Lecture 24: Ruzzo's Theorem

Date: November 16, 2023.

All NOT gates are only applied to input

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 $\mathrm{SIZE}(S(n))$ 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 $C_n(x) = 1$.

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. → Output all the gates (number, type) and output all the edges.

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 $\mathrm{poly}(n)$ size, $O((\log n)^i)$ depth, and for all $x \in \{0,1\}^n$, $x \in L$ iff $C_n(x) = 1$.

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

Proposition 1. *Let A and B be $n \times n$ Boolean matrices. There is a logspace-uniform circuit family of $\mathrm{poly}(n)$ size and $O(\log n)$ depth that computes the Boolean matrix product AB.*

Reflexive and Transitive Relations: Recall that a *relation* on a set S is a set $R \subseteq S \times S$. We say R is *reflexive* if for all $a \in S$, we have $(a,a) \in R$. We say R is transitive if for all $a,b,c \in S$, $(a,b) \in R$ and $(b,c) \in R$ implies $(a,c) \in R$.

Reflexive-Transitive Closure: The *reflexive transitive closure* R^* of R is the smallest reflexive and transitive relation containing R. Alternatively, if $R^\dagger$ is a reflexive transitive relation and $R \subseteq R^\dagger$, then $R^* \subseteq R^\dagger$.

Proposition 2. *Let S be a set of size n and let $R \subseteq S \times S$ be a relation on S. There exists a logspace-uniform circuit family of $\mathrm{poly}(n)$ size and $O(\log^2 n)$ depth that computes the reflexive transitive closure R^* of R.*

Bounding Time, Space, and Alternations: The class $\mathrm{STA}(S(n), T(n), A(n))$ is the class of all languages accepted by a Turing machine which is simultaneously $S(n)$-space-bounded, $T(n)$-time-bounded, and uses at most $A(n)$ alternations. A $*$ in a slot means no bound is imposed on that resource. We write $\Sigma A(n)$ or $\Pi A(n)$ to indicate that the alternations should start with $\vee$ or $\wedge$, respectively.

Convention : 0-alternations — ATM is deterministic

We can establish the following relations.

$$\begin{aligned}
\mathrm{L} &= \mathrm{STA}(\log n, *, 0) \\
\mathrm{NL} &= \mathrm{STA}(\log n, *, \Sigma 1) \\
\mathrm{P} &= \mathrm{STA}(\log n, *, *) = \mathrm{STA}(*, n^{O(1)}, 0) \\
\mathrm{NP} &= \mathrm{STA}(*, n^{O(1)}, \Sigma 1) \\
\Sigma_k^{\mathrm{p}} &= \mathrm{STA}(*, n^{O(1)}, \Sigma k) \\
\Pi_k^{\mathrm{p}} &= \mathrm{STA}(*, n^{O(1)}, \Pi k) \\
\mathrm{PSPACE} &= \mathrm{STA}(*, n^{O(1)}, *) = \mathrm{STA}(n^{O(1)}, *, 0)
\end{aligned}$$

$NL = STA(\log n, *, \Sigma 1)$
$\subseteq STA(\log n, *, (\log n)^{O(1)})$
$= NC$
$\subseteq STA(\log n, *, *)$
$= P.$

$NL \subseteq NC \subseteq P$

Theorem 3 (Ruzzo '81). $\mathrm{NC} = \mathrm{STA}(\log n, *, (\log n)^{O(1)})$.

($\Rightarrow$) $A \in NC$. $\exists$ $\{C_n\}_{n \in \mathbb{N}}$ where $size(C_n) \leq poly(n)$ and $depth(C_n) \leq poly(\log n)$ and $\{C_n\}_{n \in \mathbb{N}}$ solves A. and $\{C_n\}_{n \in \mathbb{N}}$ is logspace-uniform.

Goal: Find ATM M that solves A.

Given x
→ Computing $C_{|x|}(x)$

Reduced to computing value of gate d of $C_{|x|}$ on input x.

Assume NOTs are on inputs
- Name of gate d is binary string of length $O(\log n)$ (Stored logspace).
- Compute the type of gate d → Run the logspace TM that computes $C_{|x|}$.

Space needed = memory to remember d + memory to run TM that produces C_n.

(a) If d is an input gate then value is the appropriate bit of x.
(b) If d is a NOT gate then find the input wire to d and flip the appropriate bit of x.
(c) If d is $\wedge$ gate ($\vee$ gate) then we find the two incoming wires into d i.e, (c, d), (c', d) then using $\wedge$-branching ($\vee$-branching) compute value of c and value of c'.

alternations $\leq$ depth $(C_n) \leq$ poly $(\log n)$

($\Leftarrow$) $A \in STA(\log n, *, (\log n)^{O(1)})$. $\exists$ ATM M s.t. $L(M) = A$ and M is log-space bounded and # alternations $\leq (\log n)^{O(1)}$.

Goal: Find $\{C_n\}_{n \in \mathbb{N}}$ that solves A.

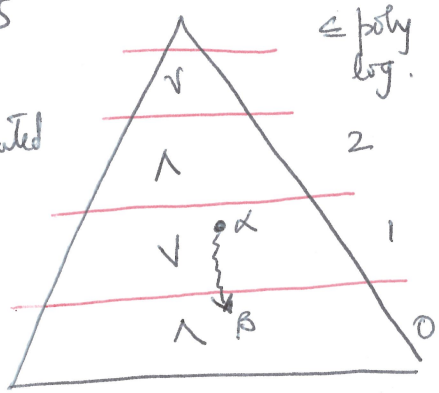
C_n on input x s.t. $|x| = n$, determine if M accepts x.

configurations of $M \leq n^c$ for some c.

Each configuration can be represented by a string of length $O(\log n)$.

$S_x = \{(\alpha, \beta) \mid \alpha \xrightarrow[x]{1}_M \beta$ and $type(\alpha) = type(\beta)\}$

↳ configurations of M on inputs of length n.



↳ Each entry of Boolean matrix S_x can be computed in log-space.

$T_x = \{(\alpha, \beta) \mid \alpha \xrightarrow[x]{1}_M \beta$ and $type(\alpha) \neq type(\beta)\}$

$S_x^* T_x = \{(\alpha, \beta) \mid \exists \alpha_1, \ldots \alpha_n \ \alpha_1 = \alpha, \alpha_n = \beta.$
$\alpha_i \xrightarrow[x]{1} \alpha_{i+1}, type(\alpha_i) = type(\alpha_i) \neq type(\alpha_n)\}$
$i < n$

Computation trees of M on x.

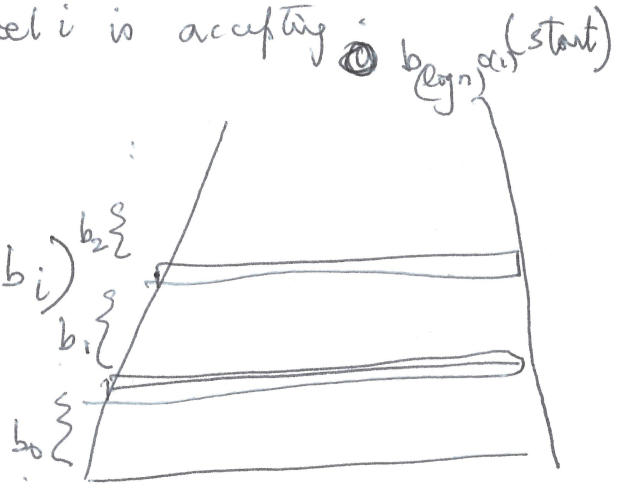
$S_x^* T_x$ can be computed by a circuit of poly size and poly log depth.

$b_i(\alpha) = \begin{cases} 0 & \text{if configuration } \alpha \text{ in level } i \text{ is not accepting} \\ 1 & \text{if configuration } \alpha \text{ in level } i \text{ is accepting} \end{cases}$

b_i — boolean vector of size n^c.

If $i+1$ is an $\vee$, $b_{i+1} = S_x^* T_x b_i$

If $i+1$ is an $\wedge$, $b_{i+1} = \neg(S_x^* T_x \neg b_i)$



To determine if M accepts x

$b_{(\log n)^{O(1)}}(start) = 1$