Circuits

Lecture 11
Uniform Circuit Complexity
Recall
Recall

- Non-uniform complexity
Recall

- Non-uniform complexity
- $P/1 \not\subseteq \text{Decidable}$
Recall

- Non-uniform complexity
  - $P/1 \not\subseteq \text{Decidable}$
  - $NP \subseteq P/\log \Rightarrow NP = P$
Recall

- Non-uniform complexity
  - $P/1 \not\subseteq \text{Decidable}$
  - $NP \subseteq P/\log \Rightarrow NP = P$
  - $NP \subseteq P/poly \Rightarrow \text{PH} = \Sigma_2^P$
Recall

- Non-uniform complexity
  - $\text{P/1} \not\subseteq \text{Decidable}$
  - $\text{NP} \subseteq \text{P/log} \Rightarrow \text{NP} = \text{P}$
  - $\text{NP} \subseteq \text{P/poly} \Rightarrow \text{PH} = \Sigma_2^p$
- Circuit Complexity
Recall

- Non-uniform complexity
  - $P/1 \not\subseteq$ Decidable
  - $NP \subseteq P/\log \Rightarrow NP = P$
  - $NP \subseteq P/poly \Rightarrow PH = \Sigma_2^P$

- Circuit Complexity
  - $\text{SIZE}(\text{poly}) = P/poly$
Recall

Non-uniform complexity

- $P/1 \not\subseteq \text{Decidable}$
- $NP \subseteq P/\log \implies NP = P$
- $NP \subseteq P/poly \implies PH = \Sigma_2^P$

Circuit Complexity

- $\text{SIZE}(\text{poly}) = P/poly$
- $\text{SIZE}$-hierarchy
Recall

- Non-uniform complexity
  - $P/1 \not\subseteq$ Decidable
  - $NP \subseteq P/\log \Rightarrow NP = P$
  - $NP \subseteq P/poly \Rightarrow PH = \Sigma_2^p$

- Circuit Complexity
  - $\text{SIZE}(\text{poly}) = P/poly$
  - $\text{SIZE}$-hierarchy
    - $\text{SIZE}(T') \subset \text{SIZE}(T)$ if $T = \Omega(t2^t)$ and $T' = O(2^t/t)$
Recall

- Non-uniform complexity
  - P/1 ⊈ Decidable
  - NP ⊆ P/log  ⇒ NP = P
  - NP ⊆ P/poly  ⇒ PH = Σ₂^P

Circuit Complexity

- SIZE(poly) = P/poly

SIZE-hierarchy

- SIZE(T') ⊆ SIZE(T) if T=Ω(t2^t) and T'=O(2^t/t)
- Most functions on t bits (that ignore last n-t bits) are in SIZE(T) but not in SIZE(T')
Uniform Circuits
Uniform Circuits

- Uniform circuit family: constructed by a TM
Uniform Circuits

- Uniform circuit family: constructed by a TM
- Undecidable languages are undecidable for these circuits families
Uniform Circuits

- Uniform circuit family: constructed by a TM
- Undecidable languages are undecidable for these circuits families
- Can relate their complexity classes to classes defined using TMs
Uniform Circuits

- Uniform circuit family: constructed by a TM
  - Undecidable languages are undecidable for these circuits families
  - Can relate their complexity classes to classes defined using TMs
- Logspace-uniform:
Uniform Circuits

- Uniform circuit family: constructed by a TM
  - Undecidable languages are undecidable for these circuits families
  - Can relate their complexity classes to classes defined using TMs
- Logspace-uniform:
  - An $O(\log n)$ space TM can compute the circuit
NC\textsuperscript{i} and AC\textsuperscript{i}
NC$^i$ and AC$^i$

- NC$^i$: class of languages decided by bounded fan-in logspace-uniform circuits of polynomial size and depth $O(\log^i n)$
NC<sub>i</sub> and AC<sub>i</sub>

- **NC<sub>i</sub>:** class of languages decided by bounded fan-in logspace-uniform circuits of **polynomial size and depth** \( O(\log^i n) \)

- **AC<sub>i</sub>:** Similar, but unbounded fan-in circuits
$\text{NC}^i$ and $\text{AC}^i$

- $\text{NC}^i$: class of languages decided by bounded fan-in logspace-uniform circuits of polynomial size and depth $O(\log^i n)$

- $\text{AC}^i$: Similar, but unbounded fan-in circuits

$\text{NC}^0$ and $\text{AC}^0$: constant depth circuits
**NC\(^i\) and AC\(^i\)**

- **NC\(^i\):** class of languages decided by bounded fan-in logspace-uniform circuits of *polynomial size and depth* \(O(\log^i n)\)

- **AC\(^i\):** Similar, but unbounded fan-in circuits

- **NC\(^0\) and AC\(^0\):** constant depth circuits

- **NC\(^0\) output** depends on only a constant number of input bits
NC\textsuperscript{i} and AC\textsuperscript{i}

- **NC\textsuperscript{i}**: class of languages decided by bounded fan-in logspace-uniform circuits of polynomial size and depth $O(\log^i n)$
  - **AC\textsuperscript{i}**: Similar, but unbounded fan-in circuits

- **NC\textsuperscript{0} and AC\textsuperscript{0}**: constant depth circuits
  - **NC\textsuperscript{0}** output depends on only a constant number of input bits
  - **NC\textsuperscript{0} \subsetneq AC\textsuperscript{0}**: Consider $L = \{1, 11, 111, \ldots\}$
NC and AC
NC and AC

\[ \text{NC} = \bigcup_{i>0} \text{NC}_i. \text{ Similarly AC.} \]
NC and AC

\[ \text{NC} = \bigcup_{i>0} \text{NC}^i. \text{ Similarly AC.} \]

\[ \text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1} \]
NC and AC

\[ \text{NC} = \bigcup_{i \geq 0} \text{NC}^i \]. Similarly AC.

\[ \text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1} \]

Clearly \( \text{NC}^i \subseteq \text{AC}^i \)
NC and AC

- NC = $\bigcup_{i>0} \text{NC}_i$. Similarly AC.

- $\text{NC}_i \subseteq \text{AC}_i \subseteq \text{NC}_{i+1}$

- Clearly $\text{NC}_i \subseteq \text{AC}_i$

- $\text{AC}_i \subseteq \text{NC}_{i+1}$ because polynomial fan-in can be reduced to constant fan-in by using a log depth tree
NC and AC

$\text{NC} = \bigcup_{i \geq 0} \text{NC}^i$. Similarly $\text{AC}$.

$\text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1}$

Clearly $\text{NC}^i \subseteq \text{AC}^i$

$\text{AC}^i \subseteq \text{NC}^{i+1}$ because polynomial fan-in can be reduced to constant fan-in by using a log depth tree

So $\text{NC} = \text{AC}$
NC \subseteq P
\( \text{NC} \subseteq \text{P} \)

- Generate circuit of the right input size and evaluate on input
\[ \text{NC} \subseteq \text{P} \]

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
\( \text{NC} \subseteq \text{P} \)

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
  - in logspace, so poly time; also circuit size is poly
$\textbf{NC} \subseteq \textbf{P}$

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
  - in logspace, so poly time; also circuit size is poly
- Evaluating the gates
$$\text{NC} \subseteq \text{P}$$

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
  - in logspace, so poly time; also circuit size is poly
- Evaluating the gates
  - Poly(n) gates
\( \text{NC} \subseteq \text{P} \)

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
  - in logspace, so poly time; also circuit size is poly
- Evaluating the gates
  - Poly(n) gates
  - Per gate takes \( O(1) \) time + time to look up output values of (already evaluated) gates
\( \text{NC} \subseteq \text{P} \)

- Generate circuit of the right input size and evaluate on input
- Generating the circuit
  - in logspace, so poly time; also circuit size is poly
- Evaluating the gates
  - Poly(n) gates
  - Per gate takes \( O(1) \) time + time to look up output values of (already evaluated) gates
- Open problem: Is \( \text{NC} = \text{P} \)?
Motivation for NC
Motivation for NC

- Fast parallel computation is (loosely) modeled as having poly many processors and taking poly-log time.
Motivation for NC

- Fast parallel computation is (loosely) modeled as having poly many processors and taking poly-log time.

- Corresponds to NC (How?)
Motivation for NC

- Fast parallel computation is (loosely) modeled as having poly many processors and taking poly-log time

- Corresponds to NC (How?)

- Depth translates to time
Motivation for NC

Fast parallel computation is (loosely) modeled as having poly many processors and taking poly-log time

Corresponds to NC (How?)

Depth translates to time

Total “work” is size of the circuit
An example
An example

PARITY in NC$^1$
An example

\[ \text{PARITY in } \mathsf{NC}^1 \]

\[ \text{PARITY} = \{ x \mid x \text{ has odd number of 1s} \} \]
An example

\[ \text{PARITY in NC}^1 \]

\[ \text{PARITY} = \{ x \mid x \text{ has odd number of 1s} \} \]

\[ \text{Circuit should evaluate } x_1 \oplus x_2 \oplus \ldots \oplus x_n \]
An example

PARITY in $\text{NC}^1$

PARITY = \{ $x$ | $x$ has odd number of 1s \}

Circuit should evaluate $x_1 \oplus x_2 \oplus \ldots \oplus x_n$

Tree of n-1 XOR gates: log n deep
An example

- **PARITY in NC¹**
  - PARITY = \{ x \mid x \text{ has odd number of 1s} \}
  - Circuit should evaluate $x_1 \oplus x_2 \oplus \ldots \oplus x_n$
  - Tree of n-1 XOR gates: log n deep
  - Each XOR gate implemented in depth 3
An example

- **PARITY in $\text{NC}^1$**
  - $\text{PARITY} = \{ x \mid x \text{ has odd number of } 1s \}$
  - Circuit should evaluate $x_1 \oplus x_2 \oplus ... \oplus x_n$
  - Tree of $n-1$ XOR gates: log $n$ deep
  - Each XOR gate implemented in depth 3
Another example
Another example

PATH ∈ AC₁
Another example

PATH ∈ AC¹

“Boolean” Matrix Multiplication
Another example

\[ \text{PATH} \in \text{AC}^1 \]

“Boolean” Matrix Multiplication

\[ Z = XY: z_{ij} = \bigvee_{k=1..n} (x_{ik} \land y_{kj}) \]
Another example

PATH $\in \mathbf{AC}^1$

"Boolean" Matrix Multiplication

$Z = XY$: $z_{ij} = \bigvee_{k=1..n} (x_{ik} \land y_{kj})$

$\mathbf{AC}^0$ circuit (OR gate with fan-in n, AND gates)
Another example

PATH ∈ AC¹

“Boolean” Matrix Multiplication

\[ Z_{ij} = \bigvee_{k=1...n} (X_{ik} \land Y_{kj}) \]

AC⁰ circuit (OR gate with fan-in n, AND gates)

If X adjacency matrix (with self-loops), \(X^t_{ij}=1\) iff path from i to j of length \(t\) or less
Another example

PATH \in AC^1

“Boolean” Matrix Multiplication

Z=XY: \[ z_{ij} = \bigvee_{k=1..n} (x_{ik} \land y_{kj}) \]

AC^0 circuit (OR gate with fan-in n, AND gates)

If X adjacency matrix (with self-loops), \( X^t_{ij} = 1 \) iff path from i to j of length t or less

\[ X^m_{ij} \text{ for } m \geq n \text{ is the transitive closure} \]
Another example

PATH ∈ AC¹

“Boolean” Matrix Multiplication

Z = XY: \( z_{ij} = \bigvee_{k=1..n} (x_{ik} \land y_{kj}) \)

AC⁰ circuit (OR gate with fan-in n, AND gates)

If X adjacency matrix (with self-loops), \( X^t_{ij} = 1 \) iff path from i to j of length \( t \) or less

\( X^m_{ij} \) for \( m \geq n \) is the transitive closure

O(log n) matrix multiplications to compute \( X^n_{ij} \)
Another example

PATH $\in \text{AC}^1$

"Boolean" Matrix Multiplication

$Z = XY$: $z_{ij} = \lor_{k=1..n} (x_{ik} \land y_{kj})$

AC$^0$ circuit (OR gate with fan-in n, AND gates)

If $X$ adjacency matrix (with self-loops), $X^t_{ij} = 1$ iff path from $i$ to $j$ of length $t$ or less

$X^{m}_{ij}$ for $m \geq n$ is the transitive closure

$O(\log n)$ matrix multiplications to compute $X^n_{ij}$

Total depth $O(\log n)$
$\mathsf{NC^1} \subseteq \mathsf{L}$
$\text{NC}^1 \subseteq \mathbb{L}$

- Generate circuit (implicitly) and evaluate
\[ \text{NC}^1 \subseteq \text{L} \]

Generate circuit (implicitly) and evaluate

cf. \( \text{NC} \subseteq \text{P} \). But now, to conserve space, a recursive evaluation (rather than bottom-up).
\[ \text{NC}^1 \subseteq \text{L} \]

- Generate circuit (implicitly) and evaluate

- cf. NC \( \subseteq \) P. But now, to conserve space, a recursive evaluation (rather than bottom-up).

- For each gate, recursively evaluate each input wire
\( \text{NC}^1 \subseteq \text{L} \)

- Generate circuit (implicitly) and evaluate
- cf. \( \text{NC} \subseteq \text{P} \). But now, to conserve space, a recursive evaluation (rather than bottom-up).
- For each gate, recursively evaluate each input wire
- Storage: A path to the current node, from the output node: since bounded fan-in, takes \( O(1) \) bits per node; since logspace uniform that is sufficient to compute the node id in logspace
\[ \text{NC}^1 \subseteq L \]

Generate circuit (implicitly) and evaluate

- cf. \( \text{NC} \subseteq \text{P} \). But now, to conserve space, a recursive evaluation (rather than bottom-up).

- For each gate, recursively evaluate each input wire

- Storage: A path to the current node, from the output node: since bounded fan-in, takes \( O(1) \) bits per node; since logspace uniform that is sufficient to compute the node id in logspace

- And at each node along the path, the input wire values evaluated results so far (again \( O(1) \) bits per node)
\( \text{NC}^1 \subseteq \text{L} \)

Generate circuit (implicitly) and evaluate

- cf. \( \text{NC} \subseteq \text{P} \). But now, to conserve space, a recursive evaluation (rather than bottom-up).

For each gate, recursively evaluate each input wire

- Storage: A path to the current node, from the output node: since bounded fan-in, takes \( O(1) \) bits per node; since logspace uniform that is sufficient to compute the node id in logspace

- And at each node along the path, the input wire values evaluated results so far (again \( O(1) \) bits per node)

- Length of path = depth of circuit = \( O(\log n) \)
$\text{NL} \subseteq \text{AC}^1$
$\text{NL} \subseteq \text{AC}^1$

Recall $\text{PATH} \in \text{AC}^1$
$\text{NL } \subseteq \text{AC}^1$

- Recall $\text{PATH } \in \text{AC}^1$
- Also recall $\text{PATH}$ is NL-complete
\( \text{NL} \subseteq \text{AC}^1 \)

- Recall \( \text{PATH} \in \text{AC}^1 \)
- Also recall \( \text{PATH} \) is NL-complete
  - with respect to log-space reductions
NL $\subseteq$ AC$^1$

- Recall PATH $\in$ AC$^1$
- Also recall PATH is NL-complete
  - with respect to log-space reductions
  - in fact, with respect to NC$^1$ reductions
\[ \text{NL} \subseteq \text{AC}^1 \]

- Recall \( \text{PATH} \in \text{AC}^1 \)
- Also recall \( \text{PATH} \) is NL-complete
  - with respect to log-space reductions
  - in fact, with respect to NC\(^1\) reductions
- For NL machine \( M \), can build (in log-space) an NC\(^1\) circuit which on input \( x \), outputs \((i,j)^{th}\) entry of the adjacency matrix of configuration graph of \( M(x) \)
Recall $\text{PATH} \in \text{AC}^1$

Also recall $\text{PATH}$ is NL-complete

with respect to log-space reductions

in fact, with respect to $\text{NC}^1$ reductions

For NL machine $M$, can build (in log-space) an $\text{NC}^1$ circuit which on input $x$, outputs $(i,j)^{th}$ entry of the adjacency matrix of configuration graph of $M(x)$

Combining the $\text{NC}^1$ circuits for reduction and the $\text{AC}^1$ circuit for $\text{PATH}$, we get an $\text{AC}^1$ circuit
Summary: NC$_i$ and AC$_i$
Summary: $NC^i$ and $AC^i$

$NC^i \subseteq AC^i \subseteq NC^{i+1} \subseteq NC = AC \subseteq P$
Summary: $\text{NC}^i$ and $\text{AC}^i$

- $\text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1} \subseteq \text{NC} = \text{AC} \subseteq \text{P}$

- $\text{NC}^0 \subseteq \text{AC}^0 \subsetneq \text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{AC}^1$
Summary: \(\text{NC}^i\) and \(\text{AC}^i\)

- \(\text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1} \subseteq \text{NC} = \text{AC} \subseteq \text{P}\)
- \(\text{NC}^0 \subseteq \text{AC}^0 \subsetneq \text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{AC}^1\)

- \(\text{AC}^0 \subsetneq \text{NC}^1\) as \(\text{PARITY} \notin \text{AC}^0\) (later)
Summary: $\text{NC}^i$ and $\text{AC}^i$

- $\text{NC}^i \subseteq \text{AC}^i \subseteq \text{NC}^{i+1} \subseteq \text{NC} = \text{AC} \subseteq \text{P}$
- $\text{NC}^0 \subsetneq \text{AC}^0 \subsetneq \text{NC}^1 \subseteq \text{L} \subseteq \text{NL} \subseteq \text{AC}^1$
- $\text{AC}^0 \subsetneq \text{NC}^1$ as $\text{PARITY} \notin \text{AC}^0$ (later)
- Open: whether $\text{NC}^i \subsetneq \text{AC}^i \subsetneq \text{NC}^{i+1}$ for larger $i$
Summary: \( NC_i \) and \( AC_i \)

- \( NC_i \subseteq AC_i \subseteq NC_{i+1} \subseteq NC = AC \subseteq P \)
- \( NC^0 \subseteq AC^0 \subseteq NC^1 \subseteq L \subseteq NL \subseteq AC^1 \)
  - \( AC^0 \subseteq NC^1 \) as \( PARITY \notin AC^0 \) (later)
- Open: whether \( NC_i \subsetneq AC_i \subsetneq NC_{i+1} \) for larger \( i \)
- Open: Is \( NC = P \)? (Can all polynomial time decidable languages be sped up to poly-log time using parallelization?)
Zoo
DC Uniform
DC Uniform

Recall Uniform circuit family: circuits in the family can be generated by a TM
Recall Uniform circuit family: circuits in the family can be generated by a TM

Suppose circuits are super-polynomially large. Cannot be logspace-uniform or $\mathsf{P}$-uniform.
Recall Uniform circuit family: circuits in the family can be generated by a TM.

Suppose circuits are super-polynomially large. Cannot be logspace-uniform or \( P \)-uniform.

DC uniform allows exponentially large circuits.
DC Uniform

Recall Uniform circuit family: circuits in the family can be generated by a TM

Suppose circuits are super-polynomially large. Cannot be logspace-uniform or P-uniform.

- DC uniform allows exponentially large circuits
- Still requires polynomial time implicit computation of the circuit
Recall Uniform circuit family: circuits in the family can be generated by a TM.

Suppose circuits are super-polynomially large. Cannot be logspace-uniform or P-uniform.

- DC uniform allows exponentially large circuits.
- Still requires polynomial time implicit computation of the circuit.
- Coincides with EXP (Why?)
O(1) depth DC Uniform
O(1) depth DC Uniform

Restricted to depth $k$, $2^{\text{poly}(n)}$ size, unbounded fan-in DC uniform circuit families decide exactly languages in $\Sigma_k^P \cup \Pi_k^P$
Restricted to depth $k$, $2^{\text{poly}(n)}$ size, unbounded fan-in DC uniform circuit families decide exactly languages in $\Sigma_k^P \cup \Pi_k^P$.

Given a DC uniform circuit (w.l.o.g alternating levels of AND and OR gates, and NOT gates only at the input level) of depth $k$, an equivalent quantified expression with $k$ alternations.
O(1) depth DC Uniform

- Restricted to depth $k$, $2^{\text{poly}(n)}$ size, unbounded fan-in DC uniform circuit families decide exactly languages in $\Sigma_k^P \cup \Pi_k^P$

- Given a DC uniform circuit (w.l.o.g alternating levels of AND and OR gates, and NOT gates only at the input level) of depth $k$, an equivalent quantified expression with $k$ alternations

- Given a quantified expression with $k$ alternations, an equivalent DC uniform circuit of depth $k$
O(1) depth DC Uniform
O(1) depth DC Uniform

- From circuit to quantified expression
O(1) depth DC Uniform

- From circuit to quantified expression

- Consider game played on the circuit: adversary picks an edge going into the AND level and Alice picks an edge going into the OR level, going through levels top to bottom
O(1) depth DC Uniform

From circuit to quantified expression

Consider game played on the circuit: adversary picks an edge going into the AND level and Alice picks an edge going into the OR level, going through levels top to bottom.

Alice wins if adversary "breaks off the path" (by picking either a non-wire edge or a wire not continuing the path), or if the path terminates at literal of value 1 (w/o breaking)
O(1) depth DC Uniform

- From circuit to quantified expression

Consider game played on the circuit: adversary picks an edge going into the AND level and Alice picks an edge going into the OR level, going through levels top to bottom.

Alice wins if adversary “breaks off the path” (by picking either a non-wire edge or a wire not continuing the path), or if the path terminates at literal of value 1 (w/o breaking).

Can check in poly time
O(1) depth DC Uniform

- From circuit to quantified expression

Consider game played on the circuit: adversary picks an edge going into the AND level and Alice picks an edge going into the OR level, going through levels top to bottom.

Alice wins if adversary “breaks off the path” (by picking either a non-wire edge or a wire not continuing the path), or if the path terminates at literal of value 1 (w/o breaking).

Input accepted by the circuit iff Alice has a winning strategy (i.e., if the quantified expression is true).

Can check in poly time.
Consider game played on the circuit: adversary picks an edge going into the AND level and Alice picks an edge going into the OR level, going through levels top to bottom. Alice wins if adversary "breaks off the path" (by picking either a non-wire edge or a wire not continuing the path), or if the path terminates at literal of value 1 (w/o breaking). Input accepted by the circuit iff Alice has a winning strategy (i.e., if the quantified expression is true). Each edge has a polynomially long label, and quantified variables take values from the same domain. Checking if edge is a correct wire in poly time (uniformity).
O(1) depth DC Uniform
O(1) depth DC Uniform

- From quantified expression to circuit:
O(1) depth DC Uniform

- From quantified expression to circuit:
- Circuit has sub-circuits evaluating the poly-time condition for each possible assignment of the quantified variables.
O(1) depth DC Uniform

From quantified expression to circuit:

- Circuit has sub-circuits evaluating the poly-time condition for each possible assignment of the quantified variables.

- Hang these sub-circuits at the leaves of a k-level AND-OR tree appropriately.
O(1) depth DC Uniform

- From quantified expression to circuit:
  - Circuit has sub-circuits evaluating the poly-time condition for each possible assignment of the quantified variables.
  - Hang these sub-circuits at the leaves of a k-level AND-OR tree appropriately.
  - Circuit can be implicitly computed in polynomial time. Size $2^{O(\text{total length of variables})}$.
Today
Today

NC^i and AC^i
Today

- $\text{NC}^i$ and $\text{AC}^i$
- DC-uniform
Today

- NC^i and AC^i
- DC-uniform
- PH levels and EXP
Today

- $\text{NC}^i$ and $\text{AC}^i$
- DC-uniform
- PH levels and EXP
- Later, more circuits and non-uniform computation (time permitting)
Today

- NC\textsuperscript{i} and AC\textsuperscript{i}
- DC-uniform
- PH levels and EXP
- Later, more circuits and non-uniform computation (time permitting)

PARITY \notin AC\textsuperscript{0}
Today

- $\text{NC}^i$ and $\text{AC}^i$
- DC-uniform
- PH levels and EXP
- Later, more circuits and non-uniform computation (time permitting)
- PARITY $\notin \text{AC}^0$
- Decision trees, Branching programs
Today

- $\text{NC}^i$ and $\text{AC}^i$
- DC-uniform
- PH levels and EXP
- Later, more circuits and non-uniform computation (time permitting)
- PARITY $\not\in \text{AC}^0$
- Decision trees, Branching programs
- Connections between circuit lowerbounds and other complexity class separations