In this lecture we will define the main computational models we will be using & establish the complexity classes that will prominently appear in our course.

Time complexity (boolean)

When studying decision problems in complexity theory (i.e. problems with a YES/NO (or 1/0) answer) each problem P yields a language $L_P \subset \{0,1\}^*$, which is simply the set of strings x s.t. $P(x)=1$. Hence, for decision problems we refer to problems and languages interchangeably.

Definition (NP): $L \subseteq \{0,1\}^*$ is in NP if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM V s.t.

$$x \in L \iff \exists\, y \in \{0,1\}^{p(|x|)} \text{ s.t. } V(x,y) = 1.$$

Definition (coNP): $L \subseteq \{0,1\}^*$ is in coNP if there is a polynomial $p : \mathbb{N} \to \mathbb{N}$ and a poly-time TM V s.t.

$$x \in L \iff \forall\, y \in \{0,1\}^{p(|x|)},\ V(x,y) = 0.$$

If we want to study problems with more than one quantifier, we can generalize NP and coNP as follows:

Definition (Σ_k^P and Π_k^P): $L \subseteq \{0,1\}^*$ is in Σ_k^P if there are polynomials $p_1, \dots, p_k : \mathbb{N} \to \mathbb{N}$ and a poly-time TM V s.t.

$$x \in L \iff \exists\, y_1 \in \{0,1\}^{p_1(|x|)}\ \forall\, y_2 \in \{0,1\}^{p_2(|x|)} \cdots Q_k\, y_k \in \{0,1\}^{p_k(|x|)}$$
$$V(x, y_1, \dots, y_k) = 1$$

where the quantifiers alternate ($\therefore Q_k = \forall$ if k is even and $Q_k = \exists$ if k odd).

Similarly $L \in \Pi_k^P$ if there are polynomials $p_1, \dots, p_k$ and poly-time TM V s.t.

$$x \in L \iff \forall\, y_1 \in \{0,1\}^{p_1(|x|)}\ \exists\, y_2 \in \{0,1\}^{p_2(|x|)} \cdots Q_k\, y_k \in \{0,1\}^{p_k(|x|)}$$
$$V(x, y_1, \dots, y_k) = 1.$$

Definition (Polynomial Hierarchy):

$$PH = \bigcup_{k \geq 0} \Sigma_k^P = \bigcup_{k \geq 0} \Pi_k^P.$$

We can also allow randomness in the verifiers, and if we do so we get a couple more important classes:

Definition (Interactive proofs): for $k \in \mathbb{N}$ (which may depend on input length) we say that language $L \in IP[k]$ if there is a poly-time PTM V that can have a k-turn interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ s.t.

$x \in L \Rightarrow \exists\, P$ s.t. $\Pr\left[\langle V, P\rangle(x) = 1\right] \geq 2/3$ (completeness)

$x \notin L \Rightarrow \forall\, P$, $\Pr\left[\langle V, P\rangle(x) = 1\right] \leq 1/3$ (soundness)

where probabilities are over randomness of V during the protocol.

Definition (AM): for $k \in \mathbb{N}$ (which may depend on input length), $L \in AM[k]$ if there is a poly-time PTM V that only sends random strings to the prover (i.e. it reveals all randomness as it uses it) and can have a k-turn interaction with a function $P : \{0,1\}^* \to \{0,1\}^*$ s.t. V always sends first message and

$x \in L \Rightarrow \exists\, P$ s.t. $\Pr\left[\langle V, P\rangle(x) = 1\right] \geq 2/3$

$x \notin L \Rightarrow \forall\, P$, $\Pr\left[\langle V, P\rangle(x) = 1\right] \leq 1/3$.

Remark: the only randomness that V can use is the one it reveals to P. For instance $AM[2]$ V will send random string r, P sends message m and V deterministically decides upon seeing x, r, m (no extra randomness).

(that is why AM is called public randomness, whereas IP is called private randomness).

Convention: we denote $AM := AM[2]$.

Definition (MA): MA is equivalent to NP with a randomized verifier.

Theorem [Goldwasser-Sipser '87]: for any $k : \mathbb{N} \to \mathbb{N}$ with $k(n)$ computable in poly(n) time

$$IP[k] \subseteq AM[k+2]$$

Theorem [Babai-Moran '88]: $\forall\, k : \mathbb{N} \to \mathbb{N}$ with $k(n) \geq 2$, $AM[2k] \subseteq AM[k]$

Corollary: $AM[O(1)] = IP[O(1)] = AM$.

Proposition: $AM \subseteq \Sigma_3^P$

Space complexity (Boolean)

When studying Turing machines, we can assume there are 3 tapes (input, work, output) (and additional read-once randomness or non-deterministic tapes) and our most important resource is the amount of space used by the machine.

Definition (Space-bounded): Let $s : \mathbb{N} \to \mathbb{N}$. We say that $L \in \text{SPACE}(s)$ if there is a constant c and a TM M deciding L where on inputs of size n M uses $\leq c \cdot s(n)$ work tape cells.

Complexity classes:

$$\text{PSPACE} := \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k)$$

$$\text{EXPSPACE} := \bigcup_{k \in \mathbb{N}} \text{SPACE}(2^{n^k})$$

Boolean circuits & Parallel Complexity

Definition (boolean circuit): a boolean circuit is a DAG where each node is labeled by an input, or the boolean operations (fan-in 2) $\wedge$, $\vee$ or the (fan-in 1) $\neg$ operation.
C has one special node called the output node and if C has n input nodes it computes a function $C : \{0,1\}^n \to \{0,1\}$.

A family of boolean circuits $\{C_n\}_{n\geq 1}$ where $C_n : \{0,1\}^n \to \{0,1\}$ computes a language L if

$$x \in L \cap \{0,1\}^n \iff C_n(x) = 1$$

A family $\{C_n\}_{n\geq 1}$ is said to be **uniform** if there is a TM M which on input 1^n constructs the circuit C_n.
A family is **non-uniform** if there are no restrictions on how these circuits are constructed.

Two measures of complexity for a circuit (and hence for circuit families):
- Size (or sequential complexity): the number of nodes in the circuit
- depth (or parallel complexity): the longest path from an input gate to the output gate.

Given circuit family $\mathcal{C} := \{C_n\}_{n\geq 1}$ we say that
$\mathcal{C} \in \text{DEPTH}(f)$ if $\text{Depth}(C_n) = O(f(n))$
and $\mathcal{C} \in \text{SIZE}(s)$ if $\text{size}(C_n) = O(s(n))$.

Complexity classes:

non uniform:
$$NC^i_{/poly} := \left\{ L \subseteq \{0,1\}^* \;\middle|\; L \text{ decided by } \mathcal{C}=\{C_n\}_n \text{ s.t. } \mathcal{C} \in \text{DEPTH}(\log^i n) \cap \text{SIZE}(\text{poly}(n)) \right\}$$

$$NC_{/poly} := \bigcup_{i\in\mathbb{N}} NC^i_{/poly}$$

and their uniform counterparts

$$NC^i := \left\{ L \subseteq \{0,1\}^* \;\middle|\; L \text{ decided by uniform } \mathcal{C} \in \text{DEPTH}(\log^i n) \cap \text{SIZE}(\text{poly}(n)) \right\}$$

$$NC := \bigcup_{i\in\mathbb{N}} NC^i.$$

Algebraic Complexity (non-uniform)

Definition (Algebraic circuits): an algebraic circuit over ring R is a DAG with nodes labeled by either a ring element (fan-in 0), an input variable (fan-in 0), a ring operation ($+, -, \times$; and inverse gates if R is a field) (fan-in 2). Each node of the graph computes a polynomial (or rational function) in the natural way.

Often times we may designate one of the nodes also as an output node, and in that case we say that a circuit C computes the polynomial (or rational function) computed by the output node.

Given a circuit C we define

$\text{SIZE}(C)$:= # nodes in graph defined by C

$\text{DEPTH}(C)$:= length of largest path from an input node an output gate

Unless stated otherwise, we'll assume that an algebraic circuit doesn't have inversion gates and computes a polynomial.

A family of circuits $\{C_n\}_{n\geq 1}$ computes a family of polynomials $\{f_n\}_{n\geq 1}$ if $C_n(\bar{x}) \equiv f_n(\bar{x})$ (as polynomials)

A family of circuits $\{C_n\}_{n\geq 1}$ is **uniform** if there is a TM M that on input 1^n computes a description of C_n. If there are no restrictions then we say that the family is **non-uniform**.

Complexity classes:

$$\text{AlgP} := \left\{ \{f_n\}_{n\geq 1} \;\middle|\; \exists \{C_n\}_{n\geq 1} \text{ s.t. } C_n \equiv f_n \text{ and } \text{SIZE}(C_n) = \text{poly}(n) \right\}$$

$$\text{VP} := \left\{ \{f_n\}_{n\geq 1} \;\middle|\; \deg f_n = \text{poly}(n) \text{ and } \{f_n\}_{n\geq 1} \in \text{AlgP} \right\}$$

$$\text{FPAR} := \left\{ \{f_n\}_{n\geq 1} \;\middle|\; \exists \text{ uniform } \{C_n\}_{n\geq 1} \text{ s.t. } C_n \equiv f_n \text{ and } \text{DEPTH}(C_n) \equiv \text{poly}(n) \right\}$$

$$\text{FNC} := \left\{ \{f_n\}_{n\geq 1} \;\middle|\; \exists \text{ (log-space) uniform } \{C_n\}_{n\geq 1} \text{ s.t. } C_n \equiv f_n \text{ and } \text{DEPTH}(C_n) = (\log n)^{O(1)}, \text{ SIZE}(C_n) = \text{poly}(n) \right\}$$

Uniform algebraic complexity

Blum-Shub-Smale model: exactly the same as the Turing model, but now computation happens over a ring R with a comparison check (either $=$ or $\geq$) to be able to branch.
Each basic computation is given by polynomials or rational functions (or a check that a certain rational function is $=$ or $\geq$ to 0).
In this setting, a Turing machine is simply a machine over $(\mathbb{F}_2, =)$.
Similarly to the boolean complexity setting, we have definitions of P_R, NP_R (where the comparison check is implicit in the definition of R).

When $R = \mathbb{C}$ (in this case we have $(\mathbb{C}, =)$) the following problem is $NP_{\mathbb{C}}$-complete:

Definition (Hilbert nullstellensatz). the Hilbert Nullstellensatz problem is given by:

$HN_{\mathbb{C}}$:
- input: $f_1, \dots, f_m \in \mathbb{C}[x_1, \dots, x_n]$
- output: YES, if $\exists\, \alpha \in \mathbb{C}^n$ s.t. $f_i(\alpha) = 0$
  NO, otherwise

A note about the input: in the above, we have not specified how the input polynomials are given.
As it turns out, one can consider any of the standard settings for describing a polynomial: dense representation, sparse representation, algebraic circuits (or equivalently straight-line programs).

Theorem [BCSS, Section 5.3, Theorem 1]. $HN_{\mathbb{C}}$ is $NP_{\mathbb{C}}$-complete.