Separating regular languages by piecewise testable and unambiguous languages

Thomas Place, Lorijn van Rooijen, Marc Zeitoun

LaBRI · Univ. Bordeaux · CNRS

September, 2013 · Highlights of Logic, Games and Automata

Separation problem

- S is a fixed class of languages.

  In this talk, $\mathsf{S} = \mathcal{B}\Sigma_1(<)$, $\mathsf{FO}^2(<)$.

Separation problem

- S is a fixed class of languages.

  In this talk, S $= \mathcal{B}\Sigma_1(<)$, $\mathsf{FO}^2(<)$.

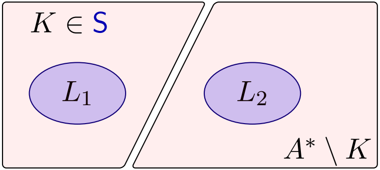
- L_1, $L_2 \in$ Reg are S-**separable** iff there exists

  $K \in$ S such that

- S is a fixed class of languages.

  In this talk, $\mathsf{S} = \mathcal{B}\Sigma_1(<)$, $\mathsf{FO}^2(<)$.

- L_1, $L_2 \in$ Reg are **S-separable** iff there exists $K \in \mathsf{S}$ such that



Separation problem

- Decision problem: Given L_1, $L_2 \in$ Reg, are they S-separable?

Separation problem

- Decision problem: Given L_1, $L_2 \in$ Reg, are they S-separable?
- Can we compute a separator?

Separation problem

- Decision problem: Given L_1, $L_2 \in$ Reg, are they S-separable?
- Can we compute a separator?
- What is the complexity of the decision problem?
  And of computing a separator?

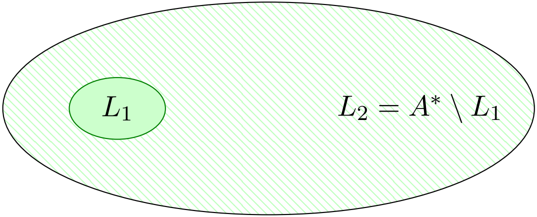
Motivation

- Captures discriminative power of logics

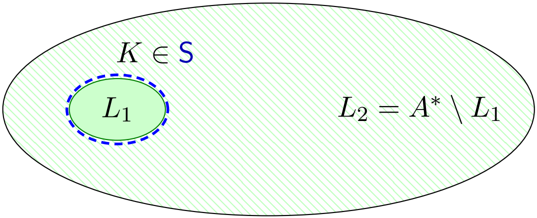
Motivation

- Captures discriminative power of logics
- Generalization of membership problem

- Captures discriminative power of logics
- Generalization of membership problem



- Captures discriminative power of logics
- Generalization of membership problem



Classes of separators

- Piecewise testable languages: definable by $\mathcal{B}\Sigma_1(<)$ formulas (recall previous talk).
- Unambiguous languages: definable by $\mathsf{FO}^2(<)$ formulas.

Classes of separators

- Piecewise testable languages: definable by $\mathcal{B}\Sigma_1(<)$ formulas (recall previous talk).
- Unambiguous languages: definable by $\mathsf{FO}^2(<)$ formulas.

Membership is decidable for both $\mathcal{B}\Sigma_1(<)$ and $\mathsf{FO}^2(<)$ languages.

Simon '75
Schützenberger '76, Thérien, Wilke '98

Stratification of S

- In general, there is no smallest separator in S.

  $\rightarrow$ Restriction of S to quantifier depth k: S$[k]$.

Stratification of S

- In general, there is no smallest separator in S.

  $\rightarrow$ Restriction of S to quantifier depth k: S$[k]$.

- If two languages are S$[k]$-separable, there is a smallest separator in S$[k]$.

Stratification of S

- In general, there is no smallest separator in S.

  $\rightarrow$ Restriction of S to quantifier depth k: S$[k]$.

- If two languages are S$[k]$-separable, there is a smallest separator in S$[k]$.

- Eg. $(a^2)^*$ and $b(b^2)^*$ have no smallest $\mathcal{B}\Sigma_1(<)$-separator, while $a^* \setminus \{a, a^3, a^5\}$ is the smallest $\mathcal{B}\Sigma_1(<)[6]$-separator.

Quantifier depth provides indexed equivalence relations for $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ languages:

Computing smallest separator for S$[k]$

Quantifier depth provides indexed equivalence relations for $\mathcal{B}\Sigma_1(<)$ resp. FO$^2(<)$ languages:

$$w_1 \sim_k w_2 \quad \Leftrightarrow \quad w_1 \text{ and } w_2 \text{ satisfy the same } \mathcal{B}\Sigma_1(<) \text{ resp. } \mathsf{FO}^2(<) \text{ formulas up to depth } k.$$

Computing smallest separator for S$[k]$

Quantifier depth provides indexed equivalence relations for $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ languages:

$$w_1 \sim_k w_2 \quad \Leftrightarrow \quad \begin{array}{l} w_1 \text{ and } w_2 \text{ satisfy the same } \mathcal{B}\Sigma_1(<) \\ \text{resp. } \mathsf{FO}^2(<) \text{ formulas up to depth } k. \end{array}$$

L is a $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ language

$$\Leftrightarrow$$

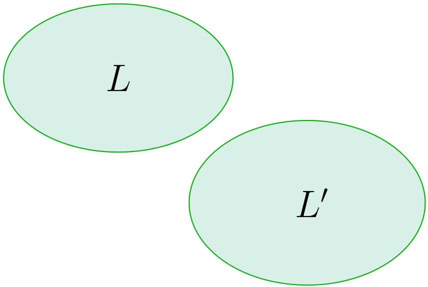
L is a union of $\sim_k$-classes for some $k \in \mathbb{N}$.

Computing smallest separator for S$[k]$

Increasing k refines the smallest potential separator:

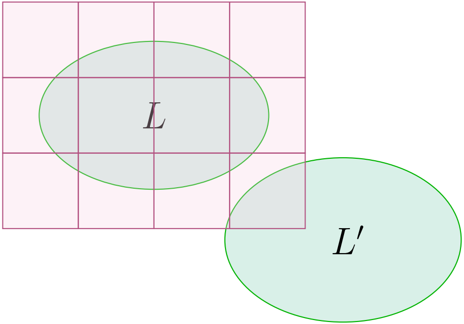
Computing smallest separator for S$[k]$

Increasing k refines the smallest potential separator:

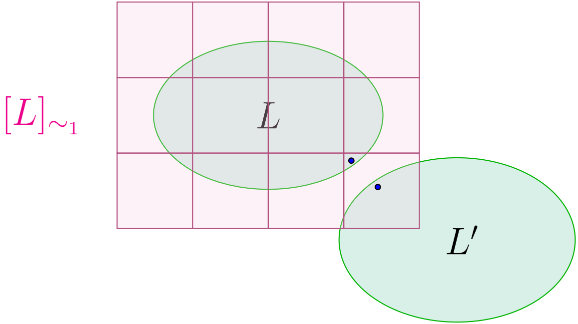


Increasing k refines the smallest potential separator:

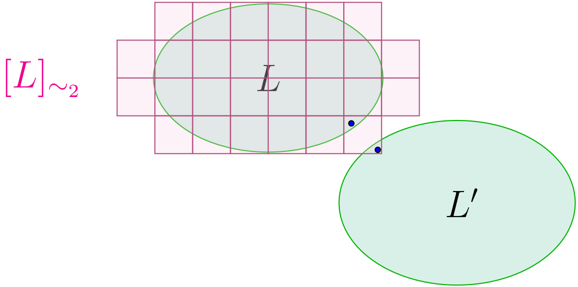
$[L]_{\sim_1}$



Increasing k refines the smallest potential separator:

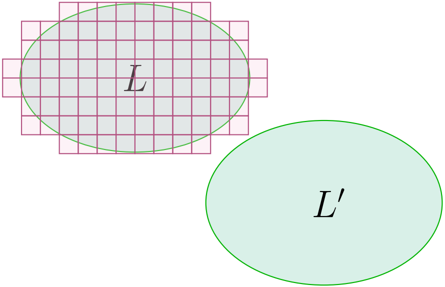


Increasing k refines the smallest potential separator:



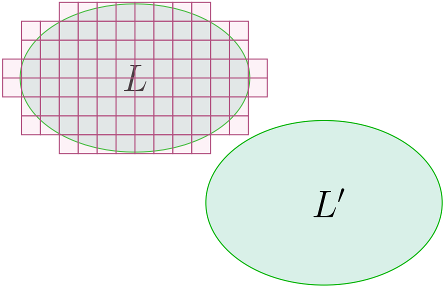
Increasing k refines the smallest potential separator:

$[L]_{\sim_3}$



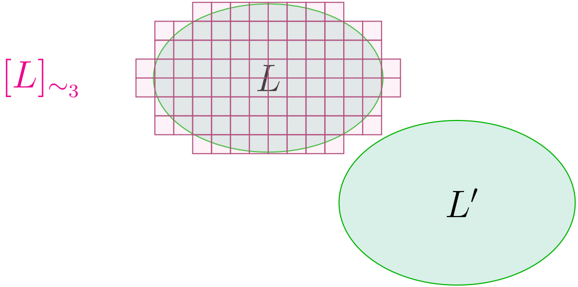
Increasing k refines the smallest potential separator:

$[L]_{\sim_3}$



Refining $[L]_{\sim_k}$ gives a semi-algorithm for separability.

Increasing k refines the smallest potential separator:



Refining $[L]_{\sim_k}$ gives a semi-algorithm for separability.

From which level of refinement can we conclude non-separability?

Testing non-separability

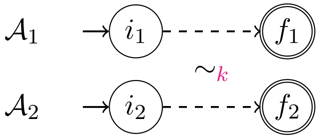
A pair of words $w_1 \in L_1, w_2 \in L_2$ is a k-witness of non-separability provided

$$w_1 \sim_k w_2.$$

A pair of words $w_1 \in L_1, w_2 \in L_2$ is a k-witness of non-separability provided

$$w_1 \sim_k w_2.$$

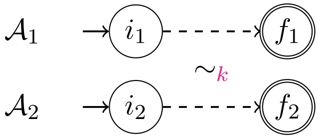
Abstraction: tuples (i_1, f_1, i_2, f_2)



A pair of words $w_1 \in L_1, w_2 \in L_2$ is a k-witness of non-separability provided

$$w_1 \sim_k w_2.$$

Abstraction: tuples (i_1, f_1, i_2, f_2)



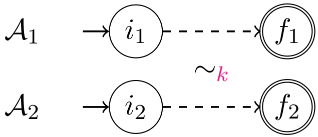
- L_1, L_2 are k-separable $\Leftrightarrow$ $\{k\text{-witnesses}\} = \emptyset$.
- $\{k+1\text{-witnesses}\} \subseteq \{k\text{-witnesses}\}$.

Testing non-separability

A pair of words $w_1 \in L_1, w_2 \in L_2$ is a k-witness of non-separability provided

$$w_1 \sim_k w_2.$$

Abstraction: tuples (i_1, f_1, i_2, f_2)



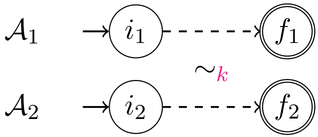
- L_1, L_2 are k-separable $\Leftrightarrow$ $\{k$-witnesses$\} = \varnothing$.
- $\{k+1$-witnesses$\} \subseteq \{k$-witnesses$\}$.
- Limit behaviour?

Testing non-separability

A pair of words $w_1 \in L_1, w_2 \in L_2$ is a k-witness of non-separability provided

$$w_1 \sim_k w_2.$$

Abstraction: tuples (i_1, f_1, i_2, f_2)



- L_1, L_2 are k-separable $\Leftrightarrow$ $\{k\text{-witnesses}\} = \varnothing$.
- $\{k+1\text{-witnesses}\} \subseteq \{k\text{-witnesses}\}$.
- Limit behaviour?

  We can compute K such that

$$\text{limit} = \varnothing \quad \Leftrightarrow \quad \{K\text{-witnesses}\} = \varnothing$$

1. Verify that the languages are not separable by any language of a sufficient, computable, index K.
2. Find a pattern in the automata producing k-witnesses for arbitrarily large k. (For $\mathcal{B}\Sigma_1(<)$: same as in previous talk)

Testing non-separability

1. Verify that the languages are not separable by any language of a sufficient, computable, index K.

2. Find a pattern in the automata producing k-witnesses for arbitrarily large k. (For $\mathcal{B}\Sigma_1(<)$: same as in previous talk)

   Patterns in the automata, independently of the bound K, provide

   - Better complexity
   - A yes/no answer for separability
   - But no separator

Main result

Theorem

For S $= \mathcal{B}\Sigma_1(<)$, $\mathsf{FO}^2(<)$, *we can compute* $K \in \mathbb{N}$ *such that TFAE*

i. L_1, L_2 *are* S*-separable.*
ii. L_1, L_2 *are* $\mathsf{S}[K]$*-separable.*
iii. $[L_1]_{\sim_K}$ *separates* L_1 *from* L_2.
iv. $\mathcal{A}_1, \mathcal{A}_2$ *do not contain a pattern witnessing non-separability.*

Main result

Condition iv. yields the following complexity results:

Theorem

Given NFAs $\mathcal{A}_1, \mathcal{A}_2$, one can determine whether the languages $L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are

- $\mathcal{B}\Sigma_1(<)$*-separable in* **PTIME**,
- $\mathsf{FO}^2(<)$*-separable in* **EXPTIME**,

with respect to $|Q_1|$,$|Q_2|$,$|A|$.

Future work

- Obtain tight bounds on the size of $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ -separators

Future work

- Obtain tight bounds on the size of $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ - separators
- Efficient computation of the separators

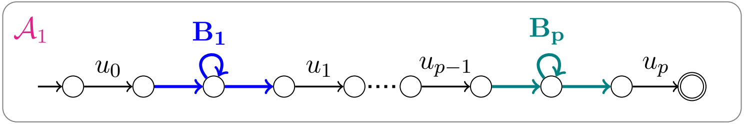
Future work

- Obtain tight bounds on the size of $\mathcal{B}\Sigma_1(<)$ resp. $\mathsf{FO}^2(<)$ - separators
- Efficient computation of the separators
- Consider other classes S

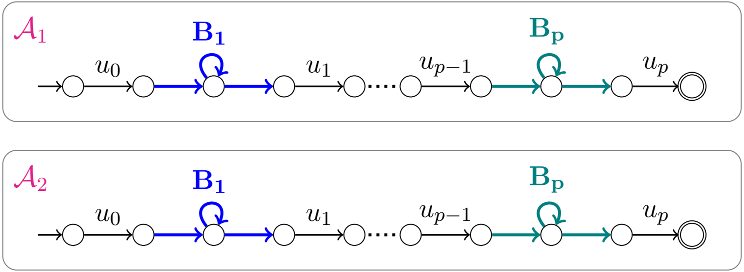
Thank you

$L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are not $\mathcal{B}\Sigma_1(<)$-separable
iff
both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path:

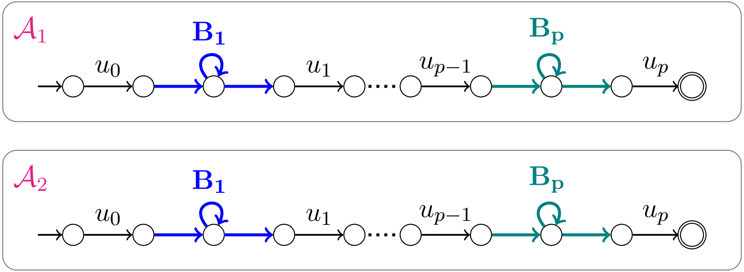
$L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are not $\mathcal{B}\Sigma_1(<)$-separable
iff
both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path:



$L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are not $\mathcal{B}\Sigma_1(<)$-separable
iff
both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path:



$L(\mathcal{A}_1)$ and $L(\mathcal{A}_2)$ are not $\mathcal{B}\Sigma_1(<)$-separable
iff
both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path:



This can be determined in PTIME$(|Q_1|, |Q_2|, |A|)$.

Detecting forbidden patterns

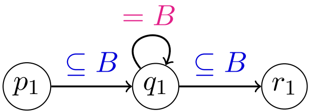
One can determine in $\mathsf{PTIME}(|Q_1|, |Q_2|, |A|)$ whether $\exists\ (\vec{u}, \vec{B})$ such that both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path.

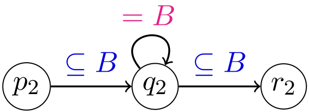
Detecting forbidden patterns

One can determine in PTIME$(|Q_1|, |Q_2|, |A|)$ whether $\exists\ (\vec{u}, \vec{B})$ such that both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path.

- By adding meta-transitions in $\mathcal{A}_i$, finding a $(\vec{u}, \vec{B})$-witness reduces to:

  Given states p_i, q_i, r_i of $\mathcal{A}_i$, is there $B \subseteq A$ st. the paths





  occur in $\mathcal{A}_1$ resp. $\mathcal{A}_2$?

Detecting forbidden patterns

One can determine in PTIME$(|Q_1|, |Q_2|, |A|)$ whether $\exists\ (\vec{u}, \vec{B})$ such that both $\mathcal{A}_1$ and $\mathcal{A}_2$ have a $(\vec{u}, \vec{B})$-path.

- By adding meta-transitions in $\mathcal{A}_i$, finding a $(\vec{u}, \vec{B})$-witness reduces to:

  Given states p_i, q_i, r_i of $\mathcal{A}_i$, is there $B \subseteq A$ st. the paths

  occur in $\mathcal{A}_1$ resp. $\mathcal{A}_2$?

- This is in PTIME: iteratively use Tarjan's algorithm.

Detecting forbidden patterns

- If B exists, $B \subseteq C_1 \stackrel{\text{def}}{=} \mathsf{alph\_scc}(q_1, \mathcal{A}_1) \cap \mathsf{alph\_scc}(q_2, \mathcal{A}_2)$ (LINEAR)
- Restrict the automata to alphabet C_1, and repeat the process:

$$C_{i+1} \stackrel{\text{def}}{=} \mathsf{alph\_scc}(q_1, \mathcal{A}_1 \restriction_{C_i}) \cap \mathsf{alph\_scc}(q_2, \mathcal{A}_2 \restriction_{C_i}).$$

- After $n \leqslant |A|$ iterations, $C_n = C_{n+1}$.
  - If $C_n = \varnothing$, the answer is no.
  - If $C_n \neq \varnothing$, it is the maximal possible B with $(= B)$-loops around q_1, q_2.
- Then, determine the remaining paths. (LINEAR)
- Overall LINEAR algorithm.