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 type: \( Q \rightarrow \{\land, \lor\} \), where \( Q \) is the set of states.

A configuration \( \alpha \) is an and-configuration, if type(\( q \)) = \( \land \), where \( q \) is the state of \( \alpha \). Similarly, \( \alpha \) is an or-configuration if type(\( q \)) = \( \lor \), where \( q \) is the state of \( \alpha \). On input \( x \), for configurations \( \alpha \) and \( \beta \), we say \( \alpha \xrightarrow{1} \beta \), the the machine can take one step from \( \alpha \) to \( \beta \).

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

- \( \alpha \) is a halting, accepting configuration,
- \( \alpha \) is an or-configuration and for some \( \beta \) such that \( \alpha \xrightarrow{1} \beta \), \( \beta \) is (inductively) labeled accepting.
- \( \alpha \) is an and-configuration and every configuration \( \beta \) such that \( \alpha \xrightarrow{1} \beta \), \( \beta \) 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 \( M \) (\( L(M) \)) 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. \( \text{ATIME}(T(n)) \) is the collection of all decision problems/languages \( A \) such that there is a \( T(n) \)-time bounded ATM \( M \) such that \( L(M) = A \).

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 \( S(|x|) \). \( \text{ASPACE}(S(n)) \) is the collection of all decision problems/languages \( A \) such that there is a \( S(n) \)-space bounded ATM \( M \) such that \( L(M) = A \).

Alternating Complexity Classes

\[
\begin{align*}
\text{ALOGSPACE} & = \text{ASPACE}(\log n) \\
\text{APTIME} & = \bigcup_k \text{ATIME}(n^k) \\
\text{APSPACE} & = \bigcup_k \text{ASPACE}(n^k) \\
\text{AEXPTIME} & = \bigcup_k \text{ATIME}(2^{n^k})
\end{align*}
\]

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

(b) \( \text{ATIME}(T(n)) \subseteq \text{DSPACE}(T(n)) \) and \( \text{NSPACE}(S(n)) \subseteq \text{ATIME}(S(n)^2) \)

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

Corollary 2. The following equivalences hold: \( \text{ALOGSPACE} = \text{P} \), \( \text{APTIME} = \text{PSPACE} \), \( \text{APSPACE} = \text{EXP} \), \( \text{AEXPTIME} = \text{EXPSPACE} \).

\[
\text{ASPACE}(\log n) \subseteq \text{P}.
\]

\[
\text{DTIME}(n^k) \subseteq \text{ASPACE}(\alpha(\log n)) = \text{ASPACE}(\log n^k)
\]

\[
1 \text{P} \subseteq \text{ASPACE}(\log n^k)
\]
RE — Existential projected Rec languages.

REC

NP — Existential projected Polytime languages.

\[
\begin{align*}
\Sigma^0_1 & = \Sigma_1^0 \cap \Pi_2^0 \\
\Sigma^0_2 & = \Sigma^0_1 \\
\Sigma^1_2 & = \Pi^0_1 \\
\Pi^0_2 & = \Sigma^0_1 \\
\Pi^0_1 & = \Sigma^0_1 \\
\Pi^1_2 & = \Pi^0_1 \\
\Sigma^0_2 & = \Sigma^0_2 \\
\Sigma^1_2 & = \Sigma^1_2 \\
\end{align*}
\]

\[
\begin{align*}
A \in \Sigma^0_k \iff & \exists y \in \text{REREC} \\
A = \exists x \left( \forall y_1 \ldots y_k \right) R(x, y_1 \ldots y_k)^2 \\
A \in \text{NP} \iff & \exists R \in \text{REC. and } c \text{ s.t.} \\
A = \exists x \left( \exists y_1 \forall y_2 \right. \\
& \left. |y_1| \leq |x|^{c} \text{ and } R(x, y_1)^3 \right)
\end{align*}
\]
Bounding Alternations: A $\Sigma_k$-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 $\Pi_k$-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 $\Sigma_0$ and $\Pi_0$-machines are deterministic TMs.

Polynomial Hierarchy:

\[
\Sigma_p^p = \{L(M) \mid M \text{ is a polynomial-time-bounded } \Sigma_k \text{-machine}\},
\]

\[
\Pi_p^p = \{L(M) \mid M \text{ is a polynomial-time-bounded } \Pi_k \text{-machine}\}.
\]

Thus, $\Sigma_0^p = P$, $\Sigma_1^p = NP$, and $\Pi_1^p = co-NP$.

**Proposition 3.** The following identities hold.

\[
\Pi_p^p = \{A \mid A \in \Sigma_k^p \},
\]

\[
\Sigma_k^p \cup \Pi_k^p \subseteq \Sigma_{k+1}^p \cap \Pi_{k+1}^p
\]

\[
P \not\subset PSPACE = \text{APTIME}
\]

**Oracle Machines:** For a language $B$, and complexity class $C$, we define

\[
P^B = \{L(M^B) \mid M \text{ is a deterministic oracle TM such that } M^B \text{ runs in polynomial time}\},
\]

\[
NP^B = \{L(M^B) \mid M \text{ is a nondeterministic oracle TM such that } M^B \text{ runs in polynomial time}\},
\]

\[
P^C = \bigcup_{B \in C} P^B.
\]

\[
NP^C = \bigcup_{B \in C} NP^B.
\]

**Theorem 4.** Let $NP_1 = NP$ and $NP_{k+1} = NP^{NP_k}$. Then $\Sigma_k^p = NP_k$ for $k \geq 1$.

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

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

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

Bounding Time and Alternation: For an ATM $M$, let $M^m_k$ 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, $M^m_k$ 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 $H_k = \{(M, x, 0^m) \mid M^m_k \text{ accepts } x\}$

**Theorem 6.** $H_k$ is $\Sigma_k^p$-complete.
PRAM - Time, # processors.

NC - polynomially many processors in poly logarithmic 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 $\land$, $\lor$, or $\neg$. A vertex labeled with $\land$, $\lor$, or $\neg$ computes the logical and, or, or negation of its inputs, respectively. We assume that vertices labeled with $\land$ or $\lor$ 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 $C_n(x) = 1$.

**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($O(n^{2^n})$).

**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 $C_n(x) = 1$.

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

We will prove

**Proposition 9.** NC $\subseteq$ P.

**Open Problem:** Is NC = P?