Lecture 23: Recap: Alternation, Polynomial Time Hierarchy, and Parallel Computation

Date: November 14, 2023.

Alternation Turing Machine (ATM) is exactly like a (multi-tape) nondeterministic Turing machine, except that there is a "type" associated with each state. That is, the formal specification of the machine includes a function $\mathsf{type} : Q \to \{\wedge, \vee\}$, where Q is the set of states.

A configuration α is an **and**-configuration, if $\mathsf{type}(q) = \wedge$, where q is the state of α. Similarly, α is an **or**-configuration if $\mathsf{type}(q) = \vee$, where q is the state of α. On input x, for configurations α and β, we say $\alpha \xrightarrow[x]{1} \beta$, the the machine can take one step from α to β.

Acceptance: We will only consider ATMs where every computation on an input x halts. The configurations α of such an ATM are labeled **accepting** if:

- α is a halting, accepting configuration,
- α is an or-configuration and for **some** β such that $\alpha \xrightarrow[x]{1} \beta$, β is (inductively) labeled accepting.
- α is an and-configuration and **every** configuration β such that $\alpha \xrightarrow[x]{1} \beta$, β is (inductively) labeled accepting.

An input x is accepted by ATM M, if the initial configuration of M on x is labeled accepting. The **language** recognized by M ($\mathbf{L}(M)$) is the set of all inputs it accepts.

Time-bounded ATMs: An ATM M is said to be $T(n)$-time bounded if on any input x, **all** computations of M on x halt in $\leq T(|x|)$ steps. ATIME$(T(n))$ is the collection of all decision problems/languages A such that there is a $T(n)$-time bounded ATM M such that $\mathbf{L}(M) = A$.

Space-bounded ATMs: An ATM M is said to be $S(n)$-space bounded if on any input x, the total number of worktape cells used in **any** computation of M on x is at most $S(|x|)$. ASPACE$(S(n))$ is the collection of all decision problems/languages A such that there is a $S(n)$-space bounded ATM M such that $\mathbf{L}(M) = A$.

Alternating Complexity Classes

$$\begin{aligned}
\text{ALOGSPACE} &= \text{ASPACE}(\log n) \\
\text{APTIME} &= \cup_k \text{ATIME}(n^k) \\
\text{APSPACE} &= \cup_k \text{ASPACE}(n^k) \\
\text{AEXPTIME} &= \cup_k \text{ATIME}(2^{n^k})
\end{aligned}$$

Theorem 1. *The following relationships hold. For items other than (a), we assume that $T(n) \geq n$ and $S(n) \geq \log n$.*

(a) ATIME$(T(n)) \subseteq$ ASPACE$(T(n))$ *and* ASPACE$(S(n)) \subseteq$ ATIME$(2^{O(S(n))})$.

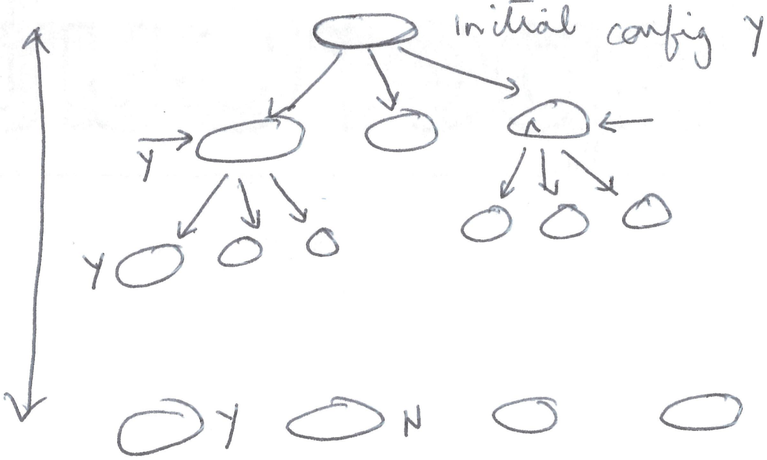
(b) ATIME$(T(n)) \subseteq$ DSPACE$(T(n))$ *and* NSPACE$(S(n)) \subseteq$ ATIME$(S(n)^2)$

(c) ASPACE$(S(n)) \subseteq$ DTIME$(2^{O(S(n))})$ *and* DTIME$(T(n)) \subseteq$ ASPACE$(\log T(n))$

Corollary 2. *The following equivalences hold:* ALOGSPACE = P, APTIME = PSPACE, APSPACE = EXP, AEXPTIME = EXPSPACE.

ASPACE(log n) ⊆ P. $\underline{\text{DTIME}(n^k)} \subseteq \text{ASPACE}(O(\log n)) = \text{ASPACE}(\log n)$

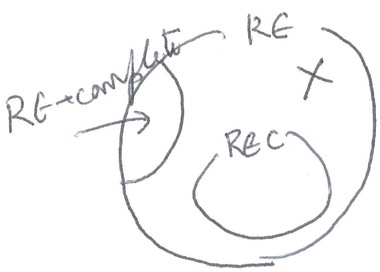
$P \subseteq \text{ASPACE}(\log n)$

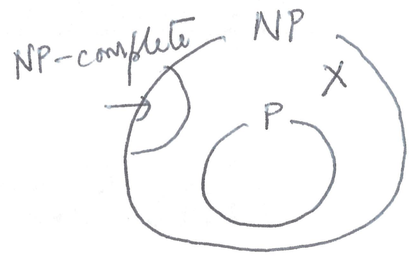


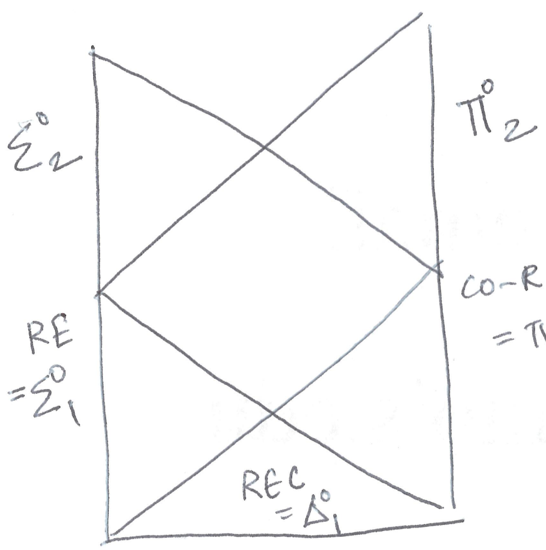
RE — Existential projected Rec languages.

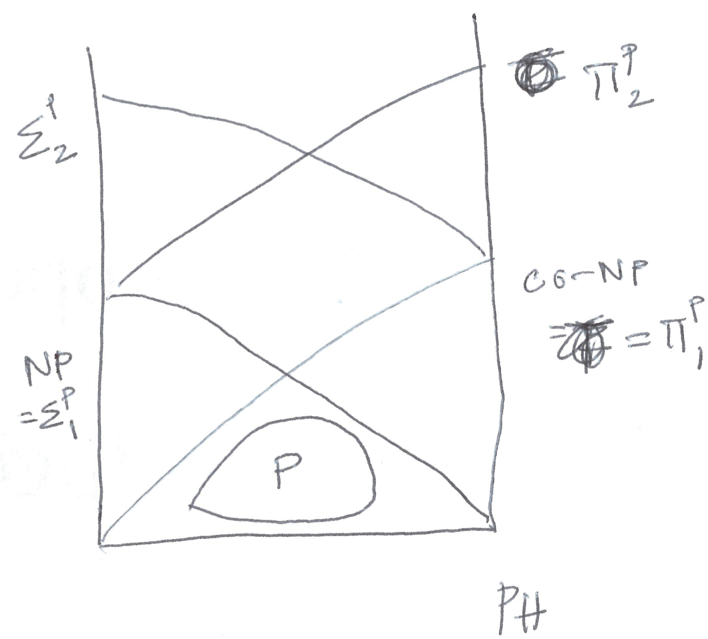
REC

NP — Existential projected Polytime languages.









$A \in \Sigma^0_k$ iff $\exists\ R \in REC$

$A = \{x \mid \exists y_1 \forall y_2 \dots Q_k y_k\ R(x, y_1 \dots y_k)\}$.

$A \in NP$ iff $\exists\ R \in P$. and c s.t

$A = \{x \mid \exists y.\ |y| \leq |x|^c \text{ and } R(x,y)\}$

Bounding Alternations: A **Σ_k-machine** is an ATM such that (a) the initial state is an or-state, and (b) on any input, every computation path, has at most $k-1$ switches between or-configurations and and-configurations.

A **Π_k-machine** is an ATM such that (a) the initial state is an and-state, and (b) on any input, every computation path, has at most $k-1$ switches between or-configurations and and-configurations.

By convention Σ_0 and Π_0-machines are deterministic TMs.

Polynomial Hierarchy:

$$\begin{aligned}
\Sigma_k^{\mathsf{p}} &= \{\mathbf{L}(M) \mid M \text{ is a polynomial-time-bounded } \Sigma_k\text{-machine}\}. \\
\Pi_k^{\mathsf{p}} &= \{\mathbf{L}(M) \mid M \text{ is a polynomial-time-bounded } \Pi_k\text{-machine}\}.
\end{aligned}$$

Thus, $\Sigma_0^{\mathsf{p}} = \Pi_0^{\mathsf{p}} = \mathrm{P}$, $\Sigma_1^{\mathsf{p}} = \mathrm{NP}$, and $\Pi_1^{\mathsf{p}} = \text{co-NP}$.

Proposition 3. *The following identities hold.*

$$\begin{aligned}
\Pi_k^{\mathsf{p}} &= \{A \mid \overline{A} \in \Sigma_k^{\mathsf{p}}\} \\
\Sigma_k^{\mathsf{p}} \cup \Pi_k^{\mathsf{p}} &\subseteq \Sigma_{k+1}^{\mathsf{p}} \cap \Pi_{k+1}^{\mathsf{p}} \\
PH = \bigcup_k \Sigma_k^{\mathsf{p}} &= \bigcup_k \Pi_k^{\mathsf{p}} \subseteq \mathrm{PSPACE} = APTIME
\end{aligned}$$

$PH \neq PSPACE$

Oracle Machines: For a language B, and complexity class $\mathcal{C}$, we define

$$\begin{aligned}
\mathrm{P}^B &= \{\mathbf{L}(M^B) \mid M \text{ is a deterministic oracle TM such that } M^B \text{ runs in polynomial time}\}. \\
\mathrm{NP}^B &= \{\mathbf{L}(M^B) \mid M \text{ is a nondeterministic oracle TM such that } M^B \text{ runs in polynomial time}\}. \\
\mathrm{P}^{\mathcal{C}} &= \cup_{B\in\mathcal{C}} \mathrm{P}^B. \\
\mathrm{NP}^{\mathcal{C}} &= \cup_{B\in\mathcal{C}} \mathrm{NP}^B.
\end{aligned}$$

$\Sigma_1^0 = RE$

$\Sigma_k^0 = RE^{\Sigma_{k-1}^0}$

Theorem 4. *Let* $\mathrm{NP}_1 = \mathrm{NP}$ *and* $\mathrm{NP}_{k+1} = \mathrm{NP}^{\mathrm{NP}_k}$. *Then* $\Sigma_k^{\mathsf{p}} = \mathrm{NP}_k$ *for* $k \geq 1$.

Theorem 5. $A \in \Sigma_k^{\mathsf{p}}$ *if and only if there is a (deterministic) polynomial time computable predicate* R *and constant* c *such that*

$$A = \{x \mid \exists y_1 \forall y_2 \cdots \mathrm{Q} y_k.\ (\bigwedge_{i=1}^{k} |y_i| \leq |x|^c) \wedge R(x, y_1, \ldots y_k)\}$$

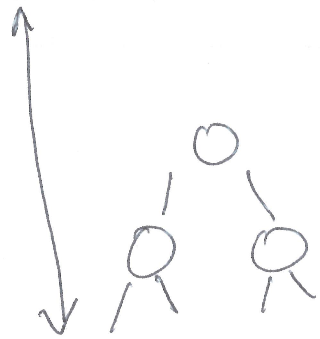
where Q *is* $\exists$ *if* k *is odd, and is* $\forall$ *if* k *is even.*

Bounding Time and Alternation: For a ATM M, let M_k^m be the same machine as M, except on an input x, if a computation takes more than m steps or has more than k alternations between "and" and "or" configurations, M_k^m halts the computation. The last configuration of such an abnormally terminated computation is accepting if it is an and-configuration, and is rejecting if it is an or-configuration.

Define $H_k = \{\langle M, x, 0^m\rangle \mid M_k^m \text{ accepts } x\}$

Theorem 6. H_k *is* Σ_k^{p}*-complete.*

PRAM. — time, # processors.



NC – polynomially many processors in polylogarithmic time.

Boolean Circuits: A *Boolean circuit C with n inputs* is a directed acyclic graph with n vertices of in-degree 0, a single vertex of out-degree 0, and whose internal vertices are all labeled with $\wedge$, $\vee$, or $\neg$. A vertex labeled with $\wedge$, $\vee$, or $\neg$ computes the logical and, or, or negation of its inputs, respectively. We assume that vertices labeled with $\wedge$ or $\vee$ have two children and vertices labeled with $\neg$ have one child. On input $x \in \{0,1\}^n$, the output of C is given by the value of the vertex of out-degree 0 and is denoted by $C(x)$.

The *size* of C is the number of gates in C. The *depth* of C is the length of the longest path from an input vertex to the output vertex.

Solving Problems using Families of Circuits: A *family of circuits $\{C_n\}_{n\in\mathbb{N}}$ of size $S(n)$* is a collection of Boolean circuits where for all n, C_n has n inputs and size at most $S(n)$. A language L is in SIZE($S(n)$) if there is a family of Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ of size $S(n)$ such that for all $x \in \{0,1\}^n$, $x \in L$ iff $\cancel{C_n(x)} = 1$.

$C_{|x|}(x) = 1$

Proposition 7. *There is a language $L \in$ SIZE($O(1)$) which is undecidable.*

Theorem 8. *Let L be an arbitrary language. Then $L \in$ SIZE($O(n2^n)$).*

Uniform Circuit Classes: A family of Boolean circuits $\{C_n\}_{n\in\mathbb{N}}$ is *logspace-uniform* if there is a logspace-bounded Turing machine that outputs the circuit C_n on input 0^n.

NC: A language L is in NC^i if there exists a logspace uniform family of circuits $\{C_n\}_{n\in\mathbb{N}}$ where C_n has poly(n) size, $O((\log n)^i)$ depth, and for all $x \in \{0,1\}^n$, $x \in L$ iff $C_n(x) = 1$.

$$\mathrm{NC} = \bigcup_{i \geq 0} \mathrm{NC}^i .$$

We will prove

Proposition 9. NC $\subseteq$ P.

Open Problem: Is NC = P?