The Separation Problem: An introduction and transfer theorems

T. Place, L. van Rooijen, Marc Zeitoun

LaBRI, Université Bordeaux, CNRS

SDA2 2015, MLV — 10/4/2015

Framework and Motivations

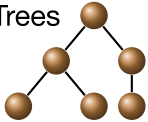
Structures

Descriptive Formalism

Words

ababcbaa

Trees



First-Order Logic (**FO**)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables **FO** (**FO**$_2$)

Fragments Σ_i, $\mathcal{B}\Sigma_i$

Locally Threshold Testable (**LTT**)

Framework and Motivations

Structures

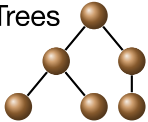
Descriptive Formalism

Express Properties

Words

ababcbaa

Trees



First-Order Logic (**FO**)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables **FO** (**FO**$_2$)

Fragments Σ_i, $\mathcal{B}\Sigma_i$

Locally Threshold Testable (**LTT**)

Framework and Motivations

Structures

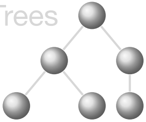
Descriptive Formalism

Express Properties

Words

ababcbaa

Trees



First-Order Logic (**FO**)

Piecewise Testable ($\mathcal{B}\Sigma_1$)

2-Variables **FO** (**FO**$_2$)

Fragments Σ_i, $\mathcal{B}\Sigma_i$

Locally Threshold Testable (**LTT**)

For this talk

First-order logic and MSO on words

First-order logic, with only the linear order '$<$'.

$$a\ b\ b\ b\ c\ a\ a\ a\ c\ a$$

First-order logic and MSO on words

First-order logic, with only the linear order '$<$'.

$$\begin{array}{cccccccccc} a & b & b & b & c & a & a & a & c & a \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

- A word is a sequence of labeled positions.
- Positions can be quantified: $\exists x \varphi$.
- Unary predicates $a(x), b(x), c(x)$ testing the label of position x.
- One binary predicate: the linear-order $x < y$.

First-order logic and MSO on words

First-order logic, with only the linear order '$<$'.

$$\begin{array}{cccccccccc} a & b & b & b & c & a & a & a & c & a \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

- A word is a sequence of labeled positions.
- Positions can be quantified: $\exists x \varphi$.
- Unary predicates $a(x), b(x), c(x)$ testing the label of position x.
- One binary predicate: the linear-order $x < y$.

Example: every a comes after some b

$$\forall x \; a(x) \Rightarrow \exists y \; (b(y) \wedge (y < x))$$

First-order logic and MSO on words

First-order logic, with only the linear order '$<$'.

$$\begin{array}{cccccccccc} a & b & b & b & c & a & a & a & c & a \\ 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{array}$$

- A word is a sequence of labeled positions.
- Positions can be quantified: $\exists x \varphi$.
- Unary predicates $a(x), b(x), c(x)$ testing the label of position x.
- One binary predicate: the linear-order $x < y$.

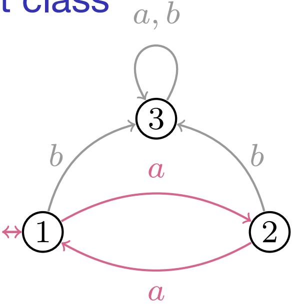
Example: every a comes after some b

$$\forall x\ a(x) \Rightarrow \exists y\ (b(y) \wedge (y < x))$$

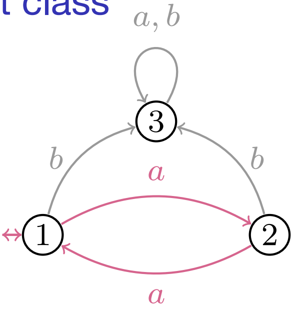
- MSO Logic: idem + quantify over sets of positions $X, Y, Z \ldots$

Regular languages: a robust class

$$L = (aa)^*$$



Regular languages: a robust class

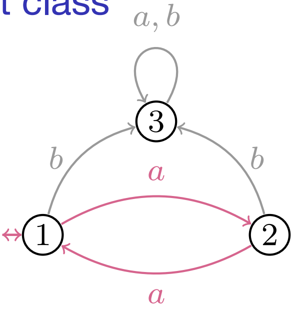


$$L = (aa)^*$$

$$\begin{aligned} &\forall x\ a(x) \ \wedge \\ \exists X\ &[\mathsf{even}(X) \ \wedge \\ &\quad \forall x\ (x \in X)] \end{aligned}$$

Regular languages: a robust class

$$L = (aa)^*$$

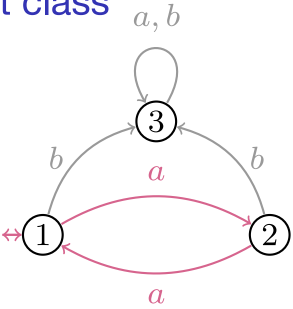


	1	2	3
$\underline{\varepsilon}$	1	2	3
$\underline{a}$	2	1	3
$\underline{b}$	3	3	3

$$\forall x\ a(x) \ \wedge$$
$$\exists X\ [\mathsf{even}(X) \ \wedge$$
$$\forall x\ (x \in X)]$$

Regular languages: a robust class

$$L = (aa)^*$$



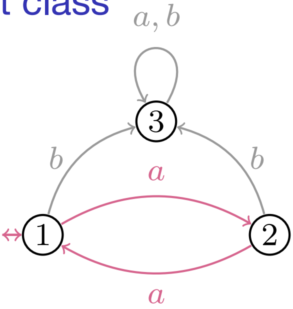
$$\forall x\, a(x) \;\wedge\; \exists X\, \big[\mathsf{even}(X) \;\wedge\; \forall x\, (x \in X)\big]$$

	1	2	3
$\underline{\varepsilon}$	1	2	3
$\underline{a}$	2	1	3
$\underline{b}$	3	3	3

$\simeq \mathbb{Z}/2\mathbb{Z} \cup \{0\}$

Regular languages: a robust class

$$L = (aa)^*$$



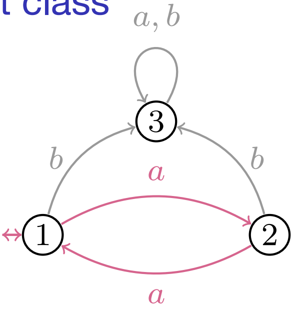
$$\begin{array}{l} \quad \forall x\ a(x) \ \wedge \\ \exists X\ [\text{even}(X)\ \wedge \\ \qquad \forall x\ (x \in X)] \end{array}$$

	1	2	3
$\underline{\varepsilon}$	1	2	3
$\underline{a}$	2	1	3
$\underline{b}$	3	3	3

$$A^* \xrightarrow{\alpha} \simeq \mathbb{Z}/2\mathbb{Z} \cup \{0\}$$

Regular languages: a robust class

$$L = (aa)^*$$



$$\forall x\; a(x) \;\wedge\; \exists X\,\big[\mathsf{even}(X) \;\wedge\; \forall x\,(x \in X)\big]$$

	1	2	3
$\underline{\varepsilon}$	1	2	3
$\underline{a}$	2	1	3
$\underline{b}$	3	3	3

$$A^* \xrightarrow{\alpha} \;\simeq \mathbb{Z}/2\mathbb{Z} \cup \{0\}$$

$$L = \alpha^{-1}(\{\varepsilon\}).$$

Kleene-Büchi-Elgot-Trakhtenbrot Theorem

$$\begin{tikzcd}
\text{Regular expressions} \arrow[r, leftrightarrow] \arrow[d, leftrightarrow] & \text{Finite automata} \arrow[d, leftrightarrow] \\
\text{MSO formulas} \arrow[r, leftrightarrow] & \begin{array}{c}\text{Recognition by a} \\ \text{finite monoid} \\ L = \alpha^{-1}(F)\end{array}
\end{tikzcd}$$

- Generic.
- Easy.

Why look at FO and fragments?

- Simple formulas are better (algorithmically).
- Some parameters making formulas complex:
  - Second order quantification,
  - Number of quantifier alternations,
  - Allowed predicates,
  - Number of variable names.

Why look at FO and fragments?

- Simple formulas are better (algorithmically).
- Some parameters making formulas complex:
  - Second order quantification,
  - Number of quantifier alternations,
  - Allowed predicates,
  - Number of variable names.

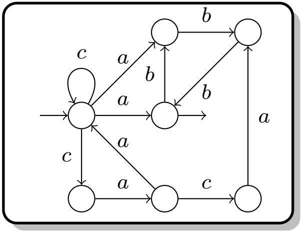
Membership Problem for a fragment $\mathcal{F}$

- **INPUT** A language L.
- **QUESTION** Is L expressible in $\mathcal{F}$?

First problem: Membership

Membership problem for a fragment $\mathcal{F}$

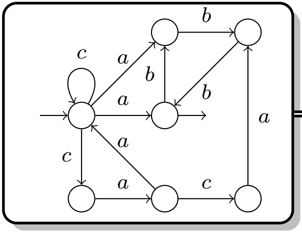
- **INPUT** A language L.
- **QUESTION** Is L expressible in $\mathcal{F}$?



First problem: Membership

Membership problem for a fragment $\mathcal{F}$

- **INPUT** A language L.
- **QUESTION** Is L expressible in $\mathcal{F}$?

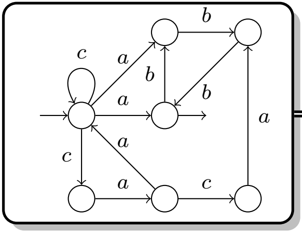


Can it be defined with an $\mathcal{F}$ formula?

First problem: Membership

Membership problem for a fragment $\mathcal{F}$

- **INPUT** A language L.
- **QUESTION** Is L expressible in $\mathcal{F}$?



Can it be defined with an $\mathcal{F}$ formula?

Schützenberger'65, McNaughton and Papert'71

For L a regular language, the following are equivalent:

- L is **FO**-definable.
- The syntactic monoid of L satisfies $u^{\omega+1} = u^{\omega}$.

Reprinted from Information and Control, Volume 8, No. 2, April 1965

INFORMATION AND CONTROL **8**, 190–194 (1965)

On Finite Monoids Having Only Trivial Subgroups

M. P. Schützenberger

An alternative definition is given for a family of subsets of a free monoid that has been considered by Trahtenbrot and by McNaughton.

I. INTRODUCTION

Let X^* be the free monoid generated by a fixed set X and let $\mathbf{Q}$ be the least family of subsets of X^* that satisfies the following conditions (K1) and (K2):

(K1). $X^* \in \mathbf{Q}$; $\{e\} \in \mathbf{Q}$ (e is the neutral element of X^*); $X' \in \mathbf{Q}$ for any $X' \subset X$.

(K2). If A_1 and A_2 belong to $\mathbf{Q}$, then $A_1 \cup A_2$,

$$A_1 \backslash A_2 \ (= \{f \in A_1 : f \notin A_2\})$$

and $A_1 \cdot A_2$ ($= \{ff' \in X^* : f \in A_1 ; f' \in A_2\}$) belong to $\mathbf{Q}$.

With different notations, $\mathbf{Q}$ has been studied in Trahtenbrot (1958) and, within a wider context, in McNaughton (1960). According to Eggan (1963), $\mathbf{Q}$ contains, for suitable X, sets of arbitrarily large star-height (cf. Section IV below).

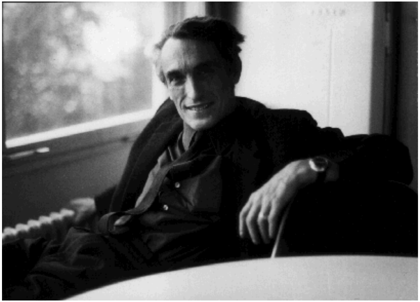
For each natural number n, let $\Gamma(n)$ denote the family of all epimorphisms γ of X^* such that **Card** $\gamma X^* \leqq n$ and that γX^* has only trivial subgroups (i.e., $\gamma f^n = \gamma f^{n+1}$ for all $f \in X^*$, cf. Miller and Clifford (1956)).

Main Property. *$\mathbf{Q}$ is identical with the union $\mathbf{Q}'$ over all n of the families*

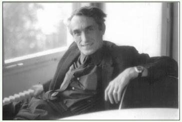
$$\mathbf{Q}'(n) = \{A \subset X^* : \gamma^{-1}\gamma A = A ; \gamma \in \Gamma(n)\}$$
$$(= \{\gamma^{-1}M' : M' \subset \gamma X^* ; \gamma \in \Gamma(n)\}).$$

As an application, if $A, A' \subset X^*$ are such that for at least one triple $f, f', f'' \in X^*$, both $\{n \in \mathbf{N} : f'f^nf'' \in A\}$ and $\{n \in \mathbf{N} : f'f^nf'' \in A'\}$ are infinite sets of integers, we can conclude that no $B \in \mathbf{Q}$ satisfies $A \subset B$ and $A' \subset X^* \backslash B$.

190



Édition des œuvres complètes de Marcel-Paul Schützenberger



Marcel-Paul Schützenberger, Oberwolfach (1975)

Marcel-Paul Schützenberger (1920-1996) est le fondateur de l'informatique théorique en France. Membre de l'Académie des Sciences, il a eu un rayonnement international important en mathématiques et en informatique.

De nombreux membres de l'Institut Gaspard-Monge sont ses anciens élèves ou disciples, ou de ses descendants. Il continue à influencer directement ou indirectement de nombreuses recherches en cours.

Jean Berstel, Alain Lascoux et Dominique Perrin ont entrepris, épaulés par les anciens élèves de Marcel-Paul Schützenberger, une édition de ses œuvres complètes. Elle se présente sous la forme de treize volumes, chacun d'environ 250 pages.

Dans un deuxième temps, une sélection de ses œuvres choisies en mathématiques et en informatique suivra, et sera proposée à la publication d'une société savante.

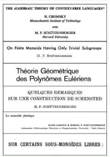
Maîtres et amis

Albert Châtelet, Georges Darmois, Maurice Fréchet, Raymond Turpin, René de Possel, Paul Dubreil, Jacques Riguet, Gilbert Gadoffre, Benoit Mandelbrot, Claude Berge

Coauteurs

Noam Chomsky, Alain Connes, André Lichnerowicz, Samuel Eilenberg, Steven Sherman, Roger Lyndon, François Blanchard, Christophe Reutenauer

Articles fondateurs



THE ALGEBRAIC THEORY OF CONTEXT-FREE LANGUAGES*
N. CHOMSKY
Massachusetts Institute of Technology
AND
M. P. SCHÜTZENBERGER
Harvard University

On Finite Monoids Having Only Trivial Subgroups
M. P. SCHÜTZENBERGER

Théorie Géométrique des Polynômes Eulériens

QUELQUES REMARQUES SUR UNE CONSTRUCTION DE SCHENSTED
M. P. SCHÜTZENBERGER

Le monoïde plaxique
ALAIN LASCOUX & MARCEL P. SCHÜTZENBERGER

SUR CERTAINS SOUS-MONOÏDES LIBRES ;

Élèves

Maurice Gross, André Lentin, Louis Nolin, Dominique Foata, Maurice Nivat, Pierre-André Picon, Christian Choffrut, Robert Cori, Michel Fliess, Jean-François Perrot, Jean-Yves Thibon, Xavier Viennot

Œuvres complètes

(13 volumes)

Marcel-Paul Schützenberger

ŒUVRES COMPLÈTES

éditées par
Jean Berstel, Alain Lascoux et Dominique Perrin

*

Tome 1 : 1943-1949

Éditeurs

Jean Berstel, Alain Lascoux, Dominique Perrin

Contact : Jean.Berstel, Alain.Lascoux, Dominique.Perrin

http://igm.univ-mlv.fr/~berstel/Schutzenberger/

29/4/2009

Quantifier alternation

Level i: Σ_i

For all i, a Σ_i formula is

$$\underbrace{\exists x_1, \dots, x_{n_1} \forall y_1, \dots, y_{n_2} \cdots\cdots}_{i \text{ blocks (starting with } \exists)} \underbrace{\varphi(\bar{x}, \bar{y}, \dots)}_{\text{quantifier-free}}$$

Quantifier alternation

Level i: Σ_i

For all i, a Σ_i formula is

$$\underbrace{\exists x_1, \dots, x_{n_1} \forall y_1, \dots, y_{n_2} \cdots\cdots}_{i \text{ blocks (starting with } \exists)} \underbrace{\varphi(\bar{x}, \bar{y}, \dots)}_{\text{quantifier-free}}$$

Σ_i is not closed under complement $\Rightarrow$ we get two other classes:

Level i: Π_i

Negation of a Σ_i formula:

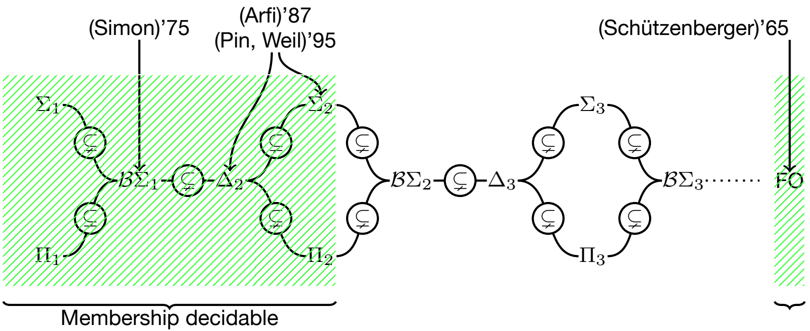
$$\underbrace{\forall x_1, \dots, x_{n_1} \exists y_1, \dots, y_{n_2} \cdots}_{i \text{ blocks (starting with } \forall)} \quad \varphi$$

Level i: $\mathcal{B}\Sigma_i$

Boolean combinations of Σ_i (and Π_i) formulas.

FO Quantifier alternation hierarchy

State of the art in 2013



Several Hierarchies

- A fragment is obtained by restricting
  - Number of quantifier alternations,
  - Allowed predicates,
  - Number of variable names.

- FO$(<)$, FO$(<, +1)$, FO$(<, +1, min, max)$: same expressiveness.

  With restricted alternation, this yields distinct fragments.

$$\Sigma_1(<), \quad \Sigma_1(<, +1), \text{ and } \Sigma_1(<, +1, min, max)$$

Why we want more than membership

If the membership answer for L

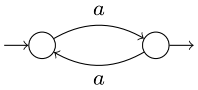
- is **YES**
  - All "subparts" of the minimal automaton of L are $\mathcal{F}$-definable.
- is **NO**, then even if $\mathcal{F}$ can talk about L:
  - We have almost no information.
  - Eg, for FO, defining L requires differentiating some u^{ω} and $u^{\omega+1}$.

Beyond membership: Separation

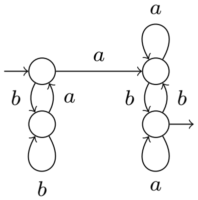
Decide the following problem:

Take **2** regular languages L_1, L_2

L_1



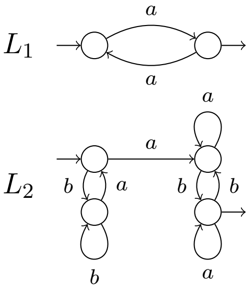
L_2



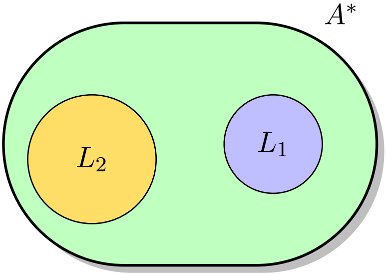
Beyond membership: Separation

Decide the following problem:

Take **2** regular languages L_1, L_2



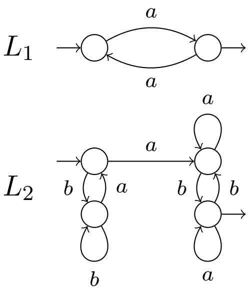
Can L_1 be separated from L_2 with an $\mathcal{F}$ formula?



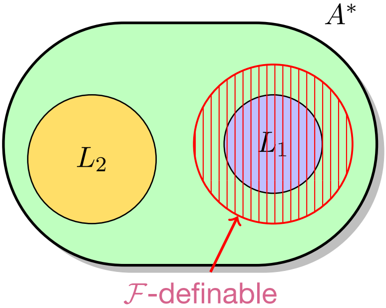
Beyond membership: Separation

Decide the following problem:

Take **2** regular languages L_1, L_2



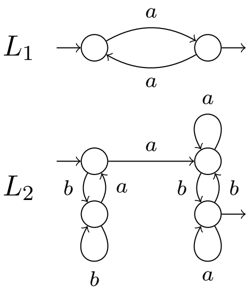
Can L_1 be separated from L_2 with an $\mathcal{F}$ formula?



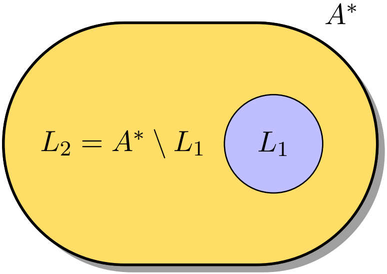
Beyond membership: Separation

Membership can be formally reduced to separation

Take **2** regular languages L_1, L_2



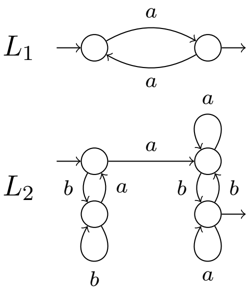
Can L_1 be separated from L_2 with an $\mathcal{F}$ formula?



Beyond membership: Separation

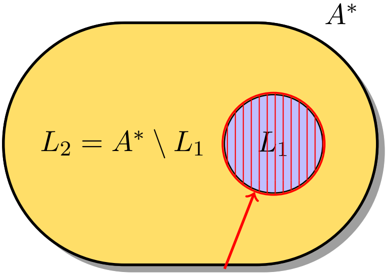
Membership can be formally reduced to separation

Take **2** regular languages L_1, L_2



$\Longrightarrow$

Can L_1 be separated from L_2 with an $\mathcal{F}$ formula?



$\mathcal{F}$-separable from complement

$\Leftrightarrow$

$\mathcal{F}$-definable

Motivations for separation

- More general: able to extract information for all languages.
- Cannot start from canonical object for the (unknown) separator.
- Therefore, may give insight to solve harder problems.

Motivations for separation

- More general: able to extract information for all languages.
- Cannot start from canonical object for the (unknown) separator.
- Therefore, may give insight to solve harder problems.

- 2 **transfer results**:
  - decidability of separation for level Σ_i of the quantifier alternation hierarchy entails decidability of membership for Σ_{i+1}.
  - decidability of separation preserved when enriching $\mathcal{F}$ with +1.

Motivations for separation

- More general: able to extract information for all languages.
- Cannot start from canonical object for the (unknown) separator.
- Therefore, may give insight to solve harder problems.

- 2 **transfer results**:
  - decidability of separation for level Σ_i of the quantifier alternation hierarchy entails decidability of membership for Σ_{i+1}.
  - decidability of separation preserved when enriching $\mathcal{F}$ with +1.

$\Rightarrow$ We shouldn't restrict ourselves to membership.

Related work

- Separation **already considered** in an algebraic framework.
- First result by K. Henckell '88 for **FO**, then for other fragments.
- Transfer $\mathcal{F} \to \mathcal{F}[+1]$: H. Straubing '85, B. Steinberg '01 (sep.).
- Purely algebraic proofs, hiding combinatorial & logical intuitions.
- Want: Simpler, combinatorial proofs.

An already known result for FO: Henckell '88

- Simple algorithm.
- Easy correctness proof.
- Intricate completeness proof.

Guide to the paper

Chapter 1. *Elementary definitions and notation* should be omitted on first reading and used as a reference as needed.

Chapter 2. *The* Pl-*functor* defines pointlike sets in a general setting and shows by an abstract compactness argument that $\mathrm{Pl}(S)$ can be computed by an aperiodic semigroup.

Chapter 3. *Definition of* $C^\omega(S)$ *and* H^ω defines $C^\omega(S)$, a collection of pointlike sets, in a constructive manner. H^ω is the 'blow-up-operator' that we will use in Chapter 5 to show $C^\omega(S) = \mathrm{Pl}(S)$. It has some examples in the end.

Chapter 4. *The Rhodes-expansion* defines the tools needed in Chapter 5.

Chapter 5. $C^\omega(S) = \mathrm{Pl}(S)$ shows the main result by actually constructing a relation $S \xrightarrow{R} \mathrm{CP}(S)$ computing $C^\omega(S)$ with $\mathrm{CP}(S)$ aperiodic. It uses H^ω, generalized to $\hat{H}^\omega$ on $\hat{C}^\omega(S)$ 'to get rid of groups by blowing up'.

A toy example: Separation for FO($=$)

- FO($=$) can just count occurrences of letters, up to threshold.
- Example: at least 2 a's: $\exists x, y \;\; x \neq y \wedge a(x) \wedge a(y)$.
- FO($=$) can express properties like

  at least 2 a's, no more than 3 b's, exactly 1 c.

- How to decide separation for FO($=$)?

A toy example: Separation for FO$(=)$

- Let $\pi(u) \in \mathbb{N}^A$ be the commutative (aka. Parikh) image of u.

$$\pi(aabad) = (3, 1, 0, 1).$$

Parikh's Theorem

For L context-free, $\pi(L)$ is (effectively) semilinear.

- For $\vec{x}, \vec{y} \in \mathbb{N}^A$, $\quad \vec{x} =_d \vec{y} \quad$ if $\quad \forall i$: $x_i = y_i$ or both $x_i, y_i \geqslant d$.

A toy example: Separation for FO$(=)$

- Let $\pi(u) \in \mathbb{N}^A$ be the commutative (aka. Parikh) image of u.

$$\pi(aabad) = (3, 1, 0, 1).$$

Parikh's Theorem

For L context-free, $\pi(L)$ is (effectively) semilinear.

- For $\vec{x}, \vec{y} \in \mathbb{N}^A$, $\quad \vec{x} =_d \vec{y} \quad$ if $\quad \forall i$: $x_i = y_i$ or both $x_i, y_i \geqslant d$.

Fact

Languages L_1, L_2 are not FO$(=)$-separable iff

$$\forall d \quad \exists u_1 \in L_1 \, \exists u_2 \in L_2, \quad \pi(u_1) =_d \pi(u_2).$$

A toy example: Separation for FO$(=)$

- Let $\pi(u) \in \mathbb{N}^A$ be the commutative (aka. Parikh) image of u.

$$\pi(aabad) = (3, 1, 0, 1).$$

Parikh's Theorem

For L context-free, $\pi(L)$ is (effectively) semilinear.

- For $\vec{x}, \vec{y} \in \mathbb{N}^A$, $\quad \vec{x} =_d \vec{y} \quad$ if $\quad \forall i$: $x_i = y_i$ or both $x_i, y_i \geqslant d$.

Fact

Languages L_1, L_2 are not FO$(=)$-separable iff

$$\forall d \quad \exists u_1 \in L_1 \, \exists u_2 \in L_2, \quad \pi(u_1) =_d \pi(u_2).$$

Proof. $\Rightarrow$ The FO$(=)$ language $\{u \mid \pi(u) \in_d \pi(L_1)\}$ contains L_1. Since L_1, L_2 are not FO$(=)$-separable, it intersects L_2.

A toy example: Separation for FO$(=)$

- Let $\pi(u) \in \mathbb{N}^A$ be the commutative (aka. Parikh) image of u.

$$\pi(aabad) = (3, 1, 0, 1).$$

Parikh's Theorem

For L context-free, $\pi(L)$ is (effectively) semilinear.

- For $\vec{x}, \vec{y} \in \mathbb{N}^A$, $\quad \vec{x} =_d \vec{y} \quad$ if $\quad \forall i$: $x_i = y_i$ or both $x_i, y_i \geqslant d$.

Fact

Languages L_1, L_2 are not FO$(=)$-separable iff

$$\forall d \quad \exists u_1 \in L_1 \, \exists u_2 \in L_2, \quad \pi(u_1) =_d \pi(u_2).$$

Proof. $\Rightarrow$ The FO$(=)$ language $\{u \mid \pi(u) \in_d \pi(L_1)\}$ contains L_1.
Since L_1, L_2 are not FO$(=)$-separable, it intersects L_2.

$\Leftarrow$ Assume there is an FO$(=)$-separator K, say of threshold d.
Then $L_1 \subseteq K \Rightarrow u_1 \in K \Rightarrow u_2 \in K$, impossible since $u_2 \in L_2$.

A toy example: Separation for FO$(=)$

Fact

Languages L_1, L_2 are not FO$(=)$-separable iff

$$\forall d \quad \exists \vec{x}_1 \in \pi(L_1)\, \exists \vec{x}_2 \in \pi(L_2), \quad \vec{x}_1 =_d \vec{x}_2.$$

Decidability of FO$(=)$-separation is then implied by

- Parikh's Theorem, and
- Decidability of Presburger logic.

Separation for FO$(=, +1)$

- FO$(=)$ can just count occurrences of letters up to a threshold.
- FO$(=, +1)$ can count occurrences of infixes up to a threshold.

  There exist at least 2 occurrences of $abba$
  and the word start with ba.

- For membership, decidability follows from a delay theorem:
  To test FO$(=, +1)$-definability, look at infixes of bounded size.

Separation for FO$(=, +1)$

- FO$(=)$ can just count occurrences of letters up to a threshold.
- FO$(=, +1)$ can count occurrences of infixes up to a threshold.

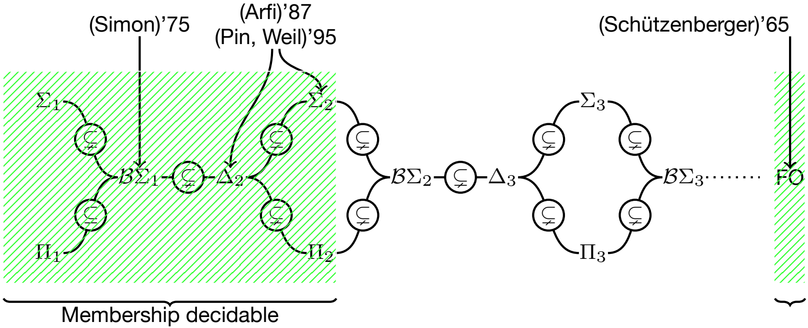
There exist at least 2 occurrences of $abba$
and the word start with ba.

- For membership, decidability follows from a delay theorem:
  To test FO$(=, +1)$-definability, look at infixes of bounded size.

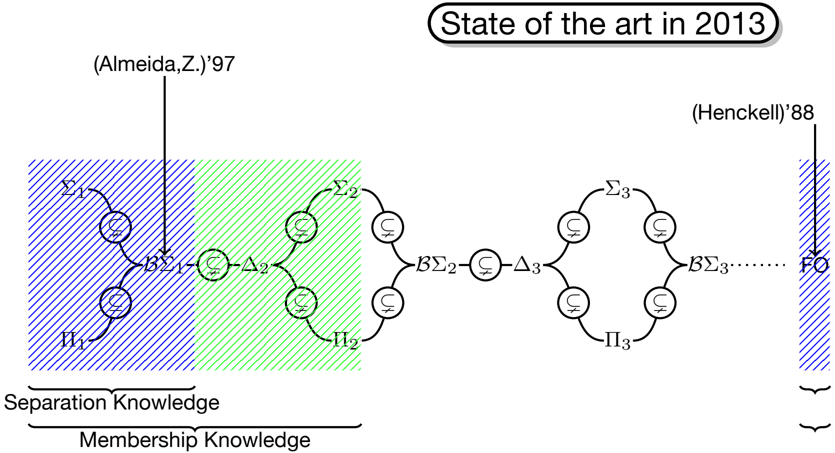
- Membership proof not trivial. Transferring separability is easier.

FO Quantifier alternation hierarchy

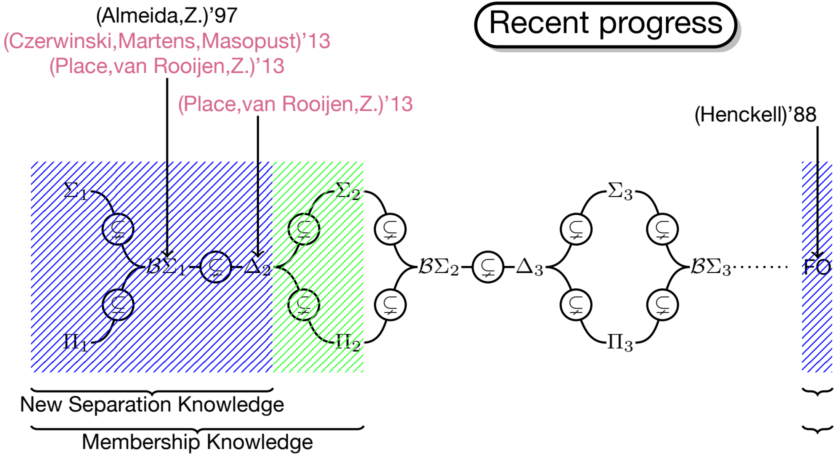
State of the art in 2013



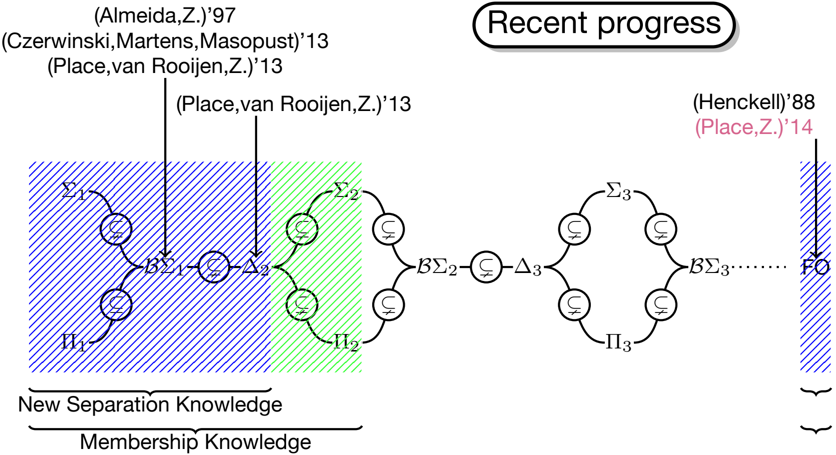
FO Quantifier alternation hierarchy



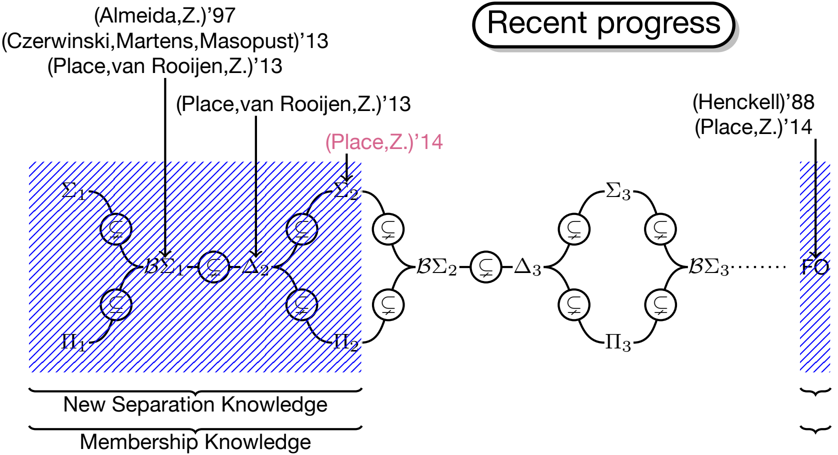
FO Quantifier alternation hierarchy



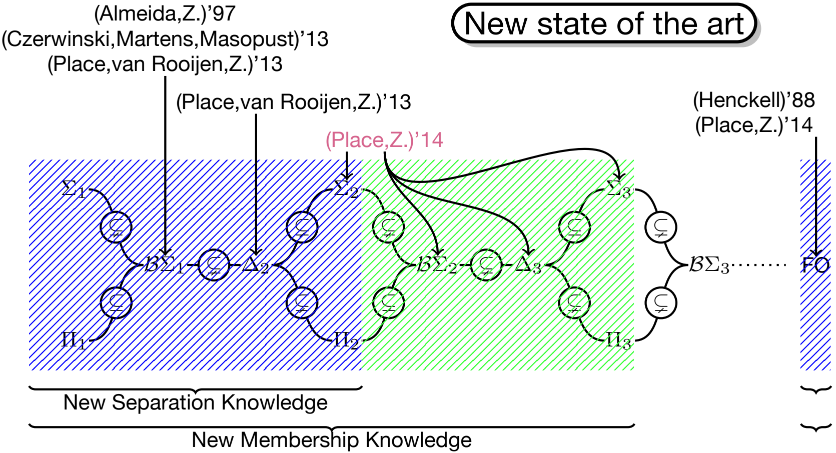
FO Quantifier alternation hierarchy



FO Quantifier alternation hierarchy

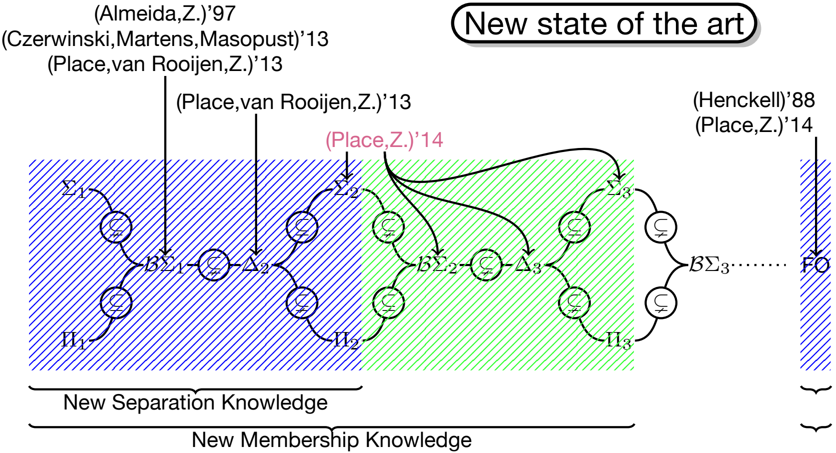


FO Quantifier alternation hierarchy



By relying on Σ_2-Analysis, one can prove
decidable characterizations for $\mathcal{B}\Sigma_2$,Δ_3,Σ_3 and Π_3.

FO Quantifier alternation hierarchy



T. Place, LICS'15

Separation for Σ_3 (hard)

Decidability for $\Delta_4, \Sigma_4, \Pi_4$

Still open for $\mathcal{B}\Sigma_3$

Summary of recent results

Specific results

- Separation reproved for FO, proved for Σ_2, Σ_3.
- Membership for $\mathcal{B}\Sigma_2$ (specific proof).

Transfer results

- Separation of Σ_n entails membership for Σ_{n+1}.
- Separation for $\mathcal{F}$ entails separation for $\mathcal{F}[+1]$.

Proofs techniques

FO is hard, let's make it easy!

Quantifier rank of a formula: Nested depth of quantifiers.

$$\exists x\ c(x) \wedge \forall x \exists y\ (a(x) \implies \exists z\ (x < z < y \wedge b(y))) \quad \text{rank } 3$$

If k fixed: finitely many **FO** properties of rank k

$\Rightarrow$ Separation is easy (test them all).

Proofs techniques

FO is hard, let's make it easy!

Quantifier rank of a formula: Nested depth of quantifiers.

$$\exists x\ c(x) \wedge \forall x \exists y\ (a(x) \implies \exists z\ (x < z < y \wedge b(y))) \quad \text{rank } 3$$

If k fixed: finitely many **FO** properties of rank k

$\Rightarrow$ Separation is easy (test them all).

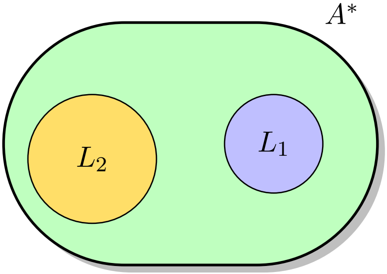
k-equivalence for FO

Let w_1, w_2 be words:

$$w_1 \approx_k w_2 \text{ iff } w_1, w_2 \text{ satisfy the same formulas of rank } k$$

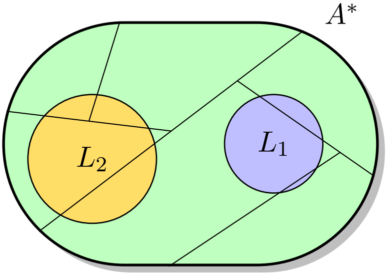
All **FO** *properties of rank k are unions of classes of $\approx_k$.*

Fixed Quantifier Rank k



Let's add the $\approx_k$-classes

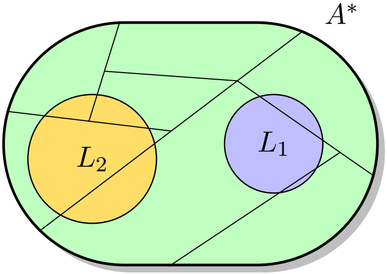
Fixed Quantifier Rank k



Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

Fixed Quantifier Rank k

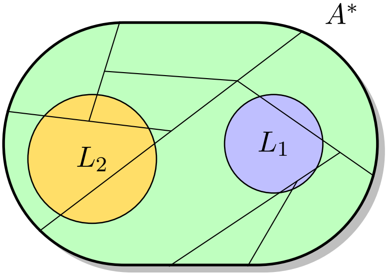


Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

When k gets larger, $\approx_k$ is refined but it never ends

Fixed Quantifier Rank k

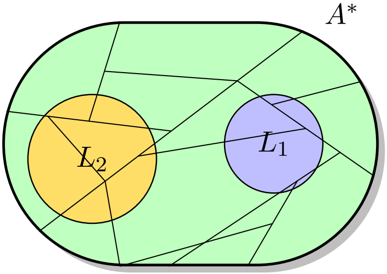


Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

When k gets larger, $\approx_k$ is refined but it never ends

Fixed Quantifier Rank k

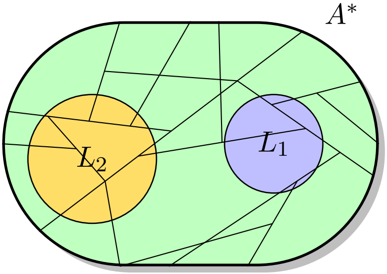


Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

When k gets larger, $\approx_k$ is refined but it never ends

Fixed Quantifier Rank k

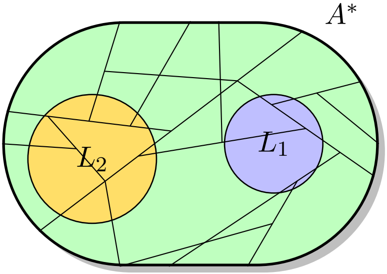


Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

When k gets larger, $\approx_k$ is refined but it never ends

Fixed Quantifier Rank k



Separable with rank k iff no $\approx_k$-class intersects both languages

For full **FO** we want to know if there exists such a k
$\Rightarrow$ Compute a 'limit' for $\approx_k$.

When k gets larger, $\approx_k$ is refined but it never ends

Idea. Abstract $\approx_k$ on a finite monoid recognizing both L_1 and L_2.

“Pair” analysis

Fix $\alpha : A^* \to M$. Compute $\mathbf{I}_k[\alpha]$, k-indistinguishable pairs.

FO-indistinguishable pairs for $\alpha : A^* \to M$

$(s_1, s_2) \in \mathbf{I}_k[\alpha]$ if

$$\exists \qquad \begin{array}{ccc} w_1 & \approx_k & w_2 \\ {\scriptstyle\alpha}\downarrow & & \downarrow{\scriptstyle\alpha} \\ s_1 & & s_2 \end{array}$$

- Smaller and smaller sets: $\mathbf{I}_{k+1}[\alpha] \subseteq \mathbf{I}_k[\alpha]$.
- Limit set: $\mathbf{I}[\alpha] = \bigcap_k \mathbf{I}_k[\alpha]$.
- Computing these pairs solves separation:

$$(s_1, s_2) \in \mathbf{I}[\alpha] \qquad \Longleftrightarrow \qquad \alpha^{-1}(s_1) \text{ and } \alpha^{-1}(s_2) \text{ not separable}$$

“Pair” analysis

- Smaller and smaller sets: $\mathbf{I}_{k+1}[\alpha] \subseteq \mathbf{I}_k[\alpha]$
- Limit set: $\mathbf{I}[\alpha] = \bigcap_k \mathbf{I}_k[\alpha]$.

What have we gained?

We work with finite semigroups $\Rightarrow$ the refinement stabilizes.

“Pair” analysis

- Smaller and smaller sets: $\mathbf{I}_{k+1}[\alpha] \subseteq \mathbf{I}_k[\alpha]$
- Limit set: $\mathbf{I}[\alpha] = \bigcap_k \mathbf{I}_k[\alpha]$.

What have we gained?

We work with finite semigroups $\Rightarrow$ the refinement stabilizes.

It may happen that $\mathbf{I}_{k+1}[\alpha] \ = \ \mathbf{I}_k[\alpha]$ before stabilization.
It may happen that

- $(r, s) \in \mathbf{I}[\alpha]$,
- $(s, t) \in \mathbf{I}[\alpha]$,
- but $(r, t) \notin \mathbf{I}[\alpha]$ (no transitivity).

The Separation Criterion

Separation Criterion

L_1, L_2 recognized by $\alpha : A^* \to M$ are not separable
iff
there are accepting elements $s_1, s_2 \in M$ for L_1, L_2 s.t. $(s_1, s_2) \in \mathbf{I}[\alpha]$.

The Separation Criterion

Separation Criterion

L_1, L_2 recognized by $\alpha : A^* \to M$ are not separable
iff
there are accepting elements $s_1, s_2 \in M$ for L_1, L_2 s.t. $(s_1, s_2) \in \mathbf{I}[\alpha]$.

Computing $\mathbf{I}[\alpha]$ suffices to solve separation.

Two approaches to compute $\mathbf{I}[\alpha]$

Brute-force

- k fixed: computing $\mathbf{I}_k[\alpha]$ easy.
- $\mathbf{I}[\alpha] = \mathbf{I}_k[\alpha]$ for some k.
- $\Rightarrow$ Prove a bound $k = f(\alpha)$,
  Compute $\mathbf{I}_k[\alpha]$.

Algorithm

Algorithm bypassing the bound k:
Direct fixpoint computation of $\mathbf{I}[\alpha]$.

Two approaches to compute $\mathbf{I}[\alpha]$

Brute-force

- k fixed: computing $\mathbf{I}_k[\alpha]$ easy.
- $\mathbf{I}[\alpha] = \mathbf{I}_k[\alpha]$ for some k.
- $\Rightarrow$ Prove a bound $k = f(\alpha)$,
  Compute $\mathbf{I}_k[\alpha]$.

Algorithm

Algorithm bypassing the bound k:
Direct fixpoint computation of $\mathbf{I}[\alpha]$.

We use approach 2.

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

> 1st Property of **FO**
>
> $$w \approx_k w$$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

$$\boxed{\begin{array}{c}\text{1st Property of } \mathbf{FO} \\ w \approx_k w\end{array}}$$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

2nd Property of FO

$w_1 \approx_k w_2$ and $u_1 \approx_k u_2 \quad \Rightarrow \quad w_1 u_1 \approx_k w_2 u_2$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

2nd Property of FO

$w_1 \approx_k w_2$ and $u_1 \approx_k u_2 \quad \Rightarrow \quad w_1u_1 \approx_k w_2u_2$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $(s_1, s_2) \in \mathbf{I}[\alpha]$ and $(t_1, t_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1t_1, s_2t_2) \in \mathbf{I}[\alpha]$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of FO

$$\forall k\ \exists n\ \forall w_1, w_2 \in A^* \qquad w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $(s_1, s_2) \in \mathbf{I}[\alpha]$ and $(t_1, t_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1t_1, s_2t_2) \in \mathbf{I}[\alpha]$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of FO

$$\forall k\ \exists n\ \forall w_1, w_2 \in A^* \qquad w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $(s_1, s_2) \in \mathbf{I}[\alpha]$ and $(t_1, t_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in \mathbf{I}[\alpha]$
3. Operation ω: $(s_1, s_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1^{\omega}, s_2^{\omega+1}) \in \mathbf{I}[\alpha]$

A first (non complete) algorithm computing $\mathbf{I}[\alpha]$

Idea. Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of FO

$$\forall k\ \exists n\ \forall w_1, w_2 \in A^* \qquad w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}$$

1. Trivial pairs: for all $w \in A^*$ $(\alpha(w), \alpha(w)) \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $(s_1, s_2) \in \mathbf{I}[\alpha]$ and $(t_1, t_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1 t_1, s_2 t_2) \in \mathbf{I}[\alpha]$
3. Operation ω: $(s_1, s_2) \in \mathbf{I}[\alpha] \Rightarrow (s_1^{\omega}, s_2^{\omega+1}) \in \mathbf{I}[\alpha]$

Correct by definition but not complete

Why it does not work

3rd Property of FO

$$w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}$$

Why it does not work

Not general enough

$\Downarrow$

3rd Property of FO

$w_1 \approx_k w_2 \Rightarrow (w_1)^n \approx_k (w_2)^{n+1}$

Needs to be replaced

$\Downarrow$

$$w_1 \approx_k w_2 \approx_k \cdots \approx_k w_m$$
$$\Downarrow$$
All large concatenations of words in $\{w_1, \ldots, w_m\}$ are $\approx_k$-equivalent.

Need for better analysis

A Generalization: FO-indistinguishable Sets for $\alpha : A^* \to M$:

- $\{s_1, s_2, \ldots, s_n\} \in \mathbf{I}_k[\alpha]$ if

$$\exists \quad \begin{array}{ccccccc} w_1 & \approx_k & w_2 & & \cdots & \approx_k & w_n \\ \downarrow{\scriptstyle\alpha} & & \downarrow{\scriptstyle\alpha} & & & & \downarrow{\scriptstyle\alpha} \\ s_1 & & s_2 & & \cdots & & s_n \end{array}$$

- Limit set: $\mathbf{I}[\alpha] = \bigcap_k \mathbf{I}_k[\alpha]$.
- Computing these sets is more general than computing pairs.

  $\Rightarrow$ also solves separation (and gives much more).

From Pairs to Sets

New Objective

We want to compute the set $\mathbf{I}[\alpha] \subseteq 2^M$ such that:

$$T \in \mathbf{I}[\alpha] \text{ iff } T \in \mathbf{I}_k[\alpha],\ \forall k \in \mathbb{N}$$

From Pairs to Sets

New Objective

We want to compute the set $\mathbf{I}[\alpha] \subseteq 2^M$ such that:

$$T \in \mathbf{I}[\alpha] \text{ iff } T \in \mathbf{I}_k[\alpha],\ \forall k \in \mathbb{N}$$

Remark

- With our new definition, we have $\mathbf{I}[\alpha] \subseteq 2^M$.
- 2^M is a monoid for operation

$$T_1 \cdot T_2 = \{t_1 t_2 \mid t_1 \in T_1\ t_2 \in T_2\}$$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

1st Property of FO

$$w \approx_k w$$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

1st Property of FO

$$w \approx_k w$$

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

2nd Property of **FO**

$w_1 \approx_k w_2$ and $u_1 \approx_k u_2 \quad \Rightarrow \quad w_1u_1 \approx_k w_2u_2$

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

2nd Property of FO

$w_1 \approx_k w_2$ and $u_1 \approx_k u_2 \quad \Rightarrow \quad w_1 u_1 \approx_k w_2 u_2$

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $T_1 \in \mathbf{I}[\alpha]$ and $T_2 \in \mathbf{I}[\alpha] \Rightarrow T_1 T_2 \in \mathbf{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \approx_k w_2 \cdots \approx_k w_m$$

$$\Downarrow$$

All large concatenations of words in $\{w_1, \cdots, w_m\}$ are $\approx_k$-equivalent.

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $T_1 \in \mathbf{I}[\alpha]$ and $T_2 \in \mathbf{I}[\alpha] \Rightarrow T_1 T_2 \in \mathbf{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \approx_k w_2 \cdots \approx_k w_m$$

$$\Downarrow$$

All large concatenations of words in $\{w_1, \cdots, w_m\}$ are $\approx_k$-equivalent.

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $T_1 \in \mathbf{I}[\alpha]$ and $T_2 \in \mathbf{I}[\alpha] \Rightarrow T_1 T_2 \in \mathbf{I}[\alpha]$
3. Operation ω: $T \in \mathbf{I}[\alpha] \Rightarrow (T^\omega \cup T^{\omega+1}) \in \mathbf{I}[\alpha]$

A new (working) Algorithm

Idea : Start with trivial pairs. Add more pairs via a fixpoint algorithm.

3rd Property of **FO**

$$w_1 \approx_k w_2 \cdots \approx_k w_m$$

$$\Downarrow$$

All large concatenations of words in $\{w_1, \cdots, w_m\}$ are $\approx_k$-equivalent.

1. Trivial sets: for all $w \in A^*$ $\{\alpha(w)\} \in \mathbf{I}[\alpha]$
2. Operation $\bullet$: $T_1 \in \mathbf{I}[\alpha]$ and $T_2 \in \mathbf{I}[\alpha] \Rightarrow T_1T_2 \in \mathbf{I}[\alpha]$
3. Operation ω: $T \in \mathbf{I}[\alpha] \Rightarrow (T^\omega \cup T^{\omega+1}) \in \mathbf{I}[\alpha]$

Correct by definition (e.g., use EF games)
Can be proved to be complete

Tools

- Ehrenfeucht-Fraïssé games.
- Combinatorial tools: Simon's Factorization Forests & Ramsey.

Conclusion

We shouldn't restrict ourselves to membership

Conclusion

We shouldn't restrict ourselves to membership, nor to separation.

Conclusion

We shouldn't restrict ourselves to membership, nor to separation.

- Freezing the framework (to membership or separation) yields limitations.
- This work is just a byproduct of the observation that one can be more demanding on the computed information.
- Generalizing the needed information is often mandatory.

Thank You!