CS 501: Formal Language Theory **Spring 2021**

Final Solutions

Note: LaTeX template courtesy of UC Berkeley EECS dept.

1. **(10 × 2 points) Unjustified True/False Questions.**

   (a) FALSE. Any w that has two 0s or 1s in a row is in DOUBLED-SUBSTRING. Any string of length ≥ 4 without 00s or 11s must contain 0101 or 1010 and thus also is in DOUBLED-SUBSTRING. So DOUBLED-SUBSTRING is co-finite and therefore regular.

   (b) TRUE. Any NFA with at most 10 states has an equivalent DFA with at most $2^{10} = 1024$ states. We can fill the states out to exactly 2021 states by adding unreachable dummy states.

   (c) TRUE. Take a DFA D for L, and create an NFA N for $\mathsf{expand}(L)$ by adding a self-loop labeled Σ to every state.

   (d) TRUE. Let P be a PDA for L. We make a new PDA P' for $\mathsf{contract}(L)$. We want P' to read the letters of u while it is simulating P on the letters of v, where u is a subset of the letters of v. So P' can either read a letter of its input, processing P for it, or nondeterministically "imagine" a new letter without reading any input, processing P for this letter. If P' accepts, the string that it simulates for P is the string v, and the string u that it actually reads and accepts is obtained by deleting letters of v.

   (e) FALSE. Let $L(G) = \{a^n b a^n : n \geq 0\}$, which is clearly a CFL (e.g. $S \to aSa|b$). Let w be a sufficiently long string in $L(G)$. No matter how we pick v and y, if both of them are simultaneously pumped with different sizes, there will be violations of $L(G)$ everywhere.

   (f) TRUE. Given G and w, note that $L(G)$ is a CFL and $w\Sigma^*$ is regular, so their intersection is a CFL. We can construct the corresponding PDA, and test whether this PDA accepts any strings at all. Each of these steps is decidable.

   (g) FALSE. Given any TM, M is Turing-recognizable by definition, so this language is the set of all TMs. Rice's Theorem does not apply because this is not a *non-trivial* property of languages.

   (h) TRUE. Let c be the number of letters in W. For any fixed n, there are $(c+1)^{n^2}$ possible candidates for the puzzle solution we are asking for. Each one can be written in $O(n^2)$ bits. In PSPACE, we can try every candidate solution, writing down one of them at a time, and it should be clear that with no other memory than the fixed word list and $O(n^2)$ bits to count the squares, we can test whether a candidate puzzle follows the rules and count the black squares in it.

   (i) TRUE. If n is large enough, we can fit every word in the list once, and we are now forbidden to repeat them. So the fewest number of black squares we can have is $n^2 - w$, where w is the sum of the lengths of the words in the list. So if $k < n^2 - w$, the answer is NO. However, if $k \geq n^2 - w$, the answer is YES. So for sufficiently large n, we're comparing two integers, which is in P.

   (j) FALSE. This is the same as STRONGLY-CONNECTED, which is NL-complete. By the Space Hierarchy Theorem, NL is a strict subset of PSPACE, so PAIRWISE-ACCESSIBLE cannot be PSPACE-complete.

2. **(5 × 6 points) Justified True/False Questions.** For each of the following questions, indicate whether it is TRUE or FALSE, *and provide a brief justification (i.e. either a proof or a counterexample).*

   (a) TRUE. 2021SAT is in NP, as the witness is the set of satisfying assignments. Given a CNF formula φ, add eleven dummy variables $y_1, \ldots, y_{11}$ to it, and consider $\varphi' = \varphi \wedge \bigwedge_{1 \leq i \leq 11}(y_i \vee \overline{y_i})$. Observe that if φ is satisfiable, then φ' has at least 2^{11} satisfying assignments, and so is in 2021SAT. Conversely, if φ' has many satisfying assignments, each of them corresponds to a satisfying assignment for φ.

(b) TRUE. Mark all sinks as LOSING, then iteratively mark nodes WINNING if they lead to some LOSING node, and LOSING if they lead to all WINNING nodes. At the end of n passes, nodes will be unmarked (DRAWING), or marked WINNING or LOSING, so the game is determined. The total time taken is $O(nm)$.

(c) TRUE. If such an X existed, choose an arbitrary PSPACE language Z, and note that $Z \leq_P X$. Because X is in NP, so Z is in NP as well. Since Z was arbitrary, this means PSPACE $\subseteq$ NP, which therefore implies NP = PSPACE.

(d) TRUE. Of course, ACYCLIC-REACH is in NL, as it is a special case of reachability. To show hardness, take an arbitrary NL machine M, equip it with a polynomial-time clock as we have seen in class, that keeps track of the number of steps in the calculation using $O(\log n)$ portion of the work tape, and rejects once the number of steps is enough for us to know we are looping. This creates an acyclic graph, so now the rest of the usual REACH proof goes through.

(e) FALSE. The problem is undecidable, by reduction from $\mathsf{ALL}_{\mathrm{CFG}}$. Given an arbitrary grammar G', construct a grammar G that accepts Σ^*, and note that $\langle G' \rangle \in \mathsf{ALL}_{\mathrm{CFG}}$ iff $\langle G, G' \rangle \in$ GRAMMAR-SUBSET.

3. **(5XC + 5 points) Product Placement.**

(a) The original entries are in $[-2^n, 2^n]$, and so they are at most n bits long. In one product with entries of at most n bits, every entry of the product is obtained by multiplying two numbers, and then adding n of these products together. We will get at most $2n$ bits in each product, then at most $2n + \log n$ bits in the new entry, as the maximum possible value of this sum is $n \cdot 2^{2n} = 2^{2n+\log n}$. We can coarsely upper bound this by $4n$. After $\log n$ many iterations of this, we are left with at most $4^{\log n} \cdot n = n^3$ bits in each final entry.

(b) We argued that a single matrix product requires multiplying two numbers, and then adding n of these products together. From lecture, we know these operations are both NC^1, so the entire process can be done in NC^1 as well. To get the n-way product, we create a binary tree of 2-way matrix products. The resulting depth of the tree is $\log n$ per 2-way matrix multiplication, and so the final depth is $O(\log^2 n)$. The size remains polynomial in n, and so this problem is in NC^2.

4. **(10 points) Symbolic Gesture.** To show that LETTER-REACHABILITY is Turing-recognizable, just observe that we can dovetail M on all of Σ^*, and keep track of the symbols M writes on its tape. As soon as the symbol a appears, we accept.

   To show LETTER-REACHABILITY is undecidable, at the outset, we observe that Rice's Theorem is not applicable. Instead, we reduce from A_{TM}. Given an instance $\langle M, w \rangle$ of A_{TM}, we construct a TM M' that on any input x, ignores the input and simulates M on w. We modify the internal simulation of M so that immediately before entering the accept state, it writes a symbol σ (that does not appear anywhere else in the tape alphabet), and then enters the accept state. This is clearly constructible, and note that $\langle M, w \rangle \in \mathsf{A}_{\mathrm{TM}}$ if and only if $\langle M', \sigma \rangle \in$ LETTER-REACHABILITY.

5. **(10 points) Separate Checks.** SEPARATED-EDGES is trivially in NP, as the certificate is the set of k pairwise separated edges.

   To show SEPARATED-EDGES is NP-hard, we reduce from INDEPENDENT-SET. Given an instance $\langle G, k \rangle$ of INDEPENDENT-SET, we construct the following graph G'. Start from G, and for each vertex v_i of G, we add a different vertex v_i' and connect it to v_i. We claim that $\langle G, k \rangle \in$ INDEPENDENT-SET if and only if $\langle G', k \rangle \in$ SEPARATED-EDGES.

   If G has an independent set of size k, then the new edges attached to those vertices form k pairwise separated edges in G'. Conversely, if G' has k pairwise separated edges, pick an endpoint in G from each of them; note that this is always possible, as each edge in G' has at least one endpoint in G; furthermore, we pick k distinct vertices in this way, as the original edges are separated. Finally, notice that these k vertices form an independent set in G. Clearly the construction is doable in polynomial time, completing the proof.

6. **(10 points) Accept the Emptiness.** Of course, $\mathsf{A}_{\mathrm{CFG}} \in \mathrm{P}$, by the CYK algorithm discussed in class.

   To show that $\mathsf{A}_{\mathrm{CFG}}$ is P-complete, we will reduce from $\overline{\mathsf{E}_{\mathrm{CFG}}}$ (and use the closure of P under complementation). Take an arbitrary instance $\langle G \rangle$ of $\overline{\mathsf{E}_{\mathrm{CFG}}}$, and go through all the rules, converting each terminal symbol that ever appears into ε. Call the resultant grammar G'. Then, we claim $\langle G \rangle \in \overline{\mathsf{E}_{\mathrm{CFG}}}$ if and only if $\langle G', \varepsilon \rangle \in \mathsf{A}_{\mathrm{CFG}}$. This should be obvious: if G parses any string, the same sequence of derivations in G' yields ε; conversely, if G does not parse any string, then G' cannot parse ε either. The reduction takes a linear sweep through the read-only input to change terminal symbols, and is therefore logspace.

7. **(6 + 9 points) A Resolved Issue.**

   (a) We can easily see that $\mathsf{TRIGRAM\text{-}SUBSTITUTION} \in \mathrm{NPSPACE}$, by nondeterministically guessing a sequence of S-moves on the input word u, and checking whether we end on v after applying each move legally. Using PSPACE = NPSPACE, we are now done.

   (b) To show PSPACE-hardness, start from an arbitrary PSPACE machine M, and an arbitrary string w. We can now modify the machine M to clean up its tape before accepting, and construct rules of the form $cqa \to cbr$ for each transition function of the form $\delta(q, a) = (r, b, R)$, and $cqa \to rcb$ for each function of the form $\delta(q, a) = (r, b, L)$. Note that these trigram substitutions capture precisely the legal moves of M. Let the set of rules constructed in this way be called S. Finally, let $z = q_{\mathrm{accept}}\#^{|w|-1}$, where $\#$ denotes the empty tape symbol. Then, observe that M accepts w if and only if $\langle w, z, S \rangle \in \mathsf{TRIGRAM\text{-}SUBSTITUTION}$. Specifying the construction clearly takes polynomial time, so we are done.