The Regular Languages of First-Order Logic with One Alternation

Corentin Barloy, Michaël Cadilhac, Charles Paperman, Thomas Zeume

LICS 2022

Expressing languages with logic

$\{a, b, c\}^* ab^*$ can be defined with:

$$\exists x, \forall y, \quad a(x) \land (y > x \Rightarrow b(y))$$

Expressing languages with logic

$\{a, b, c\}^* ab^*$ can be defined with:

$$\underbrace{\exists x, \forall y,}_{\text{one quantifier alternation}} \quad a(x) \wedge (y > x \Rightarrow b(y))$$

Expressiveness

Fragments

Numerical predicates

Expressiveness

Fragments

- First-order logic: FO

Numerical predicates

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k

Numerical predicates

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Regular predicates: REG

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Regular predicates: REG

- Many more ($xy = z$, encoding of a cat, ...)

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Regular predicates: REG

- Many more ($xy = z$, encoding of a cat, ...)

Arbitrary predicates: ARB

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Regular predicates: REG

- Many more ($xy = z$, encoding of a cat, ...)

Arbitrary predicates: ARB

$$\exists x, \forall y, a(x) \wedge (y > x \Rightarrow b(y)) \in \Sigma_2[\mathrm{REG}]$$

Expressiveness

Fragments

- First-order logic: FO
- Bounded quantifier alternation: Σ_k, Π_k
- Many more (restricting the number of variables, adding modular quantifiers,...)

Numerical predicates

- The order: $x < y$
- The successor predicate: $x + 1 = y$
- The modular predicates: $x \bmod 3 = 0$

Regular predicates: REG

- Many more ($xy = z$, encoding of a cat, ...)

Arbitrary predicates: ARB

$$\exists x, \forall y, a(x) \wedge (y > x \Rightarrow b(y)) \in \Sigma_2[\text{REG}]$$

$$\forall x, (x \text{ encodes a cat}) \Rightarrow a(x) \in \Pi_1[\text{ARB}]$$

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$ *?*

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \quad !$$

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \ \ !$$

True

- Σ_1
- FO
- FO with prime modular quantifiers

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$ *?*

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \quad !$$

True	False
▶ Σ_1 ▶ FO ▶ FO with prime modular quantifiers	▶ $\mathrm{FO} + S_5$

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \quad !$$

True	False	Open
▶ Σ_1 ▶ FO ▶ FO with prime modular quantifiers	▶ $\mathrm{FO} + S_5$	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \;\; !$$

Circuits with
composite modular gates
(AND $\in \mathrm{CC}^0_6$?)

True	False	Open
▶ Σ_1 ▶ FO ▶ FO with prime modular quantifiers	▶ $\mathrm{FO} + S_5$	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \;\; !$$

Circuits with composite modular gates ($\mathrm{AND} \in \mathrm{CC}^0_6$?)

True	False	Open
Σ_1	$\mathrm{FO} + S_5$	FO with composite modular quantifiers
FO		FO with two variables
FO with prime modular quantifiers		Σ_k, $k \geq 3$

Linear AC^0 (complexity of addition)

Central question

For $\mathcal{L}$ a fragment,

What are the regular languages of $\mathcal{L}[ARB]$?

A natural guess (Straubing):

$$\mathcal{L}[\mathrm{ARB}] \cap \mathrm{Reg} = \mathcal{L}[\mathrm{REG}] \quad !$$

Circits with
composite modular gates
(AND $\in \mathrm{CC}_6^0$?)

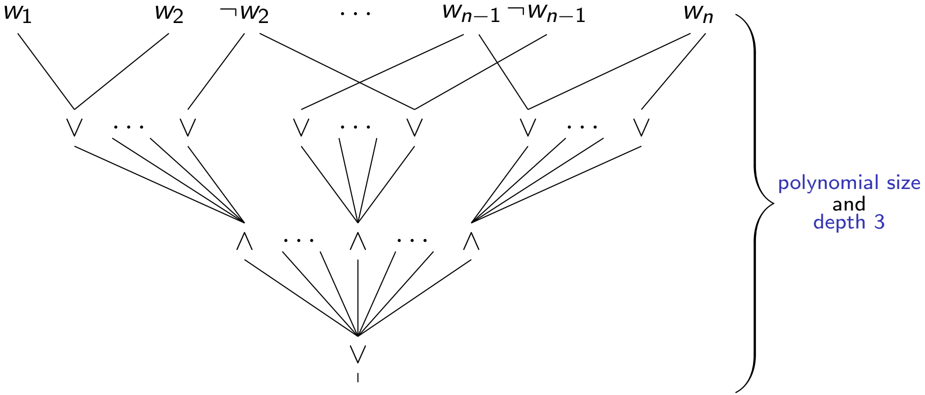
True	False	Open
▶ Σ_1 ▶ FO ▶ FO with prime modular quantifiers	▶ FO + S_5	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

Our result: $\Sigma_2[\mathrm{ARB}] \cap \mathrm{Reg} = \Sigma_2[\mathrm{REG}]$

Linear AC^0
(complexity of addition)

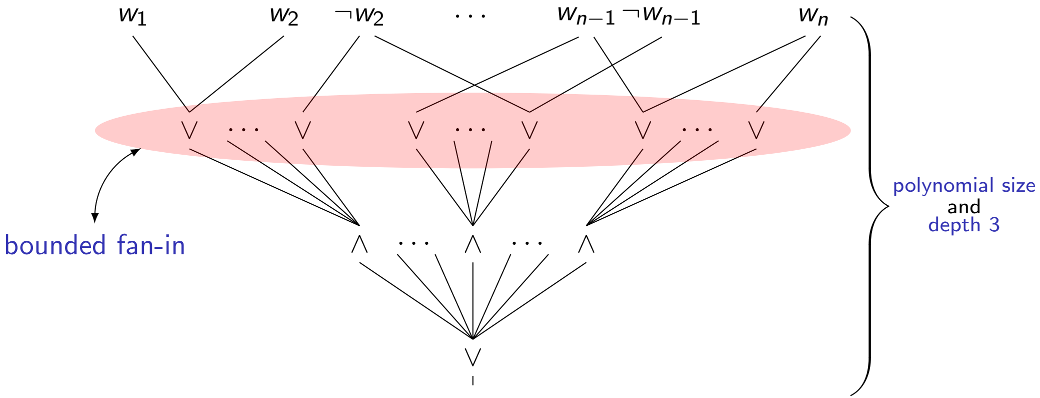
The circuit class Σ_2

$w = w_1 w_2 \cdots w_{n-1} w_n$



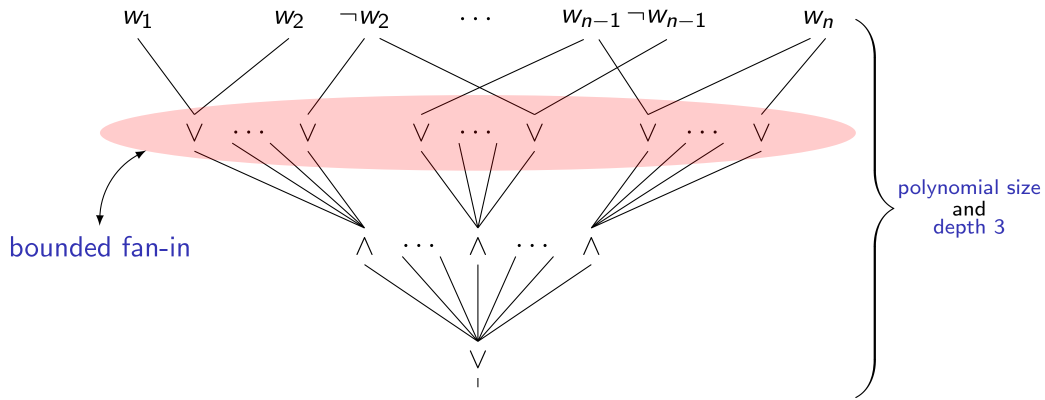
The circuit class Σ_2

$w = w_1 w_2 \cdots w_{n-1} w_n$



The circuit class Σ_2

$w = w_1 w_2 \cdots w_{n-1} w_n$



It is equivalent to Σ_2[ARB].

The main result

$$\Sigma_2[\mathsf{ARB}] \cap \mathsf{Reg} = \Sigma_2[\mathsf{REG}] \quad .$$

The main result

$$\Sigma_2[\mathrm{ARB}] \cap \mathrm{Reg} = \Sigma_2[\mathrm{REG}] \quad .$$

- $\supseteq$: Immediate.

The main result

$$\Sigma_2[\mathrm{ARB}] \cap \mathrm{Reg} = \Sigma_2[\mathrm{REG}] \quad .$$

- $\supseteq$: Immediate.
- $\subseteq$: Take a regular language not in $\Sigma_2[\mathrm{REG}]$, show that it is not in $\Sigma_2[\mathrm{ARB}]$.

The main result

$$\Sigma_2[\mathrm{ARB}] \cap \mathrm{Reg} = \Sigma_2[\mathrm{REG}] \quad .$$

- $\supseteq$: Immediate.
- $\subseteq$: Take $\underbrace{\text{a regular language not in } \Sigma_2[\mathrm{REG}]}_{\text{Algebra}}$, show that it is not in $\Sigma_2[\mathrm{ARB}]$.

The main result

$$\Sigma_2[\mathrm{ARB}] \cap \mathrm{Reg} = \Sigma_2[\mathrm{REG}] \ .$$

- $\supseteq$: Immediate.
- $\subseteq$: Take $\underbrace{\text{a regular language not in } \Sigma_2[\mathrm{REG}]}_{\text{Algebra}}$, show that $\underbrace{\text{it is not in } \Sigma_2[\mathrm{ARB}]}_{\text{Circuit lower bound}}$.

Proof sketch

Proof sketch

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

Proof sketch

Algebra | Lower bound

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

$\forall uxv \in \mathcal{L}$

$$u \quad x \quad v \in \mathcal{L}$$

Proof sketch

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

$\forall uxv \in \mathcal{L}$ such that x can be iterated

$$u \quad xxxxx \quad v \in \mathcal{L}$$

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

$\forall uxv \in \mathcal{L}$ such that x can be iterated

$$u \quad x \quad v \in \mathcal{L}$$

Proof sketch

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

$\forall uxv \in \mathcal{L}$ such that x can be iterated , then $uxyxv$ is also in $\mathcal{L}$ for every y with the same letters as x.

$$u \quad xyx \quad v \in \mathcal{L}$$

Proof sketch

Definition (limit (Sipser))

Let A be a set of words in $\mathcal{L}$.
A limit for A is a word u:

- *not in $\mathcal{L}$*
- *that can fool every $\vee$ of bounded fan-in that accepts (at least) A.*

Proof sketch

Definition (limit (Sipser))

Let A be a set of words in $\mathcal{L}$.
A limit for A is a word u:

- *not in $\mathcal{L}$*
- *that can fool every $\vee$ of bounded fan-in that accepts (at least) A.*

Proposition

If every subset of $\mathcal{L}$ big enough admits a limit, then $\mathcal{L}$ cannot be recognized by a Σ_2 circuit.

Proof: One of the $\wedge$ gates must recognize a big subset of $\mathcal{L}$.

Proof sketch

Definition (limit (Sipser))

Let A be a set of words in $\mathcal{L}$.
A limit for A is a word u:

- *not in $\mathcal{L}$*
- *that can fool every $\vee$ of bounded fan-in that accepts (at least) A.*

Proposition

If every subset of $\mathcal{L}$ big enough admits a limit, then $\mathcal{L}$ cannot be recognized by a Σ_2 circuit.

Proof: One of the $\wedge$ gates must recognize a big subset of $\mathcal{L}$.

A way of finding limits is via Erdős sunflower lemma (Håstad, Jukna, Pudlák).

Proof sketch

Definition (limit (Sipser))

Let A be a set of words in $\mathcal{L}$.
A limit for A is a word u:

- *not in $\mathcal{L}$*
- *that can fool every $\vee$ of bounded fan-in that accepts (at least) A.*

Proposition

If every subset of $\mathcal{L}$ big enough admits a limit, then $\mathcal{L}$ cannot be recognized by a Σ_2 circuit.

Proof: One of the $\wedge$ gates must recognize a big subset of $\mathcal{L}$.

A way of finding limits is via Erdős sunflower lemma (Håstad, Jukna, Pudlák).

We give here a new method of finding limits, $\underbrace{\text{specially tailored for } \Sigma_2}_{\text{of the form } uxyxv}$.

Conclusion

Also in the paper:

- Straubing's conjecture for Δ_2.

Not in the paper:

- The proof in its full generality.

Future work:

- Go higher in the hierarchy: $\mathcal{B}\Sigma_2$, Σ_3, ...
- Tackle different kind of fragments, like FO_2.