On ω-Regular Trace Languages

Classification and Synthesis

Namit Chaturvedi and Marcus Gelderie

RWTH AACHEN UNIVERSITY

GAMES 2012
September 09, 2012

Traces and trace languages

Traces as models of behaviors of distributed systems

- Traces model dependence/independence between actions
- Independent actions mean **semantic equivalence** of **swapping** in a sequence of actions

Traces and trace languages

Traces as models of behaviors of distributed systems

- Traces model dependence/independence between actions
- Independent actions mean **semantic equivalence** of **swapping** in a sequence of actions

Trace languages correspond closely with word languages

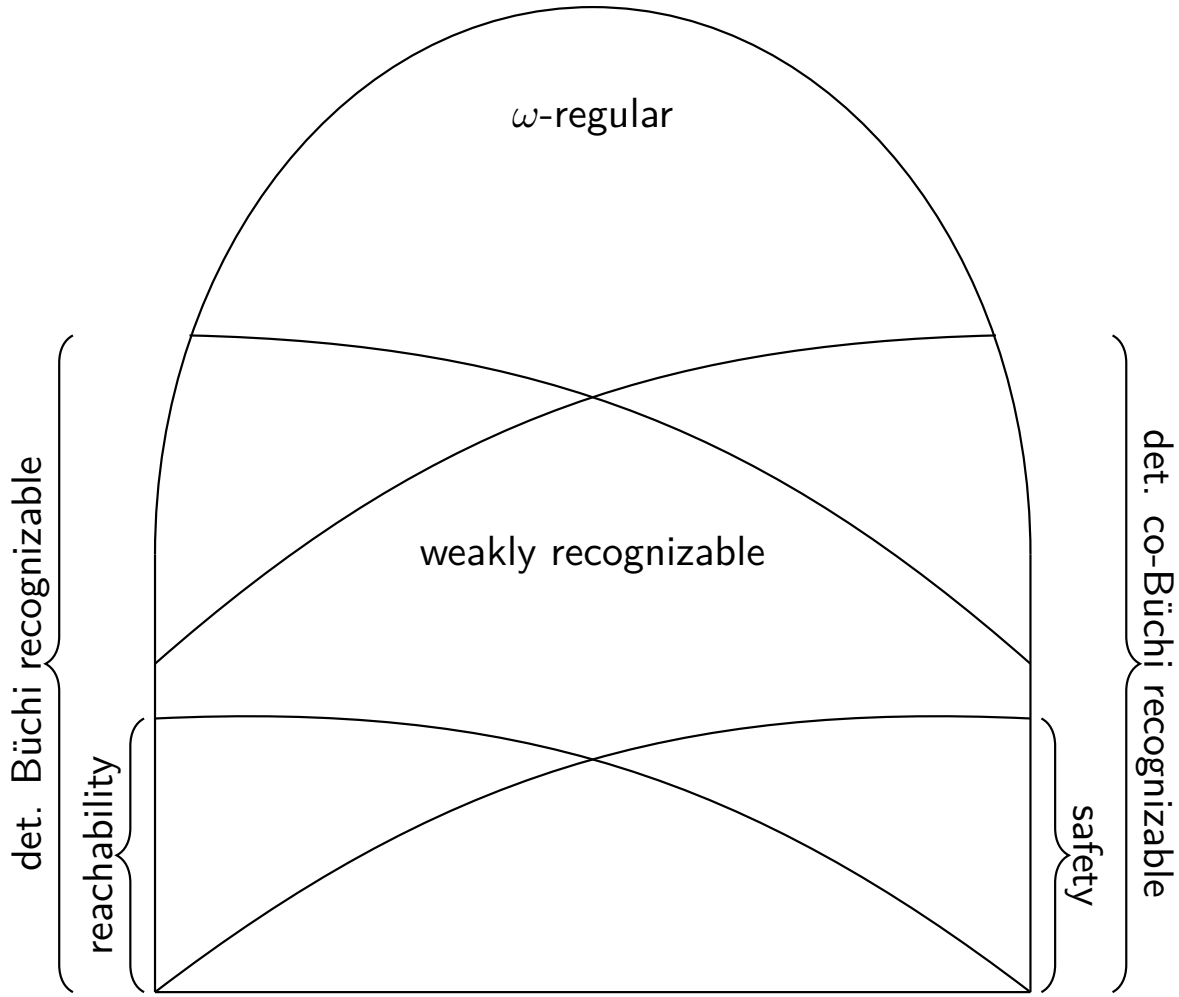
- A single **trace** represents an **equivalence class** of such partially commutative sequences
- A language of traces is **recognizable** if and only if the corresponding "trace-closed" word language is recognizable

End of similarities

Borel classification of ω-regular languages

Open problem

Can ω-regular trace languages also be classified in such a manner?



Motivation

Why Borel classification of ω-regular languages?

Motivation

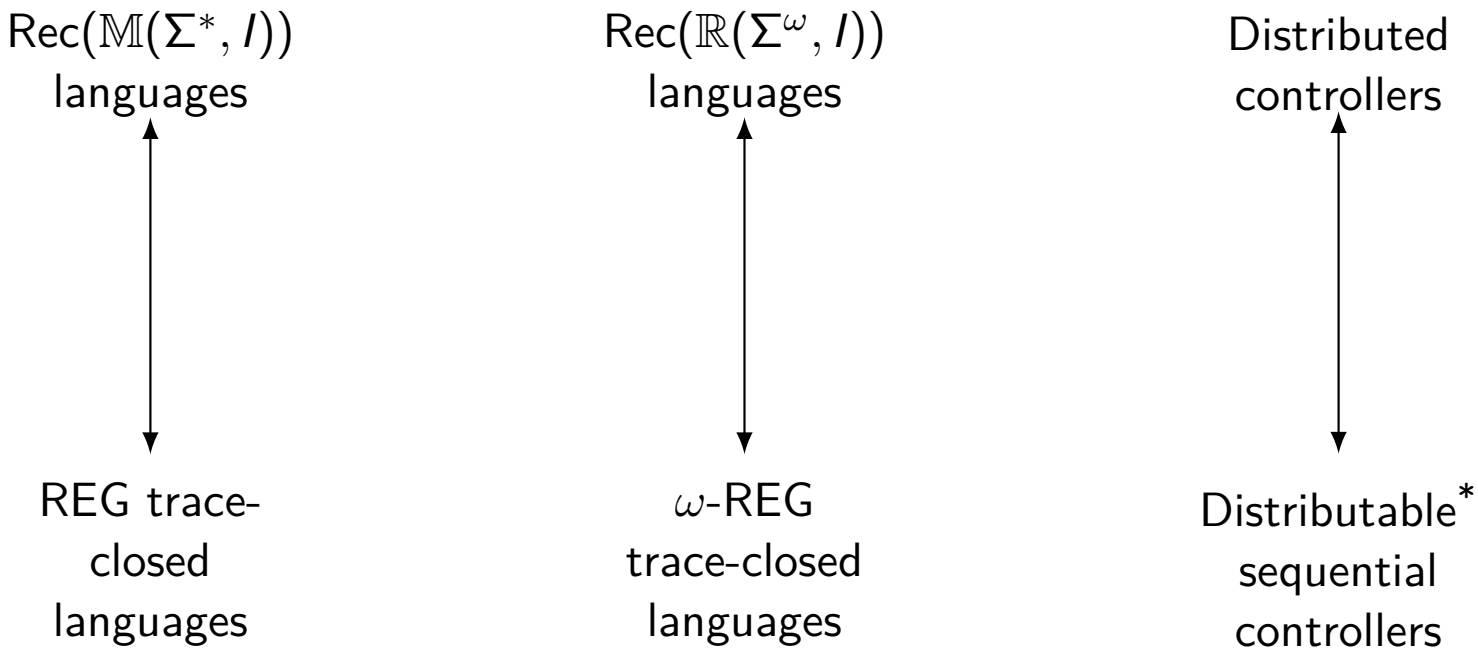
Why Borel classification of ω-regular languages?

1. Classification is good
2. Natural descriptions of ω-regular languages with the help of regular languages (using reachability, safety, and liveness conditions)
3. Easy construction of ω-automata from DFAs
4. Efficient algorithms for synthesis and verification for different classes
5. Effective classification procedure given an arbitrary ω-automaton

Motivation

Why Borel classification of ω-regular languages?

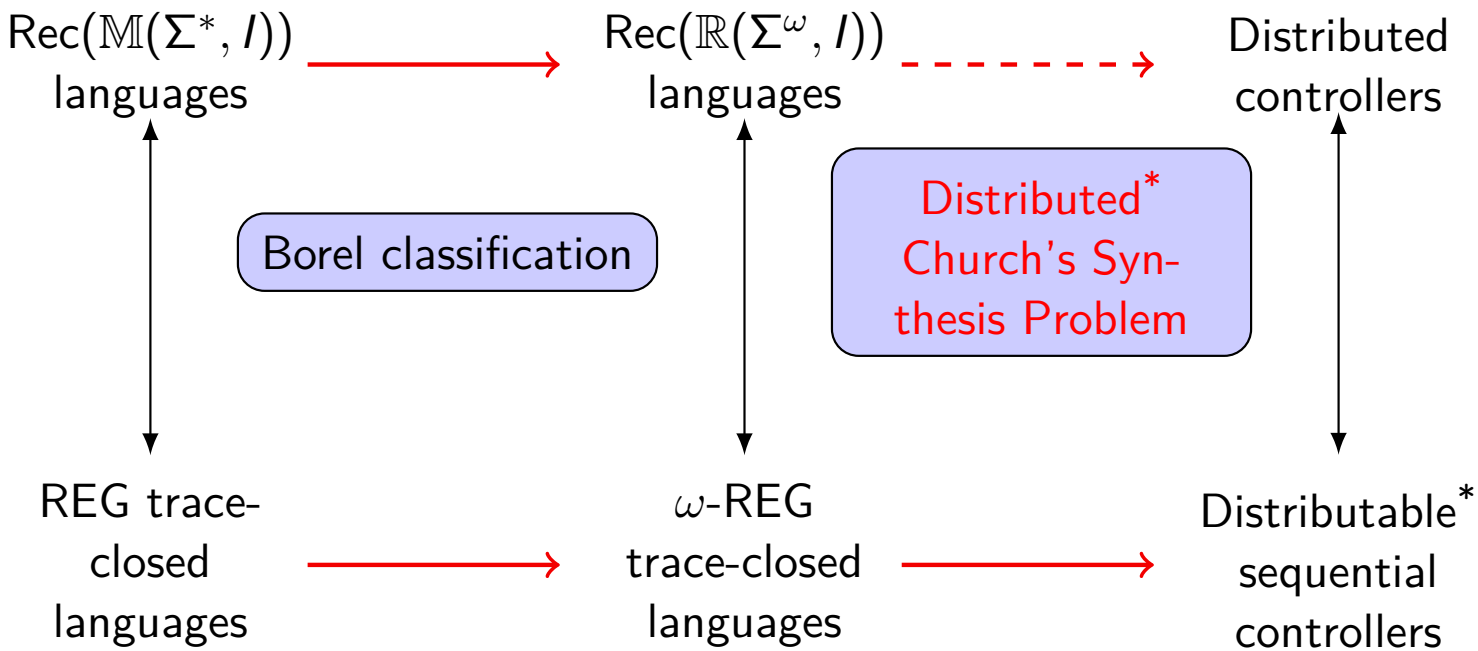
1. Classification is good
2. Natural descriptions of ω-regular languages with the help of regular languages (using reachability, safety, and liveness conditions)
3. Easy construction of ω-automata from DFAs
4. Efficient algorithms for synthesis and verification for different classes
5. Effective classification procedure given an arbitrary ω-automaton



Motivation

Why Borel classification of ω-regular languages?

1. Classification is good
2. Natural descriptions of ω-regular languages with the help of regular languages (using reachability, safety, and liveness conditions)
3. Easy construction of ω-automata from DFAs
4. Efficient algorithms for synthesis and verification for different classes
5. Effective classification procedure given an arbitrary ω-automaton



Preliminaries: Traces

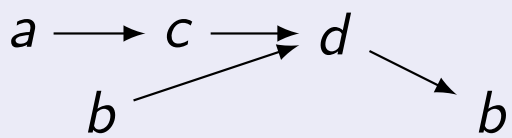
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **dependence alphabet**.

Preliminaries: Traces

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

Definition (Finite and infinite traces)

A **trace** over (Σ, I) is a labeled DAG:

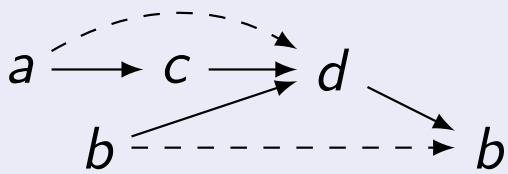


Preliminaries: Traces

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

Definition (Finite and infinite traces)

A **trace** over (Σ, I) is a labeled DAG:

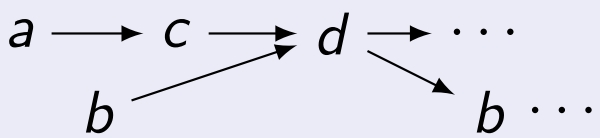


Preliminaries: Traces

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

Definition (Finite and infinite traces)

A **trace** over (Σ, I) is a labeled DAG:

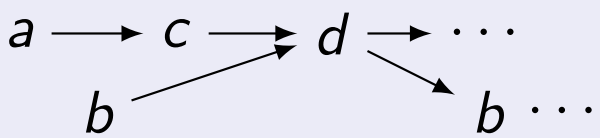


Preliminaries: Traces

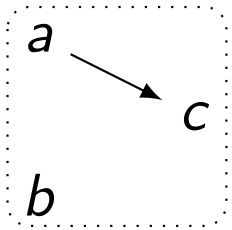
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **dependence alphabet**.

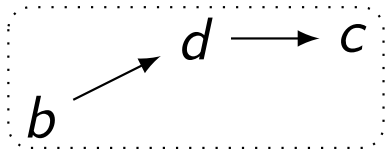
Definition (Finite and infinite traces)

A **trace** over (Σ, I) is a labeled DAG:



Assuming $a I b$ and $b I c$



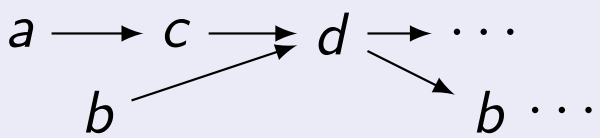


Preliminaries: Traces

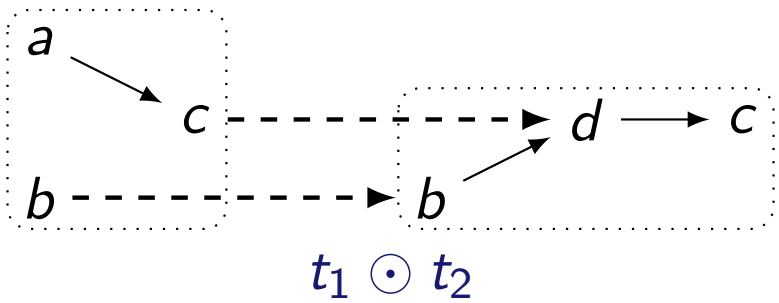
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **dependence alphabet**.

Definition (Finite and infinite traces)

A **trace** over (Σ, I) is a labeled DAG:



Assuming $a I b$ and $b I c$

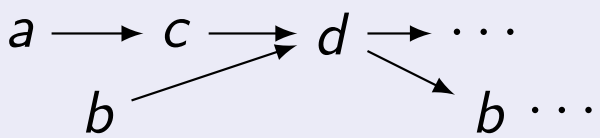


Preliminaries: Traces

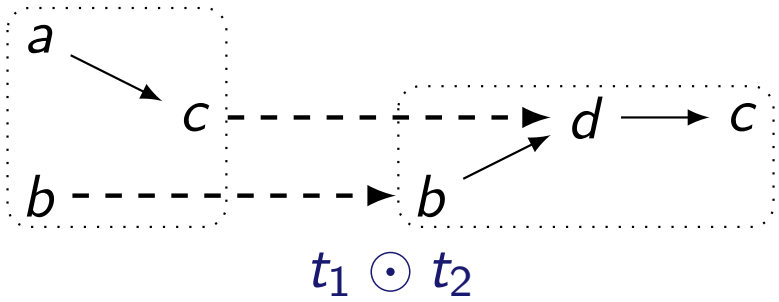
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **dependence alphabet**.

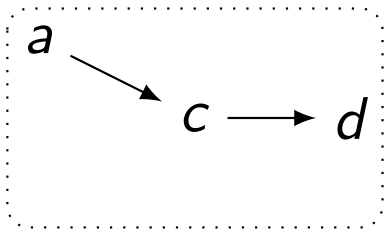
Definition (Finite and infinite traces)

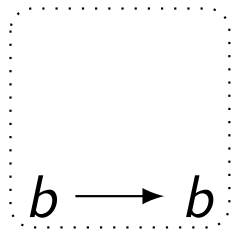
A **trace** over (Σ, I) is a labeled DAG:



Assuming $a I b$ and $b I c$





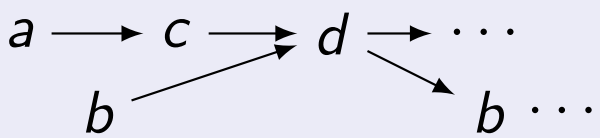


Preliminaries: Traces

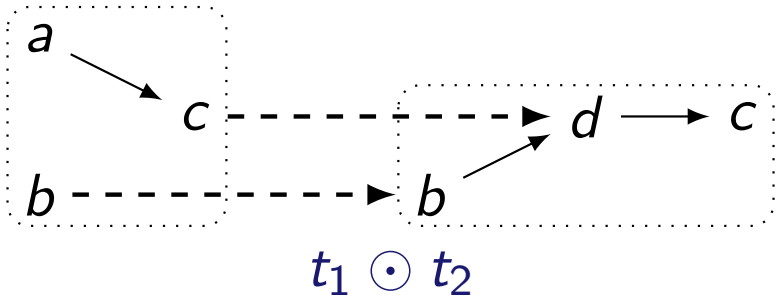
Let $I \subseteq \Sigma^2$ be a symmetric, irreflexive **independence relation**, then (Σ, I) is called a **dependence alphabet**.

Definition (Finite and infinite traces)

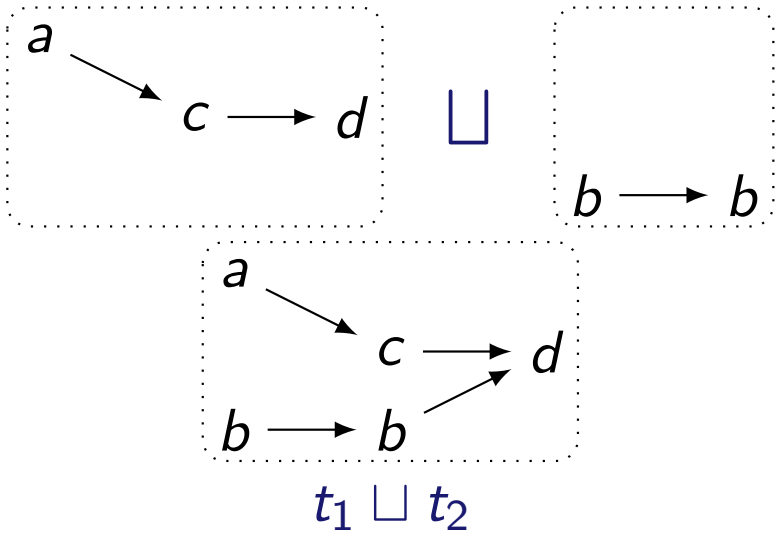
A **trace** over (Σ, I) is a labeled DAG:



Assuming $a I b$ and $b I c$



$t_1 \odot t_2$



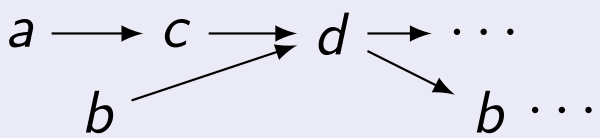
$t_1 \sqcup t_2$

Preliminaries: Traces

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

Definition (Finite and infinite traces)

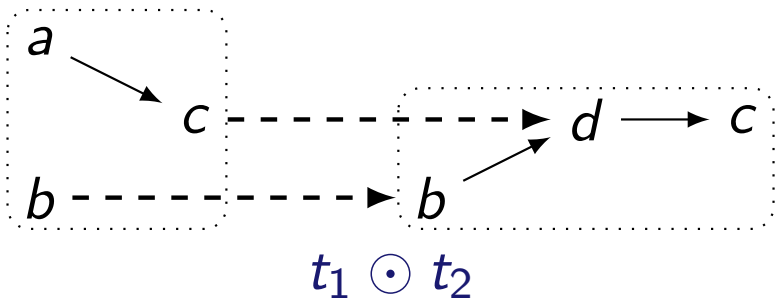
A **trace** over (Σ, I) is a labeled DAG:

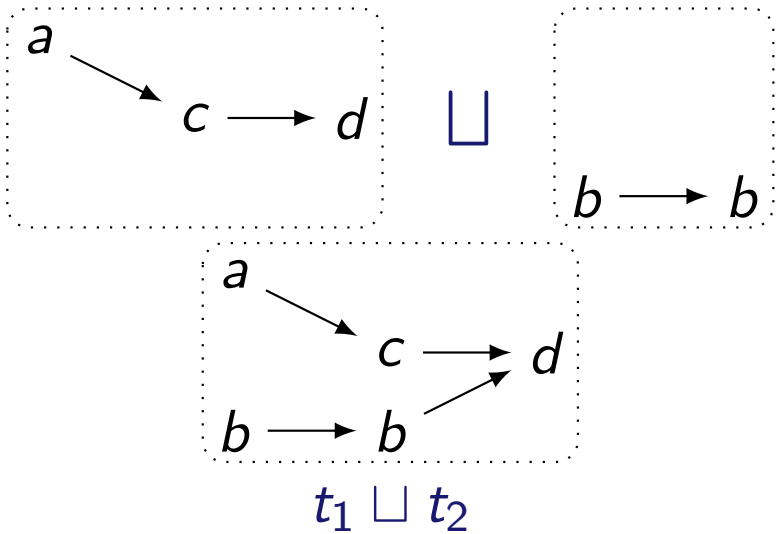


The set of all finite traces: $\mathbb{M}(\Sigma^*, I)$

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

Assuming $a I b$ and $b I c$





Preliminaries: Traces, words, languages

There is a tight connection between traces and words

- Mapping words to traces, $\Gamma : \Sigma^* \to \mathbb{M}(\Sigma^*, I)$

$$\Gamma(abcdb) = \quad \begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}$$

- Mapping traces to trace-closed sets of words

$$\Gamma^{-1}\left(\begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}\right) = \{bacdb, abcdb, acbdb\}$$

Preliminaries: Traces, words, languages

There is a tight connection between traces and words

- Mapping words to traces, $\Gamma : \Sigma^* \to \mathbb{M}(\Sigma^*, I)$

$$\Gamma(abcdb) = \begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}$$

- Mapping traces to trace-closed sets of words

$$\Gamma^{-1}\left(\begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}\right) = \{bacdb, abcdb, acbdb\}$$

- Words u and v are **trace equivalent** if $\Gamma(u) = \Gamma(v)$
- A finite or infinite language L is **trace closed** if $L = \Gamma^{-1}(\Gamma(L))$ or simply $L = [L]_{\sim_I}$

Preliminaries: Traces, words, languages

There is a tight connection between traces and words

- Mapping words to traces, $\Gamma : \Sigma^* \to \mathbb{M}(\Sigma^*, I)$

$$\Gamma(abcdb) = \begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}$$

- Mapping traces to trace-closed sets of words

$$\Gamma^{-1}\left(\begin{array}{ccccccc} a & \longrightarrow & c & \longrightarrow & d & \searrow & \\ & b \nearrow & & & & & b \end{array}\right) = \{bacdb, abcdb, acbdb\}$$

- Words u and v are **trace equivalent** if $\Gamma(u) = \Gamma(v)$
- A finite or infinite language L is **trace closed** if $L = \Gamma^{-1}(\Gamma(L))$ or simply $L = [L]_{\sim_I}$

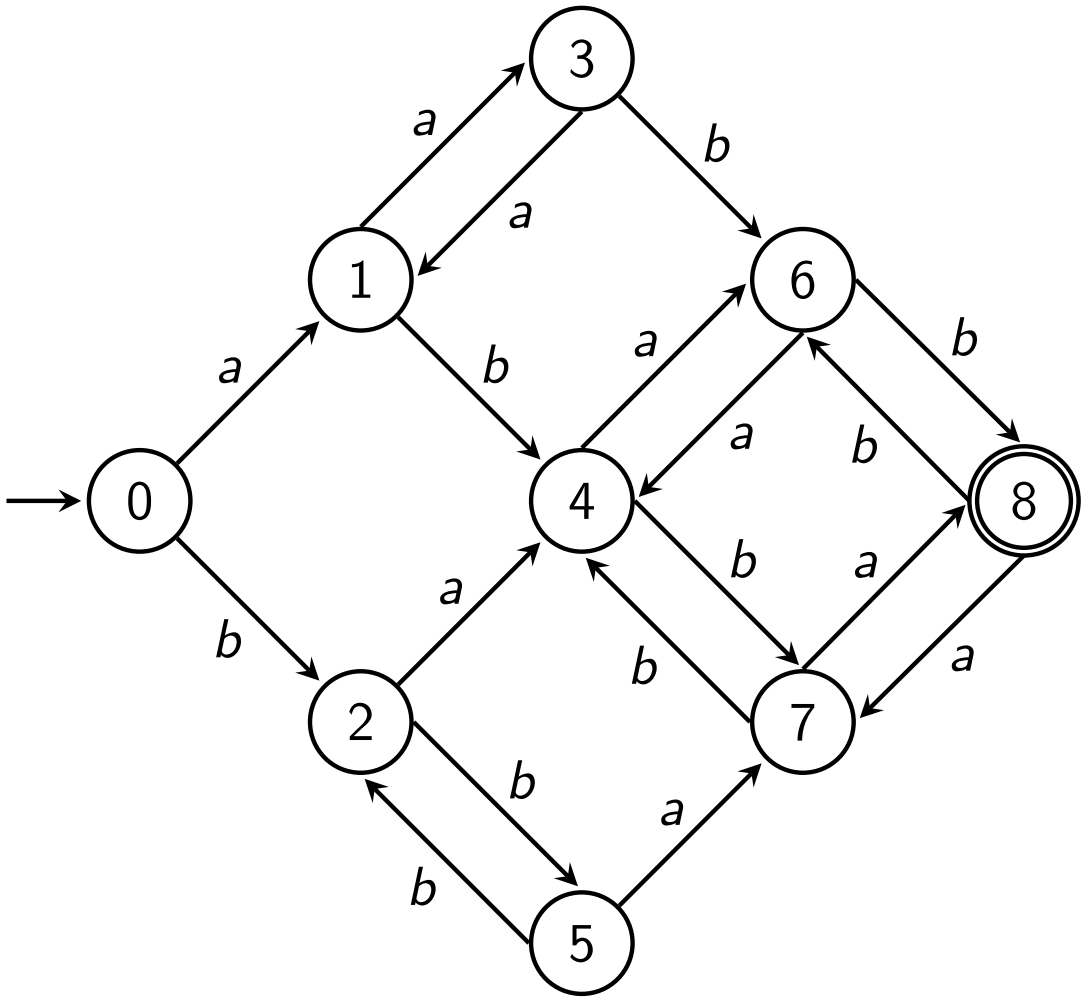
Definition

A trace language $T \subseteq \mathbb{M}(\Sigma^*, I)$ (resp. $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$) is **recognizable** iff $\Gamma^{-1}(T)$ (resp. $\Gamma^{-1}(\Theta)$) is a recognizable word language.

Preliminaries: A final note on automata

Trace-closed languages are recognized by ***I*-diamond automata**

Let $\Sigma = \{a, b\}$, $a I b$. Define $K := [(aa)^+(bb)^+]_{\sim_I}$.



Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\mathrm{ext}(K) \stackrel{?}{=} \Gamma^{-1}(\mathrm{ext}(T))$, or equivalently, $\mathrm{ext}(K) \stackrel{?}{=} [\mathrm{ext}(K)]_{\sim_I}$

Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Let $\Sigma = \{a, b\}$, $a I b$, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\mathrm{ext}(K) \neq \Gamma^{-1}(\mathrm{ext}(T))$, or equivalently, $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$

For trace-closed $K \in \mathrm{REG}$, it may hold that $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$.

Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\mathrm{ext}(K) \neq \Gamma^{-1}(\mathrm{ext}(T))$, or equivalently, $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$

  $abaabbaabb \ldots \notin \mathrm{ext}(K)$

For trace-closed $K \in \mathrm{REG}$, it may hold that $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$.

Infinitary extensions of regular trace languages

Definition

Let $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$. Its **infinitary extension** is the ω-trace language given by $\mathrm{ext}(T) := T \odot \mathbb{R}(\Sigma^\omega, I) = \bigcup_{t \in T} t \odot \mathbb{R}(\Sigma^\omega, I)$.

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

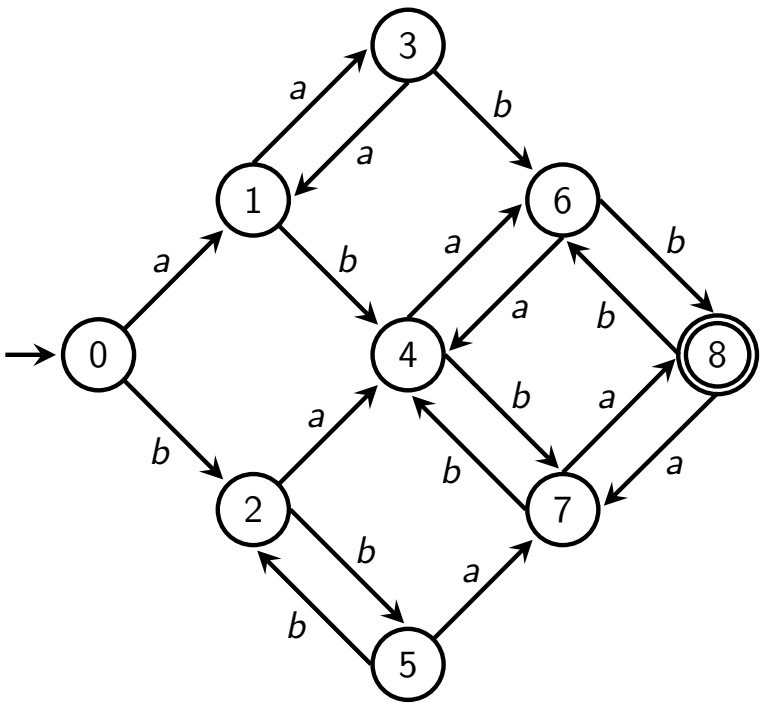
- $\mathrm{ext}(K) \neq \Gamma^{-1}(\mathrm{ext}(T))$, or equivalently, $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$

$$abaabbaabb \ldots \notin \mathrm{ext}(K)$$

$$\Gamma(ab(aabb)^\omega) = \begin{matrix} a \longrightarrow a \longrightarrow a \longrightarrow \cdots \\ b \longrightarrow b \longrightarrow b \longrightarrow \cdots \end{matrix} \in \mathrm{ext}(T)$$

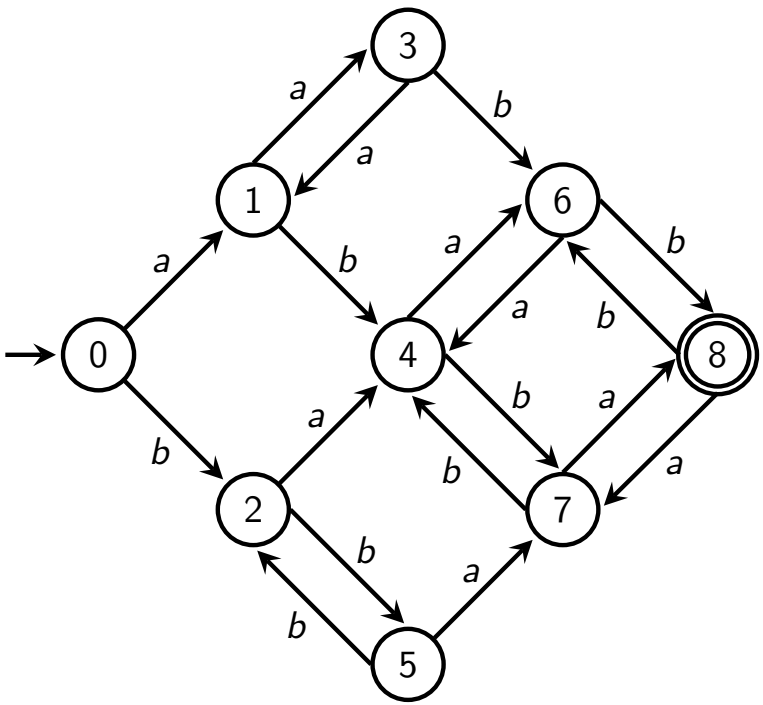
For trace-closed $K \in \mathrm{REG}$, it may hold that $\mathrm{ext}(K) \neq [\mathrm{ext}(K)]_{\sim_I}$.

Ensuring ext(K) is trace-closed



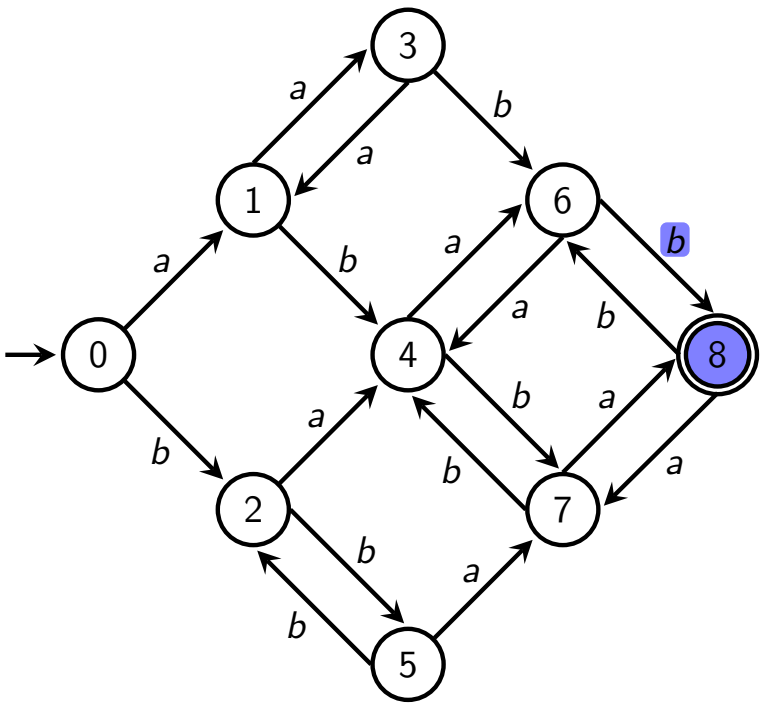
- $K = [(aa)^+(bb)^+]_{\sim_I}$ with $a I b$

Ensuring ext(K) is trace-closed



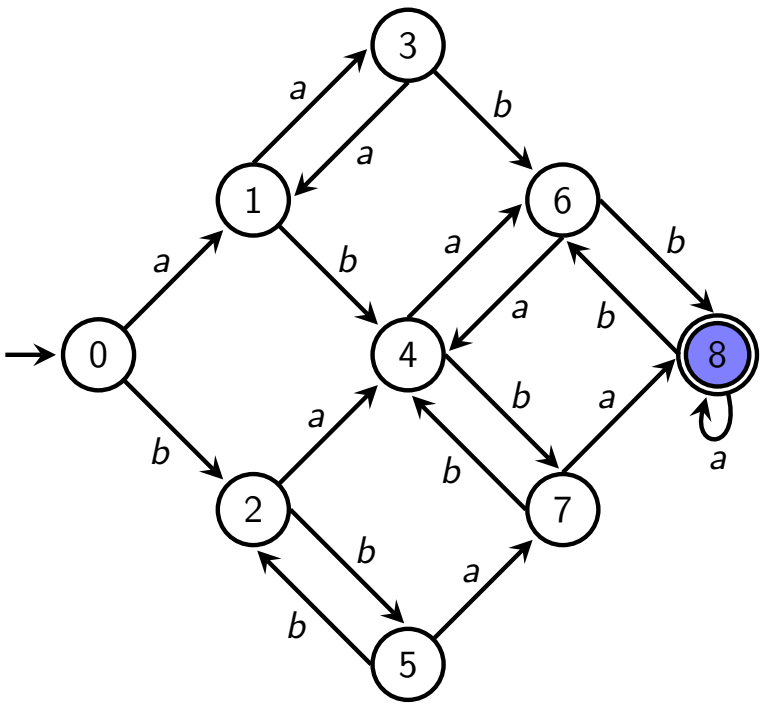
- $K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb
  - $I_b = \{a\}$, $I_a = \{b\}$

Ensuring ext(K) is trace-closed



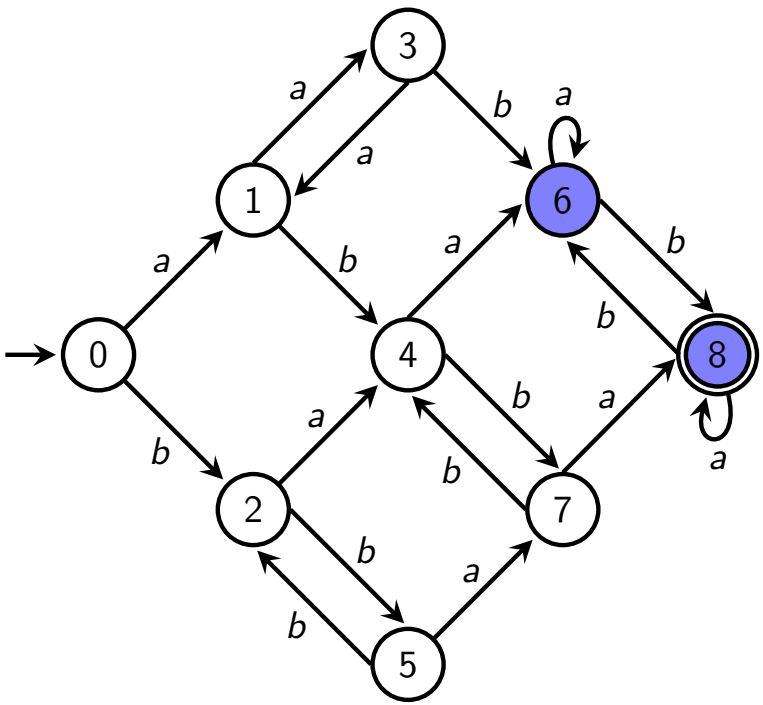
- $K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$

Ensuring ext(K) is trace-closed



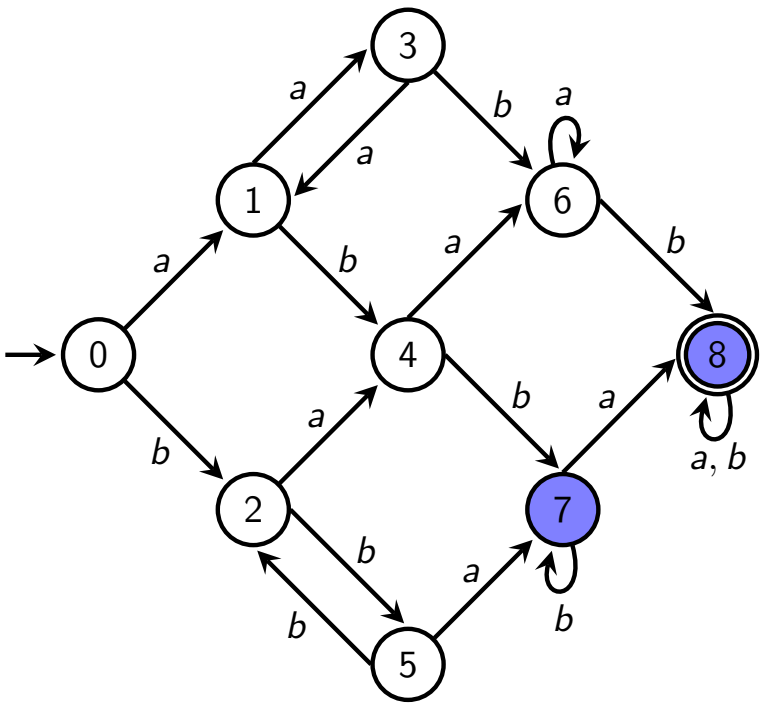
- $K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$
- $K \cup Kb^{-1}b \cdot I_b^*$

Ensuring ext(K) is trace-closed



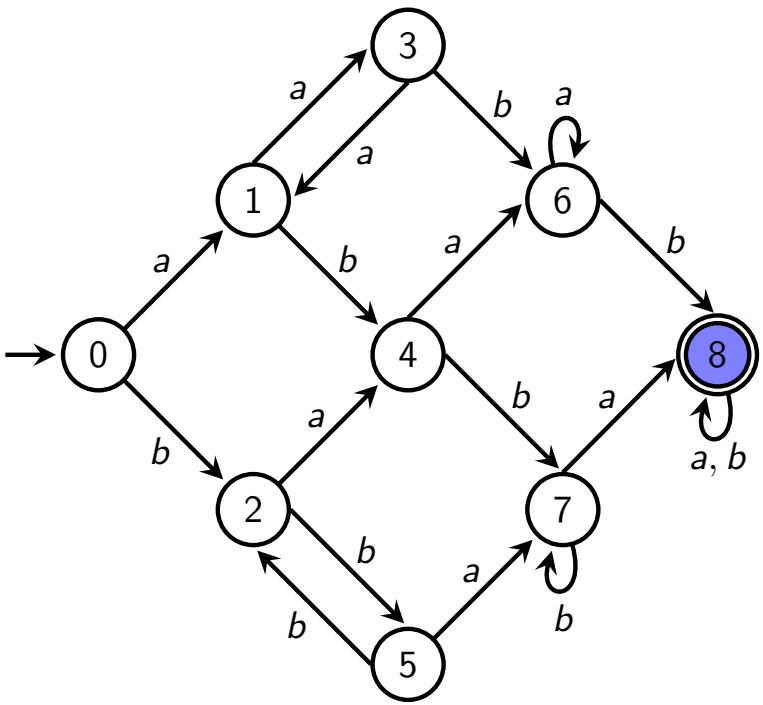
- $K = [(aa)^+(bb)^+]_{\sim_I}$ with $a I b$
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$
- $K \cup Kb^{-1}b \cdot I_b^*$
- $K_{\{b\}} = K \cup [Kb^{-1}b \cdot I_b^*]_{\sim_I}$

Ensuring ext(K) is trace-closed



- $K = [(aa)^+(bb)^+]_{\sim_I}$ with $a I b$
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$
- $K \cup Kb^{-1}b \cdot I_b^*$
- $K_{\{b\}} = K \cup [Kb^{-1}b \cdot I_b^*]_{\sim_I}$
- $K_{\{a,b\}} = K_{\{b\}} \cup [Ka^{-1}a \cdot I_a^*]_{\sim_I}$

Ensuring ext(K) is trace-closed

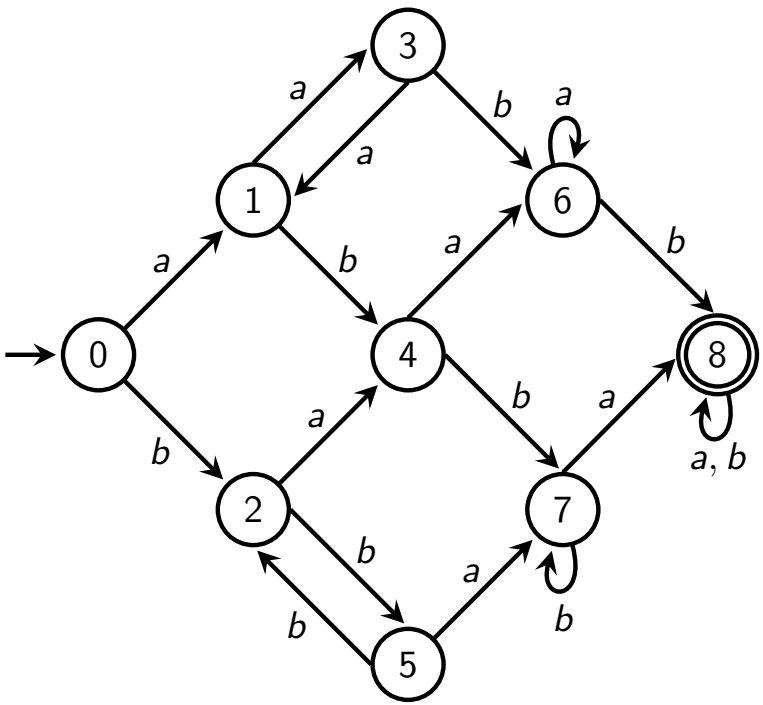


- $K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$
- $K \cup Kb^{-1}b \cdot I_b^*$
- $K_{\{b\}} = K \cup [Kb^{-1}b \cdot I_b^*]_{\sim_I}$
- $K_{\{a,b\}} = K_{\{b\}} \cup [Ka^{-1}a \cdot I_a^*]_{\sim_I}$
- $K_I = K_\Sigma$

Ensuring ext(K) is trace-closed

Definition (I-suffix extension)

For $K \in \mathrm{REG}$, trace-closed, define $K_I := K \cup \bigcup_{a \in \Sigma} [Ka^{-1}a \cdot I_a^*]_{\sim_I}$



- $K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb
  - $I_b = \{a\}$, $I_a = \{b\}$
- $Kb^{-1}b$
- $K \cup Kb^{-1}b \cdot I_b^*$
- $K_{\{b\}} = K \cup [Kb^{-1}b \cdot I_b^*]_{\sim_I}$
- $K_{\{a,b\}} = K_{\{b\}} \cup [Ka^{-1}a \cdot I_a^*]_{\sim_I}$
- $K_I = K_\Sigma$

I-suffix extension suffices

Lemma

For any trace-closed $K \in \mathrm{REG}$, *it holds that* $\mathrm{ext}(K_I) = [\mathrm{ext}(K)]_{\sim_I}$

I-suffix extension suffices

Lemma

For any trace-closed $K \in \mathrm{REG}$, *it holds that* $\mathrm{ext}(K_I) = [\mathrm{ext}(K)]_{\sim_I}$

For every $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$, the trace-closed language $\Gamma^{-1}(\mathrm{ext}(T))$ is I-diamond DWA recognizable.

I-suffix extension suffices

Lemma

For any trace-closed $K \in \mathrm{REG}$*, it holds that* $\mathrm{ext}(K_I) = [\mathrm{ext}(K)]_{\sim_I}$

For every $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$, the trace-closed language $\Gamma^{-1}(\mathrm{ext}(T))$ is I-diamond DWA recognizable.

Theorem (I-diamond DWA recognizability)

A trace-closed language $L \subseteq \Sigma^\omega$ *is recognized by an* I*-diamond DWA iff* $L \in \mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ *for a set* $\mathcal{K} \subseteq \mathrm{REG}$ *of trace-closed regular languages.*

I-suffix extension suffices

Lemma

For any trace-closed $K \in \mathrm{REG}$*, it holds that* $\mathrm{ext}(K_I) = [\mathrm{ext}(K)]_{\sim_I}$

For every $T \in \mathrm{Rec}(\mathbb{M}(\Sigma^*, I))$, the trace-closed language $\Gamma^{-1}(\mathrm{ext}(T))$ is I-diamond DWA recognizable.

Theorem (I-diamond DWA recognizability)

A trace-closed language $L \subseteq \Sigma^\omega$ *is recognized by an* I*-diamond DWA iff* $L \in \mathrm{BC}(\mathrm{ext}(\mathcal{K}))$ *for a set* $\mathcal{K} \subseteq \mathrm{REG}$ *of trace-closed regular languages.*

Definition (Weakly recognizable trace languages)

A trace language $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$ is **weakly recognizable** if $\Gamma^{-1}(\Theta)$ is I-diamond DWA recognizable.

Infinitary limits of regular trace languages

Definition

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

Infinitary limits of regular trace languages

Definition

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

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\lim(K) \neq \Gamma^{-1}(\lim(T))$

Infinitary limits of regular trace languages

Definition

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

Let $\Sigma = \{a, b\}$, aIb, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\lim(K) \neq \Gamma^{-1}(\lim(T))$
- $[\lim(K)]_{\sim_I}$ is not I-diamond DBA recognizable

Infinitary limits of regular trace languages

Definition

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

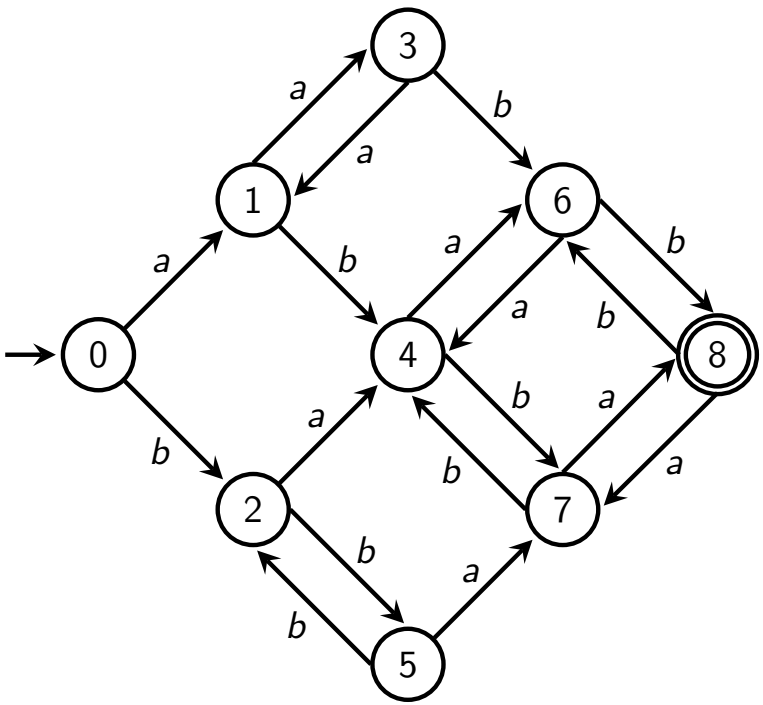
Let $\Sigma = \{a, b\}$, $a I b$, $K = [(aa)^+(bb)^+]_{\sim_I}$, and $T = \Gamma(K)$.

- $\lim(K) \neq \Gamma^{-1}(\lim(T))$
- $[\lim(K)]_{\sim_I}$ is not I-diamond DBA recognizable

Definition (Limit-stability)

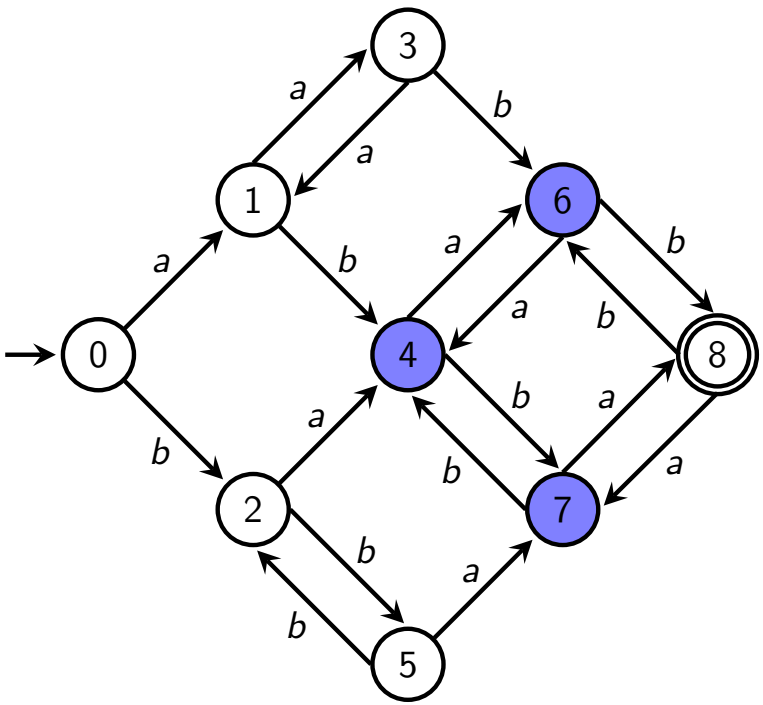
A trace-closed language $K \in \mathrm{REG}$ is **limit-stable** if $\lim(K) = [\lim(K)]_{\sim_I}$.
By extension, we say that $T \subseteq \mathbb{M}(\Sigma^*, I)$ is **limit-stable** if $\Gamma^{-1}(T)$ is.

Searching for limit-stable languages



$K = [(aa)^+(bb)^+]_{\sim_I}$ with $a I b$

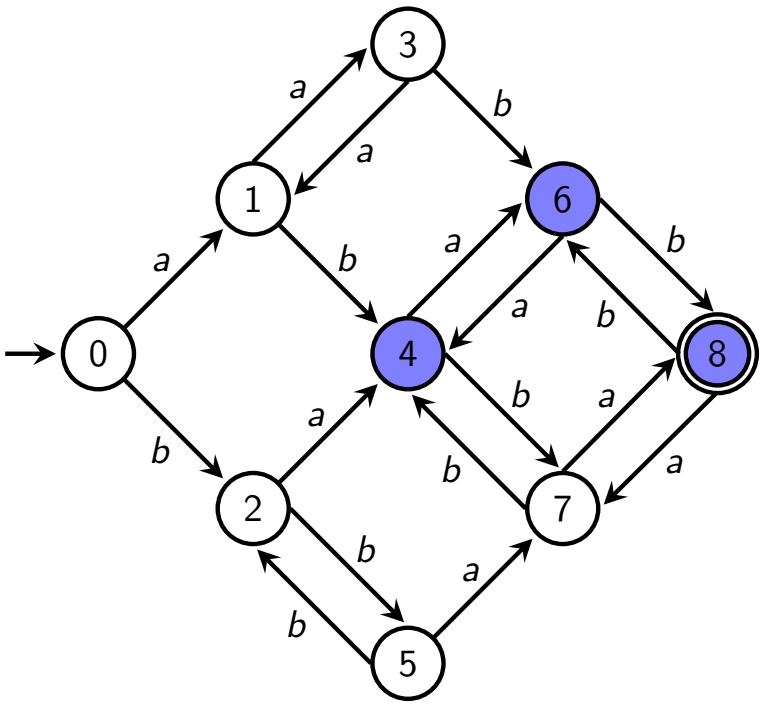
Searching for limit-stable languages



$K = [(aa)^+(bb)^+]_{\sim_I}$ with aIb

$$\Gamma(ab(aabb)^\omega) = \begin{matrix} a \to a \to a \to \cdots \\ b \to b \to b \to \cdots \end{matrix}$$

Searching for limit-stable languages



$$K = [(aa)^+(bb)^+]_{\sim_I} \text{ with } aIb$$

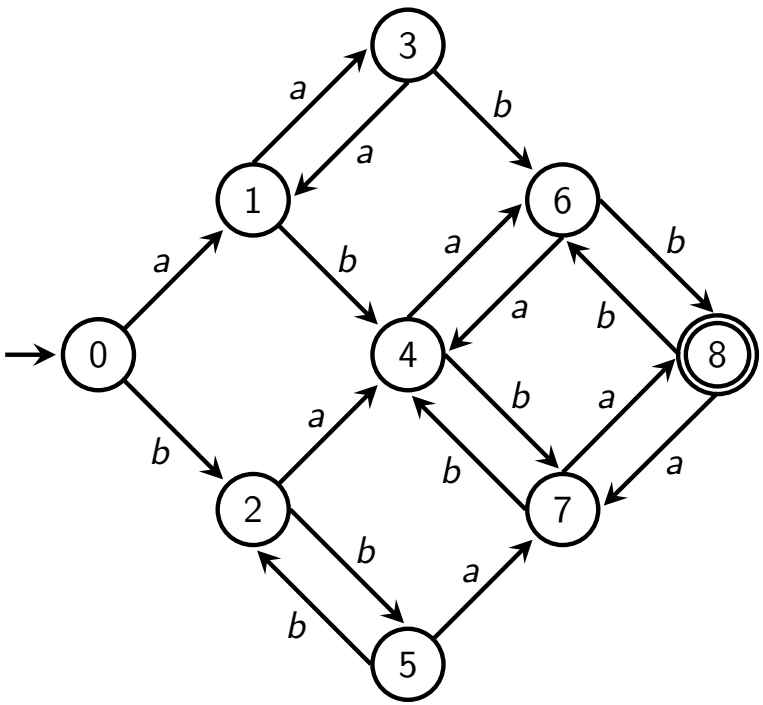
$$\Gamma(ab(aabb)^\omega) = \begin{matrix} a \to a \to a \to \cdots \\ b \to b \to b \to \cdots \end{matrix}$$

$$\Gamma(ab(abba)^\omega) = \begin{matrix} a \to a \to a \to \cdots \\ b \to b \to b \to \cdots \end{matrix}$$

Searching for limit-stable languages

Definition (F, I-cycle closure)

For a given (Σ, I), an I-diamond DFA is F, I-**cycle closed** if for all $u \sim_I v$ and all $q \in Q$, $q \xrightarrow{u}_F q$ iff $q \xrightarrow{v}_F q$.



$K = [(aa)^+(bb)^+]_{\sim_I}$ with $a I b$

$$\Gamma(ab(aabb)^\omega) = \begin{matrix} a \to a \to a \to \cdots \\ b \to b \to b \to \cdots \end{matrix}$$

$$\Gamma(ab(abba)^\omega) = \begin{matrix} a \to a \to a \to \cdots \\ b \to b \to b \to \cdots \end{matrix}$$

F, I-cycle closure is necessary and sufficient

Theorem (I-diamond DBA recognizability)

A language $K \in \mathrm{REG}$ is limit-stable if and only if K is recognized by an F, I-cycle closed DFA.

F, I-cycle closure is necessary and sufficient

Theorem (I-diamond DBA recognizability)

A language $K \in \mathrm{REG}$ is limit-stable if and only if K is recognized by an F, I-cycle closed DFA.

- F, I-cycle closure is an efficiently decidable property
- An ω language recognized by a DBA is trace-closed if and only if the DBA is F, I-cycle closed

F, I-cycle closure is necessary and sufficient

Theorem (I-diamond DBA recognizability)

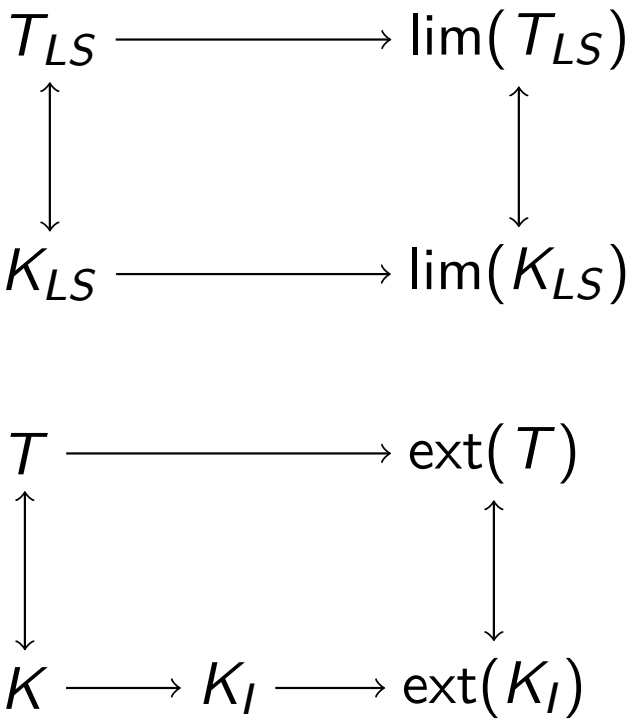
A language $K \in \mathrm{REG}$ is limit-stable if and only if K is recognized by an F, I-cycle closed DFA.

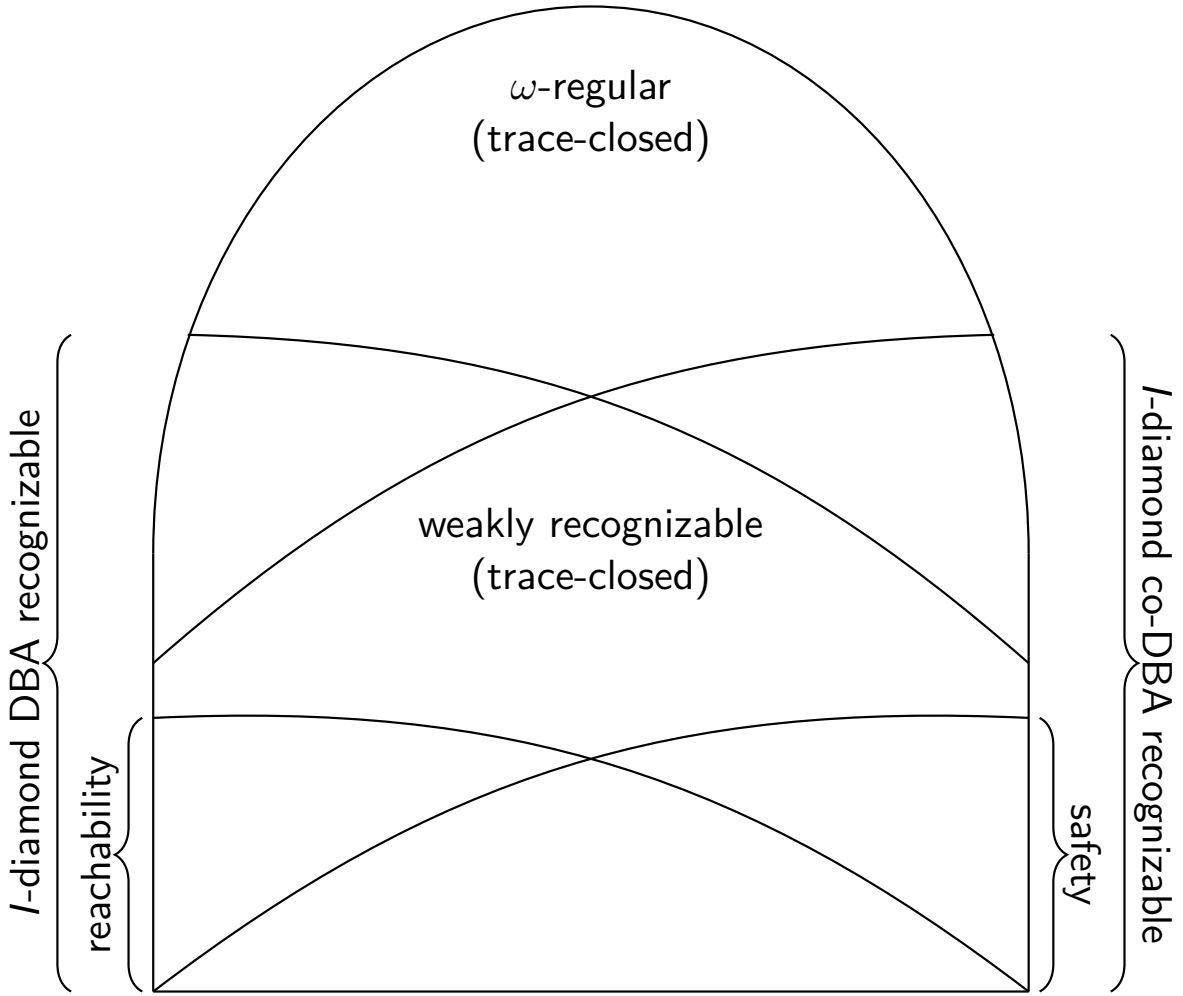
- F, I-cycle closure is an efficiently decidable property
- An ω language recognized by a DBA is trace-closed if and only if the DBA is F, I-cycle closed

Theorem (ω-regular trace languages)

A trace language Θ is ω-regular if and only if $\Gamma^{-1}(\Theta)$ is a finite Boolean combination of I-diamond DBA recognizable trace-closed languages. That is, $\mathrm{Rec}(\mathbb{R}(\Sigma^\omega, I)) = \mathrm{BC}(\lim(LS\text{-}\mathbb{M}(\Sigma^*, I)))$.

Borel hierarchy of ω-regular trace languages





Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x)\, I\, (b, y) \Leftrightarrow a I_1 b$

Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x) I (b, y) \Leftrightarrow a I_1 b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

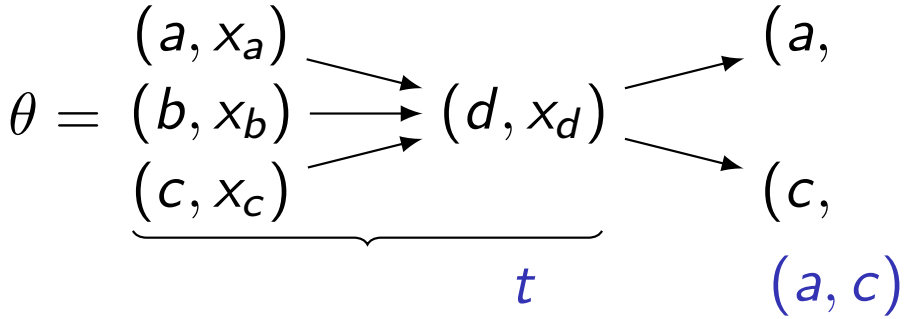
$$\theta = \underbrace{\begin{matrix} (a, x_a) & \searrow \\ (b, x_b) & \longrightarrow & (d, x_d) \\ (c, x_c) & \nearrow \end{matrix}}_{t}$$

Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x) I (b, y) \Leftrightarrow a I_1 b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

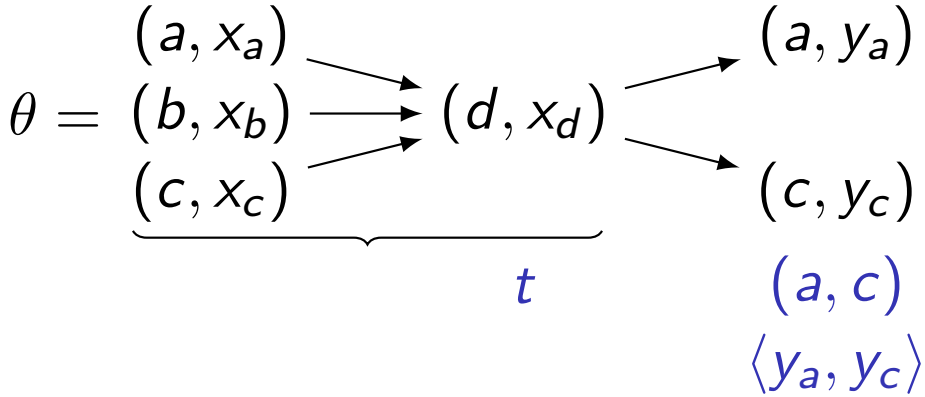


Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x) I (b, y) \Leftrightarrow a I_1 b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

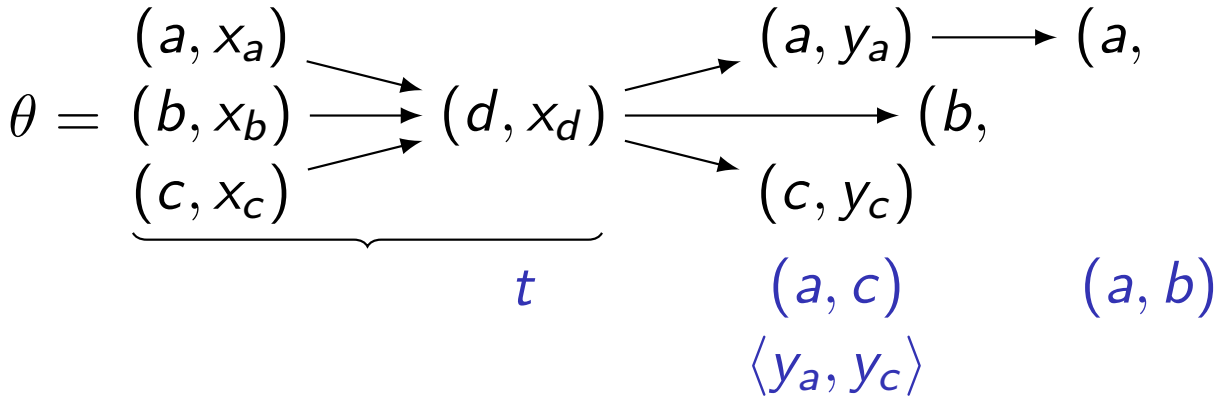


Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x)I(b, y) \Leftrightarrow aI_1b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

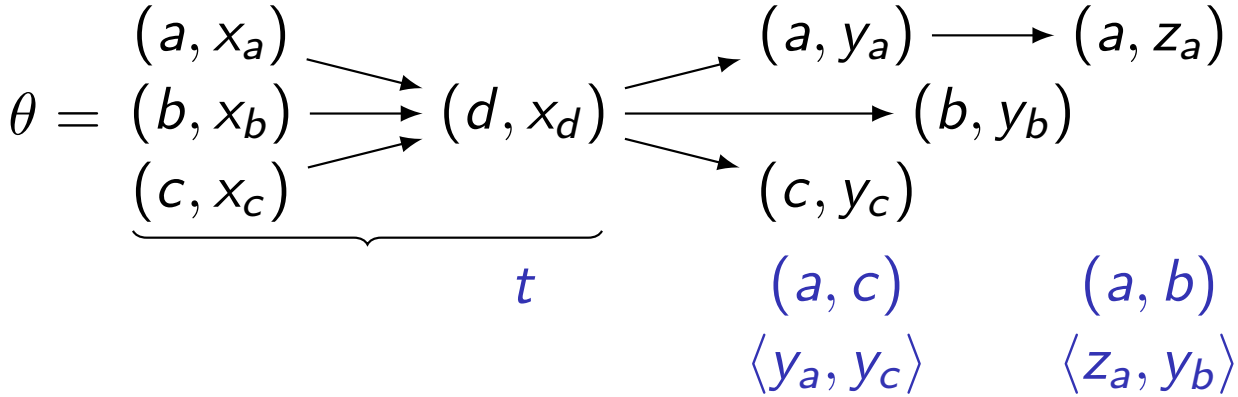


Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x)I(b, y) \Leftrightarrow aI_1b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

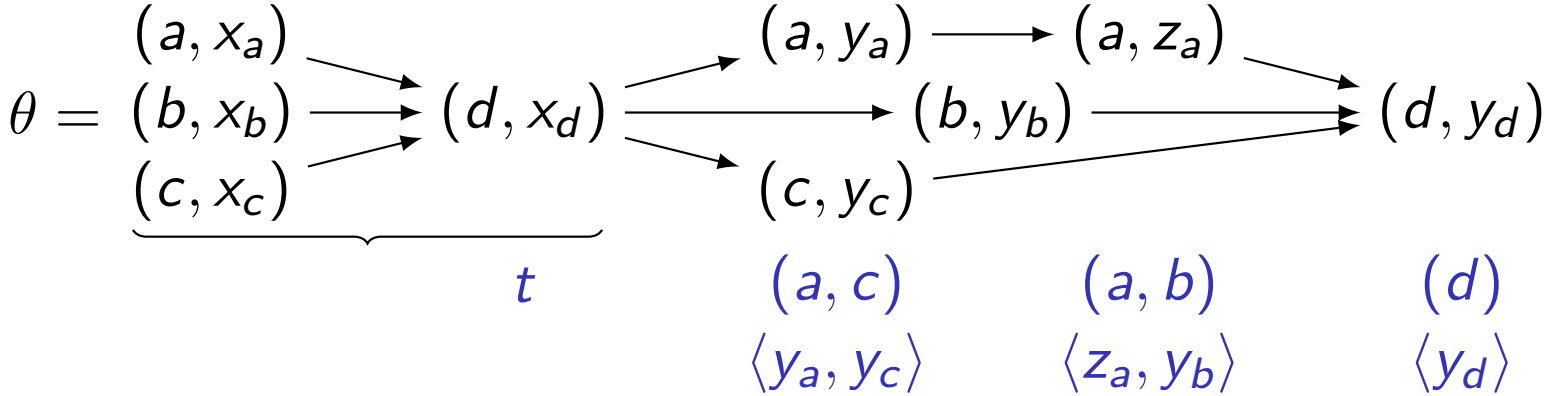


Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x)I(b, y) \Leftrightarrow aI_1b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:

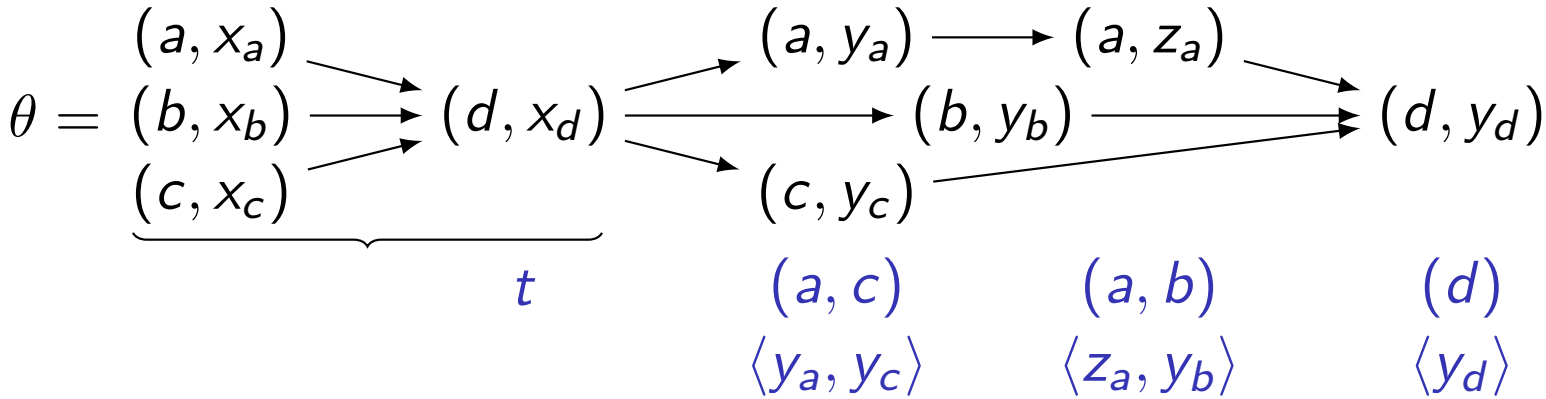


Distributed Church's Synthesis Problem

Adversary's choice of independence relation:

- $I_1 \subseteq \Sigma_1 \times \Sigma_1$ and infer $I \subseteq \Sigma \times \Sigma$ therefrom: $(a, x)I(b, y) \Leftrightarrow aI_1b$

A **play** of the distributed game over $\Sigma \subseteq \Sigma_1 \times \Sigma_2$ is a trace $\theta \in \mathbb{R}(\Sigma^\omega, I)$:



Player 1 moves (assuming Σ_1 is ordered):

- $\mathcal{C}_I := \{(a_1, \ldots, a_k) \subseteq 2^{\Sigma_1} \mid a_1 < \cdots < a_k \text{ and } a_i I a_j, i \neq j\}$

Player 2 moves:

- $\mathcal{V} := \{\langle x_1, \ldots, x_k \rangle \mid x_i \in \Sigma_2, k \leq |\Sigma_1|\}$

Distributed Church's Synthesis Problem

Game: an ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$

- If $\theta \in \Theta$ then Player 2 wins
- If $\theta \notin \Theta$ then Player 1 wins

Distributed Church's Synthesis Problem

Game: an ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$

- If $\theta \in \Theta$ then Player 2 wins
- If $\theta \notin \Theta$ then Player 1 wins

Given finite play prefixes, **strategies** guide players:

- Strategy for Player 1: $\lambda_1 : \mathbb{M}(\Sigma^*, I) \to \mathcal{C}_I$
- Strategy for Player 2: $\lambda_2 : \mathbb{M}(\Sigma^*, I) \to \mathcal{V}^{\mathcal{C}_I}$

Distributed Church's Synthesis Problem

Game: an ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$

- If $\theta \in \Theta$ then Player 2 wins
- If $\theta \notin \Theta$ then Player 1 wins

Given finite play prefixes, **strategies** guide players:

- Strategy for Player 1: $\lambda_1 : \mathbb{M}(\Sigma^*, I) \to \mathcal{C}_I$
- Strategy for Player 2: $\lambda_2 : \mathbb{M}(\Sigma^*, I) \to \mathcal{V}^{\mathcal{C}_I}$

To solve game $\Theta \in \mathrm{Rec}(\mathbb{R}(\Sigma^\omega, I))$, solve game $\Gamma^{-1}(\Theta)$.

Distributed Church's Synthesis Problem

Game: an ω-trace language $\Theta \subseteq \mathbb{R}(\Sigma^\omega, I)$

- If $\theta \in \Theta$ then Player 2 wins
- If $\theta \notin \Theta$ then Player 1 wins

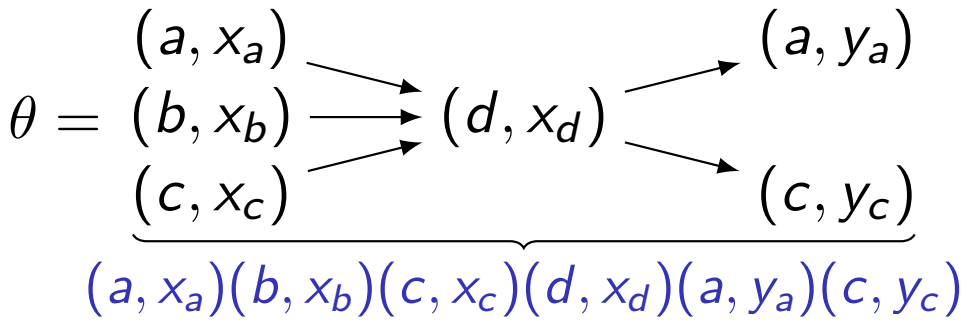
Given finite play prefixes, **strategies** guide players:

- Strategy for Player 1: $\lambda_1 : \mathbb{M}(\Sigma^*, I) \to \mathcal{C}_I$
- Strategy for Player 2: $\lambda_2 : \mathbb{M}(\Sigma^*, I) \to \mathcal{V}^{\mathcal{C}_I}$

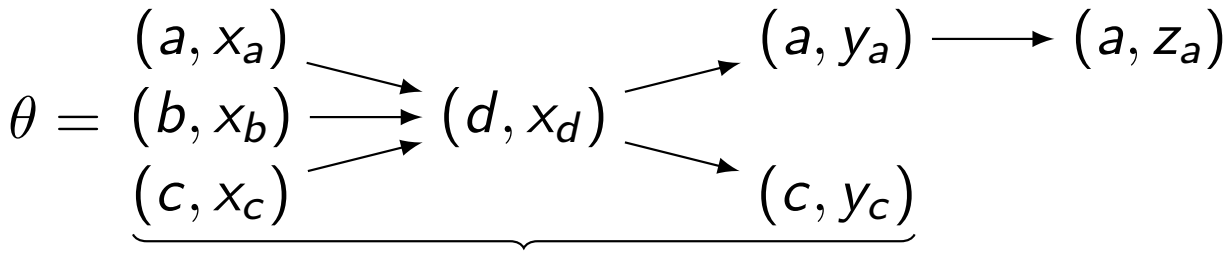
To solve game $\Theta \in \mathrm{Rec}(\mathbb{R}(\Sigma^\omega, I))$, solve game $\Gamma^{-1}(\Theta)$.

A **sequential strategy** of Player 2 is a transition system where each state $q \in Q$ has exactly one outgoing transition $(a, x_a) \in \Sigma$ for each $a \in \Sigma_1$.

Distributable sequential strategy for Player 2

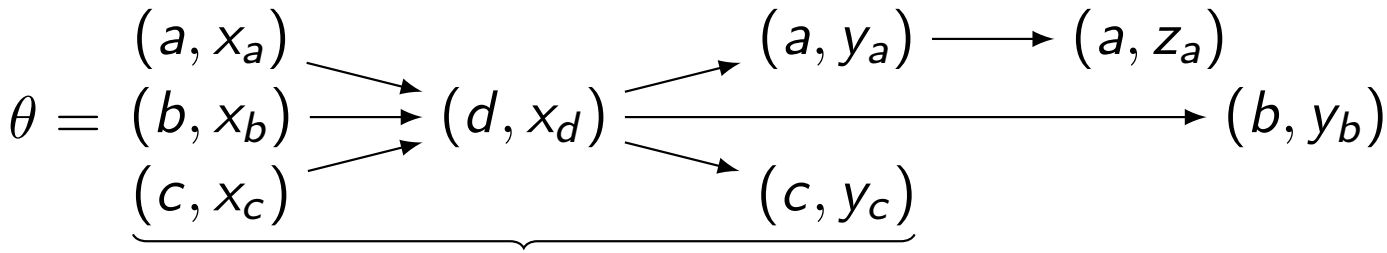


Distributable sequential strategy for Player 2



$(a, x_a)(b, x_b)(c, x_c)(d, x_d)(a, y_a)(c, y_c)$ $\quad (a)\langle z_a \rangle$

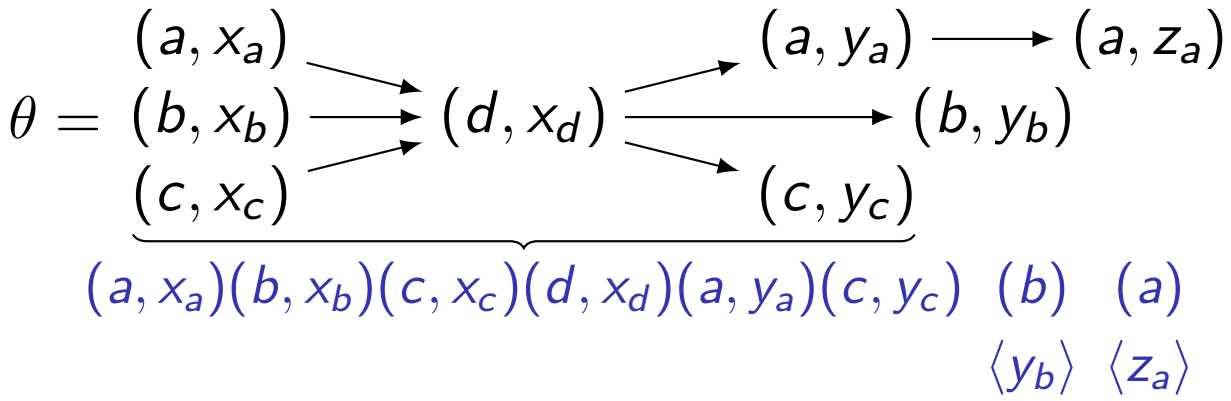
Distributable sequential strategy for Player 2



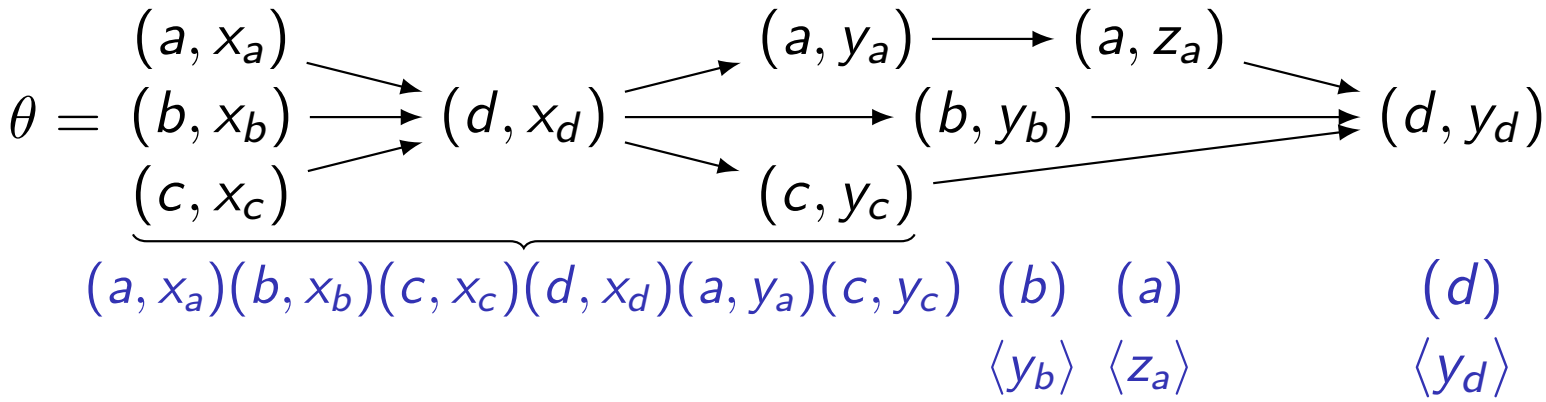
$(a, x_a)(b, x_b)(c, x_c)(d, x_d)(a, y_a)(c, y_c)$ $\quad (a)\ \ (b)$

$\langle z_a \rangle\ \ \langle y_b \rangle$

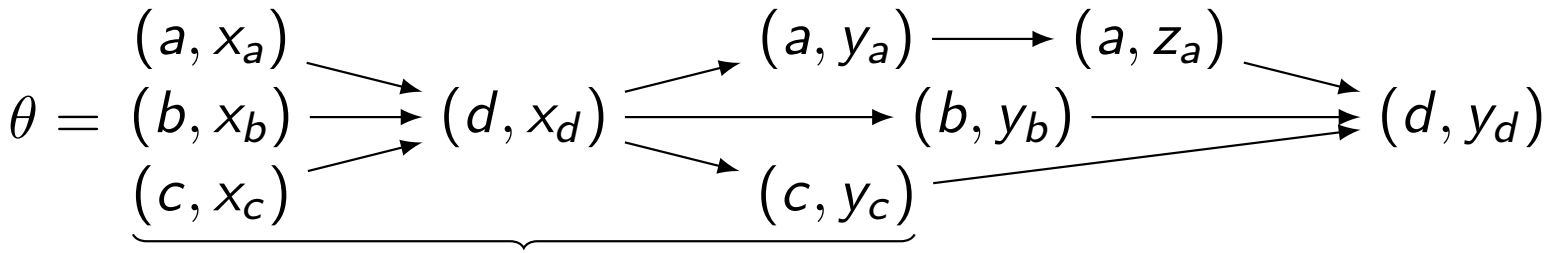
Distributable sequential strategy for Player 2



Distributable sequential strategy for Player 2



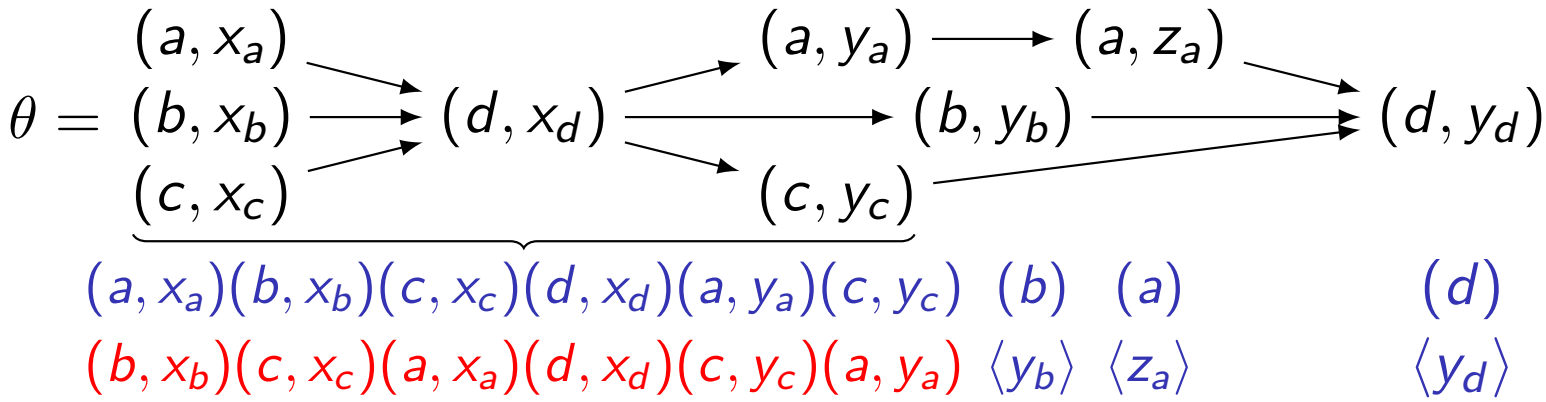
Distributable sequential strategy for Player 2



$(a, x_a)(b, x_b)(c, x_c)(d, x_d)(a, y_a)(c, y_c)$ (b) (a) (d)

$(b, x_b)(c, x_c)(a, x_a)(d, x_d)(c, y_c)(a, y_a)$ $\langle y_b \rangle$ $\langle z_a \rangle$ $\langle y_d \rangle$

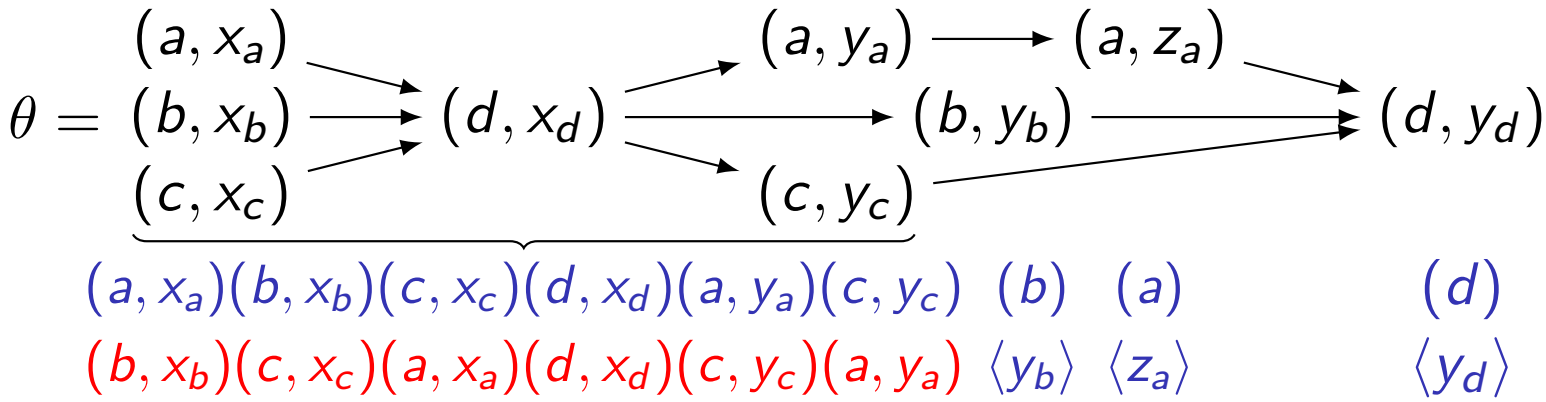
Distributable sequential strategy for Player 2



Theorem (Ștefănescu-Esparza-Muscholl, Mukund)

*There exists an **distributed strategy** for Player 2 iff there exists a **safe-branching* sequential strategy** for Player 2.*

Distributable sequential strategy for Player 2



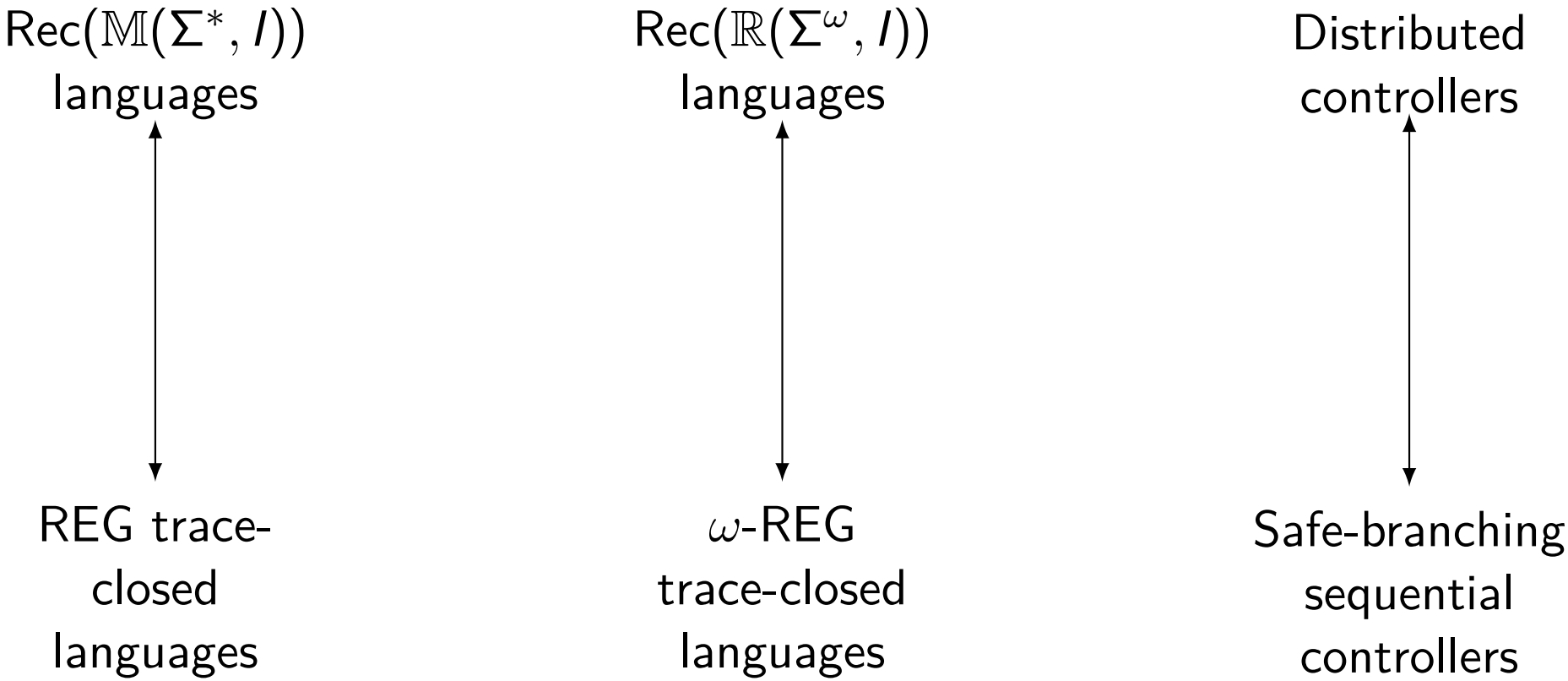
Theorem (Ștefănescu-Esparza-Muscholl, Mukund)

*There exists an **distributed strategy** for Player 2 iff there exists a **safe-branching* sequential strategy** for Player 2.*

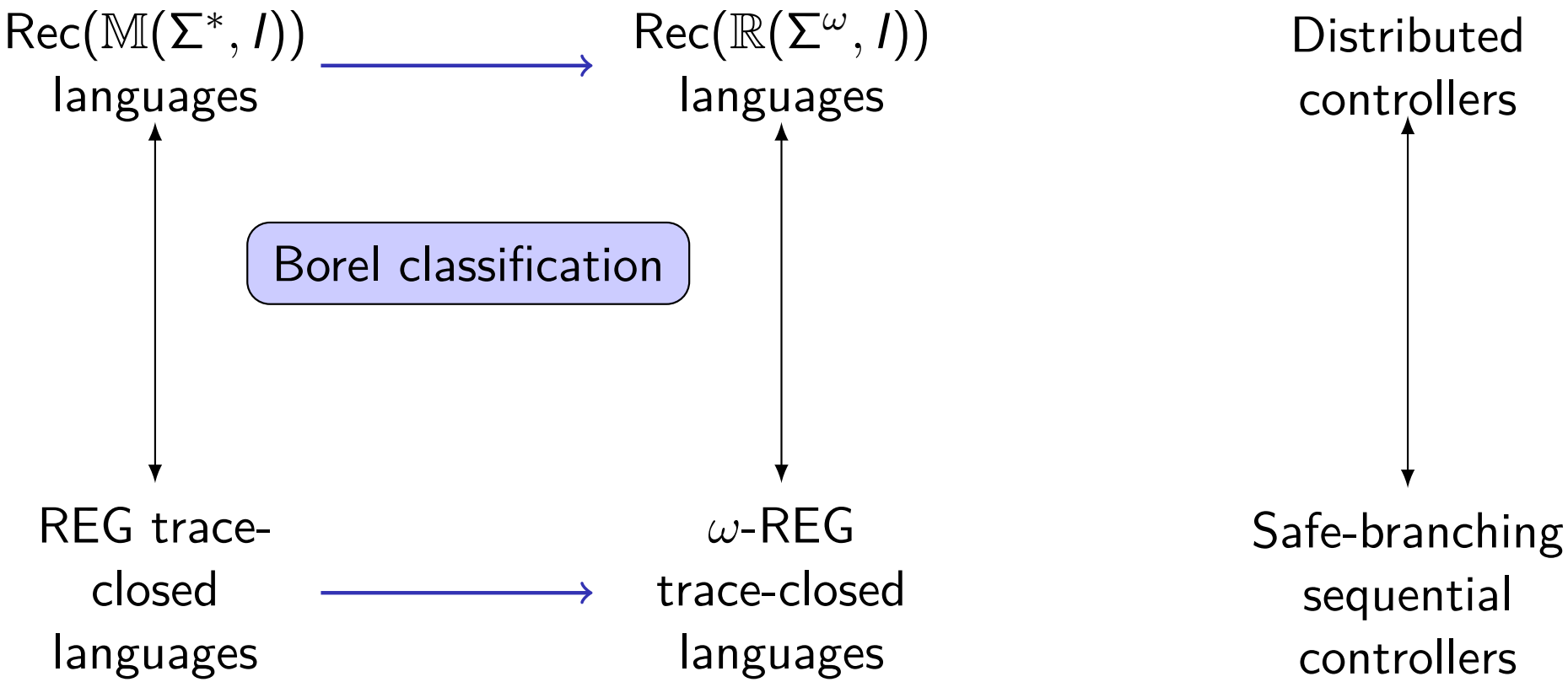
Ongoing work

Complete procedures for constructing safe-branching sequential strategies for various classes of trace-closed games.

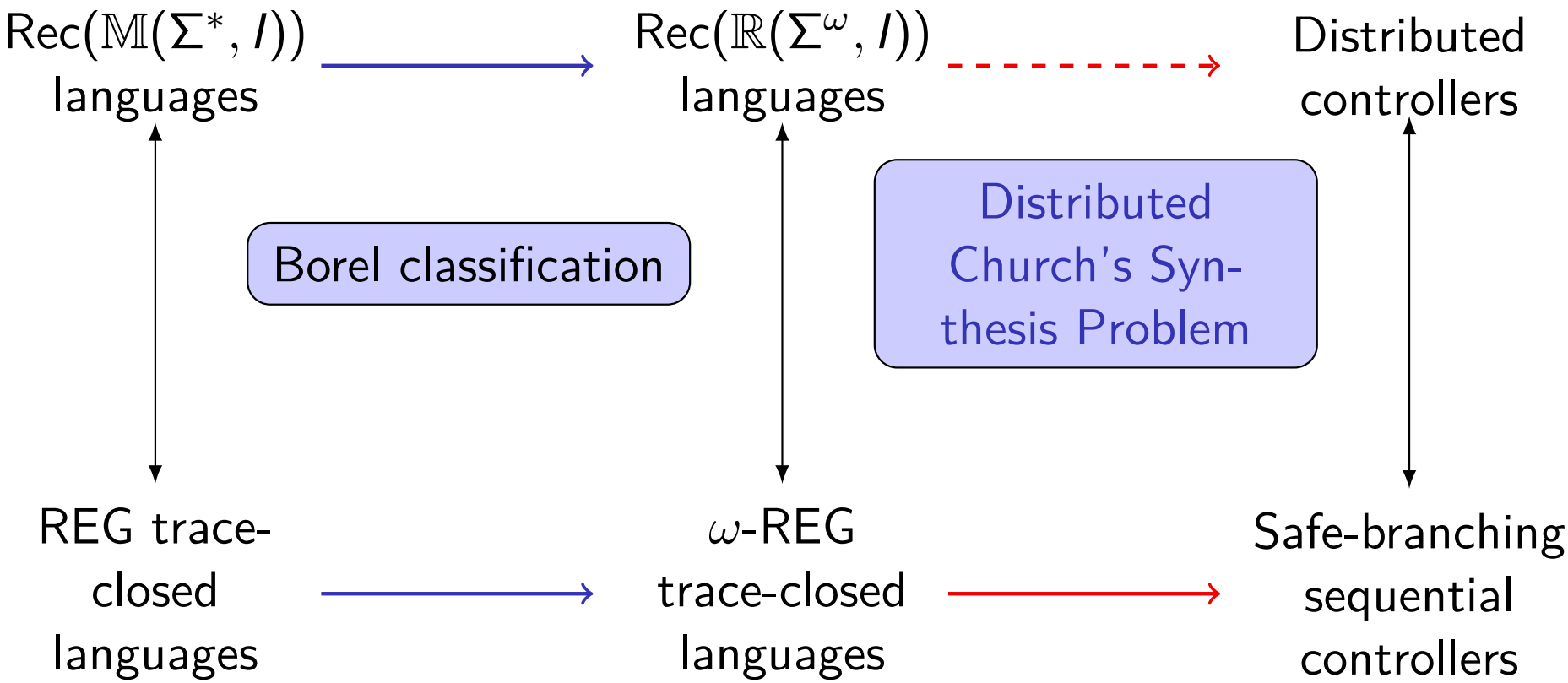
Summary


Rec($\mathbb{M}(\Sigma^*, I)$)
languages
REG trace-closed languages
Rec($\mathbb{R}(\Sigma^\omega, I)$)
languages
ω-REG trace-closed languages
Distributed controllers
Safe-branching sequential controllers

Summary


Rec($\mathbb{M}(\Sigma^*, I)$)
languages
Rec($\mathbb{R}(\Sigma^\omega, I)$)
languages
Distributed
controllers
Borel classification
REG trace-
closed
languages
ω-REG
trace-closed
languages
Safe-branching
sequential
controllers

Summary


Rec($\mathbb{M}(\Sigma^*, I)$) languages
Rec($\mathbb{R}(\Sigma^\omega, I)$) languages
Distributed controllers
Borel classification
Distributed Church's Synthesis Problem
REG trace-closed languages
ω-REG trace-closed languages
Safe-branching sequential controllers