CS 332 – Spring 2021, Assignment 8

Colaborators: None

Answer 1

(a) Theorem 9.3 of Sipser states that "For any space constructible function $f : N- > N$, a Language A exists that is decidable in O(f(n)) space but not in o(f(n)) Space." From here Sipser takes its corollary: "For any two functions $f_1, f_2 : N- > N$, where $f_1(n)$ is $o(f_2(n))$ and f_2 is space constructible, $SPACE(f_1(n)) \subset SPACE(f_2(n))$."

With this in mind, we know for a fact that n is $o(n^2)$, as such, per the corollary stated above: $SPACE(n) \subset SPACE(n^2)$. However, we should note that this containment is strict, as $SPACE(n) \subset SPACE(n^2)$, and not $SPACE(n) \subseteq SPACE(n^2)$, this means that there will be a language A in $SPACE(n^2)$ which isn't in $SPACE(n)$, which means that $SPACE(n^2) \nsubseteq SPACE(n)$.

(b) We know $P = \bigcup_k TIME(n^k)$. We also know that $2^k < n^k$ when $n > 1$ and $k > \frac{n \times \ln 2}{\ln n}$ or when $0 < n < 1$ and $\frac{n \times \ln 2}{\ln n}$, since there are natural numbers for both those k, that means that there is some $n^p \in P$ for which $2^k < n^p$, thus $P \subseteq 2^k$.

(c) We know that $EXP = \bigcup_k TIME(2^{n^k}) = TIME(2^n) \bigcup TIME(2^{n^2}) \bigcup TIME(2^{n^3}) \bigcup ...$

From here, we again know from the corollary of the hierarchy theorem as stated above: "For any two functions $f_1, f_2 : N- > N$, where $f_1(n)$ is $o(f_2(n))$ and f_2 is space constructible, $SPACE(f_1(n)) \subset SPACE(f_2(n))$.".

Furthermore, we know that 2^n is $o(2^{n^2})$, so $SPACE(2^n) \subset SPACE(2^{n^2})$, and as we seen before $SPACE(2^n) \subset SPACE(2^{n^2})$ is not $SPACE(2^n) \subseteq SPACE(2^{n^2})$, thus $SPACE(2^{n^2}) \nsubseteq SPACE(2^n)$. Which we can extrapolate to mean $EXP \nsubseteq SPACE(2^n)$.

(d) Since we know that $P \subseteq TIME(2^n)$, then $P = TIME(2^n)$. Since we have also proven that $EXP \nsubseteq SPACE(2^n)$ then $EXP \neq SPACE(2^n)$, thus $P \neq EXP$.

Answer 2

(a) $P = \bigcup_k TIME(n^k)$, if we take concatenation of two languages to be $L_{CONCAT} = \{xy | x \in L_1 \wedge y \in L_2\}$, and we assume that there are TMs T_1 to recognize L_1 and T_2 to recognize L_2, with both of these languages being in P; then to recognize L_{CONCAT} we could just run xy firstly on T_1 and before the end we'd delete every cell that was used in that TM, leaving only y, and run that on T_2.

Since both of these TMs run in P, then $T_1 = TIME(n_1^{k_1})$ and $T_2 = TIME(n_2^{k_2})$, thus $T_{CONCAT} = TIME(n_1^{k_1}) + TIME(n_2^{k_2}) = TIME(n_1^{k_1} + n_2^{k_2})$. Which is still in P, thus P is closed under concatenation.

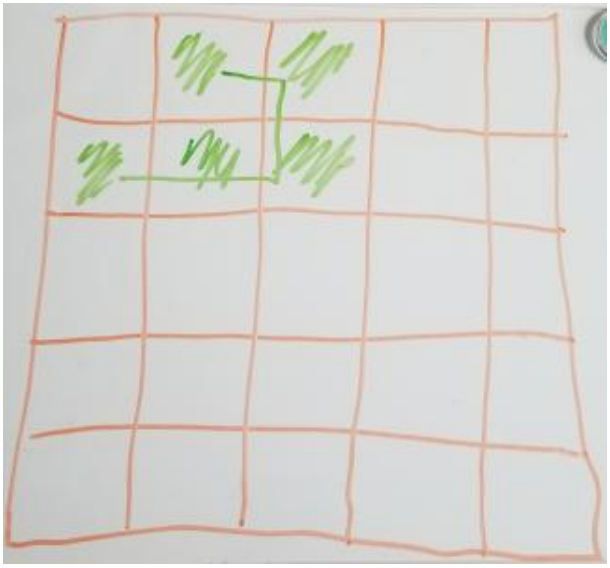
(b) $NP = \bigcup_k NTIME(n^k)$, if we take concatenation of two languages to be $L_{CONCAT} = \{xy | x \in L_1 \wedge y \in L_2\}$, and we assume that there are TMs T_1 to recognize L_1 and T_2 to recognize L_2, with both of these languages being in NP; then to recognize L_{CONCAT} we could just run xy firstly non-deterministically on T_1 and before the end we'd delete every cell that was used in that TM, leaving only y, and run that non-deterministically on T_2.

Since we know that both T_1 and T_2 can decide the languages L_1 and L_2 in $NTIME(n^k)$, then we know that at most, $L_{CONCAT} = NTIME(n_1^{k_1} + n_2^{k_2})$, which is in itself in $NTIME$, thus NP is closed under concatenation.

Answer 3

(a) Yes.

(b) There are two distinct pairs of indexes for which there is a stabled folding, $i = 0, j = 4$, both $s_0 = s_4 = 1$, and the cells are adjacent on the grid; and $i = 3, j = 4$, both $s_3 = s_4 = 1$, and the cells are adjacent on the grid.



(c) SF can be decided by a non-deterministic TM T_1 in polynomial time. This TM is:

"T_1 = On input $\langle s, d \rangle$:

1. Non-deterministically create all possible grids
2. For all grids check all possible distinct pairs i, j where $s_i = s_j = 1$ and $i < j$, count those who are stable
3. If at least one of the possible grids has more than or exactly d stable pairs accept, else reject."

Although this algorithm is wildly inefficient, all branches still run in NP, since what each branch will do is create an individual grid, which takes $TIME(k)$; then check all possible pairs, which are at most k^2; and finally accept/reject, which runs in $O(1)$ for each branch.

There are at most k starting positions, and then there are 4^{k-1} grids for each starting position, totalling $k \times 4^{k-1}$ branches, however each branch runs in $TIME(k^2)$, thus T_1 runs in $NTIME(n^2)$.

Answer 4

Our certificate for this verifier will be a list of states. The Verifier itself will be a TM V:

$V =$ On input $\langle w, c \rangle$:

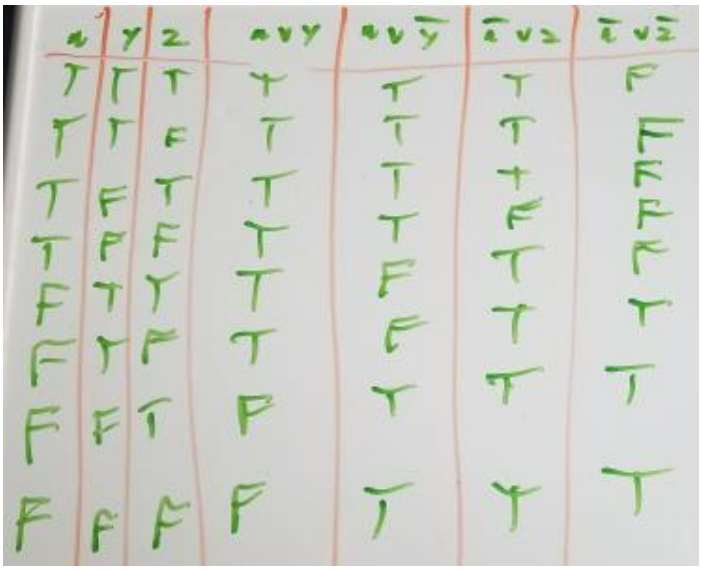
1. If $|c| > |w|^2$, or if $c_1 \neq s_0$, or if c_k isn't an accept state, then reject.
2. Loop through all the states in c and for each c_i:
    1. If you can't go from c_i to c_{i+1} using w_i then reject, else continue
3. If all checks pass, accept, else reject.

All of these checks are both deterministic, and can run in at most $O(|c|)$, since the only loop is through the states in the certifier, thus it's polynomial. In addition, this works because given a string and a possible path, we can check if: a. the String can go through that path; and b. if the path starts at the start state, and ends in an accept state. If so than that string can lead to an accept state. Furthermore, since we know that NFA's have a finite length, and that if an NFA N accept w it does it in $|w|^2$ states, thus we can also limit the size of the certificate, hence getting a deterministic and functional verifier.

Answer 5

(a) It is satisfiable, by the assignment $x = 1$, $y = 1$, $z = 0$.

(b) It is not satisfiable regardless of the assignment, as we can see from the truth-table below:



(c) We know for a fact, as seen in class, that SAT is in NP. Assuming that the TM that verifies SAT is T_1, then we can construct another TM to decide $XSAT$ as below:

"T_2 = On input $\langle \varphi_1, \varphi_2 \rangle$:

1. Non-deterministically guess all possible combinations of $c = c_1, c_2, ..., c_n \in \{0, 1\}^n$
2. For each run T_1 on $\langle \varphi_1, c \rangle$ and $\langle \varphi_2, c \rangle$
3. If one accepts, and the other reject, accept, else reject
4. If none of the combinations accept, reject, else accept."

There will be 2^n branches, however, each branch will complete it's computations in NP, since we know that if $SAT \in NP$, then $T_1 \in NP$. Thus $XSAT \in NP$.