ ,  $\mathcal{D}_p(\pi_{0|p}) = \mathcal{P}$ , and
  - ▶ for every run  $\rho$  of  $\mathfrak{T}$  and every event  $e \in \rho$ , if  $\Lambda(e) = \pi$  and  $p \in \text{dom}(e)$  then  $\text{ds}(e) = P \Leftrightarrow \mathcal{D}_p(\pi|_p) = P$

A set  $X \subseteq X_p$  of local  $p$ -states is called **homosynchronous** if for all local  $p$ -states  $x, y \in X$ :  $\mathcal{D}_p(x) = \mathcal{D}_p(y)$ . For an infinite run  $\rho$ , processes  $p$  refer to the homosynchronous, **maximal local infinity sets**

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

# 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 table  $\mathcal{F} = \{(Q_1, F_1), \dots, (Q_k, F_k)\}$  is such that

- $Q_i \subseteq \mathcal{P}$ , and
- $F_i = (F_i^p)_{p \in \mathcal{P}}$  is a tuple of homosynchronous sets  $F_i^p$ .

# 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 table  $\mathcal{F} = \{(Q_1, F_1), \dots, (Q_k, F_k)\}$  is such that

- $Q_i \subseteq \mathcal{P}$ , and
- $F_i = (F_i^p)_{p \in \mathcal{P}}$  is a tuple of homosynchronous sets  $F_i^p$ .

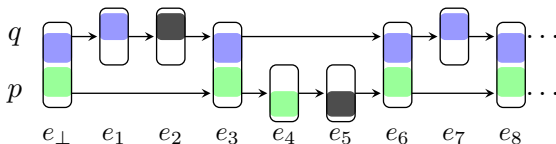
A D-SABA  $\mathfrak{A}$  **accepts a trace**  $\theta \in \mathbb{R}(\Sigma, I)$  if, for the run  $\rho$  of  $\mathfrak{A}$  on  $\theta$ , there exists a pair  $(Q_i, F_i) \in \mathcal{F}$  s.t.  $\text{dom}(\text{alphinf}(\theta)) = Q_i$  and for each process  $p \in \mathcal{P}$ :  $F_i^p \cap [\text{Inf}_p(\rho)] \neq \emptyset$ .

# 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 table  $\mathcal{F} = \{(Q_1, F_1), \dots, (Q_k, F_k)\}$  is such that

- $Q_i \subseteq \mathcal{P}$ , and
- $F_i = (F_i^p)_{p \in \mathcal{P}}$  is a tuple of homosynchronous sets  $F_i^p$ .

A D-SABA  $\mathfrak{A}$  **accepts a trace**  $\theta \in \mathbb{R}(\Sigma, I)$  if, for the run  $\rho$  of  $\mathfrak{A}$  on  $\theta$ , there exists a pair  $(Q_i, F_i) \in \mathcal{F}$  s.t.  $\text{dom}(\text{alphinf}(\theta)) = Q_i$  and for each process  $p \in \mathcal{P}$ :  $F_i^p \cap [\text{Inf}_p(\rho)] \neq \emptyset$ .



# 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 table  $\mathcal{F} = \{(Q_1, F_1), \dots, (Q_k, F_k)\}$  is such that

- $Q_i \subseteq \mathcal{P}$ , and
- $F_i = (F_i^p)_{p \in \mathcal{P}}$  is a tuple of homosynchronous sets  $F_i^p$ .

A D-SABA  $\mathfrak{A}$  **accepts a trace**  $\theta \in \mathbb{R}(\Sigma, I)$  if, for the run  $\rho$  of  $\mathfrak{A}$  on  $\theta$ , there exists a pair  $(Q_i, F_i) \in \mathcal{F}$  s.t.  $\text{dom}(\text{alphinf}(\theta)) = Q_i$  and for each process  $p \in \mathcal{P}$ :  $F_i^p \cap \lceil \text{Inf}_p(\rho) \rceil \neq \emptyset$ .

## Theorem (det. Büchi recognizable)

*A language  $\Theta \subseteq \mathbb{R}(\Sigma, I)$  is recognized by a D-SABA iff  $\Theta$  is a deterministic trace language, i.e. iff  $\Theta$  can be expressed as a finite union  $\bigcup_i \text{lim}_{A_i}(T_i)$  for regular trace languages  $T_i \subseteq \mathbb{M}(\Sigma, I)$  and  $A_i \subseteq \Sigma$ .*

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 table  $\mathcal{F} = \{F_1, \dots, F_k\}$  is such that  $F_i = (F_i^p)_{p \in \mathcal{P}}$  are tuples of homosynchronous sets  $F_i^p$ .



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 table  $\mathcal{F} = \{F_1, \dots, F_k\}$  is such that  $F_i = (F_i^p)_{p \in \mathcal{P}}$  are tuples of homosynchronous sets  $F_i^p$ .

A D-SAMA  $\mathfrak{A}$  **accepts a trace**  $\theta \in \mathbb{R}(\Sigma, I)$  if, for the run  $\rho$  of  $\mathfrak{A}$  on  $\theta$ , there exists a tuple  $F_i \in \mathcal{F}$  s.t. for each process  $p \in \mathcal{P}$ :  $[\text{Inf}_p(\rho)] = F_i^p$ .

# 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 table  $\mathcal{F} = \{F_1, \dots, F_k\}$  is such that  $F_i = (F_i^p)_{p \in \mathcal{P}}$  are tuples of homosynchronous sets  $F_i^p$ .

A D-SAMA  $\mathfrak{A}$  **accepts a trace**  $\theta \in \mathbb{R}(\Sigma, I)$  if, for the run  $\rho$  of  $\mathfrak{A}$  on  $\theta$ , there exists a tuple  $F_i \in \mathcal{F}$  s.t. for each process  $p \in \mathcal{P}$ :  $[\text{Inf}_p(\rho)] = F_i^p$ .

## Theorem ( $\omega$ -regular)

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

## closure under supersets

For tuples  $F_1 = (F_1^p)_{p \in \mathcal{P}}$ ,  $F_2 = (F_2^p)_{p \in \mathcal{P}}$  of homosynchronous sets, we say that  $F_1$  **is a superset of**  $F_2$  denoted  $F_1 \supseteq F_2$  if for each  $p \in \mathcal{P}$ ,  $F_1^p \supseteq F_2^p$ .

## closure under supersets

For tuples  $F_1 = (F_1^p)_{p \in \mathcal{P}}$ ,  $F_2 = (F_2^p)_{p \in \mathcal{P}}$  of homosynchronous sets, we say that  $F_1$  is a **superset of**  $F_2$  denoted  $F_1 \supseteq F_2$  if for each  $p \in \mathcal{P}$ ,  $F_1^p \supseteq F_2^p$ .

Table  $\mathcal{F}$  is **closed under supersets** if  $((F \in \mathcal{F}) \wedge (F' \supseteq F)) \Rightarrow (F' \in \mathcal{F})$

## closure under supersets

For tuples  $F_1 = (F_1^p)_{p \in \mathcal{P}}$ ,  $F_2 = (F_2^p)_{p \in \mathcal{P}}$  of homosynchronous sets, we say that  $F_1$  is a **superset of**  $F_2$  denoted  $F_1 \supseteq F_2$  if for each  $p \in \mathcal{P}$ ,  $F_1^p \supseteq F_2^p$ .

Table  $\mathcal{F}$  is **closed under supersets** if  $((F \in \mathcal{F}) \wedge (F' \supseteq F)) \Rightarrow (F' \in \mathcal{F})$ .

Table  $\mathcal{F}$  is said to be **closed under supersets modulo finitary acceptance sets** if

- (a) whenever a tuple  $F \in \mathcal{F}$  does not contain any “finitary acceptance sets” and there exists a tuple  $F' \supseteq F$ , then  $F' \in \mathcal{F}$ ; and
- (b) whenever a tuple  $F \in \mathcal{F}$  contains a “finitary acceptance set”  $F^p$  and there exists a tuple  $F' \supseteq F$  with  $F'^p = F^p$ , then  $F' \in \mathcal{F}$ .

# language characterization

## Theorem (L. Landweber)

*A language  $L \subseteq \Sigma^\omega$  is DBA recognizable iff for each Muller automaton recognizing  $L$  the acceptance component is closed under supersets.*

# language characterization

## Theorem (L. Landweber)

*A language  $L \subseteq \Sigma^\omega$  is DBA recognizable iff for each Muller automaton recognizing  $L$  the acceptance component is closed under supersets.*

## Theorem (characterization à la Landweber)

*A language  $\Theta$  is recognized by a D-SABA if and only if there exists a D-SAMA  $\mathfrak{A} = (\mathcal{T}, \mathcal{D}, \mathcal{F})$  recognizing  $\Theta$  whose acceptance table  $\mathcal{F}$  is closed under supersets modulo finitary acceptance sets.*

# language characterization

## Theorem (L. Landweber)

*A language  $L \subseteq \Sigma^\omega$  is DBA recognizable iff for each Muller automaton recognizing  $L$  the acceptance component is closed under supersets.*

## Theorem (characterization à la Landweber)

*A language  $\Theta$  is recognized by a D-SABA if and only if there exists a D-SAMA  $\mathfrak{A} = (\mathfrak{T}, \mathcal{D}, \mathcal{F})$  recognizing  $\Theta$  whose acceptance table  $\mathcal{F}$  is closed under supersets modulo finitary acceptance sets.*

## Theorem ( $\omega$ -regular = BC(det. Büchi recognizable))

*For any language  $\Theta \subseteq \mathbb{R}(\Sigma, I)$  of infinite traces,  $\Theta$  is D-SAMA recognizable if and only if  $\Theta$  can be expressed as a finite Boolean combination of D-SABA recognizable languages.*



# conclusion

- Generalizing the study of  $\omega$ -regular languages

- Generalizing the study of  $\omega$ -regular languages
  - ▶ A D-SABA accepts a language  $\Theta$  if and only if  $\Theta = \bigcup_{i=1}^n \lim_{A_i}(T_i)$
  - ▶ A characterization of det. trace languages in terms of a sub-class of det. Muller automata
  - ▶ An  $\omega$ -regular trace language can be expressed as a Boolean combination of D-SABA-recognizable languages

- Generalizing the study of  $\omega$ -regular languages

If the independence relation  $I = \emptyset$

- ▶ A D-SABA accepts a language  $\Theta$  if and only if  $\Theta = \bigcup_{i=1}^n \lim_{A_i}(T_i)$   
A DBA accepts a language  $L$  if and only if  $L = \bigcup_{i=1}^n \lim_{\Sigma}(K_i)$
- ▶ A characterization of det. trace languages in terms of a sub-class of det. Muller automata  
A characterization of det. languages in terms of a sub-class of det. Muller automata, leading to a decision process
- ▶ An  $\omega$ -regular trace language can be expressed as a Boolean combination of D-SABA-recognizable languages  
An  $\omega$ -regular language can be expressed as a Boolean combination of DBA-recognizable languages

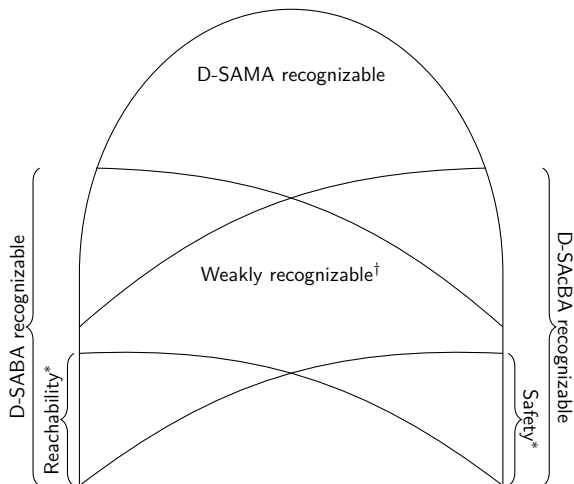
- Generalizing the study of  $\omega$ -regular languages

If the independence relation  $I = \emptyset$

- ▶ A D-SABA accepts a language  $\Theta$  if and only if  $\Theta = \bigcup_{i=1}^n \lim_{A_i}(T_i)$   
A DBA accepts a language  $L$  if and only if  $L = \bigcup_{i=1}^n \lim_{\Sigma}(K_i)$
- ▶ A characterization of det. trace languages in terms of a sub-class of det. Muller automata  
A characterization of det. languages in terms of a sub-class of det. Muller automata, leading to a decision process
- ▶ An  $\omega$ -regular trace language can be expressed as a Boolean combination of D-SABA-recognizable languages  
An  $\omega$ -regular language can be expressed as a Boolean combination of DBA-recognizable languages

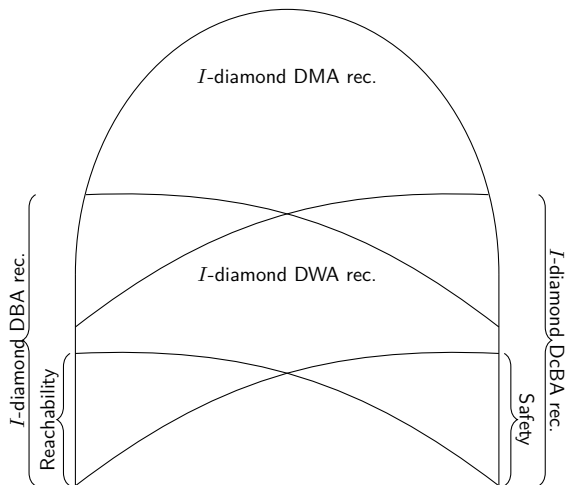
- **In progress:** deterministic, synchronization-aware weak automata

# a Borel-like hierarchy



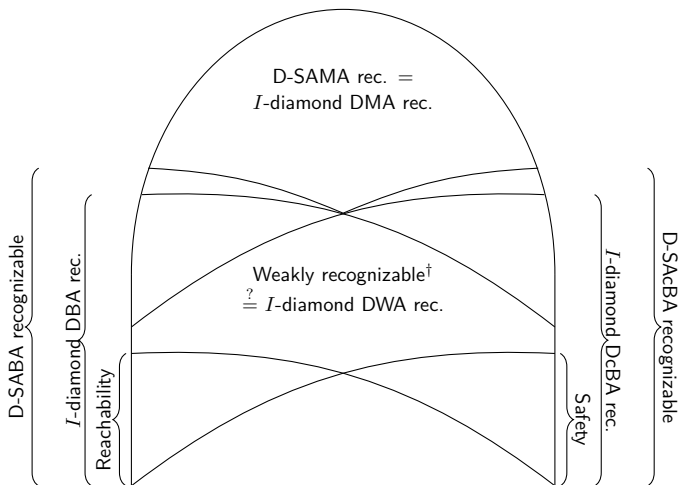
**Open:** a characterization in terms of logic

## a Borel-like hierarchy (part II)



**Open:** a characterization in terms of logic

## a Borel-like hierarchy (part II)



**Open:** a characterization in terms of logic