Formal Languages and Automata (CS452) - Homework Assignment #6

Hari Amoor, NetID: hra25

March 26, 2020

Problem 7.6, Claim: The complexity class P is closed under union, concatenation, and complementation.

Proof. Let X, Y be languages in P. We show that $X \cup Y, XY, X^{\complement} \in P$.

We know that X, Y can be recognized with complexity $\mathcal{O}(p(n)), \mathcal{O}(q(n))$ respectively, where each p, q are polynomials in n. Trivially, $X \cup Y$ can each be recognized with complexity $(p + q)(n)$, which is polynomial in n.

Now, we show that concatenation is closed in P. Let M, N be deterministic Turing machines that recognize each X, Y respectively in polynomial-time, i.e. with complexity $p(n), q(n)$ respectively. We know that $w \in XY$ iff $w = xy$ for some $x \in X, y \in Y$. Suppose $w = w_1 w_2 \dots w_n$. For each $i \in 1, 2, \dots, n$, use each M and N to see whether $w_1 \dots w_i \in X$ and $w_{i+1} \dots w_n \in Y$; return true iff this holds for any such i. This algorithm can be simulated with time-complexity $\mathcal{O}(n \cdot (p + q)(n))$, which is polynomial in n, as required.

Finally, we show that $X^{\complement}$ can be recongized with polynomial time-complexity as well. Let M be a deterministic Turing machine that recognizes X in polynomial-time. Define the Turing machine M', which accepts a word iff M doesn't. M' recongizes $X^{\complement}$ in polynomial-time; thus, $X^{\complement} \in P$.

Thus, the claim holds. $\square$

Problem 7.7, Claim: The complexity class NP is closed under union and concatenation.

Proof. Let X, Y be languages in NP. We show that $X \cup Y, XY \in NP$.

Let M, N be non-deterministic Turing machines that recongize each X, Y respectively in polynomial-time. We supply polynomial-time algorithms to recognize $X \cup Y$ and XY with machines M, N.

For a word w, run M on w. If M recognizes w, then return true. Otherwise, run N on w, and return true iff N recognizes w. This algorithm has complexity $\mathcal{O}((p+q)(n))$, where $p(n), q(n)$ are each polynomial upper-bounds for the complexity of the algorithms simulated by M, N. Since this is polynomial in n, it must be true that NP is closed under union.

To show that XY is in NP, we define a non-deterministic Turing machine K that recognizes it in polynomial time. On input w, the machine K non-deterministcally executes the decomposition $w = w_1 w_2$, and returns true iff $w_1 \in X$ and $w_2 \in Y$. Clearly, K runs with time-complexity $\mathcal{O}(\max(p(n), q(n)))$, which is polynomial in n; thus, the claim holds. □

Problem 7.15, Claim: P is closed under the Kleene star operator.

Proof. Let L be a language in P. We show that $L^\star$ is also in P.

Suppose the Turing machine M recognizes L. We define a deterministic Turing machine M' that recognizes L^{star}. On input $w = w_1 w_2 \dots w_n$, M' should do the following:

For each $j \in 1, 2, \dots n$, use M to see whether each each substring of the form $w_i \dots w_{i+j}$ is in L; return true iff, for some j, all such substrings are recognized by M. This algorithm runs in time-complexity $\mathcal{O}(n^2 p(n))$, which is polynomial in n as required. □

Problem 7.18, Claim: If $P = NP$, then every non-trivial language in P is NP-complete.

Proof. Let A, B be languages in P s.t. A is arbitrary and B is non-trivial. There must exist strings $x \in B, y \notin B$. The reduction from A to B, which suffices to prove the claim, is as follows:

Check in polynomial-time if $w \in A$. If so, return x; otherwise, return y. This reduction is in polynomial-time and holds for all non-trivial B.

Thus, the claim holds. $\square$

Problem 7.21b, Claim: $LPATH$, as defined, is NP-complete.

Proof. We assume that the Hamiltonian path problem for undirected graphs is NP-complete.

First, it is trivial that $LPATH$ is in NP. Next, we provide a reduction from $HAM - PATH$ to $LPATH$ with the following Turing machine F:

On input (G, a, b), output (G, a, b, k), where k is the number of vertices in G.

If $(G, a, b) \in HAM - PATH$, then G contains a Hamiltonian path of length k from a to b, so $(G, a, b, k) \in LPATH$. Conversely, if $(G, a, b, k) \in LPATH$, then G contains a path of length k from a to b. However, since G only has k nodes, this path must be Hamiltonian; it follows from this that $(G, a, b) \in HAM - PATH$.

This suffices to show that $LPATH$ is NP-complete. $\square$

Problem 7.38, Claim: If $P = NP$, then there exists a polynomial-time algorithm that produces a satisfying assignment to a given satisfiable boolean formula.

Proof. We assume that $P = NP$ for this problem.

There must exist a Turing machine D that solves the SAT problem in polynomial-time. Define the Turing machine B that does the following:

On input ϕ, where ϕ is a boolean formula of variables $x_1, x_2, \ldots, x_k$, run D on ϕ; if ϕ is not satisfiable, then reject it. Otherwise, for $i \in 1, 2, \ldots, k$, replace all instances of the variable x_i with $1 \in \Sigma$, and simulate D on the formula obtained therein. If D accepts this, then fix $x_i = 1$; otherwise, fix $x_i = 0$.

This runs in time-complexity $O(kp(n))$, where $p(n)$ is a polynomial upper-bound for the complexity of D. This is polynomial in n, so the claim holds. □

Problem 7.43, Claim: For a \$CNF\$-formula ϕ with m variables and c clauses, you can construct an NFA with $\mathcal{O}(cm)$ states that accepts all non-satisfying assignments, represented as boolean strings of length m. Furthermore, $P \neq NP$ implies that NFAs cannot be minimized in polynomial-time.

Proof. Let N be the required NFA. The algorithm detailed below constructs it in polynomial-time:

On input ϕ, pick each of the c clauses non-deterministcally and read the input of length m. Accept the input iff it does not satisfy the clause.

This NFA recognizes all non-satisfying assignments with $\mathcal{O}(cm)$ states as required. Furthermore, it is also constructed in polynomial-time.

Finally, run the algorithm for minimization of an NFA on N to obtain a new NFA N'. Reject ϕ iff N' contains exactly one state and accepts all binary strings. This yields a polynomial-time algorithm for $3 - SAT$, which implies that $P = NP$.

By contraposition, $P \neq NP$ implies that N cannot be minimized in polynomial-time, as required. □