19. Non-Uniform Polynomial Time

Goal:
- Show that circuits indeed have implications for the P vs. NP question.
- Introduce Turing machines that take advice.

19.1 P/Poly and Karp & Lipton's Theorem

Idea of non-uniformity:
- Use a different algorithm for every input size
- Such algorithms (for every size n) could exist even if $P \neq NP$.

Goal: Give evidence that such size-dependent algorithms are unlikely to exist.

Definition:
$P/Poly$ is the class of languages that are decidable by polynomial-sized circuit families (not necessarily uniform).
One also says the language has polynomial circuit complexity.

Example:
Let $L \subseteq \{0,1\}^*$ be a unary language ($L \subseteq \{1^n \mid n \in \mathbb{N}\}$).
Then $L \in P/Poly$.
Proof:
- For $n \in \mathbb{N}$ with $1^n \in L$, use a tree of $\wedge$-gates.
- For $n \in \mathbb{N}$ with $1^n \notin L$, use constant 0. □

Theorem: $P \subsetneq P/Poly$.
Proof: $\subseteq$
- Let $H \in P$ be decided by a TM with runtime n^c.
We construct an $n^c \times n^c$ matrix as in the proof of Ladner's theorem.

.#: The language

UNHALT := $\{1^n \mid n$'s binary representation encodes a pair (M,x) so that M halts on $x\}$.

This is the unary version of the halting-problem.
The language is undecidable and hence not in P.
It is in $P/poly$ by the example. □

Question: Does SAT have small circuits:

$SAT \in P/poly$?

Answer: No, provided the polynomial hierarchy does not collapse.
Hence, efficient non-uniform algorithms for SAT are unlikely.

Theorem (Karp & Lipton '80):

If $NP \subseteq P/poly$, then $PH = \Sigma_2^p$.

Proof:

- Recall that to show $PH = \Sigma_2^p$,
it is sufficient to show $\Pi_2^p \subseteq \Sigma_2^p$.
For the latter inclusion, it is sufficient to show
that Σ_2^p contains the Π_2^p-complete language

(1) $\Pi_2^p\text{-SAT} := \{ \underbrace{C}_{\text{unquantified}} \mid \forall u \in \{0,1\}^n \exists v \in \{0,1\}^n : C(u,v) = 1 \}$.

- If $NP \subseteq P/poly$, there exists a polynomial p
and a $p(n)$-sized circuit family $(C_n)_{n \in \mathbb{N}}$
so that for every Boolean formula C and every $u \in \{0,1\}^n$:

$C_n(C,u) = 1$ iff $\exists v \in \{0,1\}^n : C(u,v) = 1$.

- The circuits solve the decision version for SAT.
We can convert a decision algorithm for SAT into one that outputs a satisfying assignment, if it exists.
The idea is to repeatedly query SAT:

SAT (at all)
↓ yes
SAT with $x_0 = 0$
yes ( ) no
SAT with $x_1 = 0$
⋮

The procedure turns $(C_n)_{n\in\mathbb{N}}$ into the family $(C'_n)_{n\in\mathbb{N}}$.
The new family is q-sized with q a polynomial.
Moreover, for every formula ℓ and input $u \in \{0,1\}^n$

> if there is a string $v \in \{0,1\}^n$ with $\ell(u,v) = 1$,
> then $C_n(\ell, u)$ outputs such a v.

- The assumption $NP \subseteq P_{/poly}$ only guarantees the existence of the family $(C'_n)_{n\in\mathbb{N}}$.
The main idea is to guess the circuit using $\exists$-quantification.
Since the circuit outputs a satisfying assignment (if one exists), the answer can be decided directly in P:

(2) $\exists \underbrace{w \in \{0,1\}^{\overbrace{h \cdot q(n)^2}^{\text{wires}}}}_{\text{circuit}} : \forall u \in \{0,1\}^n :$ w describes a circuit C' and $\ell(u, C'(\ell,u)) = 1$.

This is a Σ_2^P-problem.

Claim: (1) $\Leftrightarrow$ (2).

- Assume (1) is false.
  Then for some u there is no v with $C(u,v) = 1$.
  Hence, (2) will be false.
- Assume (1) is true.
  Choose the right circuit.
  If for every u there is v, the circuit will output v. □

Note: The proof is another instance of Skolemization, see Section 15.2.

19.2 Circuit Lower Bounds

Summary:
- Since $P \subseteq P/poly$, if we ever prove $NP \not\subseteq P/poly$ (say by showing that the polynomial hierarchy does not collapse), we have $P \neq NP$.
- Karp & Lipton's theorem gives evidence that $NP \not\subseteq P/poly$.

Hence: To show $P \neq NP$ it would be sufficient to come up with one function

$$f^n : \{0,1\}^n \to \{0,1\}$$

that
- is in NP
- and requires large circuits.

Goal: Show that functions requiring large circuits do exist.
(Actually, almost every function needs exponential-sized circuits, but this requires a different proof technique.
None of the functions is known to be in NP.)

Theorem (Shannon '49):
For every n, there is a function $f^n : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit of size $2^n/10n$.

Proof:

The proof is by a cardinality argument.

- The number of functions from $\{0,1\}^n$ to $\{0,1\}$ is 2^{2^n}.
- Every circuit of size $\leq S$ can be represented (adjacency list) as a binary string of $9 \cdot S \cdot \log S$ bits.
  (Per gate: name, predecessors, successor, type, output.)
  Hence, the number of circuits is $\leq 2^{9 \cdot S \cdot \log S}$.
- Set $S = 2^n/10n$.
  Then the number of circuits of size S is at most

$$2^{9 \cdot S \cdot \log S} \leq 2^{9 \cdot \frac{2^n}{10n} \cdot \log \frac{2^n}{10n}}$$
$$= 2^{\frac{9 \cdot 2^n}{10n} \cdot (n - \log 10n)}$$
$$< 2^{\frac{9 \cdot 2^n}{10n} \cdot n} = 2^{\frac{9}{10} \cdot 2^n} < 2^{2^n}.$$

□

19.3 Turing Machines that Take Advice

Goal: • Characterize $P/poly$ in an equivalent way:
via Turing machines that "take advice".

- Technically, the machine has an advice string α_n which it is allowed to use, if the input has size n.

Definition:

Let $t, a : \mathbb{N} \to \mathbb{N}$ be functions.
The class of languages decidable by $t(n)$-time-bounded DTMs with $a(n)$ bits of advice,

denoted by $DTIME(t(n))/a(n)$,
contains every language L so that

there is a sequence of strings $(\alpha_n)_{n\in\mathbb{N}}$ with $\alpha_n \in \{0,1\}^{a(n)}$
and a DTM M satisfying for every $x\in\{0,1\}^n$:

$$M(x,\alpha_n) = 1 \quad \text{iff} \quad x\in L.$$

Moreover, M runs in $O(t(n))$ time.

Example:
Every unary language can be decided with 1 bit of advice.
The advice string for inputs of length n
indicates whether or not $1^n \in L$.

Theorem (Polynomial-time DTMs that take polynomial advice characterize $P/poly$):

$$P/poly = \bigcup_{c,d\in\mathbb{N}} DTIME(n^c)/n^d.$$

Proof:
"$\Rightarrow$" If $L\in P/poly$, then it is computable
by a polynomial-sized circuit family $(C_n)_{n\in\mathbb{N}}$.
We use the description of C_n
as advice string on inputs of length n.
The DTM is the polynomial-time TM that,
on input x of length n and a string representing an n-input circuit C_n
outputs $C_n(x)$.

"$\Leftarrow$" Again Ladner, Cook & Levin.

Assume L is decidable by a polynomial-time DTM M
that has access to an advice family $(\alpha_n)_{n \in \mathbb{N}}$
of size $a(n)$ for some polynomial a.

Represent the computation of $M(x, \alpha_n)$
as a circuit $D_n(x, \alpha_n)$.
We let C_n hard-wire the advice-string α_n,
so C_n only has input x. $\square$