CS452 HW10

Yuting Chen

8.30

Show that E_{DFA} is NL-complete.

Let E_{DNF} is in co-NL. Let $N = \overline{E_{DFA}}$ is NL-complete.

Consider the string accepted by N.

Check the string if accept in log-space and keep track of is current state of N and position in string.

Reduce PATH to N.

Set $\langle G(V,E), s, t \rangle$ of.

Let x be the maximum outdegree of node in Graph G.

Construct DFA $D = \{Q, \Sigma, \delta, q_0, F\}$. let y be new dead state. $\Rightarrow Q = V \cup \{y\}$.

Our alphabet will have $|\Sigma| = x$, that will take $\Sigma = \{1, 2, \cdots x\}$.

As we know we have enough symbols to label outgoing edges from given state.

When state $< x$ outgoing edges, all symbols are not be used in any transition from that state.

Add transition from that state to end state.

Set $q_0 = s$, $F = \{t\}$.

Thus, we can easily know that G has a path from s to t iff DFA D accepts at least one.

$\Rightarrow N$ is NL-complete.

Since, coNL = NL. $N = \overline{E_{DFA}}$

Thus, E_{DFA} is NL-complete

9.7

e. All strings that contain exactly 500 1's.

$\Rightarrow$ All string contains exactly 500 1's and number of 0's ≥ 0.

$\Rightarrow (1)^{500}(0)^n$, for $n \geq 0$.

f. All strings that contain at least 500 1's.

$\Rightarrow$ All strings contains 500 1's or more than 500 1's and the number of 0's ≥ 0.

$\Rightarrow 1^{500} 1^* 0^n$, for $n \geq 0$.

g. All strings that contain at most 500 1s.

$\Rightarrow$ All strings contains ≤ 500 1's and the number of 0's ≥ 0. When it's no empty, combination is equal 500.
$\Rightarrow (1 \cup \varepsilon^{500})\, 0^n$, for $n \geq 0$.

9.9

Since we can filp accept and reject states in a decider.
$\Rightarrow A \in P \Leftrightarrow A^c \in P$
Let $A \in NP$, Assume $NP = P^{SAT}$.
As the same reason, we can get that:
$A \in NP \Leftrightarrow A \in P^{SAT} \Leftrightarrow A^c \in P^{SAT} \Leftrightarrow A^c \in NP \Leftrightarrow A \in coNP$.
Thus, if $P^{SAT} = NP$, then $NP = coNP$.

9.12

Using the method of contradiction and evaluating and compare the space and time-complexity base on different scenarios.
It's not necessary that the polynomial time reductions are linear time reduction.
That can be obtained from $SAT \in TIME(n^k)$ that $NP \subseteq TIME(n^k)$.
If the reduction takes $O(n^l)$
$\Rightarrow$ The problem is reduced to SAT, It takes $O(n^{kl})$, not $O(n^k)$
Therefore, we got that the reduction produce instances of SAT of size $O(n^l)$
Thus, it's contradiction, it's proved.

9.13

As we know, A set of all strings that can be formed is known as language from some Σ^*.
Consider a machine M that decides A in time n^6.
Let M_1 be the another machine that decides pad (A, n^2).
On input w:
- w is in the form of pad $(s, |s|^2)$ for some string $s \in \Sigma^*$.
- If not, reject.

· Otherwise, simulate M on s.

The runtime: $O(|w|^3) + O(|s|^6) = O(|w|^6)$.

Thus, if $A \in TIME(n^6)$, then $pad(A, n^2) \in TIME(n^3)$.

9.14

Using the method of contra position to prove the question.

NEXPTIME $\neq$ EXPTIME $\Rightarrow$ P $\neq$ NP.

Assume that P = NP, then $L \in$ NEXPTIME.

Assume x be a positive integer like $L \in NTIME(2n^x)$.

It show that $Pad(L, 2^{n^x}) \in NP$.

Since $P = NP \Rightarrow Pad(L, 2^{n^x}) \in P$.

So, $L \in TIME(2^{O(n^x)}) \subseteq$ EXPTIME.

Thus, it follow that EXPTIME = NEXPTIME.

Thus, it may be concluded that if EXPTIME $\neq$ NEXPTIME, then P $\neq$ NP.

1

Show that if A is complete for EXPTIME under $\leq_m^P$ reductions, then there is a number $\varepsilon > 0$, such that $A \notin DTIME(2^{n^\varepsilon})$.

If $B \leq_m^P A \in DTIME(S(n))$

$\Rightarrow \exists k \ B \in DTIME(P + 2^{n^k})$

Assume that $A \in \bigcap_\varepsilon DTIME(2^{n^\varepsilon})$

We know that $\exists B \in DTIME(2^n) - DTIME(E(n))$.

$\forall \varepsilon \ B \in DTIME(P + (2^n)^\varepsilon)$

But if $\varepsilon = \frac{1}{k+1}$ $DTIME(P + (2^n)^{\frac{k}{k+1}}) \in DTIME(2^n)$

2

Show that, for all $\varepsilon > 0$, there is a set $A \in DTIME(2^{n^\varepsilon})$ that is complete for EXPTIME.

EXPTIME: $\{M|x|1^m$: M accepts x in time 2^m.

EXPTIME can be reformulated as the space class APSPACE, all problem can be solved by an alternating

Turing Machine in ploynomial space.

We know that $P \subseteq NP \subseteq PSPACE \subseteq EXPTIME \subseteq EXSPACE$.

If an NP complete problem is reduce from L, then L is at most NP-hard. L is less hard or equal in hardness to NP-complete.

Since, any problem in NPC is at least hard as any problem in less hard to NP complete is in NP. No class earsier than NP is separately defined.

By the time hierary theorem and space hierarchy theorem, $P \subsetneq EXPTIME$, $NP \subsetneq NEXPTIME$, $PSPACE \subsetneq EXPSPACE$

EXP is exponential time analogue of P. $\Rightarrow EXP = \cup DTIME(2^n)$

$P \subseteq NP \subseteq EXP$.

Thus. any algorithm takes 2^{t^n} time, $t > 0$ is some fixed constant and n is the input size.

$\boxed{3.}$ Let $A = \{(x, y) : x \in PATH, y \notin PATH\}$. Show that $A \leq_m^{log} PATH$ and $PATH \leq_m^{log} A$.

Conclude that A is NL-complete under $\leq_m^{log}$ reductions.

By the thm, EQREXT is complete for EXPSPACE under $\leq_m^P$.

EQREXT is in PSPACE by

① Given R & Q, convert them to ordinary reg. expression.

② Convert those regular expression to NFAs.

③ Use the PSPACE algorithm to determine it 2 NFA are equivalent.

If $A \leq_P B$ and $B \in NP$, then $A \in NP$.

The observation is that we can construct a certifier for A by com-posing the polynomial time reduction map and the certifier for.