Randomized complexity classes

Today: BPP (part 2) and P/poly
Today

- Circuits, $\text{P/poly}$
- Adleman’s theorem: $\text{BPP} \subseteq \text{P/poly}$
- The Karp-Lipton theorems, and consequences
Circuits

- Informally, collections of logical gates connected by wires
- Must be acyclic
- Wires can be shared
- Fan-in arbitrary here (e.g., 1=fan-in 0 and, 0=fan-in 0 or)

Remember: CIRCUIT-VALUE is P-complete (for logspace reductions)
Circuits

- Informally, collections of logical gates connected by wires
- Must be acyclic
- Wires can be shared
- Fan-in arbitrary here (e.g., 1=fan-in 0 and, 0=fan-in 0 or)

- We now consider circuits $C$ with input wires

$C[x] = \text{value of } C \text{ when fed input bits } x$
Circuits, formally: net-lists

- We encode circuits as words (net-lists), e.g.:

  wire 3 = ¬ wire 0
  wire 4 = 1 ∨ 2
  etc.

  8 is output

  3 ¬ 0
  4 ∨ 1 2
  5 ¬ 2
  6 ∧ 1 3
  7 ∨ 4 5
  8 ∨ 6 7

input wires (x)
output wire

We require wire numbers to be sorted
(implies acyclicity)
(sortedness checkable in logspace, acyclicity is NL-complete)
Reminder: **CIRCUIT-VALUE** is **P-complete**

- Encode \( p(n) \)-time TM \( M \) on input \( x \) by a circuit
- constant gates 1/0 encode initial state \( q_0 \), input \( x \), and blanks
- each inner cell depends on a constant #cells on row above  
  \( \Rightarrow \) circuit piece of cst size (replicated \( p(n)^2 \) times)
- finally, a small circuit to check **acceptance**.

output=1 iff \( M(x) \) accepts
Plenty of technical details…

- Each row encodes a config. of a **one-tape** TM \( M \)
- … in binary
- the machine **parks** the head at position 0 before accepting/rejecting
- … and continues working (doing **nothing**) forever (at least until time \( p(n) \))

**Reminder: CIRCUIT-VALUE is P-complete**

- Encode \( p(n) \)-time TM \( M \) on input \( x \) by a circuit
- constant gates \( 1/0 \) encode initial state \( q_0 \), input \( x \), and blanks
- each inner cell depends on a constant #cells on row above
  \( \Rightarrow \) circuit piece of\( \text{cst} \) size (replicated \( p(n)^2 \) times)
- finally, a small circuit to check acceptance.

Build the circuit in **logspace**: 2 nested loops from 0 to \( p(n) \), with 2 counters
An important remark

- We can **precompile** a circuit $C_n$ with $n$ free input wires — without knowing $x$, — just its length $n$, — still in logspace

- such that for every $x$ of that size $n$, $M(x)$ accepts $\iff C_n[x]=1$
A language \( L \) is in uniform \( \text{P/poly} \) iff for every \( n \), one can build a circuit \( C_n \) — in space \( O(\log n) \) — such that for every input \( x \) of size = \( n \), \( x \in L \iff C_n[x]=1 \).

Prop. \( \text{P} \subseteq \text{uniform P/poly} \).

(This is what we have just proved!)
P = Uniform P/poly

- A language L is in uniform P/poly iff for every n, one can build a circuit $C_n$ — in space $O(\log n)$ — such that for every input $x$ of size $= n$, $x \in L \iff C_n[x]=1$

- Prop. $P \subseteq$ uniform P/poly.

- In fact: Prop. $P = \text{uniform P/poly}$.

- Proof.
  Let $L \in \text{uniform P/poly}$. On input $x$ (size $n$), compute $C_n$ in space $k \log n$, hence in time $O(n^k)$. Then evaluate $C_n[x]$ in polytime. Hence $L \in P$. 
A language $L$ is in \textit{uniform} \textit{P/poly} iff for every $n$, one can build a circuit $C_n$ — in space $O(\log n)$ — such that for every input $x$ of size $= n$, $x \in L \iff C_n[x] = 1$

We no longer require to be able to compute $C_n$!

Familiarly, we say that $L$ has polynomial circuits
P/poly

- **Defn.** A language $L$ is in $\text{P/poly}$ iff there is a family $(C_n)_{n \in \mathbb{N}}$ of circuits:
  - of size $p(n)$ (for some fixed polynomial $p$)
  - such that for every input $x$ (letting $n$ be its size)
    $$x \in L \iff C_n[x]=1.$$ 

- It was initially hoped that we could prove that some $\text{NP}$-complete languages do not have polynomial circuits. That would immediately imply $\text{P} \neq \text{NP}$, since $\text{P} \subseteq \text{P/poly}$. 
P/poly is pretty weird

- **Prop.** P/poly contains some undecidable languages.

- **Proof.**
  Let $L$ be undecidable (e.g., HALT).
  Then $L' = \{\text{words } 1^n \mid \text{bin}(n) [=n \text{ written in binary}] \text{ is a word in } L\}$ is undecidable, too; and $C_n$ is…

<table>
<thead>
<tr>
<th>If bin$(n) \notin L$</th>
<th>If bin$(n) \in L$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(size $n \log n$: check the net-list!)</td>
</tr>
<tr>
<td>(ignores its input, size $O(1)$)</td>
<td>and</td>
</tr>
</tbody>
</table>

**Defn.** A language $L$ is in P/poly iff there is a family $(C_n)_{n \in \mathbb{N}}$ of circuits:
- of size $p(n)$ (for some fixed polynomial $p$)
- such that for every input $x$ (letting $n$ be its size)
  $x \in L \iff C_n[x] = 1.$
Imagine you wish to decide whether $x$ is in $L$.

... and you have a « cheat sheet » $w_n$ depending only on $n=\text{size}(x)$.

How can this help?

If $w_n$ allowed to have size $2^n$, then this helps a lot (why?)

What if $w_n$ is only allowed to have polynomial size?
Advice strings and P/poly (1/2)

- **Prop.** \( L \in \text{P/poly} \) iff there is a polytime TM \( \mathcal{M} \) and a family \( (w_n)_{n \in \mathbb{N}} \) of so-called advice strings:
  - of polysize \( p(n) \)
  - s.t. \( \forall x \) (size \( n \))
    \[ x \in L \iff \mathcal{M}(x, w_n) \text{ accepts.} \]

**Proof.**

- If \( L \in \text{P/poly} \), then let \( w_n \) be a net-list for \( C_n \)
- If \( L \) has advice strings \( w_n \), then...

(see next slide)

**Defn.** A language \( L \) is in P/poly iff there is a family \( (C_n)_{n \in \mathbb{N}} \) of circuits:
  - of size \( p(n) \) (for some fixed polynomial \( p \))
  - such that for every input \( x \) (letting \( n \) be its size)
    \[ x \in L \iff C_n[x] = 1. \]
Advice strings and P/poly (2/2)

Note: same construction as before, except... now $C_n$ includes the constant bits of $w_n$ (still not $x$.)
Adleman’s Theorem
Theorem (Prop. 1.20). $\text{BPP} \subseteq \text{P/poly}$.

Interestingly, we will be able to show the \textit{existence} of the circuits $C_n$, (or the advice strings) but we won’t be able to \textit{compute} them (efficiently).
The proof of Adleman’s Theorem (1/2)

- Let $L$ be in $\text{BPP}$.
- Among the tapes $r$ (of size $p(n)$), is there one such that for every $x$ of size $n$, $M(x,r)$ always gives the correct answer?
- Let us use the probabilistic method…

A language $L$ is in $\text{BPP}$ if and only if there is a polynomial-time TM $M$ such that for every input $x$ (of size $n$):

$$\Pr_r (M(x,r) \text{ errs}) \leq \varepsilon.$$  

error $\varepsilon = 1/2^{q(n)}$

- $\Pr_r (\exists x \text{ of size } n, M(x,r) \text{ errs}) \leq \sum_x \Pr_r (M(x,r) \text{ errs}) \leq 2^{n-q(n)}$
- … < 1 if we had the good taste to pick $q(n)=n+1$, say.
The proof of Adleman’s Theorem (2/2)

- Let $L$ be in BPP. For each size $n$, there is a tape $r_n$ (of size $p(n)$) such that for every $x$ of size $n$, $M(x,r_n)$ gives the correct answer, i.e.:
  - if $x \in L$ then $M(x,r_n)$ accepts
  - if $x \notin L$ then $M(x,r_n)$ rejects.

- ... Just use $r_n$ as advice string! $\square$
The Karp-Lipton Theorems, and consequences

(Yes, them again!)
coC

- Recall that $\Pi_{P_k} = \text{co} \Sigma_{P_k} = \text{co}$ for every $k \geq 1$.
  (coC is the class of complements of languages of C.)

- **Fact.** co is monotonic: if $C \subseteq C'$, then $\text{co}C \subseteq \text{co}C'$.

- (Already argued last time, as part of the Sipser-Gács-Lautemann theorem.)
Claim. For any class $C$, the following are equivalent:

1. $C = \text{co}C$
2. $C \subseteq \text{co}C$
3. $\text{co}C \subseteq C$.

$2 \Rightarrow 3$: let $L$ in $\text{co}C$.

Its complement is in $C$, hence in $\text{co}C$ by 2.

Therefore $L$ is also in $C$.

$3 \Rightarrow 2$, and therefore $3 \Rightarrow 1$: similar. $1 \Rightarrow 2$: obvious. □
Does PH collapse?

- We say that **PH** collapses at level 2 iff $\Sigma^p_2 = \Pi^p_2$.
  By the previous claim, equivalent to $\Pi^p_2 \subseteq \Sigma^p_2$.

- **Prop.** If $\Sigma^p_2 = \Pi^p_2$ then
  $$\Sigma^p_2 = \Pi^p_2 = \Sigma^p_3 = \Pi^p_3 = \Sigma^p_4 = \ldots = \text{PH}$$ (whence the name.)

- **Proof sketch.** Let $\exists \cdot C$ be the class of the languages
  $$\{ x \mid \exists y \text{ of poly size, } (x, y) \in L' \}, \ L' \in C.$$ 

- $\Sigma^p_3 = \exists \cdot \Pi^p_2 = \exists \cdot \Sigma^p_2 = \exists \cdot \exists \cdot \text{coNP} = \exists \cdot \text{coNP} = \Sigma^p_2$, then
  $\Pi^p_3 = \text{co} \Sigma^p_3 = \text{co} \Sigma^p_2 = \Pi^p_2 = \Sigma^p_2$, etc. □
The first Karp-Lipton theorem

- **Theorem (Prop. 1.21).** If \( \text{NP} \subseteq \text{P/poly} \), then the polynomial hierarchy collapses at level 2: \( \Pi^p_2 \subseteq \Sigma^p_2 \).

- Let me give you a **wrong** argument first. (We will repair it later.)

- Let \( L \in \Pi^p_2 \) be \( \{ x \mid \forall y \text{ of size } p(n), (x,y) \in L' \} \), \( L' \in \text{NP} \).

- \( L' \) has polynomial circuits \( C_n \), so

- \( L = \{ x \mid \forall y \text{ of size } p(n), C_{\text{size}(x,y)}[(x,y)]=1 \} \)

- \( = \{ x \mid \exists \text{poly size } C, \forall y \text{ of size } p(n), C[(x,y)]=1 \} \in \Sigma^p_2 \).

Where is the bug?

We can permute quantifiers, because \( C_{\text{size}(x,y)}=C_{n+p(n)+3} \) does **not** depend on \( y \).
The first Karp-Lipton theorem

- **Theorem (Prop. 1.21).** If $\text{NP} \subseteq \text{P/poly}$, then the polynomial hierarchy collapses at level 2: $\Pi_2^p \subseteq \Sigma_2^p$.

- Let me give you a **wrong** argument first. (We will repair it later.)

- Let $L \in \Pi_2^p$ be $\{ x \mid \forall y \text{ of size } p(n), (x,y) \in L' \}$, $L' \in \text{NP}$.

- $L'$ has polynomial circuits $C_n$, so

- $L = \{ x \mid \forall y \text{ of size } p(n), C_{\text{size}(x,y)}[(x,y)] = 1 \}$

- $\quad = \{ x \mid \exists \text{poly size } C, \forall y \text{ of size } p(n), C[(x,y)] = 1 \}$

- $\quad \in \Sigma_2^p$.

- We can permute quantifiers, because $C_{\text{size}(x,y)} = C_{n+p(n)+3}$ does **not** depend on $y$. 

Hint: this is $\Sigma^*$, not $L$ (just take the constant circuit 1 for $C$ here)
The bug

- $L = \{ x \mid \forall y \text{ of size } p(n), C_{\text{size}(x,y)}[(x,y)] = 1 \}
  
  \neq \{ x \mid \exists \text{ poly size } C, \forall y \text{ of size } p(n), C[(x,y)] = 1 \}$: here we trust some divine (all-powerful) being Merlin to give us the magical circuit $C_{\text{size}(x,y)}$ for $C$...

- ... but what prevents it from cheating? We must check that the circuit $C$ it gives us does the job.
Imagine you want to solve SAT. You are given a clause set $S$, and you ask Merlin: «is $S$ satisfiable?»

Merlin answers: «yes»

What can you conclude?

Of course, nothing.
A thought experiment

Imagine you want to solve SAT. You are given a clause set $S$, and you ask Merlin: «is $S$ satisfiable? give me a satisfying assignment $\varphi$»

Merlin answers: «yes» $\varphi$

You check $\varphi \models S$, accept if this is true, reject otherwise.

If $S$ satisfiable, then Merlin can make you accept. Otherwise, you will necessarily reject.
Self-reducibility

- Now Merlin complains he can only decide whether $S$ is satisfiable (using circuits $C_n$), not find a satisfying $\varrho$.

- You retort that SAT is self-reducible: Given an oracle $O$ deciding satisfiability, one can compute $\varrho$ such that $\varrho \models S$ (if any).

Call this the « self-reducibility machine »
Self-reducibility

- Instead of an oracle $O$, Merlin will use circuits $C_m$ on clause sets $S, S_1, S_2, \ldots$, of various sizes $m$.

- $m$ is bounded by $n=\text{size}(S)$
  
  (e.g., $S[A:=1]$ is obtained by removing clauses in which $+A$ appears, and removing $-A$ in the remaining clauses)

Call this the «**self-reducibility machine**»
A circuit for self-reducibility

- Now given (net-lists for) $C_0, C_1, \ldots, C_n$ as advice $w_0\ldots n$

- the self-reducibility machine is a poly time TM $h$ taking $(S, w)$ as input — returning an environment $\varrho$
  — satisfying $S$, if $S$ is satisfiable and Merlin is honest (i.e., plays using the above advice $w_0\ldots n$ for $w$)

- Note that, if $\text{size}(C_n) = O(n^k)$ (poly), then
  \[ \text{size}(w_0\ldots n) = O(n^{k+1}) \text{ (poly again)} \]

by the way, not quite the trick used in the lecture notes
Karp-Lipton: the proof (1/3)

- **Theorem (Prop. 1.21).** If \( \text{NP} \subseteq \text{P/poly} \), then the polynomial hierarchy collapses at level 2: \( \Pi^p_2 \subseteq \Sigma^p_2 \).

- Let \( L \in \Pi^p_2 \) be \( \{x \mid \forall y \text{ of size } p(n), (x,y) \in L'\} \), \( L' \in \text{NP} \).

- We reduce to \( \text{SAT} \)
  (this will allow us to use self-reducibility!):
  - there is a polytime function \( f / (x,y) \in L' \iff f(x,y) \in \text{SAT} \)
  - Hence \( L = \{x \mid \forall y \text{ of size } p(n), f(x,y) \in \text{SAT}\} \)
Theorem (Prop. 1.21). If $\text{NP} \subseteq \text{P}/\text{poly}$, then the polynomial hierarchy collapses at level 2: $\Pi^p_2 \subseteq \Sigma^p_2$.

$L = \{x \mid \forall y \text{ of size } p(n), f(x,y) \in \text{SAT}\}$ (from last slide)

Now use self-reducibility:
$L = \{x \mid \forall y \text{ of size } p(n), h(f(x,y),w_0...\text{size}(f(x,y))) \models f(x,y)\}$

the « self-reducibility machine »

size of advice polynomial in $n=\text{size}(x)$

a clause set $S$
Karp-Lipton: the proof (3/3)

- **Theorem (Prop. 1.21).** If \( \text{NP} \subseteq \text{P/poly} \), then the polynomial hierarchy collapses at level 2: \( \Pi^p_2 \subseteq \Sigma^p_2 \).

- \( L = \{ x \mid \forall y \text{ of size } p(n), h(f(x,y),w_0\ldots\text{size}(f(x,y))) \models f(x,y) \} \) (last slide)

- I claim that \( L = \{ x \mid \exists w, \forall y \text{ of size } p(n), h(f(x,y),w) \models f(x,y) \} \)
  (huh? that was somehow right? No, we now **check** that \( h(\ldots) \models f(x,y) \)!)  

- If \( x \in L \), then take \( w = w_0\ldots\text{size}(f(x,y)) \): \( \forall y, h(f(x,y),w) \models f(x,y) \) ✔

- If \( x \notin L \), \( \exists y, f(x,y) \) is **unsatisfiable...**
  hence whichever \( w \) we take, \( h(f(x,y),w) \not\models f(x,y) \) ✔
The second Karp-Lipton theorem

- Theorem (Prop. 1.22). If \( \text{NP} \subseteq \text{P/poly} \), then \( \text{PH} \subseteq \text{P/poly} \).

- By previous result, it suffices to show \( \Sigma^{p_2} \subseteq \text{P/poly} \).

- Let \( L = \{ x \mid \exists y \text{ of size } p(n), (x, y) \in L' \} \) where \( L' \in \text{coNP} \).

- The complement of \( L' \) has poly size advice strings, hence \( L' \) also has poly size advice strings \( w_n \).

- \( L = \{ x \mid \exists y \text{ of size } p(n), M((x, y), w_0 \ldots \text{size}(x, y)) \text{ accepts} \} \) for some poly time TM \( M \).
The second Karp-Lipton theorem

- **Theorem (Prop. 1.22).** If $\text{NP} \subseteq \text{P/poly}$, then $\text{PH} \subseteq \text{P/poly}$.

- $L = \{x \mid \exists y \text{ of size } p(n), M((x,y), w_{\text{size}(x,y)}) \text{ accepts}\}$
  for some poly time TM $M$ (from last slide)

- Let $L'' = \{(x,w) \mid \exists y \text{ of size } p(\text{size}(x)), M((x,y), w) \text{ accepts}\}$
  This is in $\text{NP}$, hence has polynomial circuits $C_n$, too!

- So $L = \{x \mid C_{\text{appropriate size}}[(x, w_{\text{size}(x,y)})]=1\}$

- size of $x + \text{cst} + \text{size of } w_{\text{size}(x,y)}$…
  polynomial in $n=\text{size}(x)$
The second Karp-Lipton theorem

- **Theorem (Prop. 1.22).** If \( \text{NP} \subseteq \text{P/poly} \), then \( \text{PH} \subseteq \text{P/poly} \).

- So \( L = \{ x \mid C_{\text{appropriate size}}[(x, w_{\text{size}(x,y)})] = 1 \} \) (from last slide)

- Hence \( L \) is decided by the circuits \( C_{\text{appropriate size}}[(\_ , w_{\text{size}(x,y)})] \)

  \[\text{size of } x + \text{cst} + \text{size of } w_{\text{size}(x,y)}\ldots\]

  \[\text{polynomial in } n=\text{size}(x)\]

- (all sizes depending only on \( n=\text{size}(x) \), not on \( x \) itself)
Conclusion
BPP cannot be too large

- **Corollary.** If BPP contains NP, then:
  - PH collapses at level 2 (unlikely)
  - and is included in P/poly.

- **Proof.**

  - **Adleman’s Theorem**
    
    Theorem (Prop. 1.20). BPP ⊆ P/poly.

  - **The first Karp-Lipton theorem**
    
    Theorem (Prop. 1.21). If NP ⊆ P/poly, then the polynomial hierarchy collapses at level 2: \( \Pi_2 \subseteq \Sigma_2 \).

  - **The second Karp-Lipton theorem**
    
    Theorem (Prop. 1.22). If NP ⊆ P/poly, then PH ⊆ P/poly.