ZHIYUAN COLLEGE, SHANGHAI JIAOTONG UNIVERSITY

CS396: Computational Complexity Homework III

Zihao Ye

December 30, 2016

EXERCISE 4.10

Consider the QBF:

$$Q = \exists x_1 \forall x_2 \exists \cdots \forall / \exists x_n \phi(x_1, x_2, \cdots, x_n)$$

- $Q = \textit{true}$ implies player 1 has a winning strategy.
- $Q = \textit{false}$ implies

$$\overline{Q} = \forall x_1 \exists x_2 \forall \cdots \exists / \forall x_n \phi(x_1, x_2, \cdots, x_n) = \textit{true}$$

Which means player 2 has a winning strategy.

EXERCISE 4.11

Suppose $L \in \text{coNSPACE}(S(n))$, then $\overline{L} \in \text{NSPACE}(S(n))$, since S is space constructible, by constructing the configuration graph of L on a turing machine, we derive

$$\forall x, x \in L \iff \overline{\text{PATH}}(G(x), s, t) = 1$$

In which the graph size is $2^{S(|x|)}$. Then according to the fact that $\overline{\text{PATH}} \in \text{NL}$, we have

$$L \in \text{NL}\left(2^{S(n)}\right) = \text{NSPACE}(S(n))$$

For the same reason, we derive

$$L \in \text{NSPACE}(S(n)) \Longrightarrow L \in \text{coNSPACE}(S(n))$$

Combine the results above, NSPACE = coNSPACE will be derived.

EXERCISE 5.5

We first prove that

$$\text{ATIME}(T(n)) \subseteq \text{SPACE}(T(n))$$

For every $L \in \text{ATIME}(T(n))$, given an exact input x, by traversing the configuration tree(Assume that $T(n)$ is time-constructible) of $M_L(x)$(which requires additional space no more than $O(T(n))$), we could decide whether $x \in L$.

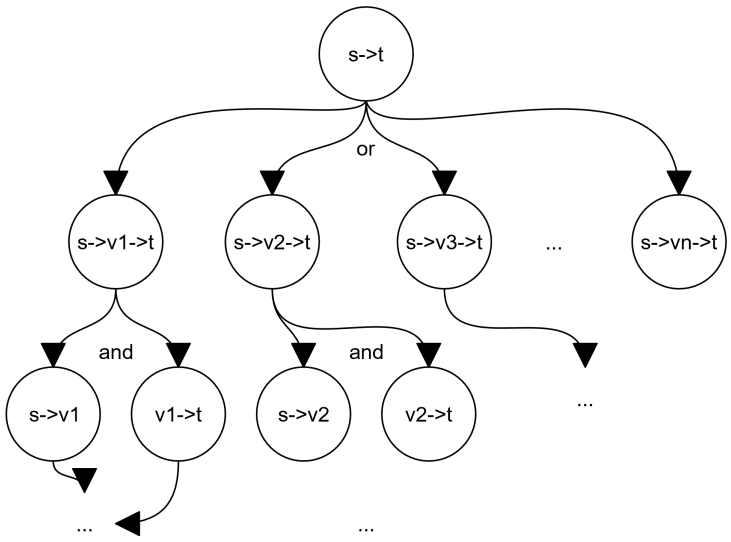
Thus we derive $\text{ATIME}(T(n)) \subseteq \text{SPACE}(T(n))$, which means

$$\text{AP} \subseteq \text{PSPACE}$$

For the other direction, for every $L \in \text{SPACE}(P(n))$, given an input x with length n, considering the **Divide and Conquer** method we used in proving *Savitch Theorem*.

The configuration graph has $M = 2^{P(n)}$ nodes. We need to construct an ATM to decide whether C_{accept} is reachable from C_{start}.

The **Divide and Conquer** algorithm could be described as:



Multiple branch(k of them at all) could be converted to $log(k)$ layers of binary branch structures.

Thus the tree's depth is no more than $O(log^2(M)) = P(n)^2$. We could decide whether $x \in L$ in $\text{ATIME}(P(n)^2) \subseteq \text{AP}$.

Combine the results above, we have

$$\text{AP} = \text{PSPACE}$$

EXERCISE 5.9

I

$$(G,k) \in \text{EXACT INDSET} \iff \forall S \subseteq V, \exists S' \in V, \left(S' \in \text{INDSET}, |S'| = k, S' \subseteq S \Longrightarrow S \notin \text{INDSET}\right)$$

II

$$(G,k) \in \text{EXACT INDSET} \iff (G,k) \in \text{INDSET} \wedge (G,k+1) \notin \text{INDSET}$$

While INDSET $\in$ NP, $\{(G,k) \mid (G,k) \notin \text{INDSET}\} \in$ coNP, thus:

$$(G,k) \in \text{DP}$$

III

For each $L \in$ DP, Suppose $L = L_1 \cap L_2$, while $L_1 \in$ NP and $L_2 \in$ coNP.

We could make a reduction from $L \in$ NP to a 3SAT : ϕ, furthermore an INDSET : (G,k). While a $L \in$ coNP could be reduced to an $\overline{\text{INDSET}}$ problem. Suppose these two reductions are $\phi_1 \to (G_1,k_1)$ and $\phi_2 \to (G_2,k_2)$, respectively.

Such reduction doesn't not suggest any lower bound for largest size of independent set. So we introduce $k_i - 1$ new nodes for each reduction, and add edges from new nodes to all existing nodes. However this augmentation guarantee that largest size of independent size can't be lower than $k_i - 1$.

Construct a new graph $G_1 \times G_2$ as described below:

$$V = V_1 \times V_2$$

$$(\langle v_1, v_2\rangle, \langle v_1', v_2'\rangle) \in E \iff (v_1, v_1') \in E_1 \vee (v_2, v_2') \in E_2$$

While this construction indicates a indset with size $|\text{IND}_1| \cdot |\text{IND}_2|$, and in this case, $|\text{IND}_1| = k_1, |\text{IND}_2| = k_2 - 1$. If $k_1 = k_2$, none of the cases could yield a EXACT_INDSET with size $k_1(k_2-1)$.

After all these conversions, we could decide whether $x \in L$ by

$$\text{EXACT\_INDSET}(G, k_1(k_2-1))$$

EXERCISE 6.3

Suppose B is the set used in the proof of *Baker-Gill-Solovay Theorem* that satisfies $U_B \in \text{NP}^B$ and $U_B \notin P^B$, in which $U_B = \{1^n \mid \exists b \in B, |b| = n\}$.

While in **Exercise 3.3**, it has been proved B could be made exponentially polynomial decidable.

$P \subseteq P^B$, thus $U_B \notin P$. While as it's a unary function, according to **Clamin 6.8**, $U_B \in P_{/poly}$.

EXERCISE 6.16

Suppose A is a $n \times n$ matrix.

Suppose $\lambda_1, \lambda_2, \cdots, \lambda_n$ are n eigenvalues of A,

$$\det(A) = \prod_{i=1}^{n} \lambda_i$$

In order to compute $\det(A)$ in a circuit with $O(log(n))$ layers, we introduce intermediate variables $\{P_i\}$, $\{E_i\}$ defined as follows:

$$P_i = \sum_{k=1}^{n} \lambda_k^i$$

$$E_i = \sum_{k_1, k_2, \cdots, k_i \in \binom{[n]}{i}} \lambda_{k_1} \lambda_{k_2} \cdots \lambda_{k_i}$$

For convenience, let E_0 be 1.

Their relations could be formulated as(*Newton's Identity*):

$$E_i = 1/i \cdot (-1)^{i+1} \sum_{k=1}^{i} P_k E_{i-k}$$

Vieta's formulas implies $\{(-1)^i E_i\}$ are the coefficients of the polynomial $\det(\lambda I - A)$, let c_i be $(-1)^i E_i$, then $(-1)^n c_n = E_n = \det(A)$.

While we could find the relation between c_i and P_i

$$-c_i = 1/i \sum_{k=1}^{i} (-1)^{i-k} P_k c_{i-k}$$

To vectorize this formula, we have to transform it to:

$$-\frac{P_i}{i} = \sum_{k=1}^{i} (-1)^{k-1} \frac{P_{i-k}}{i} c_k$$

Whose matrix form is:

$$\begin{bmatrix} -P_1 \\ -\frac{P_2}{2} \\ -\frac{P_3}{3} \\ \vdots \\ -\frac{P_n}{n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\frac{P_1}{2} & 1 & 0 & \cdots & 0 \\ -\frac{P_2}{3} & -\frac{P_1}{3} & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{P_{n-1}}{n} & -\frac{P_{n-2}}{n} & -\frac{P_{n-3}}{n} & \cdots & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix}$$

Use W to denote the weight matrix, then $P = WC$, $C = W^{-1}P$. Suppose $W = I - L$ (L is a lower-triangular matrix with all-zero diagonal elements), $W^{-1} = I + L + L^2 + \cdots + L^{n-1}$.

To show

$$\{\langle M, k \rangle \mid \det(M) = k\}$$

is actually in NC, we construct a circuit with $O(\log(n))$ layers and $O(Poly(n))$ gates as follows:

- **Addition, Subtraction, Mutliplication and Division Module**: These could be implemented by combining some 'AND' and 'OR' gates, the number of gates is $O(n)$.
- **Sum Module**: Summation of $\{a_i\}$ could be reformulated as:

$$\sum_{i=1}^{n} a_i = \sum_{i=1}^{n/2} a_i + \sum_{i=n/2+1}^{n} a_i$$

  Apply the same technique on the first half and the second half, then we derive a module with $O(\log(n))$ layers and $O(n)$ gates.
- **Matrix Multiplication Module**: Create n^3 modules to parallelly compute $A_{ik}B_{kj}$(n^3 of them). Then use n^2 **Sum Module** to parallely compute AB. This module has $O(\log(n))$ layers and $O(n^3)$ gates.
- **Matrix Power Module**: Use the same technique we used in **Sum Module**, and this could be handled with $O\left(\log^2(n)\right)$ layers and $O(n^4)$ gates.

By combining these modules we could construct a circuit to effectively calculate $\det(M)$ with $O\left(\log^c(n)\right)$ layer and $O(Poly(n))$ gates. Thus $\{\langle M,k\rangle \mid \det(M)=k\} \in \text{NC}$.

ACKNOWLEDGEMENT

Thanks Yurong You for his discussion with me.
Thanks Lequn Chen for his generous help on searching some excellent solutions online.

REFERENCE

1. http://drona.csa.iisc.ernet.in/~chandan/courses/arithmetic_circuits/notes/lec6.pdf
2. http://zoo.cs.yale.edu/classes/cs468/spr15/solutions/HW3-Solutions.pdf