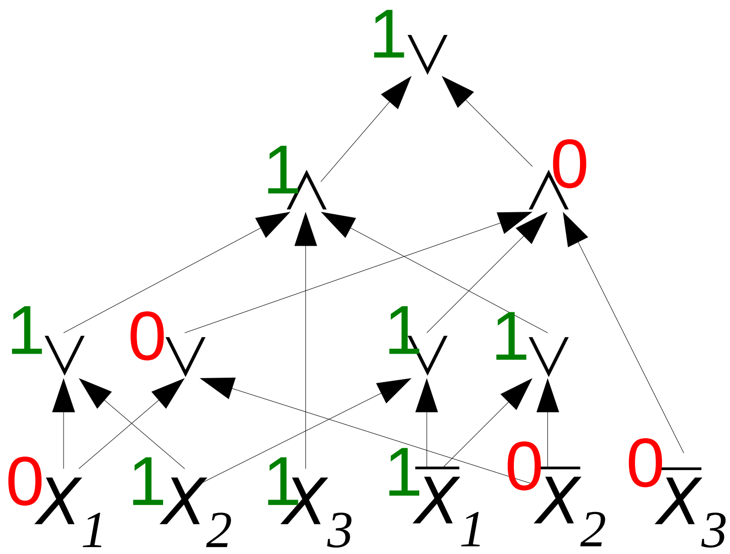
Computational complexity

lecture 5

Boolean circuits



Circuits of small depth

- class **AC**^{k} – languages recognizable by a sequence of circuits of depth $O((log(n))^k)$, and of polynomial size
- most interesting cases: **AC**^{0} (constant depth), **AC**^{1} (logarithmic depth)
- **AC**$=\bigcup_{k\in\mathbb{N}}$**AC**^{k}

Circuits of small depth

- class $\mathbf{AC}^k$ – languages recognizable by a sequence of circuits of depth $O((log(n))^k)$, and of polynomial size
- most interesting cases: $\mathbf{AC}^0$ (constant depth), $\mathbf{AC}^1$ (logarithmic depth)
- $\mathbf{AC}=\cup_{k\in\mathbb{N}}\mathbf{AC}^k$
- class $\mathbf{NC}^k$ – languages recognizable by a sequence of circuits of depth $O((log(n))^k)$, of polynomial size, and <u>of fan-in 2</u> (i.e., every gate has at most 2 predecessors)
- class $\mathbf{NC}^0$ is not interesting (only a constant number of bits is checked)
- $\mathbf{NC}=\cup_{k\in\mathbb{N}}\mathbf{NC}^k$

Circuits of small depth

Uniform variant:

- class **u-AC**^{k} – languages recognizable by a <u>uniform</u> (i.e., computable in logarithmic space) sequence of circuits of depth $O((log(n))^k)$
- **u-AC**=$\cup_{k\in\mathbb{N}}$**u-AC**^{k}
- class **u-NC**^{k} – languages recognizable by a <u>uniform</u> sequence of circuits of depth $O((log(n))^k)$ and <u>of fan-in 2</u>
- **u-NC**=$\cup_{k\in\mathbb{N}}$**u-NC**^{k}

implies polynomial size

Remark: Different names are used for these classes: **uniform-AC**^{k} or **u-AC**^{k} or **U$_L$-AC**^{k} or **AC**^{k} (i.e., some authors already in the definition of **AC**^{k} assume that the sequence of circuits is uniform)

Circuits of small depth

Example:

Binary matrix multiplication is in **u-AC**^{0}

[more precisely: the language of tuples (M,N,i,j) such that $(M \cdot N)_{i,j} = 1$]

$$(M \cdot N)_{i,j} = \bigvee_k M_{i,k} \wedge N_{k,j}$$

- level 1: compute $M_{i,k} \wedge N_{k,j}$ for every (i,j,k)
- level 2: for every (i,j) compute a big disjunction
- additional two levels: select the cell (i,j) specified on input
- it is easy to generate this circuit in logarithmic space

Circuits of small depth

Example:

Binary matrix multiplication is in **u-AC**^{0}

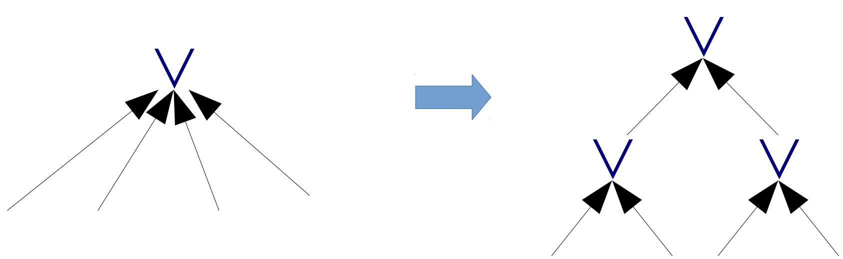
[more precisely: the language of tuples (M,N,i,j) such that $(M{\cdot}N)_{i,j}=1$]

$$(M{\cdot}N)_{i,j}=\bigvee_k M_{i,k}\wedge N_{k,j}$$

- level 1: compute $M_{i,k}\wedge N_{k,j}$ for every (i,j,k)
- level 2: for every (i,j) compute a big disjunction
- additional two levels: select the cell (i,j) specified on input
- it is easy to generate this circuit in logarithmic space

Binary matrix multiplication is in **u-NC**^{1} as well

- a disjunction of n values (on level 2) can be realized as a tree of depth $log(n)$ consisting of n-1 disjunctions of fan-in 2



Circuits of small depth

The same can be done in general:

every disjunction (conjunction) of m values can be replaced by a tree of depth $log(m) \leq c \cdot log(n)$ consisting of m-1 disjunctions (conjunctions) of fan-in 2

Thus we obtain that:

AC$^k \subseteq$**NC**$^{k+1}$ & **u-AC**$^k \subseteq$**u-NC**$^{k+1}$

By definition we also have that:

NC$^k \subseteq$**AC**^{k} & **u-NC**$^k \subseteq$**u-AC**^{k}

Thus in particular:

AC=**NC** & **u-AC**=**u-NC**

Circuits of small depth

Intuition: **u-NC** contains problems, which can be quickly solved by parallel algorithm

An open problem: does **u-NC**≠**P**?

Circuits of small depth

Intuition: **u-NC** contains problems, which can be quickly solved by parallel algorithm

An open problem: does **u-NC**≠**P**?

We have a sequence of inclusions:

$\mathbf{u\text{-}AC}^0 \subseteq \mathbf{u\text{-}NC}^1 \subseteq \mathbf{u\text{-}AC}^1 \subseteq \mathbf{u\text{-}NC}^2 \subseteq \ldots \subseteq \mathbf{u\text{-}AC} = \mathbf{u\text{-}NC} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$

It is <u>conjectured</u> that all of them are strict, but it is only known that:

- $\mathbf{u\text{-}AC}^0 \neq \mathbf{u\text{-}NC}^1$
- **u-NC**≠**PSPACE**

Circuits of small depth

Intuition: **u-NC** contains problems, which can be quickly solved by parallel algorithm

An open problem: does **u-NC≠P**?

We have a sequence of inclusions:

$\mathbf{u\text{-}AC}^0 \subseteq \mathbf{u\text{-}NC}^1 \subseteq \mathbf{u\text{-}AC}^1 \subseteq \mathbf{u\text{-}NC}^2 \subseteq \ldots \subseteq \mathbf{u\text{-}AC} = \mathbf{u\text{-}NC} \subseteq \mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PSPACE}$

It is <u>conjectured</u> that all of them are strict, but it is only known that:

- $\mathbf{u\text{-}AC}^0 \neq \mathbf{u\text{-}NC}^1$
- **u-NC≠PSPACE**

Why **u-NC≠PSPACE**?

Follows from the hierarchy theorem, because $\mathbf{u\text{-}NC} \subseteq \mathbf{polyL}$

(on tutorials you will prove that $\mathbf{u\text{-}NC}^1 \subseteq \mathbf{L}$)

Why $\mathbf{u\text{-}AC}^0 \neq \mathbf{u\text{-}NC}^1$?

<u>Following slides</u>

The parity language

PARITY – the language of those words $\{0,1\}$ in which the number of ones is even

Fact: PARITY$\in$**u-NC**^{1}

We count ones modulo 2 – circuit of tree-like shape.

Theorem (1986): PARITY$\notin$**AC**^{0}

Proof – the following part of the lecture

- It is one of quite rare nontrivial proofs saying that some problem cannot be solved in some complexity class.
- (Mostly hardness theorems are relative – if a problem A is hard, then a problem B is hard, e.g. NP-completeness)

PARITY$\notin$**AC**^{0}

- We are going to consider multi-variable polynomials over the field $\mathbb{Z}_3=\{0,1,2\}$ (we will use them to approximate the behavior of a circuit)
- A polynomial p (of n variables) is called *proper* if for arguments in $\{0,1\}^n$ it gives results in $\{0,1\}$ (we are interested only in such polynomials - they define a boolean function of n variables, like circuits)

PARITY$\notin$**AC**^{0}

- We are going to consider multi-variable polynomials over the field $\mathbb{Z}_3=\{0,1,2\}$ (we will use them to approximate the behavior of a circuit)
- A polynomial p (of n variables) is called *proper* if for arguments in $\{0,1\}^n$ it gives results in $\{0,1\}$ (we are interested only in such polynomials - they define a boolean function of n variables, like circuits)
- The *total degree* of a polynomial p is defined as the sum of exponents in a monomial in p, e.g., $x^4y^1+x^1y^2z^3$ has degree 6

PARITY$\notin$**AC**^{0}

- We are going to consider multi-variable polynomials over the field $\mathbb{Z}_3=\{0,1,2\}$ (we will use them to approximate the behavior of a circuit)
- A polynomial p (of n variables) is called *proper* if for arguments in $\{0,1\}^n$ it gives results in $\{0,1\}$ (we are interested only in such polynomials - they define a boolean function of n variables, like circuits)
- The *total degree* of a polynomial p is defined as the sum of exponents in a monomial in p, e.g., $x^4y^1+x^1y^2z^3$ has degree 6

Fix a depth d. We will prove that PARITY cannot be recognized by a sequence (even not necessarily uniform) of circuits of depth d and polynomial size.

PARITY$\notin$**AC**^{0}

- We are going to consider multi-variable polynomials over the field $\mathbb{Z}_3=\{0,1,2\}$ (we will use them to approximate the behavior of a circuit)
- A polynomial p (of n variables) is called *proper* if for arguments in $\{0,1\}^n$ it gives results in $\{0,1\}$ (we are interested only in such polynomials - they define a boolean function of n variables, like circuits)
- The *total degree* of a polynomial p is defined as the sum of exponents in a monomial in p, e.g., $x^4y^1+x^1y^2z^3$ has degree 6

Fix a depth d. We will prove that PARITY cannot be recognized by a sequence (even not necessarily uniform) of circuits of depth d and polynomial size.

General idea:

- Every circuit of small depth can be approximated by a proper polynomial of low degree (Lemma 1)
- The parity function cannot be approximated by a polynomial of low degree (Lemma 2)

PARITY $\notin$ **AC**^{0}

<u>Lemma 1.</u> For every $t>0$ and n, for every circuit C with n input gates and depth d there exists a proper polynomial of n variables and total degree $\leq (2t)^d$, which differs from C on at most $\frac{|C|}{2^t} 2^n$ inputs (where $|C|$ denotes the number of gates in C)

PARITY$\notin$**AC**^{0}

<u>Lemma 1.</u> For every $t>0$ and n, for every circuit C with n input gates and depth d there exists a proper polynomial of n variables and total degree $\leq(2t)^d$, which differs from C on at most $\frac{|C|}{2^t}2^n$ inputs (where $|C|$ denotes the number of gates in C)

We will use this lemma with $2t=n^{1/(2d)}$

Then we obtain polynomials of degree $\leq\sqrt{n}$, while the fraction $|C|/2^t$ tends to 0 when $|C|$ is polynomial in n, and d is constant.

PARITY$\notin$**AC**^{0}

<u>Lemma 1.</u> For every $t>0$ and n, for every circuit C with n input gates and depth d there exists a proper polynomial of n variables and total degree $\leq (2t)^d$, which differs from C on at most $\frac{|C|}{2^t}2^n$ inputs (where $|C|$ denotes the number of gates in C)

We will use this lemma with $2t=n^{1/(2d)}$

Then we obtain polynomials of degree $\leq\sqrt{n}$, while the fraction $|C|/2^t$ tends to 0 when $|C|$ is polynomial in n, and d is constant.

<u>Lemma 2.</u> For large enough n every polynomial of n variables and total degree $\leq\sqrt{n}$ differs from the parity function on at least $\frac{1}{100}2^n$ inputs.

Lemma 1 + Lemma 2 $\rightarrow$ polynomial circuits of constant depth cannot recognize PARITY

Proof of Lemma 1 (∗)

<u>Lemma 1.</u> For every $t>0$ and n, for every circuit C with n input gates and depth d there exists a proper polynomial of n variables and total degree $\leq (2t)^d$, which differs from C on at most $\frac{|C|}{2^t} 2^n$ inputs (where $|C|$ denotes the number of gates in C)

<u>Proof.</u>

- Fix n, t and a circuit C of depth d.
- Assume w.l.o.g. that C uses only OR and NOT gates.
- To every gate of C we will assign a proper polynomial of n variables $x_1,\ldots,x_n$, by induction on the depth of the gate, so that it will compute the value of this gate C for relatively many inputs

Proof of Lemma 1 (∗)

To every gate of C we will assign a proper polynomial of n variables $x_1,\ldots,x_n$, by induction on the depth of the gate, so that it will compute the value of this gate C for relatively many inputs:

- i-th input gate – take the polynomial x_i, which always computes a correct value
- NOT gate. If we have assigned a polynomial p to its predecessor, we take polynomial $1-p$, which computes a correct value precisely when p computed a correct value
- it remains to handle OR gates – the only nontrivial case

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we could take the polynomial: $1-(1-p_1)\cdot\ldots\cdot(1-p_k)$
- it works well whenever $p_1,\dots,p_k$ worked well
- but its degree is too large: if $p_1,\dots,p_k$ have degrees at most s, then its degree is ks – we rather need to obtain $\leq 2ts$, as then on the output gate we will have degree $(2t)^d$
- we thus have to proceed in a more clever way

Proof of Lemma 1 (∗)

Consider an <u>OR gate</u> of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we could take the polynomial: $1-(1-p_1)\cdot\ldots\cdot(1-p_k)$
- it works well whenever $p_1,\dots,p_k$ worked well
- but its degree is too large: if $p_1,\dots,p_k$ have degrees at most s, then its degree is ks – we rather need to obtain $\leq 2ts$, as then on the output gate we will have degree $(2t)^d$
- we thus have to proceed in a more clever way
- in a moment, we will appropriately choose sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$
- we will take the polynomial:

$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t)$ where $q_i=(\sum_{j\in S_i}p_j)^2$

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- in a moment, we will appropriately choose sets $S_1,\dots,S_t \subseteq \{1,\dots,k\}$
- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=\left(\sum_{j\in S_i} p_j\right)^2$$

- p is proper, since $\{0^2,1^2,2^2\}=\{0,1\}$

Proof of Lemma 1 (⋆)

Consider an <u>OR gate</u> of fan-in k. To its arguments we have assigned some polynomials $p_1,\ldots,p_k$.

- in a moment, we will appropriately choose sets $S_1,\ldots,S_t \subseteq \{1,\ldots,k\}$
- we will take the polynomial:

  $p=1-(1-q_1)\cdot\ldots\cdot(1-q_t)$ where $q_i=(\sum_{j\in S_i} p_j)^2$
- p is proper, since $\{0^2,1^2,2^2\}=\{0,1\}$
- if degrees of $p_1,\ldots,p_k$ are $\leq s$, then the degree of p is $\leq 2ts$;

  then for the output gate of C we obtain degree $\leq (2t)^d$ – as required in the lemma
- it remains to see that p approximates well the value of the gate (for an appropriate choice of the sets $S_1,\ldots,S_t$)

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\textstyle\sum_{j\in S_i}p_j)^2$$

Fix some input (of the whole circuit C) on which all $p_1,\dots,p_k$ give correct values. Let us randomly choose sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ (every list of sets has the same probability)

Proof of Lemma 1 (⋆)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\ldots,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\textstyle\sum_{j\in S_i}p_j)^2$$

Fix some input (of the whole circuit C) on which all $p_1,\ldots,p_k$ give correct values. Let us randomly choose sets $S_1,\ldots,S_t\subseteq\{1,\ldots,k\}$ (every list of sets has the same probability)

- If all p_j give value 0, then p also gives value 0 – correctly

Proof of Lemma 1 (∗)

Consider an <u>OR gate</u> of fan-in k. To its arguments we have assigned some polynomials $p_1,\ldots,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\textstyle\sum_{j\in S_i} p_j)^2$$

Fix some input (of the whole circuit C) on which all $p_1,\ldots,p_k$ give correct values. Let us randomly choose sets $S_1,\ldots,S_t\subseteq\{1,\ldots,k\}$ (every list of sets has the same probability)

- If all p_j give value 0, then p also gives value 0 – correctly
- If some p_j gives value 1, then for a chosen set S_i the polynomial q_i gives value 1 if in this set S_i the number of polynomials with value 1 is not divisible by 3. This is the case for at least half of choices of S_i. Thus the probability that for a random S_i the polyno-mial q_i gives value 1 is ≥ 0.5 (then the whole p also gives value 1).

Proof of Lemma 1 (∗)

Consider an <u>OR gate</u> of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\textstyle\sum_{j\in S_i} p_j)^2$$

Fix some input (of the whole circuit C) on which all $p_1,\dots,p_k$ give correct values. Let us randomly choose sets $S_1,\dots,S_t \subseteq \{1,\dots,k\}$ (every list of sets has the same probability)

- If all p_j give value 0, then p also gives value 0 – correctly
- If some p_j gives value 1, then for a chosen set S_i the polynomial q_i gives value 1 if in this set S_i the number of polynomials with value 1 is not divisible by 3. This is the case for at least half of choices of S_i. Thus the probability that for a random S_i the polyno-mial q_i gives value 1 is ≥ 0.5 (then the whole p also gives value 1).
- Thus, if the sets $S_1,\dots,S_t \subseteq \{1,\dots,k\}$ are chosen randomly, the proba-bility that p will give an incorrect value is at most $1/2^t$

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\ldots,p_k$.

- we will take the polynomial:

  $p=1-(1-q_1)\cdot\ldots\cdot(1-q_t)$ where $q_i=(\sum_{j\in S_i}p_j)^2$

- For a fixed input, for which all $p_1,\ldots,p_k$ give correct values, and for sets $S_1,\ldots,S_t\subseteq\{1,\ldots,k\}$ chosen randomly, the probability that p gives an incorrect value is at most $1/2^t$

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we will take the polynomial:

  $p=1-(1-q_1)\cdot\ldots\cdot(1-q_t)$ where $q_i=(\sum_{j\in S_i} p_j)^2$

- For a fixed input, for which all $p_1,\dots,p_k$ give correct values, and for sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ chosen randomly, the probability that p gives an incorrect value is at most $1/2^t$
- Thus: for an input chosen randomly among those inputs for which all $p_1,\dots,p_k$ give correct values, and for sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ chosen randomly, the probability that p gives an incorrect value is at most $1/2^t$

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,\dots,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot\ldots\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\textstyle\sum_{j\in S_i} p_j)^2$$

- For a fixed input, for which all $p_1,\dots,p_k$ give correct values, and for sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ chosen randomly, the probability that p gives an incorrect value is at most $1/2^t$
- Thus: for an input chosen randomly among those inputs for which all $p_1,\dots,p_k$ give correct values, and for sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ chosen randomly, the probability that p gives an incorrect value is at most $1/2^t$
- Thus: there exist sets $S_1,\dots,S_t\subseteq\{1,\dots,k\}$ such that for an input chosen randomly among those inputs for which all $p_1,\dots,p_k$ give correct values, the probability that p gives an incorrect value is at most $1/2^t$

Proof of Lemma 1 (∗)

Consider an OR gate of fan-in k. To its arguments we have assigned some polynomials $p_1,...,p_k$.

- we will take the polynomial:

$$p=1-(1-q_1)\cdot...\cdot(1-q_t) \qquad \text{where} \qquad q_i=(\sum_{j\in S_i} p_j)^2$$

- Thus: there exist sets $S_1,...,S_t\subseteq\{1,...,k\}$ such that for an input chosen randomly among those inputs for which all $p_1,...,p_k$ give correct values, the probability that p gives an incorrect value is at most $1/2^t$
- We take an arbitrary list of sets having this property
- The considered gate introduces a mistake on at most $2^n/2^t$ inputs
- Altogether, the value will be incorrect (for some gate) for at most $|C|\cdot 2^n/2^t$ inputs

[THE END OF THE PROOF OF LEMMA 1]

PARITY$\notin$**AC**^{0}

General idea:

- Every circuit of small depth can be approximated by a proper polynomial of low degree (Lemma 1 – already showed)
- The parity function cannot be approximated by a polynomial of low degree (Lemma 2 – now)

Proof of Lemma 2 (∗)

<u>Lemma 2.</u> For large enough n every polynomial of n variables and total degree $\leq\sqrt{n}$ differs from the parity function on at least $\frac{1}{100}2^n$ inputs.

A general idea:

- We assume that there exists a polynomial of low degree which agrees with the parity function on a large set S of inputs.
- Using this polynomial, for every function we will construct a polynomial of low degree which agrees with this function on the same set S.
- There are many functions, but significantly less polynomials.
- Thus the set S cannot be too large.

Proof of Lemma 2 (∗)

Lemma 2. For large enough n every polynomial of n variables and total degree $\leq\sqrt{n}$ differs from the parity function on at least $\frac{1}{100}2^n$ inputs.

- Let $PAR(x_1,...,x_n)$ denote the parity function
- Consider the „shifted” parity function $PAR':\{-1,1\}^n\rightarrow\{-1,1\}$ $PAR'(x_1,...,x_n)=PAR(x_1-1,...,x_n-1)+1=x_1\cdot x_2\cdot...\cdot x_n$

Proof of Lemma 2 (∗)

<u>Lemma 2.</u> For large enough n every polynomial of n variables and total degree $\leq\sqrt{n}$ differs from the parity function on at least $\frac{1}{100}2^n$ inputs.

- Let $PAR(x_1,\ldots,x_n)$ denote the parity function
- Consider the „shifted" parity function $PAR':\{-1,1\}^n\rightarrow\{-1,1\}$ $PAR'(x_1,\ldots,x_n)=PAR(x_1-1,\ldots,x_n-1)+1=x_1\cdot x_2\cdot\ldots\cdot x_n$
- If there exists a polynomial which agrees with PAR on some set of inputs, then there exists a polynomial of the same degree, which agrees with PAR' on the same set
- Thus take a polynomial p of degree $\leq\sqrt{n}$ approximating PAR' Let $S\subseteq\{-1,1\}^n$ be the set of those inputs in which p agrees with PAR'.

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- Take any function $f{:}S\rightarrow\mathbb{Z}_3$
- We can always represent f as a polynomial:

$$p_f(x_1,\ldots,x_n)=\sum_{(y_1,\ldots,y_n)\in S} f(y_1,\ldots,y_n)\cdot(2-x_1y_1)\cdot\ldots\cdot(2-x_ny_n)$$

- This polynomial has degree n, too large for us
- We will correct it so that the degree will be $\leq n/2+\sqrt{n}$

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- Take any function $f:S\rightarrow\mathbb{Z}_3$
- We can always represent f as a polynomial:

$$p_f(x_1,\ldots,x_n)=\sum_{(y_1,\ldots,y_n)\in S} f(y_1,\ldots,y_n)\cdot(2-x_1y_1)\cdot\ldots\cdot(2-x_ny_n)$$

- This polynomial has degree n, too large for us
- We will correct it so that the degree will be $\leq n/2+\sqrt{n}$
- To this end, in p_f we replace every monomial $\prod_{i\in T}x_i$ of degree $|T|>n/2$ by $p(x_1,\ldots,x_n)\cdot\prod_{i\notin T}x_i$

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- Take any function $f:S\rightarrow\mathbb{Z}_3$
- We can always represent f as a polynomial:

$$p_f(x_1,\ldots,x_n)=\sum_{(y_1,\ldots,y_n)\in S} f(y_1,\ldots,y_n)\cdot(2-x_1y_1)\cdot\ldots\cdot(2-x_ny_n)$$

- This polynomial has degree n, too large for us
- We will correct it so that the degree will be $\leq n/2+\sqrt{n}$
- To this end, in p_f we replace every monomial $\prod_{i\in T}x_i$ of degree $|T|>n/2$ by $p(x_1,\ldots,x_n)\cdot\prod_{i\notin T}x_i$
- This modification does not change the result, as for $(x_1,\ldots,x_n)\in S$ we have $p(x_1,\ldots,x_n)=x_1\cdot\ldots\cdot x_n$ and $(x_1)^2=1$
- Now the degree is indeed $\leq n/2+\sqrt{n}$

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- Take any function $f{:}S\rightarrow\mathbb{Z}_3$
- We can always represent f as a polynomial:

$$p_f(x_1,\ldots,x_n)=\sum_{(y_1,\ldots,y_n)\in S} f(y_1,\ldots,y_n)\cdot(2\text{-}x_1y_1)\cdot\ldots\cdot(2\text{-}x_ny_n)$$

- This polynomial has degree n, too large for us
- We will correct it so that the degree will be $\leq n/2+\sqrt{n}$
- To this end, in p_f we replace every monomial $\prod_{i\in T}x_i$ of degree $|T|>n/2$ by $p(x_1,\ldots,x_n)\cdot\prod_{i\notin T}x_i$
- This modification does not change the result, as for $(x_1,\ldots,x_n)\in S$ we have $p(x_1,\ldots,x_n)=x_1\cdot\ldots\cdot x_n$ and $(x_1)^2=1$
- Now the degree is indeed $\leq n/2+\sqrt{n}$
- Thus (using the hypothetical polynomial p) for every function $f{:}S\rightarrow\mathbb{Z}_3$ we have constructed a polynomial of degree $\leq n/2+\sqrt{n}$, which on S gives the same values as f

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- For every function $f{:}S\rightarrow\mathbb{Z}_3$ we have constructed a polynomial of degree $\leq n/2+\sqrt{n}$, which on S gives the same values as f
- For inputs in $\{-1,1\}^n$ we have that $x^2=1$, so we can assume that in the polynomial there are no exponents greater than 1.

Proof of Lemma 2 (∗)

- A polynomial p of degree $\leq\sqrt{n}$ agrees with PAR' on a set $S\subseteq\{-1,1\}^n$.
- For every function $f{:}S\rightarrow\mathbb{Z}_3$ we have constructed a polynomial of degree $\leq n/2+\sqrt{n}$, which on S gives the same values as f
- For inputs in $\{-1,1\}^n$ we have that $x^2=1$, so we can assume that in the polynomial there are no exponents greater than 1.

Let us compute the number of such polynomials:

- For large enough n, there are $\leq 0.99\cdot 2^n$ <u>monomials</u> of n variables and degree $\leq n/2+\sqrt{n}$, using every variable at most once (next slide)
- Thus the number of <u>polynomials</u> is $\leq 3^{0.99\cdot 2^n}$
- The number of functions $f{:}S\rightarrow\mathbb{Z}_3$ is $3^{|S|}$, to each of them we have assigned a different polynomial
- Thus $|S|\leq 0.99\cdot 2^n$

Proof of Lemma 2 (∗)

Why the number of monomials (using variables $x_1,\ldots,x_n$, each of them either with exponent 0 or 1) of degree $\leq n/2+\sqrt{n}$ is $\leq 0.99\cdot 2^n$, for large enough n?

- Choose a monomial in random
- Let X_i=(does x_i appear in the monomial)
- Random variables X_i are independent and $P(X_i=0)=P(X_i=1)=0.5$
- <u>Central limit theorem</u>: for every $z\in\mathbb{R}$, $P(Z_n\leq z) \xrightarrow{n\to\infty} \Phi(z)$
  where
  $$Z_n = \frac{\sum_{i=1}^{n}(X_i-\mu)}{\sqrt{n}\sigma}$$
  and $\mu=EX_i=0.5$, $\sigma=sd(X_i)=0.5$, and Φ is the cumulative distribution function of the normal distribution $N(0,1)$
- Notice that $X_1+\ldots+X_n\leq n/2+\sqrt{n} \Leftrightarrow Z_n\leq 2$, and $\Phi(2)\approx 0{,}97725$
- Thus for large enough n, the probability that the degree is $\leq n/2+\sqrt{n}$ i.e., $P(Z_n\leq 2)$ is at most $0{,}99$

[THE END OF THE PROOF OF LEMMA 2]

Extensions of **AC**^{0}

Consider circuits like in **AC**^{0}, where additionally we can use the XOR gate. Then we can recognize PARITY.
Is it enough to recognize, e.g., all regular languages?

Extensions of **AC**^{0}

Consider circuits like in **AC**^{0}, where additionally we can use the XOR gate. Then we can recognize PARITY.
Is it enough to recognize, e.g., all regular languages?

- Class **AC**$^0[m]$ – like **AC**^{0}, but where we can additionally use gates counting the number of ones modulo m
- It is known that: if p,q are different <u>prime</u> numbers, then **AC**$^0[p]$ cannot count modulo q
- An open problem: we cannot show any language, even from **NP**, which cannot be recognized in **AC**$^0[6]$
  (gates „mod 6” ⇔ gates „mod 2” i gates „mod 3”)