Non-Uniform Computation & Circuits

Lecture 10
Wherein every language can be decided
Non-Uniform Computation
Non-Uniform Computation

Uniform: Same program for all (the infinitely many) inputs
Non-Uniform Computation

- Uniform: Same program for all (the infinitely many) inputs
- Non-uniform: A different “program” for each input size
Non-Uniform Computation

- Uniform: Same program for all (the infinitely many) inputs
- Non-uniform: A different “program” for each input size
  - Then complexity of building the program and executing the program
Non-Uniform Computation

Uniform: Same program for all (the infinitely many) inputs

Non-uniform: A different “program” for each input size

Then complexity of building the program and executing the program

Sometimes will focus on the latter alone
Non-Uniform Computation

- Uniform: Same program for all (the infinitely many) inputs
- Non-uniform: A different “program” for each input size
  - Then complexity of building the program and executing the program
  - Sometimes will focus on the latter alone
  - Not entirely realistic if the program family is uncomputable or very complex to compute
Non-uniform advice
Non-uniform advice

Program: TM M and advice strings \{A_n\}
Non-uniform advice

- Program: TM $M$ and advice strings $\{A_n\}$
- $M$ given $A_{|x|}$ along with $x$
Non-uniform advice

- Program: TM $M$ and advice strings \{${A}_n$\}
  - $M$ given $A_{|x|}$ along with $x$
  - $A_n$ can be the program for inputs of size $n$
Non-uniform advice

- Program: TM $M$ and advice strings $\{A_n\}$
- $M$ given $A_{|x|}$ along with $x$
- $A_n$ can be the program for inputs of size $n$
- $|A_n| = 2^n$ is sufficient
Non-uniform advice

- Program: TM $M$ and advice strings $\{A_n\}$
- $M$ given $A_{|x|}$ along with $x$
- $A_n$ can be the program for inputs of size $n$
- $|A_n|=2^n$ is sufficient
- But $\{A_n\}$ can be uncomputable (even if just one bit long)
Non-uniform advice

- Program: TM $M$ and advice strings $\{A_n\}$
  - $M$ given $A_{|x|}$ along with $x$
  - $A_n$ can be the program for inputs of size $n$
  - $|A_n|=2^n$ is sufficient
  - But $\{A_n\}$ can be uncomputable (even if just one bit long)
  - e.g. advice to decide undecidable unary languages
P/poly and P/log
P/poly and P/log

\[ \text{DTIME}(T)/a \]
P/poly and P/log

\[ \text{DTIME}(T)/a \]

Languages decided by a TM in time \( T(n) \) using non-uniform advice of length \( a(n) \)
P/poly and P/log

- DTIME(T)/a
  - Languages decided by a TM in time T(n) using non-uniform advice of length a(n)
- P/poly = \( \bigcup_{c,d,k>0} \text{DTIME}(kn^c)/kn^d \)
**P/poly and P/log**

- **DTIME(T)/a**
  - Languages decided by a TM in time $T(n)$ using non-uniform advice of length $a(n)$

- **P/poly**
  - $\bigcup_{c,d,k>0} \text{DTIME}(kn^c)/kn^d$

- **P/log**
  - $\bigcup_{c,k>0} \text{DTIME}(kn^c)/k \log n$
NP vs. P/log, P/poly
NP vs. P/log, P/poly

- P/log (or even DTIME(1)/1) has undecidable languages
NP vs. P/log, P/poly

- P/log (or even DTIME(1)/1) has undecidable languages
- e.g. unary undecidable languages
NP vs. P/log, P/poly

- P/log (or even DTIME(1)/1) has undecidable languages
  - e.g. unary undecidable languages
- So P/log cannot be contained in any of the uniform complexity classes
NP vs. P/log, P/poly

- P/log (or even DTIME(1)/1) has undecidable languages
  - e.g. unary undecidable languages
- So P/log cannot be contained in any of the uniform complexity classes
- P/log contains P
NP vs. P/log, P/poly

- P/log (or even DTIME(1)/1) has undecidable languages
  - e.g. unary undecidable languages
  - So P/log cannot be contained in any of the uniform complexity classes

- P/log contains P

- Does P/log or P/poly contain NP?
NP ⊆ P/log ⇒ NP=P
NP \subseteq P/\text{log} \Rightarrow NP=P

\medskip

Recall finding witness for an NP language is Turing reducible to deciding the language.
Search using Decision

Suppose given “oracles” for deciding all NP languages, can we easily find certificates?

Yes! So, if decision easy (decision-oracles realizable), then search is easy too!

Say, given $x$, need to find $w$ s.t. $(x,w) \in L'$ (if such $w$ exists)

consider $L_1$ in NP: $(x,y) \in L_1$ iff $\exists z$ s.t. $(x,yz) \in L'$. (i.e., can $y$ be a prefix of a certificate for $x$).

Query $L_1$-oracle with $(x,0)$ and $(x,1)$. If $\exists w$, one of the two must be positive: say $(x,0) \in L_1$; then first bit of $w$ be 0.

For next bit query $L_1$-oracle with $(x,00)$ and $(x,01)$
Search using Decision

Suppose given “oracles” for deciding all NP languages, can we easily find certificates?

Yes! So, if decision easy (decision-oracles realizable), then search is easy too!

Say, given $x$, need to find $w$ s.t. $(x,w) \in L'$ (if such $w$ exists).

Consider $L_1$ in NP: $(x,y) \in L_1$ iff $\exists z$ s.t. $(x,yz) \in L'$. (i.e., can $y$ be a prefix of a certificate for $x$).

Query $L_1$-oracle with $(x,0)$ and $(x,1)$. If $\exists w$, one of the two must be positive: say $(x,0) \in L_1$; then first bit of $w$ be 0.

For next bit query $L_1$-oracle with $(x,00)$ and $(x,01)$.
NP ⊆ P/log ⇒ NP=P

Recall finding witness for an NP language is Turing reducible to deciding the language
\( \text{NP} \subseteq \text{P/log} \implies \text{NP}=\text{P} \)

- Recall finding witness for an NP language is Turing reducible to deciding the language.

- If \( \text{NP} \subseteq \text{P/log} \), then for each \( L \) in \( \text{NP} \), there is a poly-time TM with log advice which can find witness (via self-reduction).
\[ \text{NP} \subseteq \text{P/log} \Rightarrow \text{NP}=\text{P} \]

- Recall finding witness for an NP language is Turing reducible to deciding the language.

- If \( \text{NP} \subseteq \text{P/log} \), then for each \( L \) in \( \text{NP} \), there is a poly-time TM with log advice which can find witness (via self-reduction).

- Guess advice (poly many), and for each guessed advice, run the TM and see if it finds witness.
NP \subseteq P/\log \Rightarrow NP=P

- Recall finding witness for an NP language is Turing reducible to deciding the language

- If NP \subseteq P/\log, then for each L in NP, there is a poly-time TM with log advice which can find witness (via self-reduction)

- Guess advice (poly many), and for each guessed advice, run the TM and see if it finds witness

- If no advice worked (one of them was correct), then input not in language
$\text{NP} \subseteq \text{P/poly} \Rightarrow \text{PH} = \Sigma_2^P$
NP \subseteq \text{P/poly} \implies \text{PH} = \Sigma_2^P

Will show \( \Pi_2^P = \Sigma_2^P \)
\[ \text{NP} \subseteq \mathsf{P/poly} \implies \mathsf{PH}=\Sigma_2^P \]

\[ \text{Will show } \Pi_2^P = \Sigma_2^P \]

\[ \text{Consider } L = \{ x | \forall w_1 \ (x,w_1) \in L' \} \in \Pi_2^P \text{ where } \]
\[ L' = \{(x,w_1) | \exists w_2 \ F(x,w_1,w_2) \} \in \mathsf{NP} \]
\[ \text{NP} \subseteq \text{P/poly} \implies \text{PH} = \Sigma_2^P \]

- Will show $\Pi_2^P = \Sigma_2^P$

- Consider $L = \{x| \forall w_1 (x,w_1) \in L' \} \in \Pi_2^P$ where $L' = \{(x,w_1)| \exists w_2 F(x,w_1,w_2)\} \in \text{NP}$

- If NP $\subseteq$ P/poly then consider $M$ with advice $\{A_n\}$ which finds witness for $L'$: i.e. if $(x,w_1) \in L'$, then $M(x,w_1; A_n)$ outputs a witness $w_2$ s.t. $F(x,w_1,w_2)$
\[ \text{NP} \subseteq \text{P/poly} \Rightarrow \text{PH}=\Sigma^{P}_{2} \]

- Will show \( \Pi^{P}_{2} = \Sigma^{P}_{2} \)

- Consider \( L = \{ x | \forall w_{1} \ (x, w_{1}) \in L' \} \in \Pi^{P}_{2} \) where 
  \( L' = \{ (x, w_{1}) | \exists w_{2} \ F(x, w_{1}, w_{2}) \} \in \text{NP} \)

- If \( \text{NP} \subseteq \text{P/poly} \) then consider \( M \) with advice \( \{ A_{n} \} \) which finds witness for \( L' \): i.e. if \( (x, w_{1}) \in L' \), then \( M(x, w_{1}; A_{n}) \) outputs a witness \( w_{2} \) s.t. \( F(x, w_{1}, w_{2}) \)

- \( L = \{ x | \exists z \forall w_{1} \ F(x, w_{1}, M(x, w_{1}; z)) \} \)
Boolean Circuits
Boolean Circuits

Non-uniformity: circuit family \( \{C_n\} \)
Boolean Circuits

- Non-uniformity: circuit family \( \{C_n\} \)
- Given non-uniform computation \((M,\{A_n\})\), can define equivalent \( \{C_n\} \)
Boolean Circuits

- Non-uniformity: circuit family \( \{ C_n \} \)
- Given non-uniform computation \((M, \{ A_n \})\), can define equivalent \( \{ C_n \} \)
Boolean Circuits

- Non-uniformity: circuit family \( \{C_n\} \)
- Given non-uniform computation \((M,\{A_n\})\), can define equivalent \( \{C_n\} \)
- Advice \(A_n\) is hard-wired into circuit \(C_n\)
Boolean Circuits

- Non-uniformity: circuit family \( \{C_n\} \)
- Given non-uniform computation \((M,\{A_n\})\), can define equivalent \( \{C_n\}\)
- Advice \(A_n\) is hard-wired into circuit \(C_n\)
- Size of circuit polynomially related to running time of TM
Boolean Circuits

- Non-uniformity: circuit family \( \{C_n\} \)
- Given non-uniform computation \((M, \{A_n\})\), can define equivalent \( \{C_n\} \)
- Advice \( A_n \) is hard-wired into circuit \( C_n \)
- Size of circuit polynomially related to running time of TM

Size = no. of wires
Boolean Circuits

- Non-uniformity: circuit family \( \{C_n\} \)
- Given non-uniform computation \((M,\{A_n\})\), can define equivalent \( \{C_n\} \)
- Advice \( A_n \) is hard-wired into circuit \( C_n \)
- Size of circuit polynomially related to running time of TM
- Conversely, given \( \{C_n\} \), can use description of \( C_n \) as advice \( A_n \) for a “universal” TM
Boolean Circuits

- Non-uniformity: circuit family \{C_n\}

- Given non-uniform computation \((M,\{A_n\})\), can define equivalent \{C_n\}

Advice \(A_n\) is hard-wired into circuit \(C_n\)

- Size of circuit polynomially related to running time of TM

- Conversely, given \{C_n\}, can use description of \(C_n\) as advice \(A_n\) for a "universal" TM

- \(|A_n|\) comparable to size of circuit \(C_n\)
SIZE(T)
SIZE(T)

SIZE(T): languages solved by circuit families of size T(n)
SIZE(T)

- SIZE(T): languages solved by circuit families of size $T(n)$
- $P/poly = SIZE(poly)$
SIZE(T)

- SIZE(T): languages solved by circuit families of size $T(n)$
- $P/poly = SIZE(poly)$
  - $SIZE(poly) \subseteq P/poly$: Size $T$ circuit can be described in $O(T \log T)$ bits (advice). Universal TM can evaluate this circuit in poly time
SIZE(T)

- SIZE(T): languages solved by circuit families of size T(n)
- P/poly = SIZE(poly)
  - SIZE(poly) ⊆ P/poly: Size T circuit can be described in O(T log T) bits (advice). Universal TM can evaluate this circuit in poly time
  - P/poly ⊆ SIZE(poly): Transformation from Cook’s theorem, with advice string hardwired into circuit
SIZE bounds
SIZE bounds

* All languages (decidable or not) are in SIZE(T) for $T = O(n^{2^n})$
SIZE bounds

- All languages (decidable or not) are in SIZE(T) for $T=O(n2^n)$
- Circuit encodes truth-table
SIZE bounds

- All languages (decidable or not) are in $\text{SIZE}(T)$ for $T=O(n2^n)$

- Circuit encodes truth-table

- Most languages need circuits of size $\Omega(2^n/n)$
SIZE bounds

- All languages (decidable or not) are in SIZE(T) for $T=O(n2^n)$
- Circuit encodes truth-table
- Most languages need circuits of size $\Omega(2^n/n)$
- Number of circuits of size $T$ is at most $T^{2T}$
SIZE bounds

- All languages (decidable or not) are in SIZE(T) for T=O(n2ⁿ)
  
  - Circuit encodes truth-table

- Most languages need circuits of size Ω(2ⁿ/n)

- Number of circuits of size T is at most T²^T

- If T = 2ⁿ/4n, say, T²^T < 2(2ⁿ)/2
SIZE bounds

- All languages (decidable or not) are in SIZE(T) for T = O(n2^n)
  - Circuit encodes truth-table
- Most languages need circuits of size Ω(2^n/n)
  - Number of circuits of size T is at most T^{2T}
- If T = 2^n/4n, say, T^{2T} < 2^{(2^n)/2}
- Number of languages = 2^{2^n}
SIZE hierarchy
SIZE hierarchy

\( \text{SIZE}(T') \subsetneq \text{SIZE}(T) \) if \( T = \Omega(t2^t) \) and \( T' = O(2^t/t) \), for \( t(n) \leq n \)
SIZE hierarchy

SIZE(T') ⊊ SIZE(T) if T=\Omega(t2^t) and T'=O(2^t/t), for t(n)≤n

e.g., T(n) = n \log n and T'(n) = n/log n
SIZE hierarchy

- \( \text{SIZE}(T') \subseteq \text{SIZE}(T) \) if \( T = \Omega(t2^t) \) and \( T' = O(2^t/t) \), for \( t(n) \leq n \)
- e.g., \( T(n) = n \log n \) and \( T'(n) = n/\log n \)
- Consider functions on \( t \) bits (ignoring \( n-t \) bits)
SIZE hierarchy

\[ \text{SIZE}(T') \subseteq \text{SIZE}(T) \text{ if } T = \Omega(t2^t) \text{ and } T' = O(2^t/t), \text{ for } t(n) \leq n \]

- e.g., \( T(n) = n \log n \) and \( T'(n) = n / \log n \)
- Consider functions on \( t \) bits (ignoring \( n-t \) bits)
- All of them in \( \text{SIZE}(T) \), most not in \( \text{SIZE}(T') \)
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$^i$ and AC$^i$
NCᵢ and ACᵢ

NCᵢ: class of languages decided by bounded fan-in logspace-uniform circuits of polynomial size and depth $O(\log^i n)$
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^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^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\(^i\) 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\(^0\) \subseteq AC\(^0\)**: Consider \(L = \{1,11,111,...\}\)
NC and AC
NC and AC

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

\( \text{NC} = \bigcup_{i>0} \text{NC}^i \). 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

\( \text{NC} = \bigcup_{i>0} \text{NC}^i \). 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

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

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

Clearly \( NC^i \subseteq AC^i \)

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

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

- 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
NC ⊆ 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
NC ⊆ 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 NC = 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
Today
Today

Non-uniform complexity
Today

- Non-uniform complexity
- \( P/1 \not\subseteq \text{Decidable} \)
Today

- Non-uniform complexity

- $P/1 \not\subseteq$ Decidable

- $NP \subseteq P/log \Rightarrow NP = P$
Today

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

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

- Non-uniform circuit Complexity
Today

- Non-uniform complexity
  - \( P/1 \not\subseteq \text{Decidable} \)
  - \( \text{NP} \subseteq P/\log \Rightarrow \text{NP} = P \)
  - \( \text{NP} \subseteq P/poly \Rightarrow \text{PH} = \Sigma_2^P \)
- Non-uniform circuit Complexity
  - \( \text{SIZE}(\text{poly}) = P/poly \)
Non-uniform complexity

- \( \text{P}/1 \not\subseteq \text{Decidable} \)
- \( \text{NP} \subseteq \text{P}/\log \Rightarrow \text{NP} = \text{P} \)
- \( \text{NP} \subseteq \text{P}/\text{poly} \Rightarrow \text{PH} = \Sigma_2^P \)

Non-uniform circuit Complexity

- \( \text{SIZE}(\text{poly}) = \text{P}/\text{poly} \)
- \( \text{SIZE}-\text{hierarchy}: \text{SIZE}(T') \subset \text{SIZE}(T) \text{ if } T = \Omega(t^{2^t}), T' = O(2^t/t) \)
Today

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

- Non-uniform circuit Complexity
  - $\text{SIZE}(\text{poly}) = P/poly$
  - $\text{SIZE}$-hierarchy: $\text{SIZE}(T') \subsetneq \text{SIZE}(T)$ if $T=\Omega(t2^t)$, $T'=O(2^t/t)$

- Uniform Circuit Complexity
Today

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

Non-uniform circuit Complexity

- $\text{SIZE}(\text{poly}) = P/poly$
- $\text{SIZE}$-hierarchy: $\text{SIZE}(T') \subset \text{SIZE}(T)$ if $T = \Omega(2^t)$, $T' = O(2^t/t)$

Uniform Circuit Complexity

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