The Regular Languages in Circuit Classes

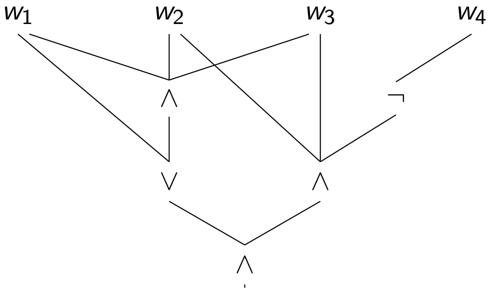
Corentin Barloy

Université de Lille

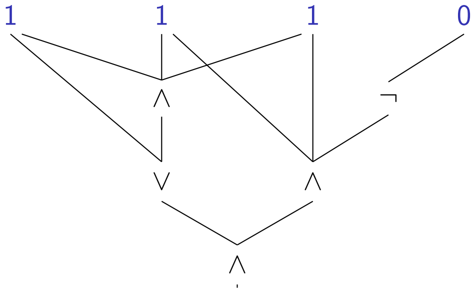
Joint work with Michaël Cadilhac, Charles Paperman and Thomas Zeume

Bordeaux 2022

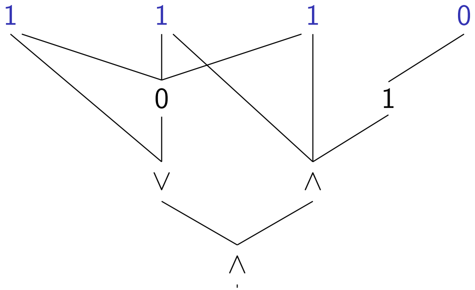
Circuits



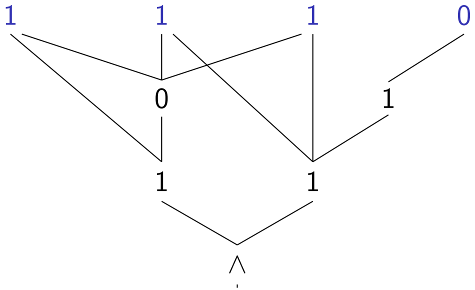
Circuits



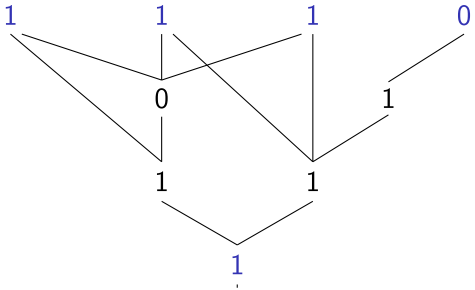
Circuits



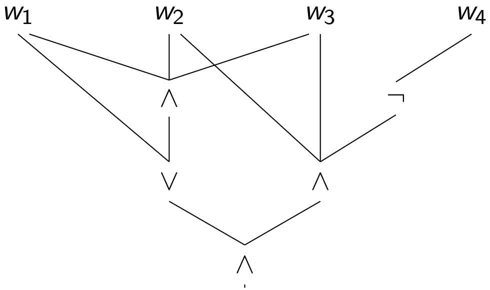
Circuits



Circuits

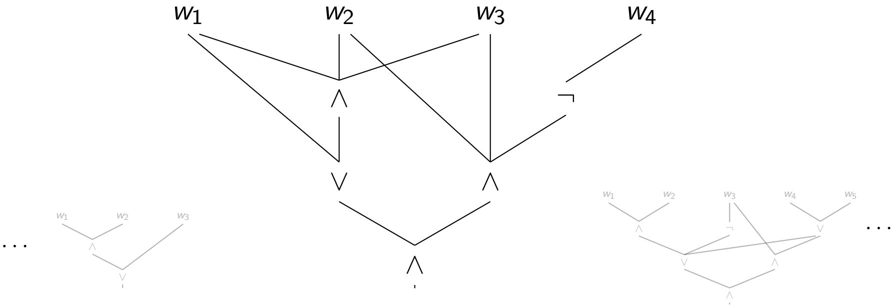


Circuits



$L_4 = \{1110, \ldots\}$

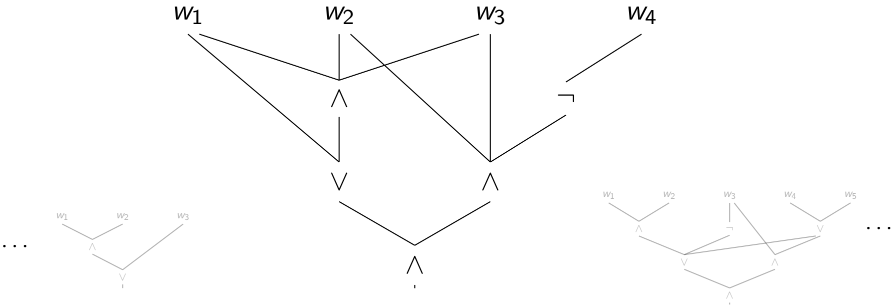
Circuits



$L_4 = \{1110, \dots\}$ $\qquad$ $L = \bigcup_{i \in \mathbb{N}} L_i$

Circuits

size = number of gates

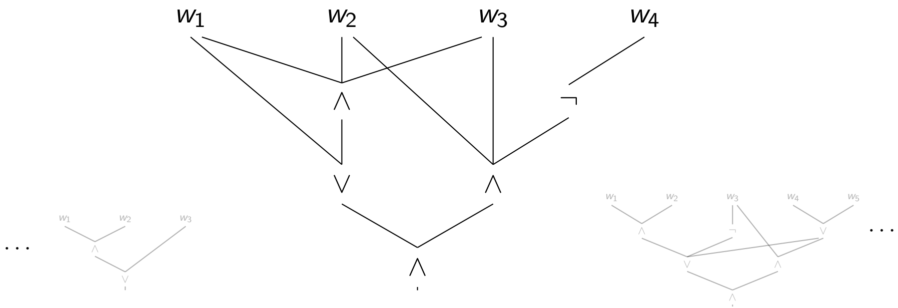


$L_4 = \{1110, \dots\}$ $\quad$ $L = \bigcup_{i \in \mathbb{N}} L_i$

Circuits

size $=$ number of gates

depth $=$ maximal length of a path

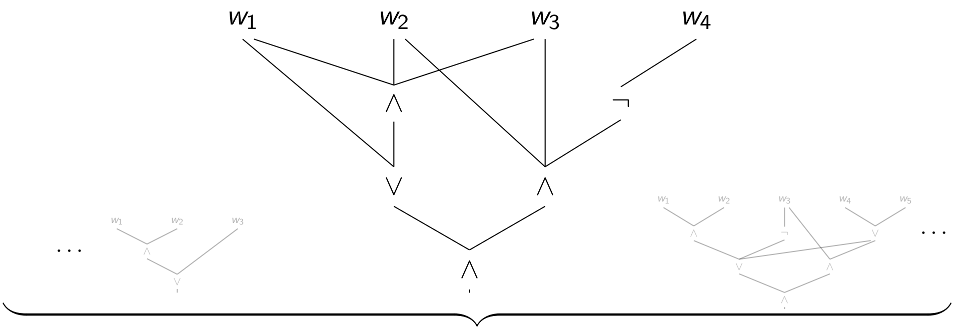


$L_4 = \{1110, \ldots\}$ $\quad$ $L = \bigcup_{i \in \mathbb{N}} L_i$

Circuits

depth = maximal length of a path

size = number of gates



Uniformity = definable in P, LOGSPACE, ...

$L_4 = \{1110, \dots\}$ $\qquad$ $L = \bigcup_{i \in \mathbb{N}} L_i$

Motivations

P vs NP

Motivations

$$\mathsf{P} \subseteq \mathsf{P/poly} \qquad \text{vs} \qquad \mathsf{NP}$$

Motivations

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P}/\mathsf{poly} \quad \textbf{vs} \quad \mathsf{NP}$$

Motivations

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P/poly} \qquad \textbf{vs} \qquad \mathsf{NP}$$

circuits with
polynomially many gates

Motivations

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P/poly} \qquad \text{vs} \qquad \mathsf{NP}$$

circuits with
polynomially many gates

P-uniform circuits with
polynomially many gates

Motivations

LOGSPACE-uniform circuits with
polynomially many gates

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P/poly} \qquad \text{vs} \qquad \mathsf{NP}$$

P-uniform circuits with
polynomially many gates

circuits with
polynomially many gates

Motivations

LOGSPACE-uniform circuits with
polynomially many gates

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P}/\mathsf{poly} \quad \textbf{vs} \quad \mathsf{NP}$$

P-uniform circuits with
polynomially many gates

circuits with
polynomially many gates

Shannon: Almost every function requires exponential size.

Motivations

LOGSPACE-uniform circuits with
polynomially many gates

algorithms in P
with polynomial advice

$$\mathsf{P} \subseteq \mathsf{P/poly} \qquad \text{vs} \qquad \mathsf{NP}$$

P-uniform circuits with
polynomially many gates

circuits with
polynomially many gates

(Close to hardware)

Shannon: Almost every function requires exponential size.

Classes of small circuits

$$\begin{tikzcd}
 & \mathsf{ACC}^0[\mathcal{P}] \arrow[dr] & & \mathsf{TC}^0 \arrow[dr] & \\
\mathsf{AC}^0 \arrow[ur] & & \mathsf{ACC}^0 \arrow[ur] & & \mathsf{NC}^1
\end{tikzcd}$$

Classes of small circuits

$$\mathsf{AC}^0 \rightarrow \mathsf{ACC}^0[\mathcal{P}] \rightarrow \mathsf{ACC}^0 \rightarrow \mathsf{TC}^0 \rightarrow \mathsf{NC}^1$$

polynomial size
constant depth
unbounded fan-in

Classes of small circuits

$$
\begin{tikzcd}
 & \mathrm{ACC}^0[\mathcal{P}] \arrow[dr] & & \mathrm{TC}^0 \arrow[dr] & \\
\mathrm{AC}^0 \arrow[ur] \arrow[d, leftrightarrow] & & \mathrm{ACC}^0 \arrow[ur] & & \mathrm{NC}^1 \arrow[d, leftrightarrow] \\
\begin{array}{c}\text{polynomial size}\\ \text{constant depth}\\ \text{unbounded fan-in}\end{array} & & & & \begin{array}{c}\text{polynomial size}\\ \text{logarithmic depth}\\ \text{fan-in } 2\end{array}
\end{tikzcd}
$$

Classes of small circuits

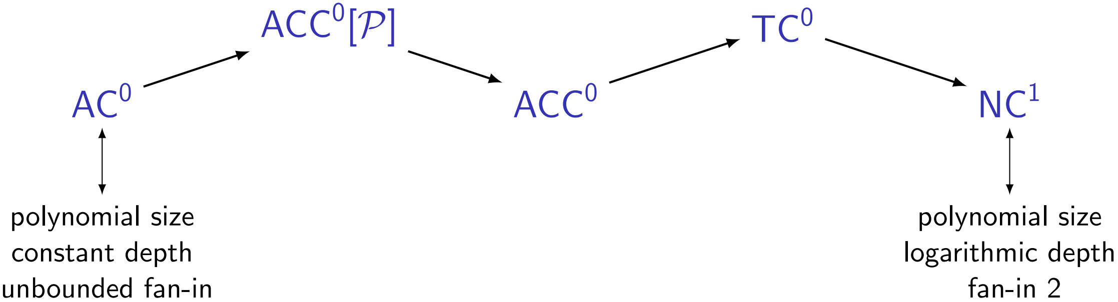
$$\mathsf{AC}^0 \rightarrow \mathsf{ACC}^0[\mathcal{P}] \rightarrow \mathsf{ACC}^0 \rightarrow \mathsf{TC}^0 \rightarrow \mathsf{NC}^1$$

$\mathsf{AC}^0 \leftrightarrow$ polynomial size
constant depth
unbounded fan-in

$\mathsf{NC}^1 \leftrightarrow$ polynomial size
logarithmic depth
fan-in 2

$\boxed{\text{MOD}}$: number of $1 \equiv_m 0$?

Classes of small circuits



$\boxed{\text{MOD}}$: number of 1 $\equiv_m$ 0 ?

$\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

Classes of small circuits

$$AC^0 \to ACC^0[\mathcal{P}] \to ACC^0 \to TC^0 \to NC^1$$

AC^0 ↔ polynomial size
constant depth
unbounded fan-in

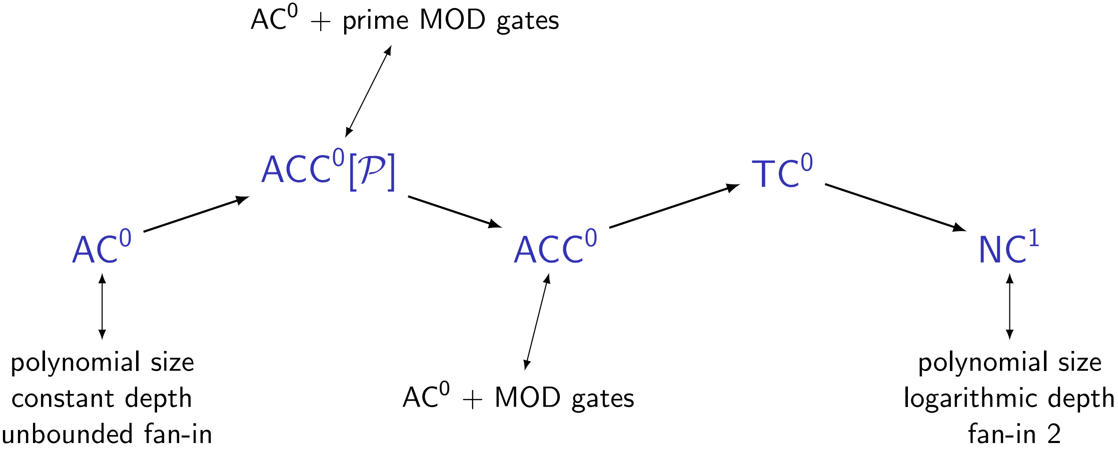
ACC^0 ↔ AC^0 + MOD gates

NC^1 ↔ polynomial size
logarithmic depth
fan-in 2

$\boxed{\text{MOD}}$: number of $1 \equiv_m 0$?

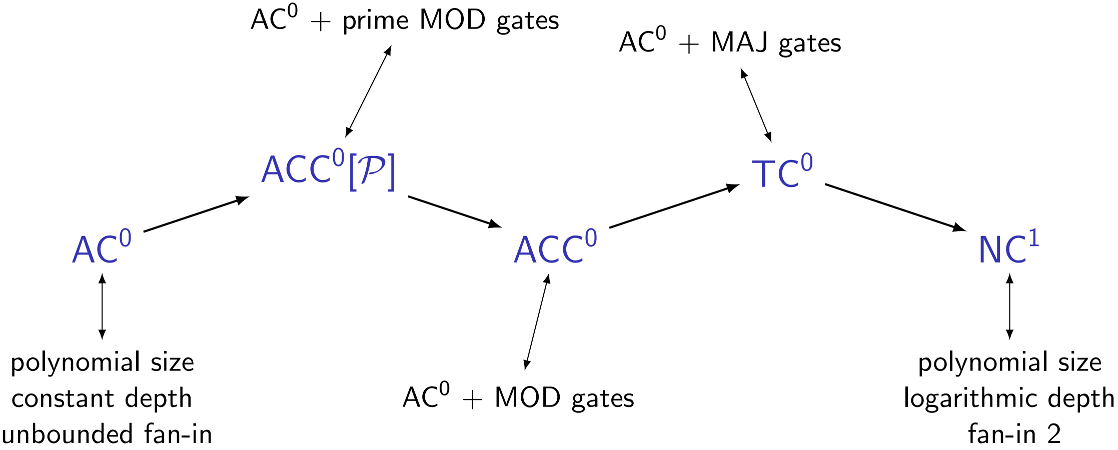
$\boxed{\text{MAJ}}$: number of $1 \geq$ number of 0 ?

Classes of small circuits



$\boxed{\text{MOD}}$: number of 1 $\equiv_m$ 0 ? $\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

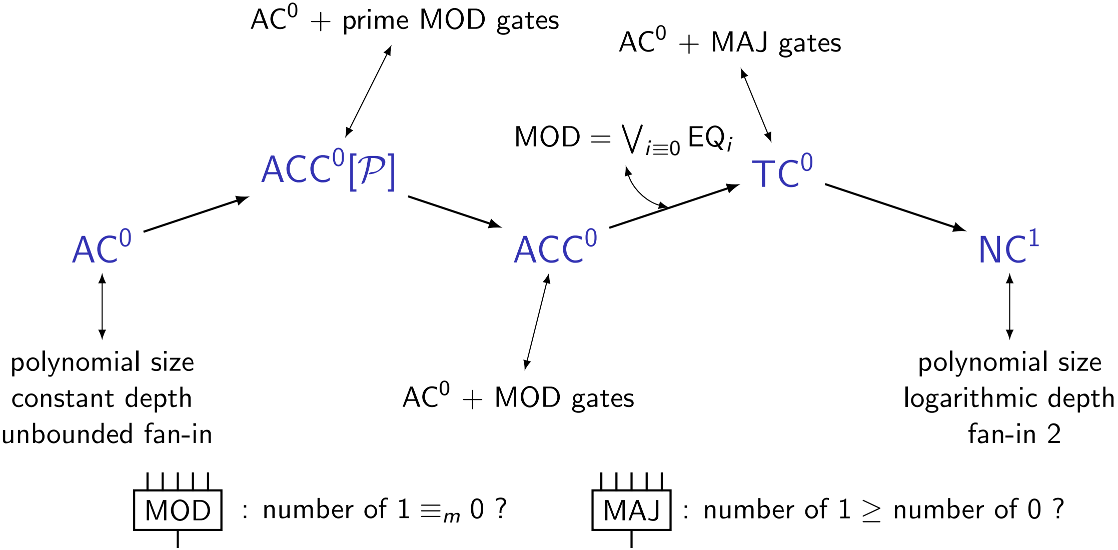
Classes of small circuits



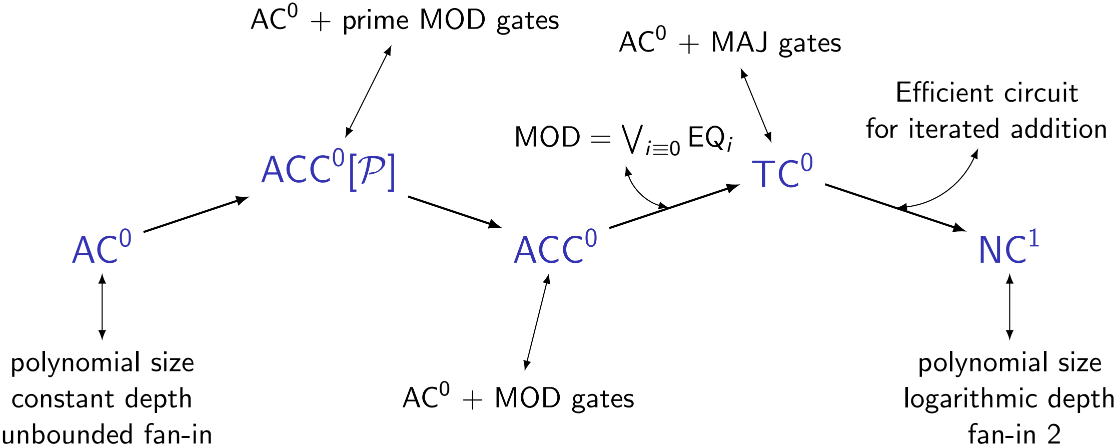
$\boxed{\text{MOD}}$: number of 1 $\equiv_m$ 0 ?

$\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

Classes of small circuits

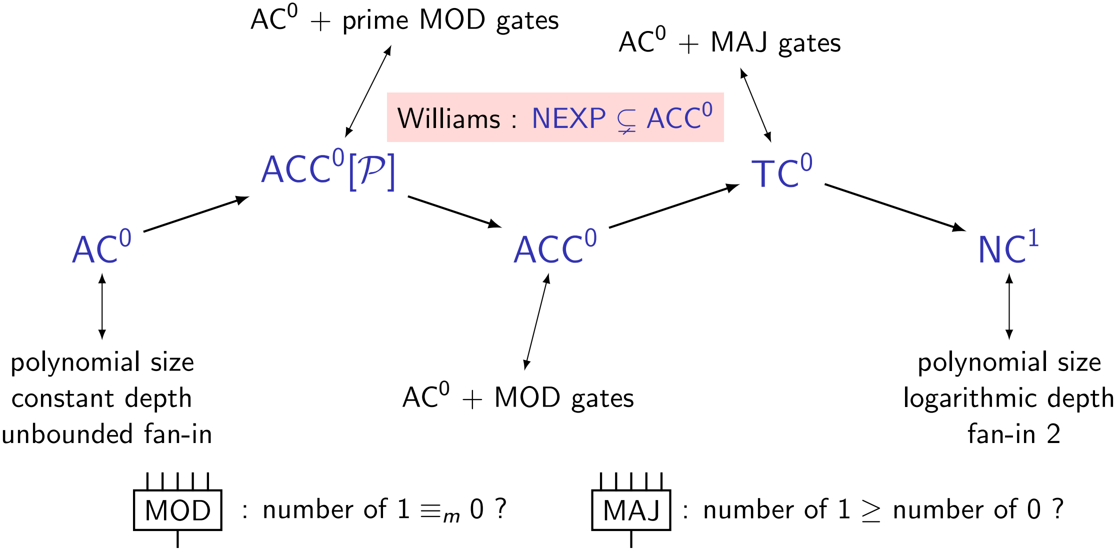


Classes of small circuits

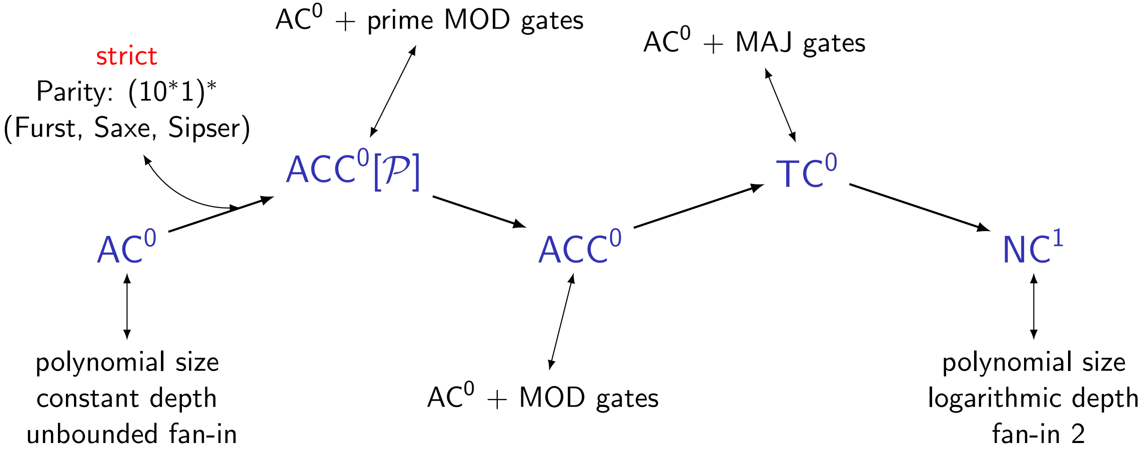


$\boxed{\text{MOD}}$: number of 1 $\equiv_m 0$? $\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

Classes of small circuits

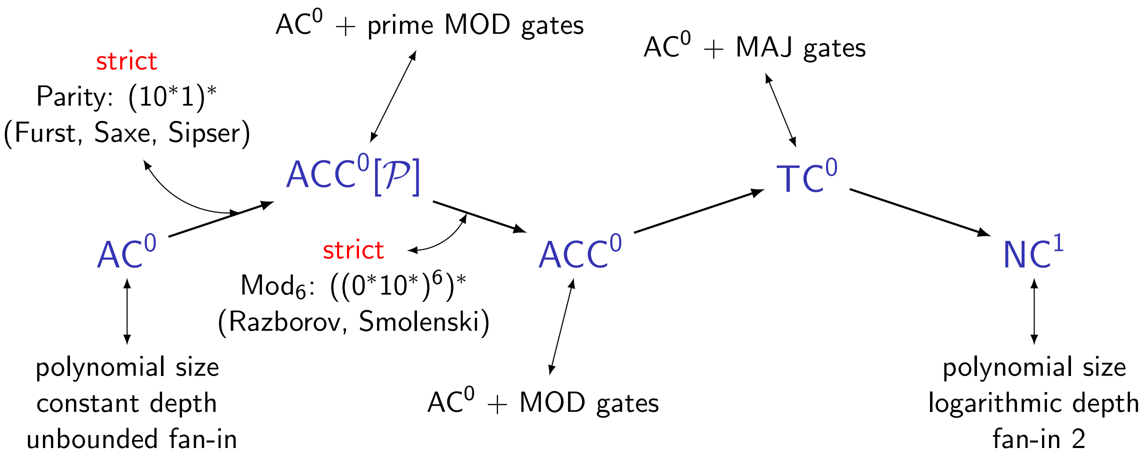


Classes of small circuits



$\boxed{\text{MOD}}$: number of 1 $\equiv_m$ 0 ? $\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

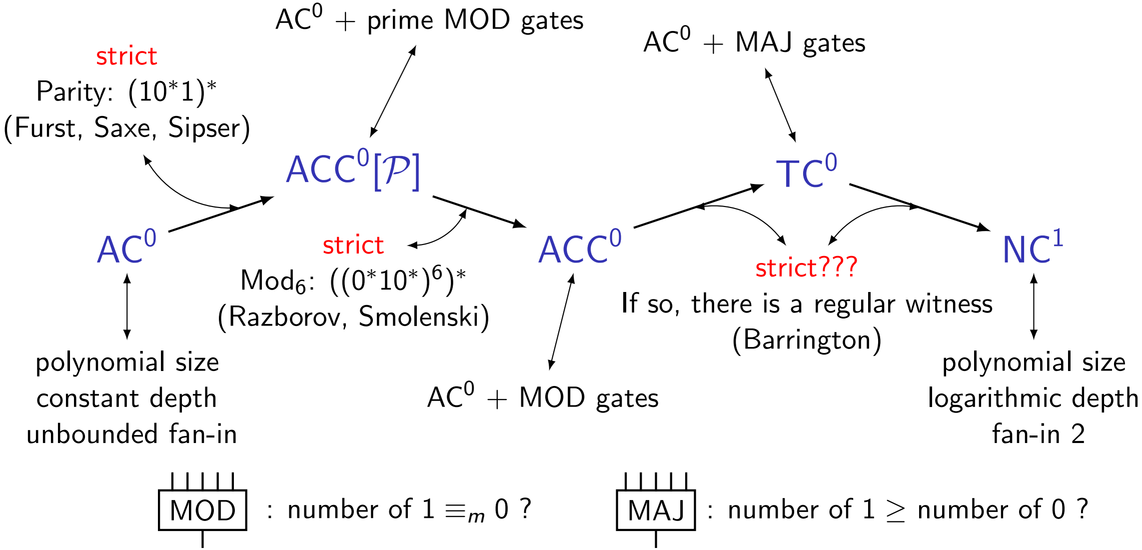
Classes of small circuits



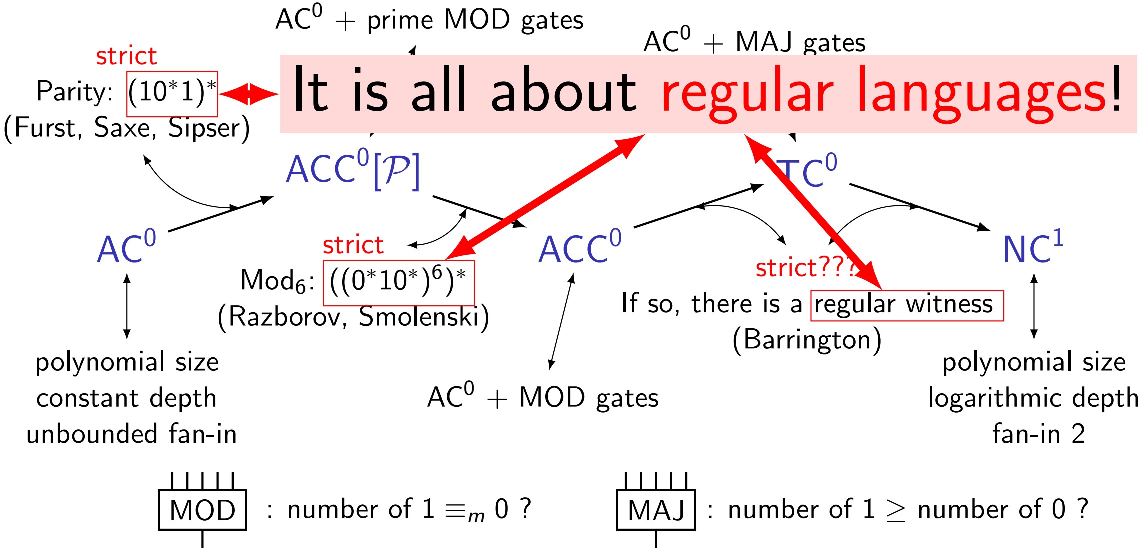
$\boxed{\text{MOD}}$: number of 1 $\equiv_m$ 0 ?

$\boxed{\text{MAJ}}$: number of 1 $\geq$ number of 0 ?

Classes of small circuits



Classes of small circuits



The case of NC^1

Theorem

All regular languages are in NC^1.

The case of NC^1

Theorem
All regular languages are in NC^1.

$\mathcal{A}$ an automaton: finitely many functions $Q \to Q$

The case of NC^1

Theorem

All regular languages are in NC^1.

$\mathcal{A}$ an automaton: finitely many functions $Q \to Q$

Bounded size circuits for:

- Given a letter a, its transition function
- The composition of two functions
- Whether a function maps the initial state to a final state

The case of NC^1

Theorem

All regular languages are in NC^1.

$\mathcal{A}$ an automaton: finitely many functions $Q \to Q$

Bounded size circuits for:

- Given a letter a, its transition function
- The composition of two functions
- Whether a function maps the initial state to a final state

Divide and conquer.

Expressing languages with logic

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

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

Expressing languages with logic

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

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

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

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

$\forall x,$  $(x) \Rightarrow a(x)$

$$\in \Pi_1[\mathrm{ARB}]$$

Circuits and formulas

$$\begin{tikzcd}
 & \mathsf{ACC}^0[\mathcal{P}] \arrow[dr] & & \mathsf{TC}^0 \arrow[dr] & \\
\mathsf{AC}^0 \arrow[ur] & & \mathsf{ACC}^0 \arrow[ur] & & \mathsf{NC}^1
\end{tikzcd}$$

Circuits and formulas

$$\mathsf{AC}^0 \to \mathsf{ACC}^0[\mathcal{P}] \to \mathsf{ACC}^0 \to \mathsf{TC}^0 \to \mathsf{NC}^1$$

FO[ARB]

Circuits and formulas

$\mathrm{AC}^0 \to \mathrm{ACC}^0[\mathcal{P}] \to \mathrm{ACC}^0 \to \mathrm{TC}^0 \to \mathrm{NC}^1$

AC^0: FO[ARB]

$\mathrm{ACC}^0[\mathcal{P}]$: FO+MOD[ARB] (prime)

ACC^0: FO+MOD[ARB]

$\exists^{mod}_{q,r}$ = # satisfiyng assignements is $\equiv$ r mod q

Circuits and formulas

$$
\begin{tikzcd}[column sep=small]
 & \mathrm{ACC}^0[\mathcal{P}] \arrow[dr] & & \mathrm{TC}^0 \arrow[dr] & \\
\mathrm{AC}^0 \arrow[ur] & \mathrm{FO{+}MOD}[\mathrm{ARB}] & \mathrm{ACC}^0 \arrow[ur] & \mathrm{FO{+}MAJ}[\mathrm{ARB}] & \mathrm{NC}^1 \\
 & \text{(prime)} & & & \\
\mathrm{FO}[\mathrm{ARB}] & & \mathrm{FO{+}MOD}[\mathrm{ARB}] & &
\end{tikzcd}
$$

$\exists^{mod}_{q,r}$ = # satisfiyng assignements is $\equiv$ r mod q

$\exists^{maj}$ = a majority of the assignements are satisfying

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):

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

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

- Σ_1
- FO $\longleftrightarrow$ AC^0
- FO with prime modular quantifiers

$\mathrm{ACC}^0[\mathcal{P}]$

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 $\longleftrightarrow$ AC^0 ▶ FO with prime modular quantifiers	▶ $\mathrm{FO} + S_5$

$\mathrm{ACC}^0[\mathcal{P}]$

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

- Σ_1
- $\mathrm{FO} \longleftrightarrow \mathrm{AC}^0$
- FO with prime modular quantifiers

$\mathrm{ACC}^0[\mathcal{P}]$

False

- $\mathrm{FO} + S_5$

Open

- 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):

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

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

True	False	Open
▶ Σ_1 ▶ FO $\longleftrightarrow$ AC^0 ▶ FO with prime modular quantifiers	▶ $\mathrm{FO} + S_5$	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

$\mathrm{ACC}^0[\mathcal{P}]$

Central question

For $\mathcal{L}$ a fragment,

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

A natural guess (Straubing):

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

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

True	False	Open
▶ Σ_1 ▶ FO $\longleftrightarrow$ AC^0 ▶ FO with prime modular quantifiers	▶ FO + S_5	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

$\mathrm{ACC}^0[\mathcal{P}]$

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):

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

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

True	False	Open
▶ Σ_1 ▶ FO ⟷ AC^0 ▶ FO with prime modular quantifiers	▶ FO + S_5	▶ FO with composite modular quantifiers ▶ FO with two variables ▶ Σ_k, $k \geq 3$

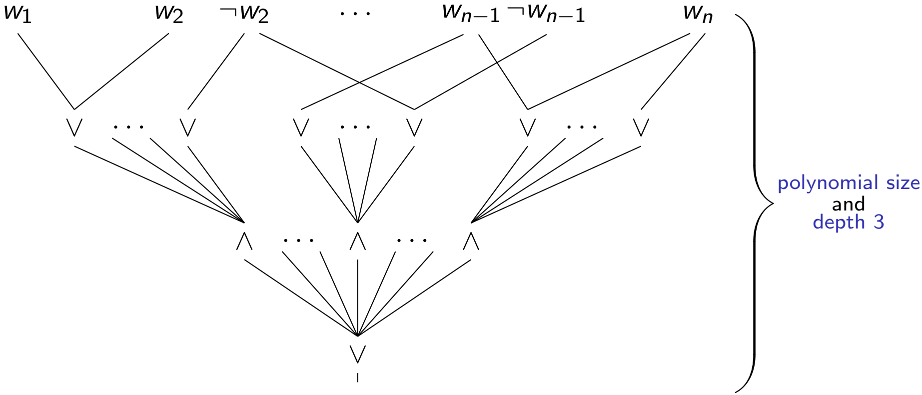
$\mathrm{ACC}^0[\mathcal{P}]$

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

Linear AC^0 (complexity of addition)

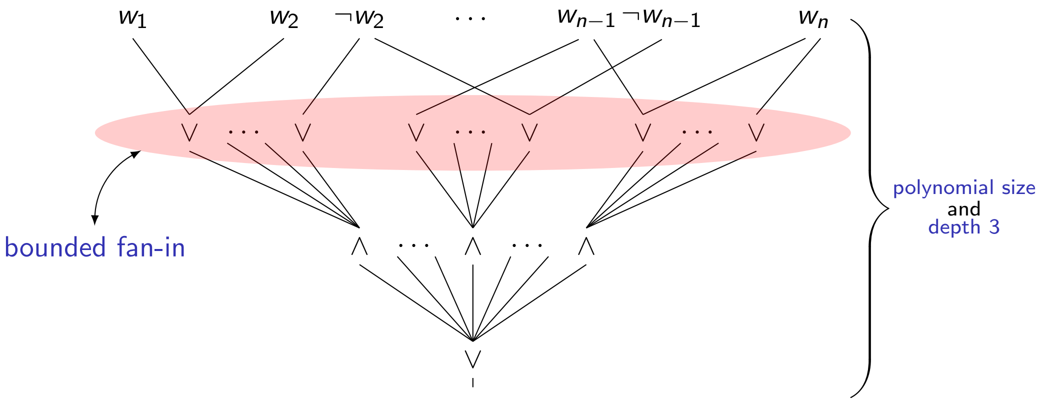
The circuit class Σ_2

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



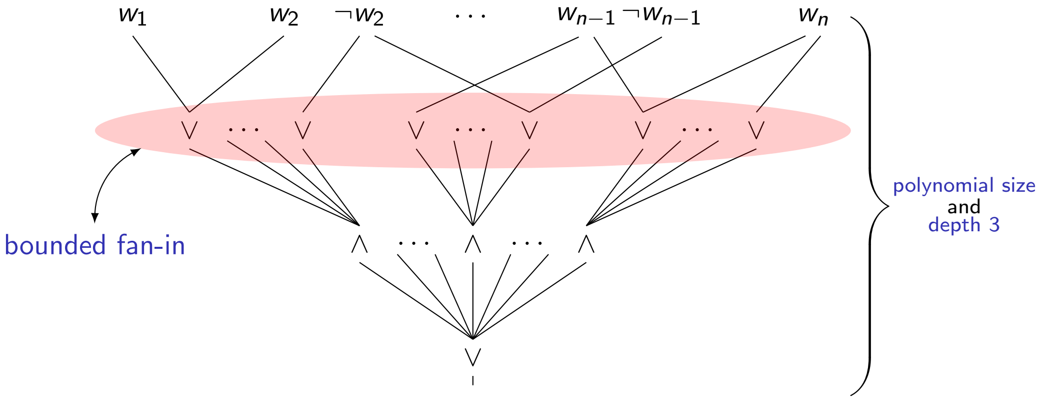
The circuit class Σ_2

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



The circuit class Σ_2

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



It is equivalent to Σ_2[ARB].

The case of Σ_2 (B,Cadilhac, Paperman, Zeume)

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

The case of Σ_2 (B,Cadilhac, Paperman, Zeume)

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

- $\supseteq$: Immediate.

The case of Σ_2 (B,Cadilhac, Paperman, Zeume)

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

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

The case of Σ_2 (B,Cadilhac, Paperman, Zeume)

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

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

The case of Σ_2 (B,Cadilhac, Paperman, Zeume)

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

- $\supseteq$: 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

$\boxed{\text{Algebra}}$ $\boxed{\text{Lower bound}}$

Proof sketch

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

Proof sketch

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

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

Proof sketch

Theorem (Pin, Weil)

$\mathcal{L}$ in Σ_2[REG] iff:

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

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

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.