

Circuit Lower-bounds

Lecture 25

Weak circuits are indeed weak

Circuit Lower-bounds

Circuit Lower-bounds

- Today:

Circuit Lower-bounds

- Today:
 - **PARITY $\notin AC^0$**

Circuit Lower-bounds

- Today:
 - **PARITY $\notin AC^0$**
 - Two different proofs! (Latter generalizes to ACC^0)

Circuit Lower-bounds

- Today:
 - **PARITY $\notin AC^0$**
 - Two different proofs! (Latter generalizes to ACC^0)
 - **CLIQUE cannot be decided by poly-sized monotone circuits**

Circuit Lower-bounds

- Today:
 - **PARITY $\notin AC^0$**
 - Two different proofs! (Latter generalizes to ACC^0)
 - **CLIQUE cannot be decided by poly-sized monotone circuits**
- (Only sketches/partial proofs. See textbook or lecture-notes from linked courses)

PARITY \notin AC⁰

PARITY \notin AC⁰

- Recall AC⁰

PARITY \notin AC⁰

- Recall AC⁰
 - Poly size, constant depth (unbounded fan-in)

PARITY \notin AC⁰

- Recall AC⁰
 - Poly size, constant depth (unbounded fan-in)
 - Today, non-uniform AC⁰

PARITY \notin AC⁰

- Recall AC⁰
 - Poly size, constant depth (unbounded fan-in)
 - Today, non-uniform AC⁰
 - How powerful can AC⁰ be?

PARITY \notin AC⁰

- Recall AC⁰
 - Poly size, constant depth (unbounded fan-in)
 - Today, non-uniform AC⁰
 - How powerful can AC⁰ be?
- Recall PARITY

PARITY \notin AC⁰

- Recall AC⁰
 - Poly size, constant depth (unbounded fan-in)
 - Today, non-uniform AC⁰
 - How powerful can AC⁰ be?
- Recall PARITY
 - How shallow can a poly-sized circuit family for PARITY be?

A Switching Argument

A Switching Argument

- For the sake of contradiction, suppose constant depth (say $\leq d$, d being minimal) circuits for PARITY

A Switching Argument

- For the sake of contradiction, suppose constant depth (say $\leq d$, d being minimal) circuits for PARITY
 - Show depth $d-1$ circuits for every input size n : start from depth d circuit for a larger n' , and construct one for the smaller n .

A Switching Argument

- For the sake of contradiction, suppose constant depth (say $\leq d$, d being minimal) circuits for PARITY
 - Show depth $d-1$ circuits for every input size n : start from depth d circuit for a larger n' , and construct one for the smaller n .
 - By “restricting” to n inputs

A Switching Argument

- For the sake of contradiction, suppose constant depth (say $\leq d$, d being minimal) circuits for PARITY
 - Show depth $d-1$ circuits for every input size n : start from depth d circuit for a larger n' , and construct one for the smaller n .
 - By “restricting” to n inputs
 - And showing how to rewrite with depth $d-1$, staying poly sized

AND-OR trees

AND-OR trees

- Any function can be written as depth 2 AND-OR tree or an OR-AND tree

AND-OR trees

- Any function can be written as depth 2 AND-OR tree or an OR-AND tree
 - But exponential size

AND-OR trees

- Any function can be written as depth 2 AND-OR tree or an OR-AND tree
 - But exponential size
- Any circuit can be rewritten as an AND-OR tree (each leaf has a literal, possibly shared with other leaves)

AND-OR trees

- Any function can be written as depth 2 AND-OR tree or an OR-AND tree
 - But exponential size
- Any circuit can be rewritten as an AND-OR tree (each leaf has a literal, possibly shared with other leaves)
 - If polynomial size and constant depth (AC^0), stays so

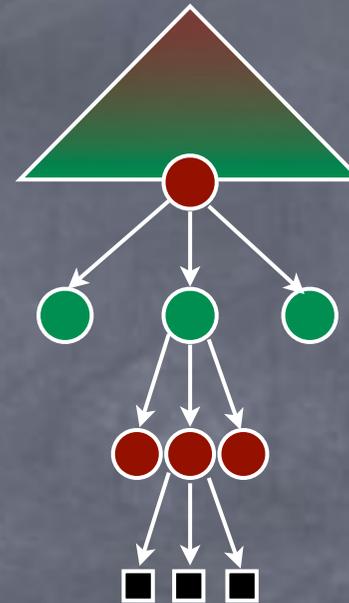
Switching

Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size

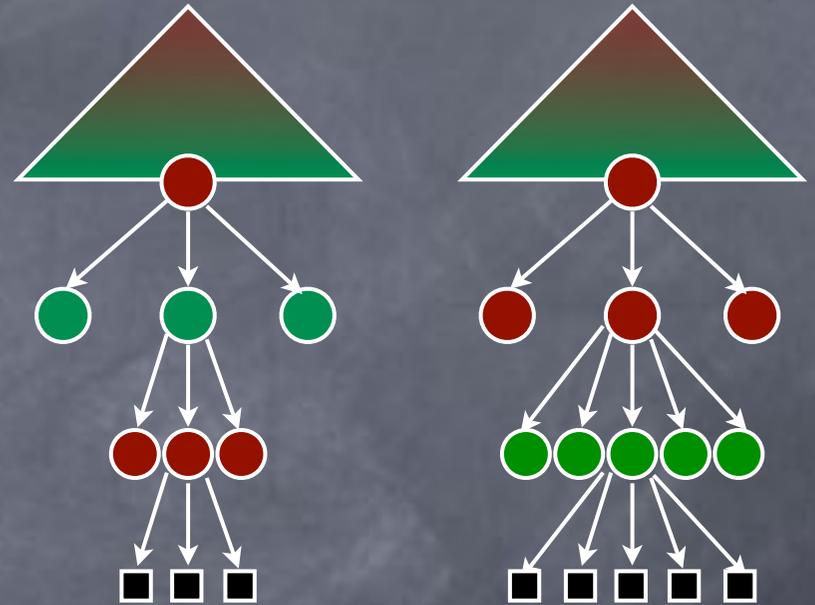
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size



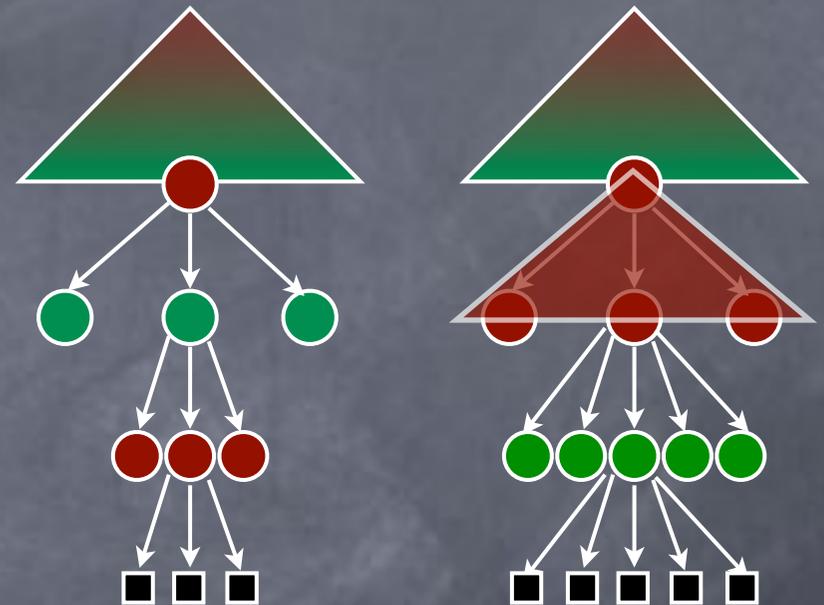
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size



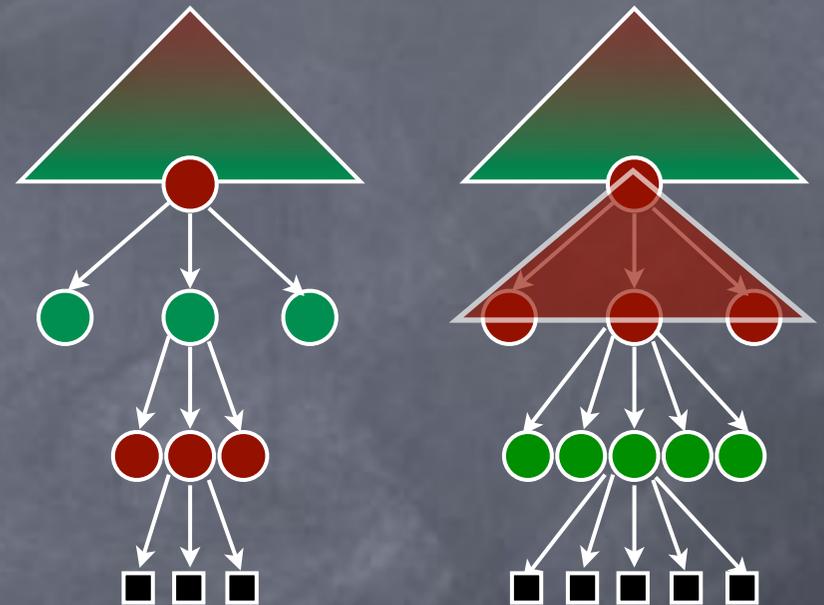
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size



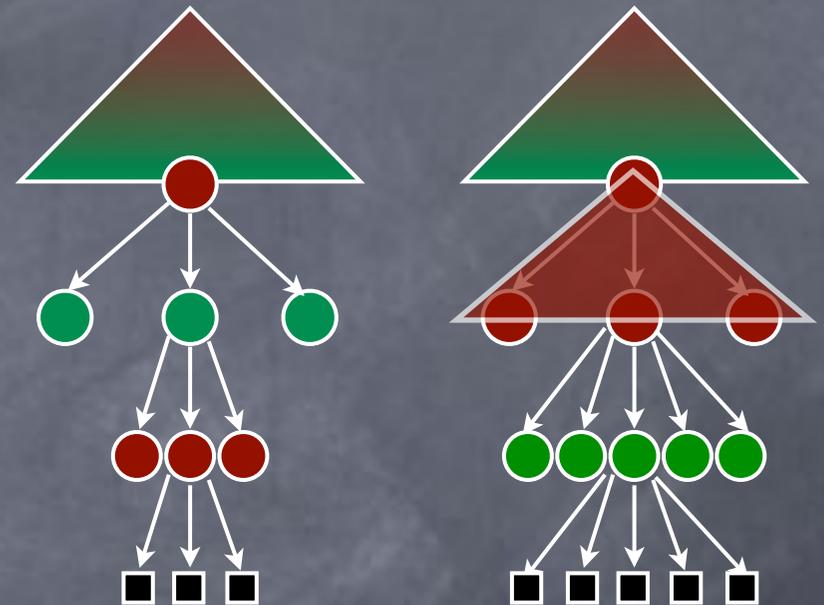
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size
- A depth d AC^0 circuit changes into depth $d-1$



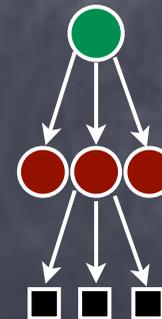
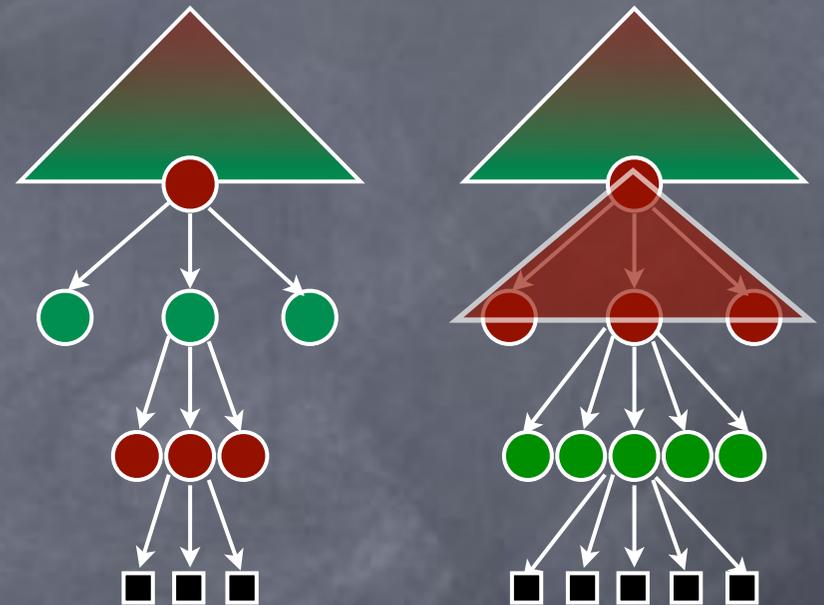
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size
 - A depth d AC^0 circuit changes into depth $d-1$
- When is switching possible?



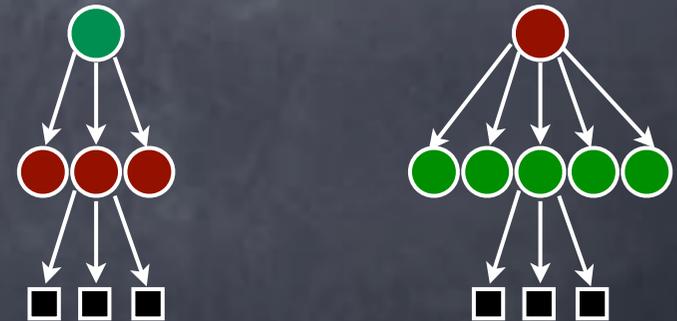
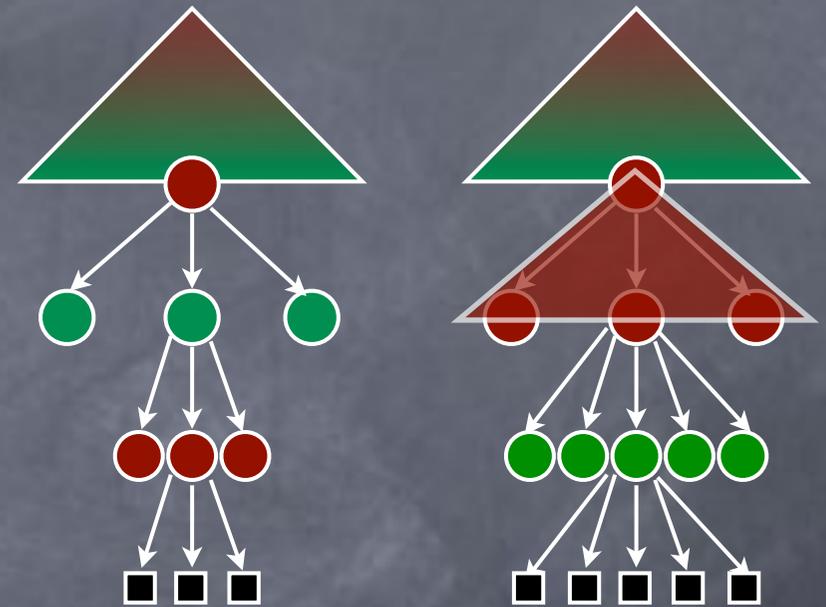
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size
 - A depth d AC^0 circuit changes into depth $d-1$
- When is switching possible?



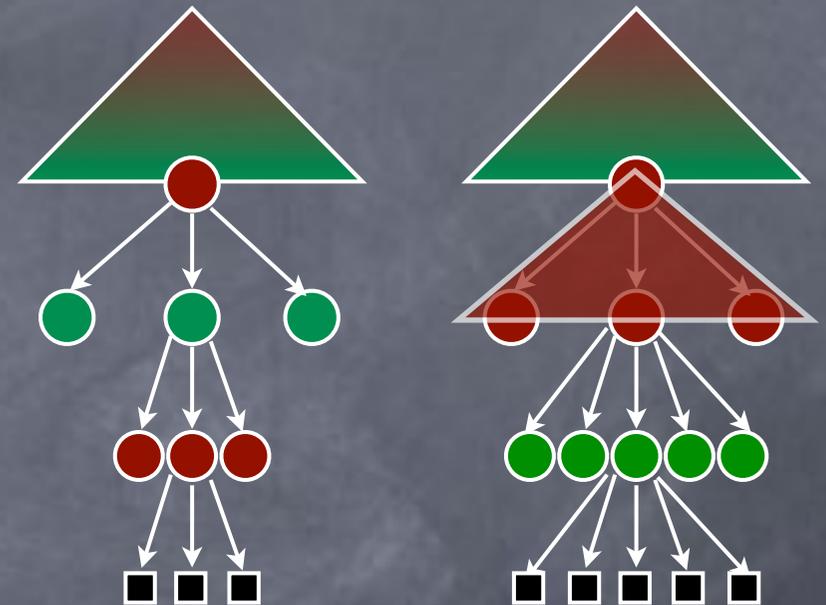
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size
 - A depth d AC^0 circuit changes into depth $d-1$
- When is switching possible?



Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size



- A depth d AC^0 circuit changes into depth $d-1$
- When is switching possible?

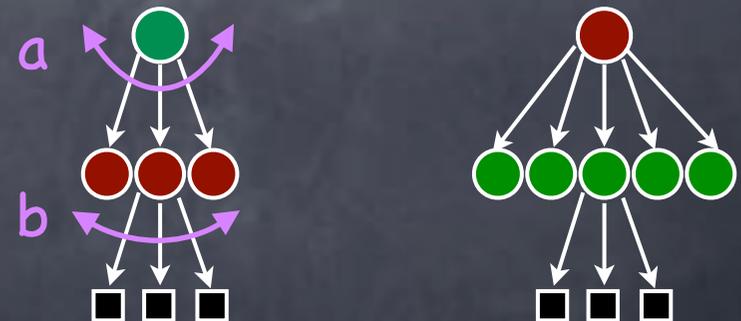
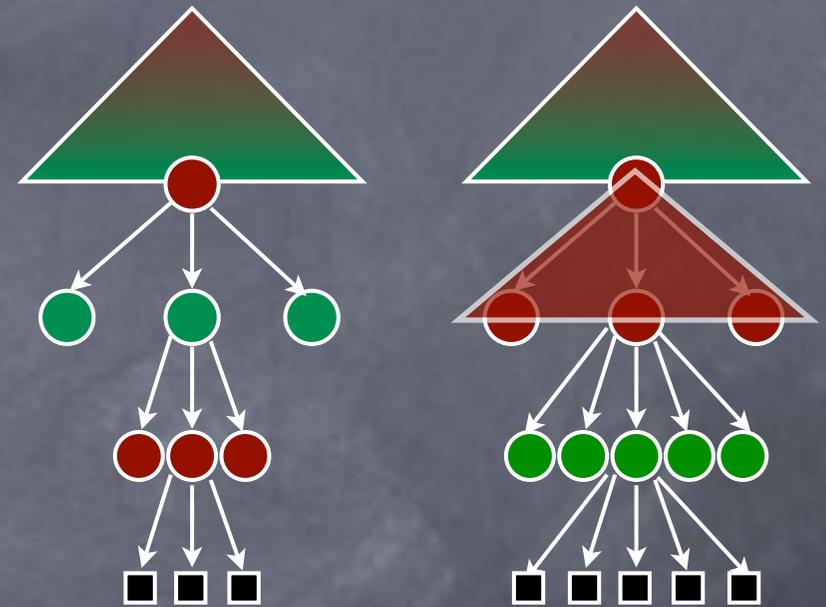
- Distributivity



Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size
 - A depth d AC^0 circuit changes into depth $d-1$
- When is switching possible?

- Distributivity



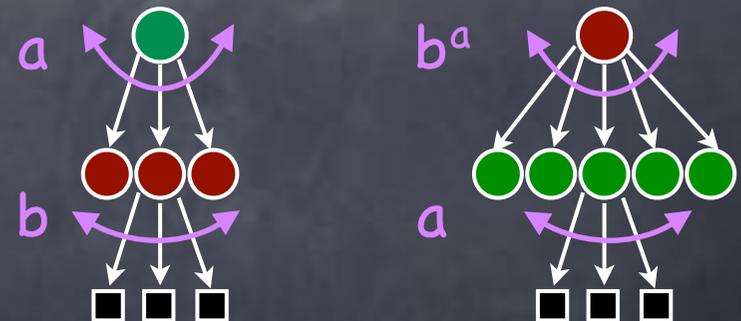
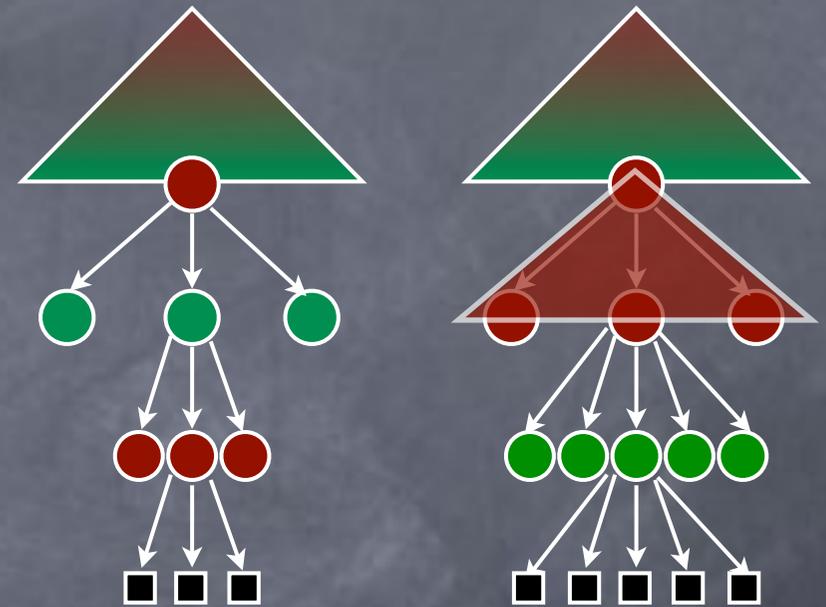
Switching

- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size

- A depth d AC^0 circuit changes into depth $d-1$

- When is switching possible?

- Distributivity



Switching

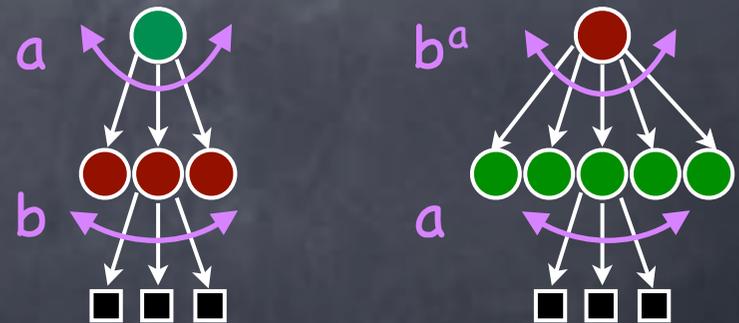
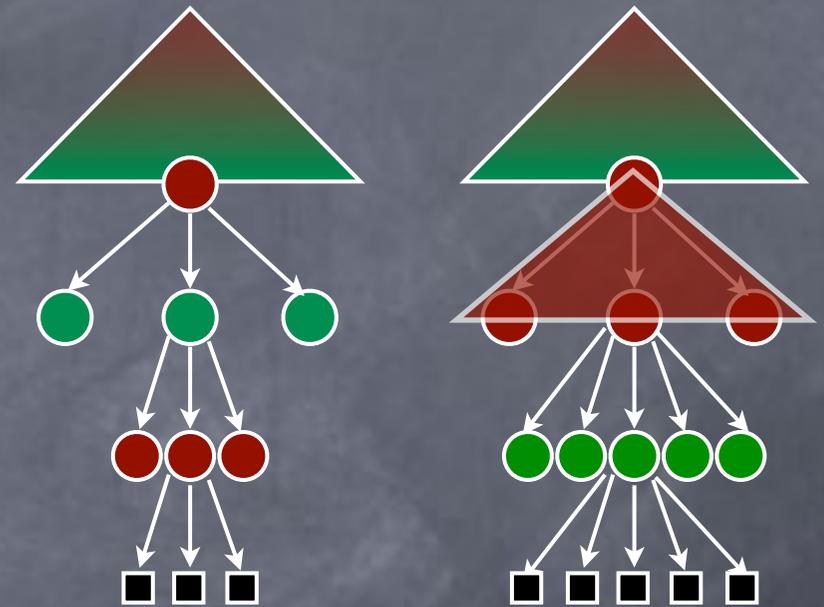
- In an AND-OR tree, if bottom two levels can be replaced by equivalent two levels with switched AND-OR order, and polynomial size

- A depth d AC^0 circuit changes into depth $d-1$

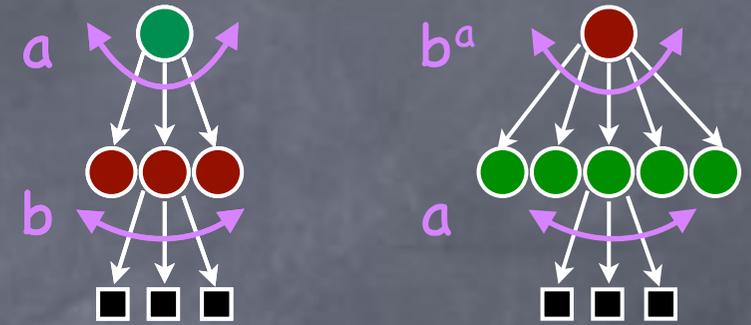
- When is switching possible?

- Distributivity

- But may increase size to exponential

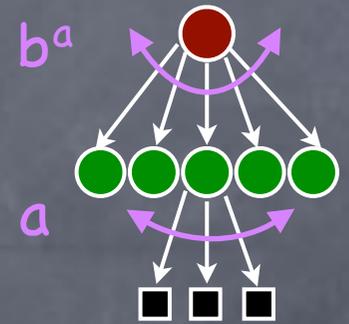
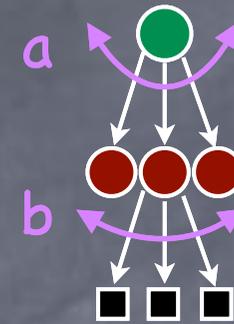


Switching Lemma



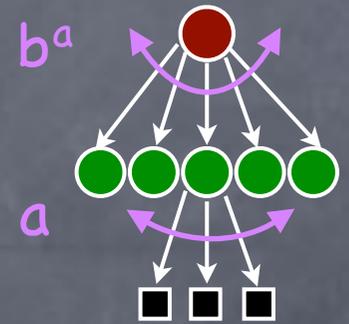
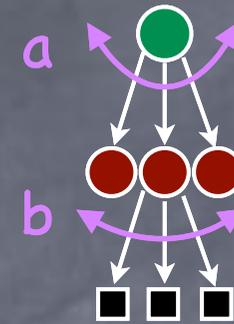
Switching Lemma

- A **modified** function has a switched circuit



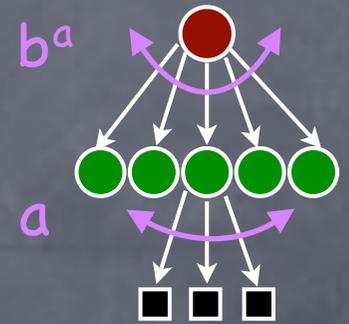
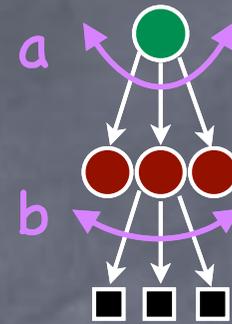
Switching Lemma

- A **modified** function has a switched circuit
- Size stays polynomial even after switching



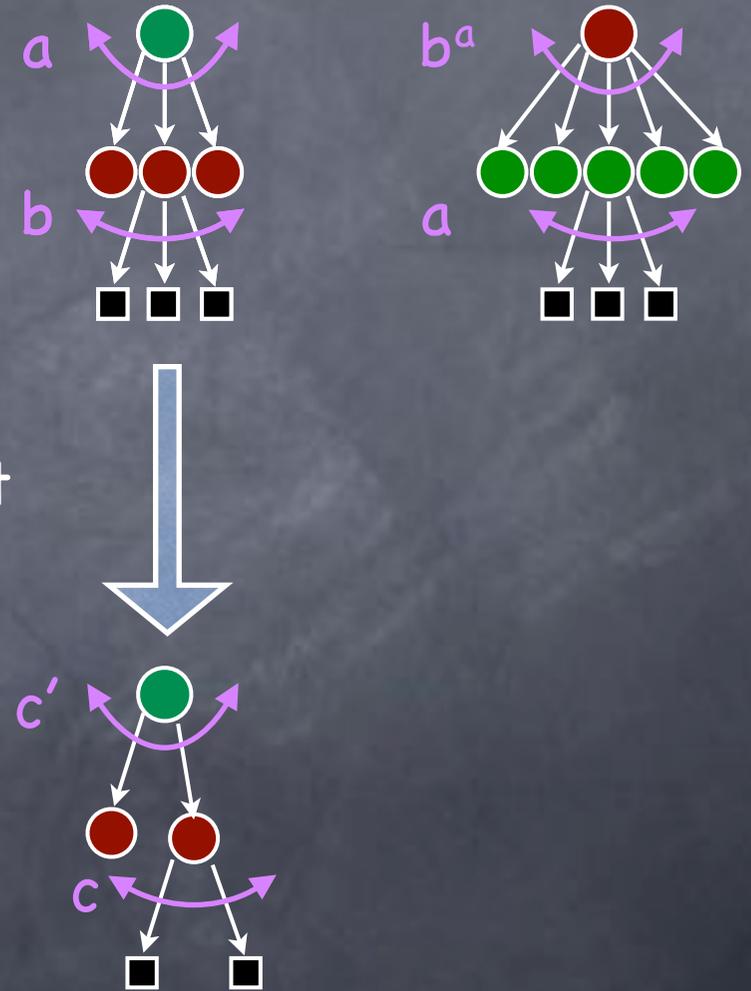
Switching Lemma

- A **modified** function has a switched circuit
- Size stays polynomial even after switching
- Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)



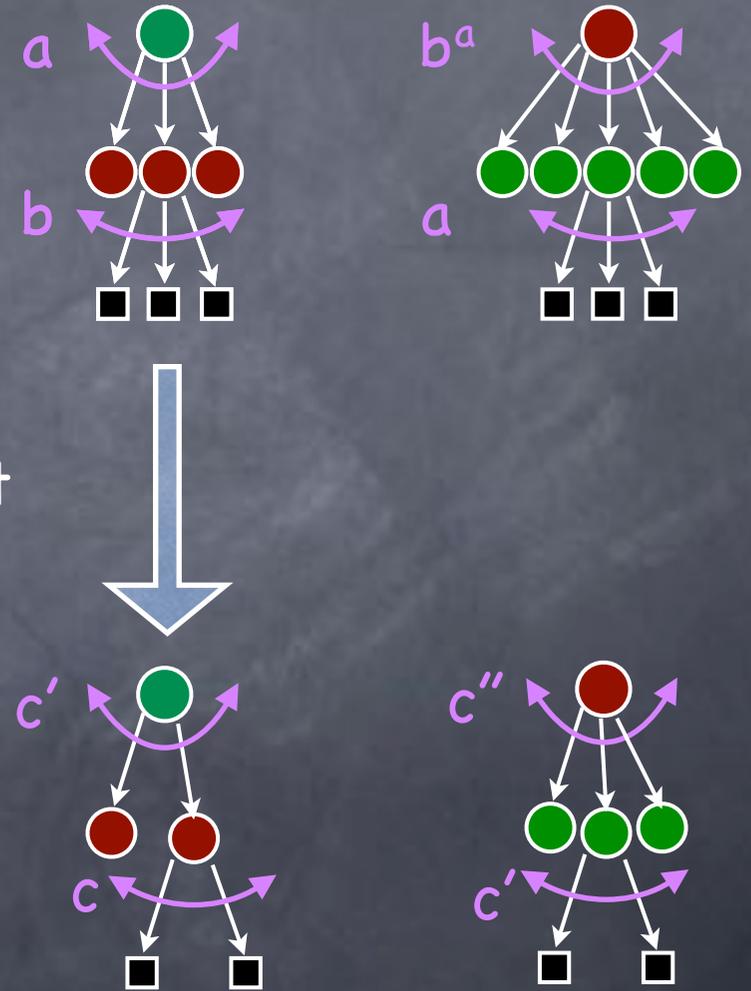
Switching Lemma

- A **modified** function has a switched circuit
- Size stays polynomial even after switching
- Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)



Switching Lemma

- A **modified** function has a switched circuit
- Size stays polynomial even after switching
- Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)



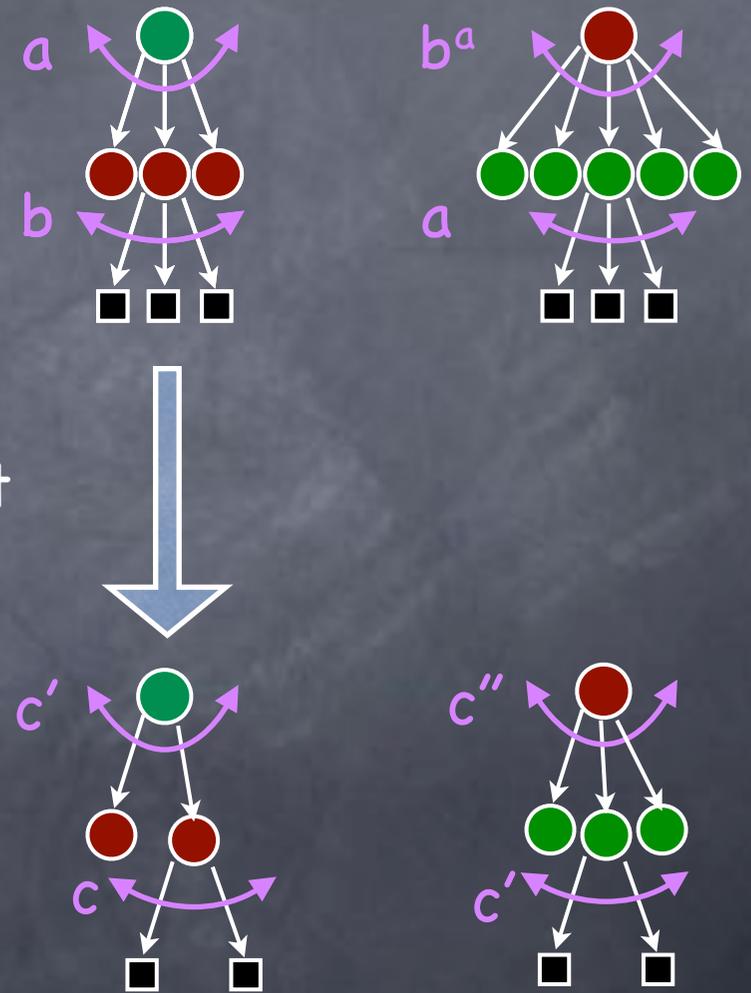
Switching Lemma

- A **modified** function has a switched circuit

- Size stays polynomial even after switching

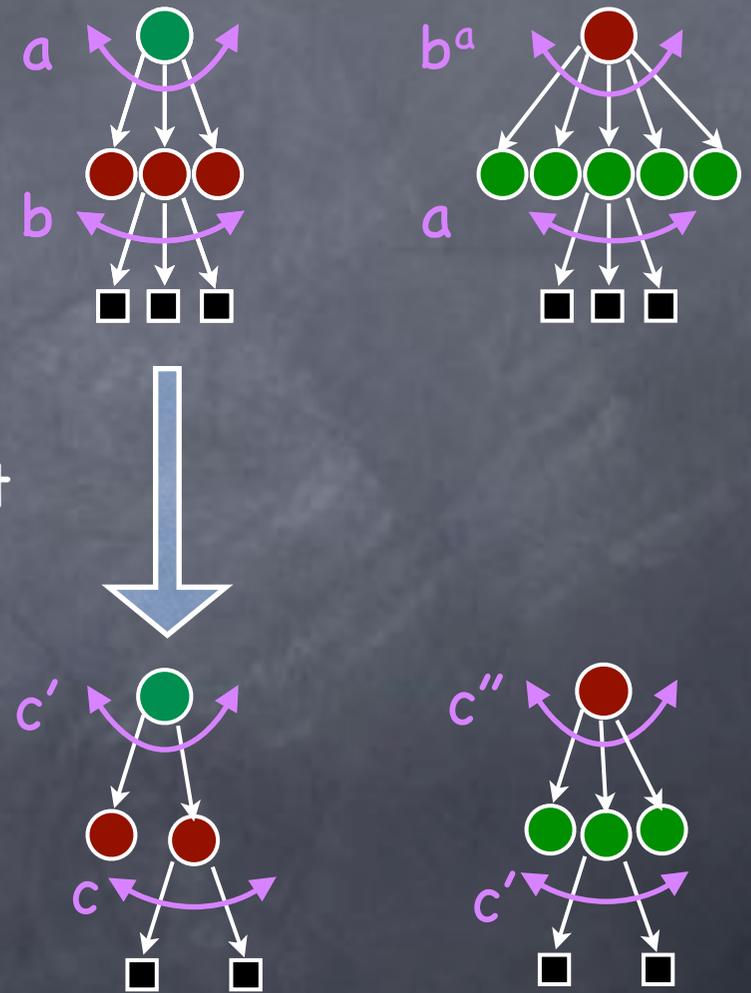
- Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)

- Still function of many variables



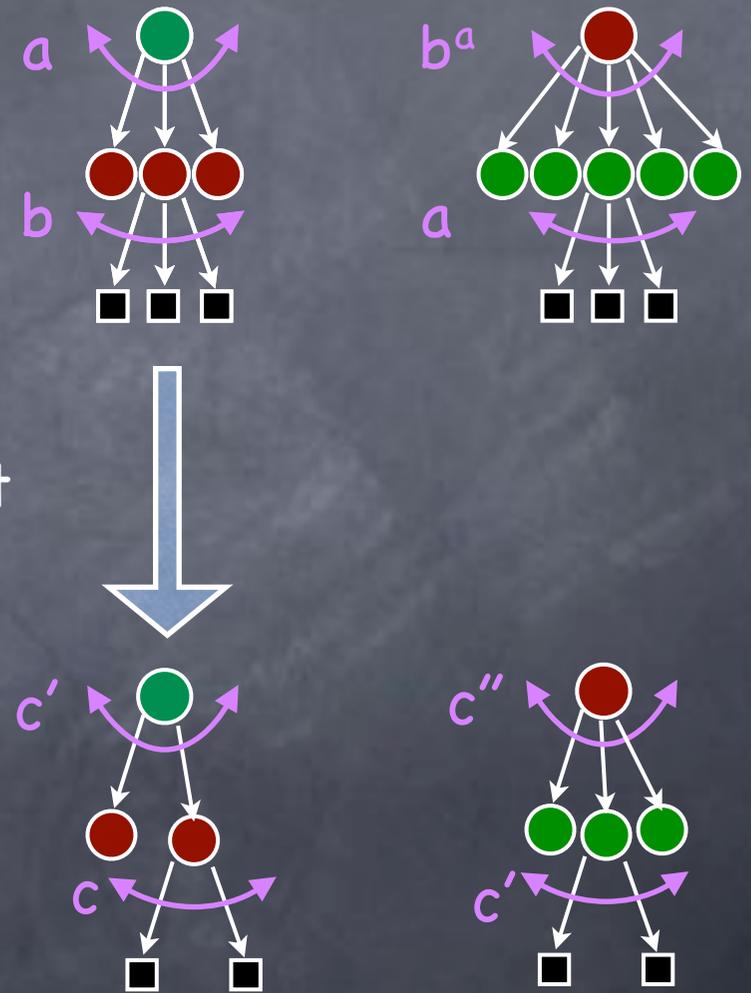
Switching Lemma

- A **modified** function has a switched circuit
- Size stays polynomial even after switching
- Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)
- Still function of many variables
- How to find such a modified function



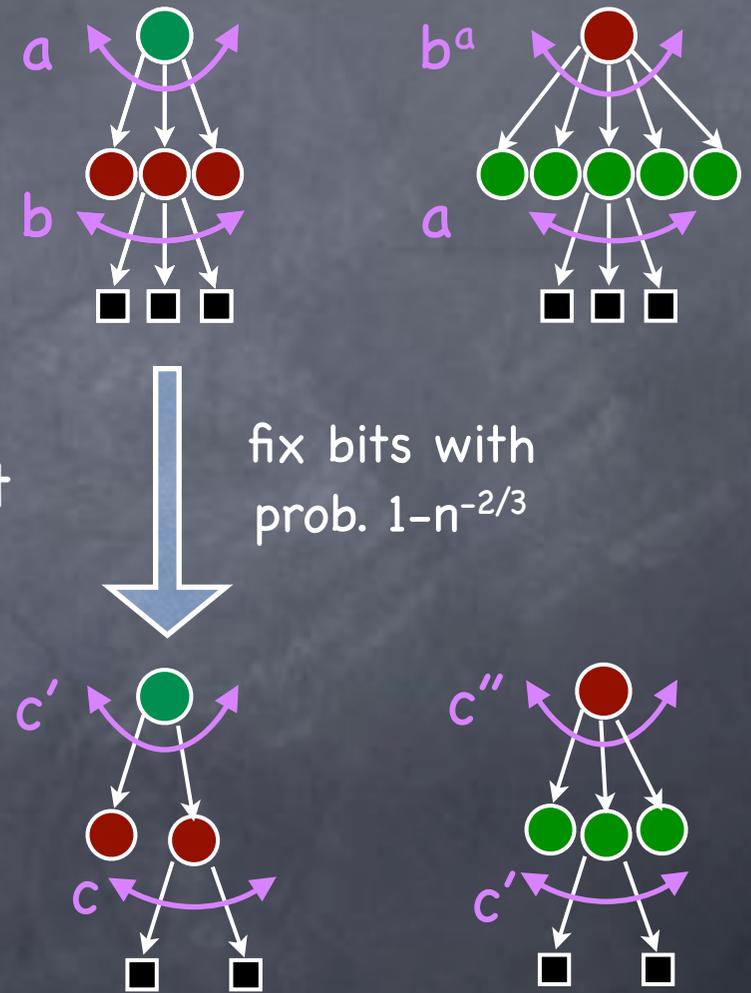
Switching Lemma

- A **modified** function has a switched circuit
 - Size stays polynomial even after switching
 - Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)
 - Still function of many variables
- How to find such a modified function
 - Random restriction



Switching Lemma

- A **modified** function has a switched circuit
 - Size stays polynomial even after switching
 - Computes same function as before but with most variables already set to specific values (a "**restriction**" of the original function)
 - Still function of many variables
- How to find such a modified function
 - Random restriction



Switching Lemma

Switching Lemma

- Random restriction. With positive probability:

Switching Lemma

- Random restriction. With positive probability:
 - can switch bottom levels, **staying poly sized**

Switching Lemma

- Random restriction. With positive probability:
 - can switch bottom levels, **staying poly sized**
 - with high probability for each switched node (this in fact, is the switching lemma); then union bound

Switching Lemma

- Random restriction. With positive probability:
 - can switch bottom levels, **staying poly sized**
 - with high probability for each switched node (this in fact, is the switching lemma); then union bound
 - computes PARITY for $n^{2/3}$ variables (w.h.p. by Chernoff)

Switching Lemma

- Random restriction. With positive probability:
 - can switch bottom levels, **staying poly sized**
 - with high probability for each switched node (this in fact, is the switching lemma); then union bound
 - computes PARITY for $n^{2/3}$ variables (w.h.p. by Chernoff)
- Depth $d-1$, poly-sized circuit family for PARITY

Switching Lemma

- Random restriction. With positive probability:
 - can switch bottom levels, **staying poly sized**
 - with high probability for each switched node (this in fact, is the switching lemma); then union bound
 - computes PARITY for $n^{2/3}$ variables (w.h.p. by Chernoff)
- Depth $d-1$, poly-sized circuit family for PARITY
 - Contradiction: started with minimal depth!

Razborov-Smolensky

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p
- Plan:

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p
- Plan:
 - Given a circuit C , can find a polynomial s.t.

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p
- Plan:
 - Given a circuit C , can find a polynomial s.t.
 - Polynomial has “low degree”

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p
- Plan:
 - Given a circuit C , can find a polynomial s.t.
 - Polynomial has “low degree”
 - Polynomial agrees with C on most inputs

Razborov–Smolensky

- An alternate proof that $\text{PARITY} \notin \text{AC}^0$
 - Generalizes to $\text{ACC}^0(p)$ for odd primes p
- Plan:
 - Given a circuit C , can find a polynomial s.t.
 - Polynomial has “low degree”
 - Polynomial agrees with C on most inputs
 - Show that no low degree polynomial can agree with PARITY on that many inputs

Polynomial from Circuit

Polynomial from Circuit

- Assume circuit has OR, NOT gates

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial
 - If we do this faithfully, degree will be large

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial
 - If we do this faithfully, degree will be large
 - Large enough to evaluate PARITY

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial
 - If we do this faithfully, degree will be large
 - Large enough to evaluate PARITY
 - So allow polynomials which err on some inputs

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial
 - If we do this faithfully, degree will be large
 - Large enough to evaluate PARITY
 - So allow polynomials which err on some inputs
 - At each gate will pick polynomial from a distribution

Polynomial from Circuit

- Assume circuit has OR, NOT gates
 - Replace gates by polynomials (over some field), and compose together into one big polynomial
 - If we do this faithfully, degree will be large
 - Large enough to evaluate PARITY
 - So allow polynomials which err on some inputs
 - At each gate will pick polynomial from a distribution
 - Composed polynomial will be good with prob. > 0

Polynomials for OR, NOT and PARITY

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!
 - We will work over $GF(q)$, $q > 2$

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!
 - We will work over $GF(q)$, $q > 2$
 - $PARITY = [1 - (1 - 2x_1)(1 - 2x_2) \dots (1 - 2x_n)] / 2$ (if $2 \neq 0$)

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!
 - We will work over $GF(q)$, $q > 2$
 - $PARITY = [1 - (1 - 2x_1)(1 - 2x_2) \dots (1 - 2x_n)] / 2$ (if $2 \neq 0$)
 - $NOT = 1 - x$.

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!
 - We will work over $GF(q)$, $q > 2$
 - $PARITY = [1 - (1 - 2x_1)(1 - 2x_2) \dots (1 - 2x_n)] / 2$ (if $2 \neq 0$)
 - $NOT = 1 - x$.
 - $OR = 1 - (1 - x_1) \dots (1 - x_n)$

Polynomials for OR, NOT and PARITY

- Want that PARITY is complex (high degree) while OR, NOT are simple (low degree)
 - If over $GF(2)$, PARITY is just sum (degree 1)!
 - We will work over $GF(q)$, $q > 2$
 - $PARITY = [1 - (1 - 2x_1)(1 - 2x_2) \dots (1 - 2x_n)] / 2$ (if $2 \neq 0$)
 - $NOT = 1 - x$.
 - $OR = 1 - (1 - x_1) \dots (1 - x_n)$
 - But high degree! Need OR to be simple!

Approximate Polynomials

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $OR(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $OR(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$
- If $OR(x_1, \dots, x_n) = 1$

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $\text{OR}(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$
- If $\text{OR}(x_1, \dots, x_n) = 1$
 - $\Pr_a[a_1x_1 + \dots + a_nx_n = 0] \leq 1/q$ (in fact " $=$ " " $=$ ") (why?)

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $\text{OR}(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$
- If $\text{OR}(x_1, \dots, x_n) = 1$
 - $\Pr_a[a_1x_1 + \dots + a_nx_n = 0] \leq 1/q$ (in fact " $=$ " " $=$ ") (why?)
 - Recall in $\text{GF}(q)$, $u^{q-1} = 1$ unless $u=0$ (since non-0 elements form a group of order $q-1$ under multiplication)

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $OR(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$
- If $OR(x_1, \dots, x_n) = 1$
 - $\Pr_a[a_1x_1 + \dots + a_nx_n = 0] \leq 1/q$ (in fact " $=$ " " $=$ ") (why?)
 - Recall in $GF(q)$, $u^{q-1} = 1$ unless $u=0$ (since non-0 elements form a group of order $q-1$ under multiplication)
 - i.e. $\Pr_a[(a_1x_1 + \dots + a_nx_n)^{q-1} = 1] \geq 1-1/q$

Approximate Polynomials

- Consider (random) polynomial $p(x_1, \dots, x_n) = (a_1x_1 + \dots + a_nx_n)^{q-1}$ where a_i are picked at random from the field
- If $\text{OR}(x_1, \dots, x_n) = 0$ then $p(x_1, \dots, x_n) = 0$
- If $\text{OR}(x_1, \dots, x_n) = 1$
 - $\Pr_a[a_1x_1 + \dots + a_nx_n = 0] \leq 1/q$ (in fact " $=$ " " $=$ ") (why?)
 - Recall in $\text{GF}(q)$, $u^{q-1} = 1$ unless $u=0$ (since non-0 elements form a group of order $q-1$ under multiplication)
 - i.e. $\Pr_a[(a_1x_1 + \dots + a_nx_n)^{q-1} = 1] \geq 1-1/q$
- Can boost to $1-1/q^t$ by doing (exact) OR t times: $\text{deg} < qt$

Approximate Polynomials

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial
 - Substitute child polynomials into the gate-poly's variables

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial
 - Substitute child polynomials into the gate-poly's variables
 - Degree multiplies: depth d circuit gives deg **$O(\log 1/\epsilon)^d$**

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial
 - Substitute child polynomials into the gate-poly's variables
 - Degree multiplies: depth d circuit gives deg **$O(\log 1/\epsilon)^d$**
 - Error adds (by union bound): size s circuit gives **error $< s\epsilon$**

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial
 - Substitute child polynomials into the gate-poly's variables
 - Degree multiplies: depth d circuit gives deg **$O(\log 1/\epsilon)^d$**
 - Error adds (by union bound): size s circuit gives **error $< s\epsilon$**
- Using $\epsilon=1/(4s)$, **degree $O(\log s)^d$ polynomial**, correct w.p. $> 3/4$

Approximate Polynomials

- OR: a random polynomial of **degree $O(\log 1/\epsilon)$** , such that it is correct with **prob. $> 1-\epsilon$**
- Composing gate-polynomials into circuit-polynomial
 - Substitute child polynomials into the gate-poly's variables
 - Degree multiplies: depth d circuit gives deg **$O(\log 1/\epsilon)^d$**
 - Error adds (by union bound): size s circuit gives **error $< s\epsilon$**
- Using $\epsilon=1/(4s)$, **degree $O(\log s)^d$ polynomial**, correct w.p. $> 3/4$
 - One polynomial, **correct on $> 3/4$ fraction of inputs (why?)**

How about PARITY?

How about PARITY?

- Can PARITY also be **approximated** (i.e., calculated correctly for some large input set S) by a low-degree polynomial?

How about PARITY?

- Can PARITY also be **approximated** (i.e., calculated correctly for some large input set S) by a low-degree polynomial?
 - PARITY is essentially $\prod_{i=1 \text{ to } n} x_i$, for inputs from $\{+1, -1\}^n$

How about PARITY?

- Can PARITY also be **approximated** (i.e., calculated correctly for some large input set S) by a low-degree polynomial?
 - PARITY is essentially $\prod_{i=1}^n x_i$, for inputs from $\{+1,-1\}^n$
 - If can calculate $\prod_{i=1}^n x_i$ (for $S \subseteq \{+1,-1\}^n$) using degree D , then can calculate (for S) any polynomial using degree $D+n/2$ polynomial (why?)

How about PARITY?

- Can PARITY also be **approximated** (i.e., calculated correctly for some large input set S) by a low-degree polynomial?
 - PARITY is essentially $\prod_{i=1}^n x_i$, for inputs from $\{+1, -1\}^n$
 - If can calculate $\prod_{i=1}^n x_i$ (for $S \subseteq \{+1, -1\}^n$) using degree D , then can calculate (for S) any polynomial using degree $D+n/2$ polynomial (why?)
 - But if S large (say $3/4 \cdot 2^n$), then too many polynomials that are distinct for S

How about PARITY?

- Can PARITY also be **approximated** (i.e., calculated correctly for some large input set S) by a low-degree polynomial?
 - PARITY is essentially $\prod_{i=1}^n x_i$, for inputs from $\{+1, -1\}^n$
 - If can calculate $\prod_{i=1}^n x_i$ (for $S \subseteq \{+1, -1\}^n$) using degree D , then can calculate (for S) any polynomial using degree $D+n/2$ polynomial (why?)
 - But if S large (say $3/4 \cdot 2^n$), then too many polynomials that are distinct for S
 - Need $D = \Omega(\sqrt{n})$ to have enough degree $D+n/2$ polys.

PARITY \notin ACC(q)⁰

PARITY \notin ACC(q)⁰

- Given depth d , size s circuit C , there is a polynomial of degree $O(\log(s))^d$ which agrees with C on $3/4$ of inputs

PARITY \notin ACC(q)⁰

- Given depth d , size s circuit C , there is a polynomial of degree $O(\log(s))^d$ which agrees with C on $3/4$ of inputs
 - Using approximate OR polynomials

PARITY \notin ACC(q)⁰

- Given depth d , size s circuit C , there is a polynomial of degree $O(\log(s))^d$ which agrees with C on $3/4$ of inputs
 - Using approximate OR polynomials
 - Even if circuit has Mod_q (boolean) gates: $(x_1 + \dots + x_n)^{q-1}$

PARITY \notin ACC(q)⁰

- Given depth d , size s circuit C , there is a polynomial of degree $O(\log(s))^d$ which agrees with C on $3/4$ of inputs
 - Using approximate OR polynomials
 - Even if circuit has Mod_q (boolean) gates: $(x_1 + \dots + x_n)^{q-1}$
- PARITY needs degree $\Omega(\sqrt{n})$ polynomial for approximation

PARITY \notin ACC(q)⁰

- Given depth d , size s circuit C , there is a polynomial of degree $O(\log(s))^d$ which agrees with C on $3/4$ of inputs
 - Using approximate OR polynomials
 - Even if circuit has Mod_q (boolean) gates: $(x_1 + \dots + x_n)^{q-1}$
- PARITY needs degree $\Omega(\sqrt{n})$ polynomial for approximation
- $\log(s) = \Omega(\sqrt{n})^{1/d}$ or $s = 2^{\Omega(n)^{1/2d}}$: i.e., if depth is constant then size not poly (in fact exponential)

Monotone Circuits

Monotone Circuits

- Another restricted class for which strong lower-bounds are known

Monotone Circuits

- Another restricted class for which strong lower-bounds are known
 - Monotone circuits: no NOT gate (and no neg. literal)

Monotone Circuits

- Another restricted class for which strong lower-bounds are known
 - Monotone circuits: no NOT gate (and no neg. literal)
 - For monotonic functions f

Monotone Circuits

- Another restricted class for which strong lower-bounds are known
 - Monotone circuits: no NOT gate (and no neg. literal)
 - For monotonic functions f
 - To show that f has no poly-sized monotone circuit family

Monotone Circuits

- Another restricted class for which strong lower-bounds are known
 - Monotone circuits: no NOT gate (and no neg. literal)
 - For monotonic functions f
 - To show that f has no poly-sized monotone circuit family
 - Still possible that f may have a more efficient non-monotone circuit family (or even be in P)

CLIQUE

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits
 - A way to turn a circuit into an approximately correct circuit, gate by gate (AND/OR gate \rightarrow "approximation gate")

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits
 - A way to turn a circuit into an approximately correct circuit, gate by gate (AND/OR gate \rightarrow "approximation gate")
 - Will consider effect of this change on some Yes examples and some No examples

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits
 - A way to turn a circuit into an approximately correct circuit, gate by gate (AND/OR gate \rightarrow "approximation gate")
 - Will consider effect of this change on some Yes examples and some No examples
 - Converting each gate to approximation makes only a few extra examples go wrong

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits
 - A way to turn a circuit into an approximately correct circuit, gate by gate (AND/OR gate \rightarrow "approximation gate")
 - Will consider effect of this change on some Yes examples and some No examples
 - Converting each gate to approximation makes only a few extra examples go wrong
 - A circuit with only approximation gates errs on a large number of the examples

CLIQUE

- $\text{CLIQUE}_{n,k}$ does not have poly-sized monotone circuits
 - A way to turn a circuit into an approximately correct circuit, gate by gate (AND/OR gate \rightarrow "approximation gate")
 - Will consider effect of this change on some Yes examples and some No examples
 - Converting each gate to approximation makes only a few extra examples go wrong
 - A circuit with only approximation gates errs on a large number of the examples
 - Original circuit must have been large

Proof Sketch

Proof Sketch

- Input sets

Proof Sketch

- Input sets
 - Yes set: graphs with no edges except a single k -clique.
 - No set: complete $(k-1)$ -partite graphs

Proof Sketch

- Input sets
 - Yes set: graphs with no edges except a single k -clique.
No set: complete $(k-1)$ -partite graphs
- Since monotone circuit, we can label each gate with a set of subgraphs which will make the gate's output 1

Proof Sketch

- Input sets
 - Yes set: graphs with no edges except a single k -clique.
No set: complete $(k-1)$ -partite graphs
- Since monotone circuit, we can label each gate with a set of subgraphs which will make the gate's output 1
 - Input gates: edges

Proof Sketch

- Input sets
 - Yes set: graphs with no edges except a single k -clique.
No set: complete $(k-1)$ -partite graphs
- Since monotone circuit, we can label each gate with a set of subgraphs which will make the gate's output 1
 - Input gates: edges
 - OR: take union of the two sets of input-subsets

Proof Sketch

- Input sets
 - Yes set: graphs with no edges except a single k -clique.
 - No set: complete $(k-1)$ -partite graphs
- Since monotone circuit, we can label each gate with a set of subgraphs which will make the gate's output 1
 - Input gates: edges
 - OR: take union of the two sets of input-subsets
 - AND: take set of pair-wise unions of input-subsets

Proof Sketch

Proof Sketch

- Approximation gate: output wire labeled by a sample of M cliques of at most t vertices. Value 1 if at least one of those M cliques is present in the graph

Proof Sketch

- Approximation gate: output wire labeled by a sample of M cliques of at most t vertices. Value 1 if at least one of those M cliques is present in the graph
 - Input gates: edges

Proof Sketch

- Approximation gate: output wire labeled by a sample of M cliques of at most t vertices. Value 1 if at least one of those M cliques is present in the graph
 - Input gates: edges
 - OR: take union of the two sets of subsets, and “prune” to M subsets

Proof Sketch

- Approximation gate: output wire labeled by a sample of M cliques of at most t vertices. Value 1 if at least one of those M cliques is present in the graph
 - Input gates: edges
 - OR: take union of the two sets of subsets, and “prune” to M subsets
 - AND: take set of pair-wise unions which are at most t vertices, and “prune” to M subsets

Proof Sketch

- Approximation gate: output wire labeled by a sample of M cliques of at most t vertices. Value 1 if at least one of those M cliques is present in the graph
 - Input gates: edges
 - OR: take union of the two sets of subsets, and “prune” to M subsets
 - AND: take set of pair-wise unions which are at most t vertices, and “prune” to M subsets
 - Pruning uses “sunflower lemma”: find a sunflower and replace petals by core

Proof Sketch

Proof Sketch

- Converting each gate to approximation makes only a few more examples go wrong

Proof Sketch

- Converting each gate to approximation makes only a few more examples go wrong
 - Bounding new **false positives among No sets** and **false negatives among Yes sets** introduced by pruning

Proof Sketch

- Converting each gate to approximation makes only a few more examples go wrong
 - Bounding new **false positives among No sets** and **false negatives among Yes sets** introduced by pruning
- A circuit with only approximation gates errs on a large number of the examples

Proof Sketch

- Converting each gate to approximation makes only a few more examples go wrong
 - Bounding new **false positives among No sets** and **false negatives among Yes sets** introduced by pruning
- A circuit with only approximation gates errs on a large number of the examples
 - If output identically “No” then errs on entire Yes set. Else, output wire’s label has some subset X , $|X| \leq t = O(\sqrt{k})$, and then a constant fraction of No-examples get accepted