CSE396, Spring 2018 **Final Exam** **May 15, 2018**

One notes binder allowed, no Internet, closed neighbors, 170 minutes. The exam has **seven** problems and totals 240 pts., subdivided as shown. *Show your work*—this may help for partial credit. *Please write in the exam books only*—if you need more paper you may ask. Notation and terms are as in the text, except for the following alternative terminology:

- *Turing-recognizable* language and *c.e. or r.e.* language are synonyms. The class of such languages is denoted by RE.
- *Decidable* language and *recursive* language are synonyms. The class of such languages is denoted by REC.
- Angle brackets $\langle M \rangle$ denote the encoding of a Turing machine M as a string over the alphabet ASCII, which can be further coded over $\Sigma = \{0, 1\}$. Fine details of such encodings are not important, and *languages using such encodings may be assumed not to be CFLs.* Similar remarks apply to the use of (e.g.) $\langle P, w \rangle$ to encode a program P and an input w as a single string.
- The notation $P(x)$ means "the computation of P on input x."

As usual, x^R stands for the reversal of the string x, and $\#c(x)$ stands for the number of times the character c occurs in x. For any character c and number n, c^n stands for a run of n consecutive c's. The complement of a language A is denoted by $\tilde{A}$, difference of sets A and B by $A \setminus B$, and ϕ stands for a Boolean formula. Alphabets default to being $\Sigma = \{0, 1\}$ unless otherwise specified.

You may cite theorems and facts that were covered in lectures and/or text without further proof, so long as the cited item is clearly stated and your use of it is clear. You may refer to the following (not exhaustive!) list of languages and their classifications:

$\{ a^n b^n : n \geq 0 \}$	DCFL, but not regular
$\{ x \in \{ a, b \}^* : x = x^R \}$	CFL and co-CFL, but not a DCFL
$\{ a^n b^n c^n : n \geq 0 \}$	Co-CFL and in P, but not a CFL
$\{ a^m b^n a^m b^n : m, n \geq 0 \}$	Co-CFL and in P, but not a CFL
$\{ ww : w \in \{ a, b \}^* \}$	Co-CFL and in P, but not a CFL
$\{ \langle G, w \rangle : G$ is a CFG and $w \in L(G) \}$	A_{CFG}: In P but not a CFL (or co-CFL).
$\{ x\#y\#z : x = y \ \wedge \ y \neq z \}$	In P, but not CFL or co-CFL
$\{ \phi : (\exists a \in \{ 0, 1 \}^n)\ \phi(a) = 1 \}$	SAT: In NP, believed not in P
$\{ \phi : (\forall a \in \{ 0, 1 \}^n)\ \phi(a) = 1 \}$	TAUT: In co-NP, not in NP unless NP = co-NP
$\{$ poly-time NTMs $N : \langle N \rangle \notin L(N) \}$	D_{NP}: Decidable but not in NP
$\{ \langle M, w \rangle : M$ is a TM and accepts $w \}$	A_{TM}: R.e. but not co-r.e.
$\{ \langle M \rangle : M$ is a TM and does not accept $\langle M \rangle \}$	D_{TM}: Co-r.e. but not r.e.
$\{ \langle M \rangle :$ for all inputs x, $M(x)$ halts $\}$	TOT: Neither r.e. nor co-r.e.
$\{ \langle M \rangle : L(M) = \Sigma^* \}$	ALL_{TM}: Neither r.e. nor co-r.e.

(1) (50 pts.)

Classify each of the following languages $L_1, \ldots, L_{10}$ over $\Sigma = \{0, 1\}$ according to whether it is

(a) regular;

(b) a DCFL but not regular;

(c) a CFL but not a DCFL;

(d) in P but not a CFL;

(e) decidable but definitely not in P or at least NP-hard;

(f) c.e. but not decidable;

(g) co-c.e. but not c.e.; or

(h) neither c.e. nor co-c.e.

You need not justify your answers, but *brief* justifications may help for partial credit. With $r = 1^*01^*01^*$, the predicates for L_1 through L_4 are equivalent to, respectively, $L(M) \supseteq L(r)$, $L(M) \subseteq L(r)$, $L(M) = L(r)$, and $L(M) \cap L(r) \neq \emptyset$. Strings in L_7 have length a multiple of 3 and are indexed $1 \ldots 3m$ for some $m > 0$.

1. $L_1 = \{ \langle M \rangle : M$ accepts all strings that have exactly two 0s in them $\}$.
2. $L_2 = \{ \langle M \rangle :$ all strings that M accepts have exactly two 0s in them $\}$.
3. $L_3 = \{ \langle M \rangle : M$ accepts exactly the strings that have exactly two 0s in them $\}$.
4. $L_4 = \{ \langle M \rangle : M$ accepts some string that has exactly two 0s in them $\}$.
5. $L_5 =$ the set of strings that have exactly two 0s in them.
6. $L_6 =$ the set of even-length strings that have exactly two 0s and they are in the two center positions.
7. $L_7 =$ the set of strings x with exactly two 0s and they are in the places indexed m and $2m$ out of $3m$.
8. $L_8 =$ the complement of L_7.
9. $L_9 = \{ \langle N, x \rangle : N$ is an NFA and N accepts $x \}$.
10. $L_{10} = \{ \langle N, x \rangle : N$ is an NFA and N accepts all strings of the same length as $x \}$.

Please write your answers in this form: if L_{11} were the language of the Halting Problem, you could write "11. g" or "11. (g)" or (safest) "L_{11} is (g) r.e. but undecidable."

(2) 9 × 5 = 45 pts. *Multiple Choice*: Indicate clearly the one best answer for each in the form (number). (letter).

1. If A is a regular language and $B = \{xx : x \in A\}$ then B
   (a) is always regular;
   (b) is always a CFL but might not be regular;
   (c) is always decidable but might not be a CFL;
   (d) can be undecidable.

2. If A is a regular language and B is a DCFL then $A \cup B$
   (a) is always regular;
   (b) is always a DCFL but might not be regular;
   (c) is always a CFL but might not be a DCFL;
   (d) is always decidable but might not be a CFL.

3. To prove a language B is not c.e., it would suffice to:
   (a) Show $A_{TM} \leq_m B$;
   (b) Show $SAT \leq_m B$;
   (c) Show $E_{TM} \leq_m B$;
   (d) Show $B \leq_m NE_{TM}$.

4. Given an NFA N, an efficient way to decide whether $L(N) \cap 0^* = \emptyset$ is:
   (a) Convert N to an equivalent DFA M, do Cartesian product to get a DFA M' such that $L(M') = L(M) \cap 0^*$, and run the E_{DFA} algorithm on M';
   (b) Convert N to an equivalent regular expression r and accept if and only if r does not have any '1' characters in it;
   (c) Remove all arcs from N that process characters other than '0' and accept if and only if the start state is not live in the resulting NFA N';
   (d) The problem is NP-hard so there probably is no efficient way.

5. The Church-Turing Thesis includes the assertion that:
   (a) every yes/no decision problem is decidable;
   (b) some yes/no decision problems are neither c.e. nor co-c.e.;
   (c) every decision problem decidable in any high-level programming language is decidable by a Turing machine;
   (d) P = NP.

6. The concatenation $0^* \cdot (0 \cup 1)^*$ equals:
   (a) $(0 \cup 1)^*$;
   (b) 0^*;
   (c) 0^*1^*;
   (d) $\emptyset$.

7. Which of the following is NOT a property of the language B of balanced-parenthesis strings:
   (a) For all $x, y \in B$, $x \cdot y \in B$;
   (b) For all $x \in B$, $x^R \in B$;
   (c) For all $x \in B$, the number of '(' in x equals the number of ')' in x;
   (d) For all $x \in B$, $|x|$ is even.

8. If $A = L(M)$ for some DFA M with k states, then the language $A^R = \{x^R : x \in A\}$:
   (a) Equals $L(M^R)$ for some DFA M^R with k states.
   (b) Is regular but might not have a DFA with k states;
   (c) Is always decidable but might not be regular;
   (d) Could be undecidable.

9. In the CFG $G = S \longrightarrow abS \mid Sba \mid b$,
   (a) The string abb is ambiguous;
   (b) The string bba is ambiguous;
   (c) The string $abbba$ is ambiguous;
   (d) No string in $L(G)$ is ambiguous.

(3) 5 × 3 = 15 pts. *True/false*. You must write out the word `true` *or* `false` *in full*, and justifications are not needed. (The five questions are overleaf.)

(a) *True/false?:* The CFG $G = S \longrightarrow abS \mid Sba \mid a$ is ambiguous.

(b) *True/false?:* If A and B are c.e., then $A \setminus B$ is always c.e.

(c) *True/false?:* The problem of whether two regular expressions r_1, r_2 denote the same language (i.e., whether $L(r_1) = L(r_2)$) is decidable.

(d) *True/false?:* The problem of whether two context-free grammars G_1, G_2 generate the same language (i.e., whether $L(G_1) = L(G_2)$) is decidable.

(e) *True/false?:* Every CFL belongs to NP.

(4) (12 + 18 + 12 = 42 pts.)

Let $\Sigma = \{ a, b, c \}$. Let

$$\begin{aligned} A &= \{ x \in \Sigma^* : \text{between every two } b\text{'s in } x \text{ there is at least one } c \} \\ B &= \{ x \in \Sigma^* : \text{between every two } a\text{'s in } x \text{ there is at least one } b \} \\ E &= A \cap B. \end{aligned}$$

(a) For each of the following strings, say yes/no whether it belongs to E (2 pts. each):

(i) *baccc* (ii) ϵ (iii) *abacbacba* (iv) *abacaba* (v) *abacbab* (vi) cx, for any $x \in E$.

(b) Design a DFA M such that $L(M) = E$. For full credit, you must either use a "strategy" to build M or give a well-commented arc-node diagram. (18 pts.)

(c) Find a regular expression r such that $L(r)$ equals **the complement of** E. You may either work from complementing your DFA M or from negating the above definitions of A, B, E. (12 pts.)

(5) (18 pts.)

Define L to be the language of binary strings of even length that have exactly two 0s, in which one 0 is in the first half of the string and the other 0 is in the second half. Prove via the Myhill-Nerode Theorem that L is not a regular language.

(6) (6 + 6 + 18 + 12 = 42 pts.)

Consider strings x with both parentheses ( ) and braces { }. Say that x is "quasi-balanced" if erasing the braces in x leaves a string of balanced parens and erasing the parens in x leaves a string of balanced braces. For example, $x =$ ((\{\{))\}\} is quasi-balanced even though the two kinds of brackets are not correctly nested with each other.

This language Q of quasi-balanced strings is not a CFL: strings of the form $x = (^n\{^n)^n\}^n$ lead to a CFL Pumping Lemma proof highly like the one on Prelim II. But let us consider the following attempt to approach Q by a context-free grammar:

$$\begin{aligned} S &\to (A\}S \mid \{B)S \mid (S)S \mid \{S\}S \mid \epsilon \\ A &\to \{S) \mid (A) \mid)\{ \\ B &\to (S\} \mid \{B\} \mid \}(\end{aligned}$$

(a) Give parse trees for the strings ({)} and {(}) in this grammar.

(b) Show by a direct argument that the illustrative string $x =$ ((\{\{))\}\} *cannot* be derived in G. (problem continues...)

(c) Prove by structural induction that $L(G) \subset Q$. Reasonable shortcuts and appeals to symmetry are fine. Part of the answer is given below.

(d) Describe in words a two-tape Turing machine M such that $L(M) = Q$. Can you make M obey the rules of a DPDA except for one leftward rewinding "pass" of its input tape head?

Part answer to (c): To get the ball rolling, we'll take P_S to be the basic property that strings derived from S belong to Q and handle the rules that don't involve the variables A or B: $S \to \epsilon$ is immediate since $\epsilon \in Q$. Suppose $S \Longrightarrow^* x$ using the rule $S \to (S)S$ first. Then $x = (y)z$ where $S \Longrightarrow^* y$ and $S \Longrightarrow^* z$. By IH P_S on the RHS (twice), y and z are balanced in parens if you erase their braces. Then $x = (y)z$ remains balanced in parens on erasing its braces. It also remains balanced in braces if you erase its parens because that involves just concatenating the balanced braces in y and the balanced braces in z. This upholds P_S on LHS. The rule $S \to \{S\}S$ is handled symmetrically. You take it from there by defining appropriate properties P_A, P_B and analyzing the other rules...

(7) (4 + 18 + 6 = 28 pts.)

Prove that the following decision problem is undecidable, via a mapping reduction from the language of a problem we've already proved undecidable:

INSTANCE: A one-tape Turing machine M.

QUESTION: On any input x, whenever M has made two consecutive right (R) moves, are they immediately followed by a left (L) move?

First give a formal definition of the language L of this problem beginning "$L = \{\langle M \rangle : \dots\}$." Second, prove that L is undecidable. Third, also answer "is L c.e.?" with a brief justification. *For a hint*, note that the restriction in the QUESTION is not a practical burden, because if we want to move a TM M_0 rightward a bunch of times we can do $RRLR$ and continue RLR and RLR again as often as we like—the restriction just slows down a simulation $M_0(w)$ by a time factor of up to 3.

END OF EXAM.