Open book, open notes, closed neighbors, 170 minutes. The exam has **six** problems and totals **267** pts., subdivided as shown.

(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 $P$ but not a CFL;
(e) in $NP$, but not in $P$ unless $NP = P$.
(f) decidable, but not in $NP$ unless $NP = co-NP$.
(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 $L(G) \neq \emptyset \}$. **Answer:** (d) In $P$ but not a CFL; essentially the complement of $E_{CFG}$.

2. $L_2 = \{ \langle G \rangle : G$ is a context-free grammar and $L(G) \neq \Sigma^* \}$. **Answer:** (g) R.e. but not decidable; essentially the complement of $ALL_{CFG}$; is c.e. because $A_{CFG}$ is decidable and one can guess a string that $G$ doesn’t derive.

3. $L_3 = \{ \langle \phi \rangle : The$ Boolean formula $\phi$ is not a tautology $\}$. **Answer:** (e) In $NP$ and complete since it is essentially the complement of $TAUT$, which is equivalent to $SAT$.

4. $L_4 = \{ \langle N \rangle : N$ is an NFA and $L(N) = \Sigma^* \}$. **Answer:** (f) Decidable but not in $NP$ unless $NP = co-NP$, since its complement was shown $NP$-hard on PS10.

5. $L_5 = \{ x \in \{ a, b \}^* : \#a(x) \geq \#b(x) \}$. **Answer:** (b) DCFL but not regular: DCFL because a DPDA can maintain the count of $\#a(y) - \#b(y)$ for prefixes $y$ of $x$ by pushing either $a$ or a character meaning “IOU an $a$” if the string starts with a $b$, for instance. Not regular by $S = a^*$ using “wlog. $m < n$” given $x = a^m$ and $y = a^n$ taking $z = b^{m+1}$.

6. $L_6 = \{ x \in \{ a, b \}^* : \#a(x) - \#b(x)$ is a multiple of 4 $\}$. **Answer:** (a) Regular since a DFA with 4 states can keep track of the congruences.

7. $L_7 = \{ a^ib^ja^k : i \neq k \lor j \neq k \}$. **Answer:** (c) CFL but not a DCFL: CFL since union of two DCFLs seen in class; not CFL since complement (when intersected with the regular set $a^*b^*c^*$) is $\{ a^ib^jc^i \}$—or at least has the same CFL Pumping Lemma proof as seen in class.
8. \( L_8 = \{ \text{Java programs } P: \text{on some input } x, P(x) \text{ prints “Hello World!”} \} \). Answer: (g) R.e. but not decidable; transparently equivalent to \( NE_{TM} \).

9. \( L_9 = \{ \text{Java programs } P: \text{for all inputs } x, P(x) \text{ prints “Hello World!”} \} \). Answer: (h) Not r.e., in fact neither r.e. nor co-r.e., since transparently equivalent to \( ALL_{TM} \).

10. \( L_{10} = \{ \langle M_1, M_2 \rangle: M_1 \text{ and } M_2 \text{ are DFAs and } L(M_1) \cap L(M_2) = \emptyset \} \). Answer: (d) In \( P \) but not a CFL: in \( P \) via Cartesian product and \( E_{DFA} \) algorithm, not a CFL since uses encodings.

(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 \( L(G) = L(D) \). Answer: True, since every DPDA recognizes a CFL and every CFL has a grammar in ChNF (with special allowance for \( \epsilon \) if needed).

(b) True/false?: The intersection of two CFLs is always a CFL. Answer: False—e.g. \( \{a^n b^n c^n\} \) can arise this way.

(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 \( M_\alpha(x) \). Answer: False because this can have exponential blowup; better is to convert the regexp to an NFA \( N \) as in class then simulate \( N(x) \) directly as on PS10.

(d) True/false?: The union of two regular languages is always a DCFL. Answer: True because it is regular and every regular language is 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 \( L(M) = \Sigma^* \). Answer: True because \( E_{DFA} \) has a polynomial-time (indeed, truly efficient) algorithm.

(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 \( L(M) = \Sigma^* \). Answer: False because \( E_{TM} \) is undecidable.

(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 \( S = (aa \cup bb)^* \).

Answer: (b)—anything allowing \( x = (bb)^m \) and \( y = (bb)^n \) will fail.
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 \( z = a(bb)^m \).

   \( \text{Answer: (c) need the extra } a \text{ to make an even number } m+1 \text{ of } a. \)

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

   \( \text{Answer: (a)—the others allow or force } B \text{ not to be in } \text{NP}. \)

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.

   \( \text{Answer: (d)—} D \cup \tilde{D} = \Sigma^* \text{ is an example of (a), the complement of } \{ a^n b^n c^n \} \text{ an example of (c)}. \)

5. For every language \( A \), the concatenation \( A^*A^* \) equals:
   (a) \( (AA)^* \);
   (b) \( \Sigma^* \);
   (c) \( A^* \);
   (d) None of the above.

   \( \text{Answer: (c) The } \epsilon \text{ in } A^* \text{ keeps all “singles.”} \)

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.

   \( \text{Answer: (b)—again } \epsilon \text{ allows every word in } A \text{ and } \tilde{A} \text{ to pass thru.} \)

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 million states.

Answer: (d)—and the limit is reached when \( L = \{ x : \#a(x), \#b(x), \#c(x) \text{ all } \equiv 0 \pmod{100} \} \).

8. In the CFG \( G = S \rightarrow aS | bS | a \ni b : \)
   (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.

Answer: (d)—this is the basic unambiguous (“right-linear”) list grammar.

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

Answer: (a)—by the same basic algorithm as for \( E_{\text{CFG}} \).

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

Answer: (b)—string (a) isn’t in the language and string (c) can’t work when the “adversary” picks \( vxy = c \).

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.

Answer: (b)—not (a) since the Church-Turing thesis is “universal”; not (c) since projects to solve tough halting problems have succeeded; not (d) because those models are equivalent to TMs in computability.
(4) (39 pts.)

A person who likes to keep positive and negative values separate once wrote the following context-free grammar $G = (V, \Sigma, R, S)$ with $V = \{N, S\}$ and rules:

$$S \rightarrow S + S \mid N + S \mid S - (N) \mid 0 \mid 1$$
$$N \rightarrow -(S) \mid (N - S)$$

(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 (b) 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.

**Answer:**

(a) The tree has $S \Rightarrow S - (N)$ at the root. The $N$ derives $-(S)$, which shows how the leading $-$ sign can come in. The two $S$-es then derive $1+0$ and $1+1$, respectively.

(b) It can occur: $S \Rightarrow S + S \Rightarrow S + N + S \Rightarrow S + -(S) + S$ and so on.

(c) This can occur too: $S \Rightarrow N + S \Rightarrow (N - S) + S \Rightarrow (N - N + S) + S \Rightarrow (N - (S) + S) + S$ . . .

(d) The two main ways were to change $N \rightarrow -(S)$ into $N \rightarrow (-S)$ or to add parens around the right-hand side $N + S$ (or just around the $N$). The latter main one gives this $G'$:

$$S \rightarrow S + S \mid (N + S) \mid S - (N) \mid 0 \mid 1$$
$$N \rightarrow -(S) \mid (N - S)$$

With reference to the new grammar, define:

- $P_S$ = “Every $x$ I derive has no $++$ or $--$ and begins and ends with a parenthesis or constant.”
- $P_N$ = “Every $y$ I derive has no $++$ or $--$ and ends with a parenthesis.” (Notes: It does not matter if you vacuously added, “or constant.” If you did the $N \rightarrow (-S)$ change instead then $P_N \equiv P_S$. It was also enough to say “. . . does not begin/end with an operator.”)

Doing three rules is (more than) enough to make the picture plain:

- $S \rightarrow (N + S)$: Suppose $S \Rightarrow^* x$ utrf. Then $x = (y + z)$ where $N \Rightarrow^* y$ and $S \Rightarrow^* z$. By IH $P_S$ on RHS, $z$ does not begin with $-$, and by both $P_S$ and $P_N$ on RHS, neither $y$ nor $z$ has internal substring issues. This covers all ways $++$ or $--$ could possibly occur. That $x$ immediately begins and ends with parentheses upholds the added clause of $P_S$ as well.
• $S \rightarrow S - (N)$: Suppose $S \Rightarrow^* x \text{ utrf}$. Then $x = y - (z)$ where $S \Rightarrow^* y$ and $N \Rightarrow^* z$. By IH $P_S$ on RHS, $y$ begins with '(' hence so does $x$. The other extra point needed is that by $P_S$ on RHS, $y$ cannot end in $-$ or $\ast$.

• $N \rightarrow (N - S)$: Suppose $N \Rightarrow y \text{ utrf}$. Then $y = wz$ where $N \Rightarrow^* w$ and $S \Rightarrow^* z$. By IH $P_N$ on RHS, $w$ ends in $)$. As before, $z$ does not begin with $-$ (or $\ast$)—so any new danger is averted—and $y$ has external parentheses which upholds the added clause of $P_N$.

This covers all three “danger cases,” and the safety of the other rules is entirely similar.

(5) $(15 + 6 + 6 + 24 = 51 \text{ pts.})$

Let $A = \{a^n b^n : n \geq 1\}$. 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 $aaba$, 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 $(\{S, T, U\}, \{a, b\}, \mathcal{R}, S)$, where the rules in $\mathcal{R}$ are:

\[
\begin{align*}
S & \rightarrow aSb \mid aTU \mid UTb \\
T & \rightarrow aTb \mid \epsilon \\
U & \rightarrow a \mid b.
\end{align*}
\]

(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) $aaabab$.

(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.

Answer: (a) (i) no, $\epsilon \notin L(G)$ since every rule for $S$ has a terminal; (ii) yes: $S \Rightarrow UTb \Rightarrow bTb \Rightarrow bb$; (iii) no: the length is odd; (iv) yes: $S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaTUbb \Rightarrow aaaaUbb \Rightarrow aaaaabb$.

(b) No: it requires changing at least those two characters to match the $a^+b^+$ form required for $A$, and that is one change too many. [Note that it was incorrect to say that the grammar couldn’t derive it; we have not yet established that $G$ is comprehensive.]

(c) Since $x = awa$ and $x = bwb$ each use up the one allowed “error,” $w$ must have no more errors: it must either belong to $A$ or be the empty string.

(d) Prove $(\forall n)P(n)$, where $P(n)$ is for each $x \in \{a, b\}^n$, if $x \in E$ then $S \Rightarrow^* x$. From the answer to part (c) we see that $T$ drives exactly the strings in $A \cup \{\epsilon\} = \{a^n b^n : n \geq 0\}$. We can use that as a “lemma” to avoid braiding $T$ into the induction, and since $U$ obviously derives just either $a$ or $b$, we can leave it out too. So no “enhanced $P(n)$”—just go with the old $P(n)$. Note that $P(0)$ holds vacuously since $\epsilon \notin E$, and $P(n)$ is also vacuous for odd $n$. So we can think of $n = 2$ as the “real basis.”

Basis ($n = 2$): The strings $aa, ab, bb$ all belong to $E$. We derives $bb$ above and get $S \Rightarrow aTU \Rightarrow aU \Rightarrow aa$ and $S \Rightarrow aTU \Rightarrow aU \Rightarrow ab$ for the other two. So $P(2)$ holds.
Induction (meaningfully $n \geq 4$): Assume (III) the statements $P(m)$ for all $m < n$ (again noting that $P(m)$ for odd $m$ are vacuous but don’t hurt anything). Let $x \in E$ with $|x| = n$. Then considering the first and last chars of $x$ we have that for some string $w \in \Sigma^*$, $|w| = n - 2$, either (i) $x = awa$, (ii) $x = awb$, or (iii) $x = bwb$. Note that $x = bwa$ is ruled out by part (b), and in all cases $w \neq \epsilon$ since $n \geq 4$.

Case (i) $x = awa$: Then $w \in A \cup \{\epsilon\}$ by part (c), so by the “lemma” for $T$, $T \Longrightarrow^* w$. Then we build the derivation $S \Longrightarrow aTU \Longrightarrow^* awu \Longrightarrow^* awa = x$. So $S \Longrightarrow^* x$ in this case.

Case (ii) $x = awb$: Then $w \in E$ and $|w| = n - 2 < n$, so $P(n - 2)$ applies inductively to give $S \Longrightarrow^* w$. So we build: $S \Longrightarrow aSb \Longrightarrow^* awb = x$.

Case (iii) $x = bwb$: Similarly to case (i), we get $T \Longrightarrow^* w$ by part (c) and the “lemma” $L_T = A \cup \{\epsilon\}$. So $S \Longrightarrow UTb \Longrightarrow bTb \Longrightarrow^* bwb = x$.

Since we get $S \Longrightarrow^* x$ in each of three cases that are mutually exhaustive for the language $E$, we deduce $P(n)$, so $(\forall n)P(n)$ follows by strong induction. This shows $E \subseteq L(G)$. [The soundness of these rules is pretty transparent, so $E = L(G)$ does follow.]

(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.)

Answer: (a) $z = aabbbbaaa = a \cdot bbbbaaa$. Also good is $abababa = a \cdot bbababa$, but $z = abbbaaaa$ is intuitively the most extreme case and suggests the proof in part (c).

(b) $z = aaaaabbb$. Any one of the internal breakpoints works.

(c) Take $S = ab^+$. Let any $x, y \in S$, $x \neq y$, be given. Then there are numbers $m, n \geq 1$ where wlog. $m < n$ such that $x = ab^m$ and $y = ab^n$. Take $z = a^{n-1}$ (which is well-defined because $n \geq 1$). Then $yz = ab^m a^{n-1}$ is in $A \cdot B$—indeed with the unique breakdown shown in part (a). But $xz = ab^m a^{n-1} \notin A \cdot B$ because $n - 1 \geq m$, so no suffix has more $b$’s than $a$’s. Thus $L(xz) \neq L(yz)$, and since $x, y \in S$ are arbitrary, $L$ is non-regular by the Myhill-Nerode Theorem.
(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.

Answer: (a) $L = \{ \langle M \rangle : L(M) \neq \emptyset \land L(M) \neq \Sigma^* \}$.

(b) Given any instance $x = \langle M, w \rangle$ of the Acceptance Problem, define $f(x)$ to be the code of a TM $M'$ that works as follows on any input $y$: “Simulate $M(w)$ open-endedly. If and when $M(w)$ halts, accept $y$ if and only if $y \neq \epsilon$ [or, to tie this in to PS10, iff $y = 0^n1^n$ for some $n \geq 1$].” (If $x$ is not a valid code, map it to some TM $M_0$ such that $L(M_0) = \emptyset$ so $\langle M_0 \rangle \notin L$. Or ignore this issue.) Then:

$$\langle M, w \rangle \in A_{TM} \iff M(w) \downarrow = 1 \iff L(M') = (\Sigma^* \setminus \{\epsilon\}) \text{ (or } = \{0^n1^n\} \text{) } \implies L(M') \neq \emptyset \land L(M') \neq \Sigma^* \implies f(x) = \langle M' \rangle \in L;$$

$$\langle M, w \rangle \notin A_{TM} \iff L(M') = \emptyset \implies f(x) = \langle M' \rangle \notin L.$$

So $A_{TM} \leq_m L$, so $L$ is undecidable. (The only hitch was realizing that the simple “all-or-nothing switch” fails because it jumps into the “$= \Sigma^*$” clause. Reductions testing $x == w$ also worked fine.)

(c) $L$ is not recognizable. The intuitive reason is that it still involves checking some kind of “all.” To prove that $L$ is in fact neither c.e. nor co-c.e., it suffices to show $D_{TM} \leq_m L$. This actually follows by the solution to PS10, problem (1), as in fact did part (b).