One notes binder allowed, no Internet, closed neighbors, 170 minutes. The exam has six problems and totals 267 pts., subdivided as shown. Show your work—this may help for partial credit. Please write on these sheets only—if you need more paper you may ask.

Notation and terminology are as in the text, except for the following cases in which lectures gave 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. 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 $\bar{A}$, and $\phi$ stands for a Boolean formula.

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:

$$\begin{align*}
\{ a^n b^n : n \geq 0 \} & \text{ DCFL, but not regular} \\
\{ x \in \{ a, b \}^* : x = x^R \} & \text{ CFL and co-CFL, but not a DCFL} \\
\{ a^n b^n c^n : n \geq 0 \} & \text{ Co-CFL and in P, but not a CFL} \\
\{ a^n b^n a^m b^n : m, n \geq 0 \} & \text{ Co-CFL and in P, but not a CFL} \\
\{ \langle G, w \rangle : G \text{ is a CFG and } w \in L(G) \} & \text{ Co-CFL and in P, but not a CFL} \\
\{ x \# y \# z : x = y \land y \neq z \} & \text{ ACFG: In P but not a CFL (or co-CFL).} \\
\{ \phi : (\exists a \in \{ 0, 1 \}^n) \phi(a) = 1 \} & \text{ In P, but not CFL or co-CFL} \\
\{ \phi : (\forall a \in \{ 0, 1 \}^n) \phi(a) = 1 \} & \text{ SAT: In NP, believed not in P} \\
\{ \langle G, w \rangle : G \text{ is a CFG and } w \in L(G) \} & \text{ TAUT: In co-NP, not in NP unless NP = co-NP} \\
\{ \langle M, w \rangle : M \text{ is a TM and accepts } w \} & \text{ DNP: Decidable but not in NP} \\
\{ \langle M \rangle : M \text{ is a TM and does not accept } \langle M \rangle \} & \text{ R.e. but not co-r.e.} \\
\{ \langle M \rangle : \text{ for all inputs } x, M(x) \text{ halts} \} & \text{ Co-r.e. but not r.e.} \\
\{ \langle M \rangle : \text{ for all inputs } x, M(x) \text{ does not halt} \} & \text{ Neither r.e. nor co-r.e.}
\end{align*}$$
(1) (50 pts.)
Classify each of the following languages $L_1, \ldots, L_{10}$ by whether it is [currently known to be]

(a) regular;
(b) a DCFL but not regular;
(c) a CFL but not a DCFL;
(d) in $\mathbb{P}$ but not a CFL;
(e) in $\mathbb{NP}$, but not in $\mathbb{P}$ unless $\mathbb{NP} = \mathbb{P}$.
(f) decidable, but not in $\mathbb{NP}$ unless $\mathbb{NP} = \mathbb{co-NP}$.
(g) r.e. but not decidable.
(h) not r.e.—i.e., not recognizable.

You need not justify your answers, but brief justifications may help for partial credit—especially with some “close” answers. The languages are:

1. $L_1 = \{ \langle G \rangle : G$ is a context-free grammar and $L(G) \neq \emptyset \}$.
2. $L_2 = \{ \langle G \rangle : G$ is a context-free grammar and $L(G) \neq \Sigma^* \}$.
3. $L_3 = \{ \langle \phi \rangle :$ The Boolean formula $\phi$ is not a tautology $\}$.
4. $L_4 = \{ \langle N \rangle : N$ is an NFA and $L(N) = \Sigma^* \}$.
5. $L_5 = \{ x \in \{ a, b \}^* : \#a(x) \geq \#b(x) \}$.
6. $L_6 = \{ x \in \{ a, b \}^* : \#a(x) - \#b(x)$ is a multiple of 4 $\}$.
7. $L_7 = \{ a^i b^j a^k : i \neq k \lor j \neq k \}$.
8. $L_8 = \{ \text{Java programs } P: \text{on some input } x, P(x) \text{ prints “Hello World!” } \}$.
9. $L_9 = \{ \text{Java programs } P: \text{for all inputs } x, P(x) \text{ prints “Hello World!” } \}$.
10. $L_{10} = \{ \langle M_1, M_2 \rangle : M_1 \text{ and } M_2 \text{ are DFAs and } L(M_1) \cap L(M_2) = \emptyset \}$.

(2) $6 \times 6 = 36$ pts. True/false with justifications. You must write out the word true or false in full (3 pts.), and then write a brief justification—it need not be a proof (3 pts.).

(a) True/false?: For every DPDA $D$ that does not accept $\epsilon$, there is a context-free grammar $G$ in Chomsky normal form such that $L(G) = L(D)$.
(b) True/false?: The intersection of two CFLs is always a CFL.
(c) True/false?: The generally quickest way to tell if a given string $x$ matches a given regular expression $\alpha$ is to convert $\alpha$ into an equivalent DFA $M_\alpha$ and then run $M_\alpha(x)$.
(d) True/false?: The union of two regular languages is always a DCFL.
(e) True/false?: It is possible to write a “version 2.0” of Turing Kit such that whenever the user draws a DFA $M$, there is an efficient menu option to tell whether $L(M) = \Sigma^*$. 

(f) True/false?: It is possible to write a “version 3.0” of Turing Kit such that whenever the user draws a deterministic Turing machine $M$, there is an efficient menu option to tell whether $L(M) = \Sigma^*$.

(3) $11 \times 5 = 55$ pts. Multiple Choice: Circle clearly the one best answer for each. This time no justifications are needed, though they could help for partial credit.

1. In a Myhill-Nerode proof that the language $L = \{a^{2n}b^{2n} : n \geq 0\}$ is non-regular, the proof can begin with:
   (a) Take $S = a^*b^*$;
   (b) Take $S = (aa)^*$;
   (c) Take $S = (bb)^*$;
   (d) Take $S = (aa \cup bb)^*$.

2. In a Myhill-Nerode proof that the same language $L = \{a^{2n}b^{2n} : n \geq 0\}$ is non-regular, suppose we took $S = a^*$ instead—this is slightly inferior but workable. Suppose the “adversary” gives you $x = a^m$, $y = a^n$ where $m$ and $n$ are odd. Then your next step can be:
   (a) Take $z = b^m$.
   (b) Take $z = b^{2m}$;
   (c) Take $z = ab^{m+1}$; or
   (d) Take $z = a(bb)^m$.

3. To prove a language $B$ is $\text{NP}$-complete, after first showing $B \in \text{NP}$, we could:
   (a) Show SAT $\leq_p B$.
   (b) Show TAUT $\leq_p B$.
   (c) Show $\text{A}_{\text{TM}} \leq_p B$;
   (d) Show $B \leq_p \text{A}_{\text{TM}}$.

4. The union of two non-regular DCFLs can possibly be:
   (a) Regular;
   (b) A non-regular DCFL;
   (c) A CFL that is not a DCFL;
   (d) Any of the above.

5. For every language $A$, the concatenation $A^*A^*$ equals:
   (a) $(AA)^*$;
   (b) $\Sigma^*$;
   (c) $A^*$;
   (d) None of the above.
6. For every language $A$, the concatenation $(A \cup \{\epsilon\}) \cdot (\tilde{A} \cup \{\epsilon\})$ equals:
   (a) $AA$;
   (b) $\Sigma^*$;
   (c) $A^*$;
   (d) None of the above.

7. If $M_1$, $M_2$, and $M_3$ are DFAs with 100 states each, then $L(M_1) \cap L(M_2) \cap L(M_3)$ is:
   (a) Always empty;
   (b) Always recognized by a DFA with 100 states;
   (c) Possibly non-regular;
   (d) Always regular, but the smallest DFA that recognizes it might need 1,000,000 states.

8. In the CFG $G = S \rightarrow aS | bS | a | b$:
   (a) The string $abb$ is ambiguous;
   (b) The string $aba$ is ambiguous;
   (c) The empty string $\epsilon$ is ambiguous;
   (d) No string in $L(G)$ is ambiguous.

9. The language $\{ \langle G \rangle : G$ is a CFG and $\epsilon \in L(G) \}$ is (known to be):
   (a) In $P$;
   (b) $NP$-complete;
   (c) Equal to $\{ \epsilon \}$;
   (d) Undecidable.

10. In a proof that $\{a^i b^j c^k : i < j < k \}$ is not a CFL, upon being given a “pumping length” $p$, you can start by taking:
    (a) $s = a^p b^p c^p$;
    (b) $s = a^p b^{p+1} c^{p+2}$;
    (c) $s = a^p b^{2p} c^{3p}$;
    (d) Any of the above—they all work.

11. The undecidability of the Halting Problem, noting also the Church-Turing thesis, means that:
    (a) Extraterrestrial civilizations may be able to build computers that can solve it, even though human beings cannot;
    (b) There is no program that solves every given instance of the Halting Problem with a yes/no answer;
    (c) Human beings should not even bother to try to solve any instances of the Halting Problem;
    (d) Turing machines are too weak a model of computation to solve it; random-access machines or quantum computers are needed to solve it.
A person who likes to keep positive and negative values separate once wrote the following context-free grammar $G = (V, \Sigma, R, S)$ with $V = \{N, S\}$, $\Sigma = \{0, 1, (, ), +, -\}$, and rules:

$$
S \rightarrow S + S | N + S | S - (N) | 0 | 1 \\
N \rightarrow -(S) | (N - S).
$$

(a) Give a derivation tree for the string $1+0-(-(1+1))$. (6 pts.)

(b) Is it true that for every $x \in L(G)$, the substring $++$ does not occur in $x$? If so, prove it by structural induction; if not, demonstrate the existence of strings in $L(G)$ that do have this substring.

(c) Same question as (c) for the substring $--$.

(d) A hint: at least one of your answers in (b),c will be “no”—that is, the substring can occur. But it is possible to change one rule in $R$ such that neither substring can occur, in any $x \in L(G)$, and without using more than one pair of parentheses in any rule. Make the change to obtain a new grammar $G'$, and then prove this fact about your $G'$.

**Important Note:** You may and should make liberal but reasonable use of the phrase “This case is similar to a previous one” in order to shorten the proof(s). Parts (b)–(d) total 33 pts., but saying how they are subdivided would give too much away.

### (5) (15 + 6 + 6 + 24 = 51 pts.)

[Spring 2017: This is just FYI for you; comprehensiveness proofs by multi-threaded induction were not covered this year. See next “alternate” problem.] Let $A = \{a^nb^n : n \geq 1\}$. Define $E$ to be the language of strings that differ in at most one place from a string in $A$. An example of a string in $E$ is $aba$, since changing the last $a$ to $b$ gives a string in $A$. Note that $E$ contains $A$, and that the strings in $E$ have the same lengths as strings in $A$. Define $G$ to be the context-free grammar $(\{S, T, U\}, \{a, b\}, R, S)$, where the rules in $R$ are:

$$
S \rightarrow aSb | aTU | UTb \\
T \rightarrow aTb | \epsilon \\
U \rightarrow a | b.
$$

(a) For each of the following strings, say whether it belongs to $E$, and if so, give a parse tree or leftmost derivation for it in $G$: (i) $\epsilon$, (ii) $bb$, (iii) $aaabb$, (iv) $aaaabb$.

(b) Can any string $x$ that begins with $b$ and ends with $a$ belong to $E$? Justify your answer briefly.

(c) If $x \in E$ and $x = awa$ or $x = bwb$ for some string $w$, then what must be true about $w$?

(d) Prove by induction that $E \subseteq L(G)$. In fact the languages are equal, but you only need to prove the containment. Your answers to (b) and (c) may help you simplify the induction.

### (5’) (36 pts.)

[Spring 2017: This alternative problem was considered for last year’s exam and is more representative for you.] Let $\Sigma = \{a, b\}$. Consider the following two languages, both of which are nonregular.

$$
A = \{ x \in \Sigma^* : a(x) > b(x) \}, \\
B = \{ x \in \Sigma^* : a(x) < b(x) \}.
$$
(a) Give an example of a string $z$ of length 8 such that there is exactly one way to write $z = xy$ with $x \in A$ and $y \in B$. (That is, there is a unique way to break $z$ into a string $x$ that has more $a$'s than $b$'s followed by a string $y$ that has more $b$'s than $a$'s. (6 pts.)

(b) Give an example of a string $z$ of length 8 such that there are 7 different ways to write $z = xy$ with $x \in A$ and $y \in B$. (6 pts.)

(c) Let $D = A \cdot B$. Prove that $D$ is non-regular. (Note: I do not know how to do this with the version of the “Pumping Lemma” given in the text, but it is possible with a careful Myhill-Nerode argument that applies some of the ideas from your answers to (a) and/or (b). 24 pts.)

(6) (6 + 24 + 6 = 36 pts.)

Consider the following decision problem:

*Instance:* A deterministic Turing machine $M$.

*Question:* Do there exist strings $x, y \in \Sigma^*$ such that $M$ accepts $x$ but $M$ does not accept $y$?

(a) Using set notation, formalize this problem as a language $L$.

(b) Prove that $L$ is undecidable, by reduction from a known undecidable problem such as $A_{TM}$ or $K$.

(c) Is $L$ recognizable? Justify your answer briefly. (A full proof using another reduction is worth 6 pts. exam extra-credit.) END OF EXAM.