(1) (5 × 3 = 15 pts.) True/False.

Please write out the words true and/or false in full. No justifications are needed. Be sure to write your answers in the exam books.

(a) The union of two context-free languages is always a context-free language.
(b) There is a regular language \( L \) whose complement \( \tilde{L} \) is not a context-free language.
(c) Let \( G = (V, \Sigma, R, S) \) be a context-free grammar with the rule \( S \rightarrow SS \). If a given string \( x \) is ambiguous in \( G \), then so is \( xx \).
(d) Some context-free languages cannot be accepted by any deterministic Turing machine.
(e) Every context-free grammar in Chomsky normal form is unambiguous.

Answer:

(a) True by \( S \rightarrow S_1 \mid S_2 \) etc.
(b) False: \( \tilde{L} \) is always regular too, and every regular language is a CFL.
(c) True: If \( x \) has two different parse trees from \( S \), then \( xx \) has (at least) \( 2 \times 2 = 4 \) different parse trees from \( SS \), and \( S \Rightarrow SS \) so \( xx \) is in fact ambiguous.
(d) False: Every CFL is computably enumerable, indeed decidable.
(e) False: The rule \( S \rightarrow SS \) (or \( A \rightarrow AA \) for other variables \( A \)) is allowed in ChNF, and any grammar with that rule is going to be ambiguous (except for the trivial possibility that \( S \) or \( A \) does not derive any terminal string at all).

(2) (24 pts. total)

Let \( E \) be the language of all strings over \( \Sigma = \{a, b\} \) that do not have the substring \( bb \), and let \( G \) be the following context-free grammar:

\[
\begin{align*}
S & \rightarrow \epsilon \mid b \mid BS \mid SA \\
A & \rightarrow aS \mid AA \\
B & \rightarrow a \mid bAaB
\end{align*}
\]

(a) Show that the string \( babab \) is ambiguous in the grammar \( G \), by giving two different parse trees:

(b) Is any other variable besides \( S \) nullable? Give one(s) if so.

(c) Do any unit rules occur during the conversion to Chomsky normal form? Give one(s) if so—but do not do any more of the conversion.

(d) Prove by analysis of individual rules that \( L(G) \subseteq E \). Hint: Ask yourself what additional properties, besides not allowing a \( bb \) substring themselves, must the variables \( A \) and \( B \) maintain?
(a) **Answer in prose:** Start \( S \to SA \) in both cases. In the first, derive the \( S \) to \( b \) and \( A \) to \( aS \). This gives \( baS \), which repeats to give \( babaS \) and finally \( babab \). The second derives the \( A \) to \( AA \) instead, giving \( bAA \). The next level of the tree has \( baS aS \), and deriving both \( S \)’es to \( b \) completes the tree. (There are other parses too.)

(b) **Answer:** No; in particular, \( A \) is not nullable since it eventually must derive at least one \( a \).

(c) **Answer:** Yes—the rules \( S \to B \) and \( S \to A \) occur. The rule \( A \to a \) is also added, but it does not count as a unit rule.

(d) The extra properties you want to enunciate are that every string derived from \( A \) begins with ‘\( a \)’ and every string derived from \( B \) ends in ‘\( a \)’ The former immediately in the rule \( A \to a \) and is preserved in the rule \( A \to AA \) “by structural induction” because of the leading \( A \) on the right-hand side. Similarly, the rule \( B \to bAA \) preserves the property “…ends in ‘\( a \)” by structural induction because there is a recursive \( B \) on the right-hand side. With these properties in hand, we can now analyze every rule that has a juncture where a \( bb \) could possibly occur, including the two rules we just examined:

- \( S \to BS \): Because the \( B \) supplies a final \( a \) at the juncture with \( S \), and because \( B \) and \( S \) individually speak the property of having no \( bb \) inside strings they derive, this rule upholds that property.
- \( S \to SA \): This time the \( A \) supplies an \( a \) to lead the right-hand side of the juncture, so it is safe from introducing a \( bb \).
- \( A \to AA \): The second \( A \) on the right-hand side of the rule makes sure the juncture is safe. (Note that the first \( A \) could supply a final \( b \) via the derivation \( A \Rightarrow aS \Rightarrow ab \).)
- \( B \to bAA \): Again the \( A \) inductively supplies a leading \( a \) to safely follow the initial \( b \) in the body of the rule. The other two junctures are covered by the literal ‘\( a \)’ in that rule body. So it is safe.
- The other rules cannot cause a \( bb \) by themselves.

Hence \( L(G) \subseteq E \). (In fact the grammar is comprehensive, i.e. \( L(G) = E \), but this wasn’t asked. See also a note at the end on how this kind of material was presented in previous years.)

(3) (30 pts.)

Define \( L = \{ a^i b^j c^k : i = j + k, \ i,j,k \geq 0 \} \). Do (a) and **Exactly One** of (b) or (c). In whichever case you choose, you must write some prose comments that explain how your \( G \), \( M_1 \), or \( M_2 \) works—and you need not prove correctness formally.

(a) Design a context-free grammar \( G \) such that \( L(G) = L \). Then modify your \( G \) into a context-free grammar \( G' \) such that \( L(G') = L \setminus \{ \epsilon \} \) and \( G' \) has no \( \epsilon \)-rules.

(b) Design a single-tape deterministic Turing machine \( M_1 \) such that \( L(M_1) = L \). OR,

(c) Design a deterministic pushdown automaton \( M_2 \), coded as a two-tape deterministic Turing machine that obeys the pushdown restrictions, such that \( L(M_2) = L \).

**Answer:** For (a), the strategy is to generate one \( c \) in back for each \( a \) in front. At some point switch to a new variable that does one \( b \) in back per \( a \) in front. Finally zap that new variable. The following rules carry out this strategy:
Given a breakdown \( s = w^i x y^i z \) into five substrings that “survives”—i.e., is such that for all \( i \geq 0 \), \( w^i x y^i z \) \textbf{does} belong to \( L \). Briefly explain why. (There are multiple correct answers.)

\[
\begin{align*}
  u &= \text{______________} & v &= \text{_____} & x &= \text{_____} & y &= \text{_____} & z &= \text{______________} \\
  \text{Answer:} \quad &\text{One of many ways is to take} \\
  &u = a^{p-1} & v = a & x = \epsilon & y = b & z = b^p c^{p+2}
\end{align*}
\]

(either abstractly or concretely with \( p = 5 \) was fine). Ditto taking \( vxy = bc \) on the other side instead, and actually the OR allows you to give practically any breakdown with \( |vxy| \leq 5 \). The reason in the above particular case is that the property of having one less \( a \) than \( b \) doesn’t change in \( w^i x y^i z \) for any \( i \). [The only unsuccessful choices would be things like \( v = aa \) and \( y = b \), where \( i = 2 \) makes the counts all equal, or \( v = b \) and \( y = cc \), where \( i = 0 \) gives \( a^p b^p c^p \notin L \). The all-too-common errors were forgetting \( |vxy| \leq p \) by e.g. \( v = a, x = bbb, y = b \) and choosing \( v = ab \) (etc.) where already with \( i = 2, s^{(i)} = w^2 x y^i z \) is no longer in \( a^* b^* c^* \) so not in \( L \). And not making \( w^i x y^i z = s \) to begin with.]

\(^1\)A grammar starts with \( S \to S_1C \mid AS_2 \), makes \( A \) derive \( a^* \) and \( C \) derive \( c^* \), and makes \( S_1 \) derive \( \{a^i b^j : j > i \} \) by ways we have seen in lectures and notes and answer keys, likewise \( S_2 \) deriving \( \{b^j c^k : k > j \} \). You can answer the question without caring about these details.
(5) (21 pts.)

Prove that every DFA $M$ such that $L(M) = (a + ab)^*(b + ba)^*$ needs at least 5 states, by exhibiting a PD set of size 5 for this language. [Added for Spring 2017, replacing the problem labeled (X) below which represents material not covered, but this may seem longer.]

Answer: The shortest string not belonging to $L(M)$ is $baa$. In fact it is a dead string, so we only need to find 4 $L$-distinct strings among $\epsilon, a, b, aa, ab, ba, bb$. We separate $x = \epsilon$ from $y = a$ by $z = baa$ itself because $yz = abaa = ab \cdot a \cdot a$ does belong to $L$. We separate $b$ from both $\epsilon$ and $a$ by $z = aa$. And finally we separate $ba$ from all three strings by $z = a$. So $S = \{\epsilon, a, b, ba, baa\}$ is a PD set of size 5. Hence any DFA $M$ such that $L(M) = (a + ab)^*(b + ba)^*$ needs at least 5 states, by the finite case of the Myhill-Nerode Theorem. (And there is such a 5-state DFA.)

(X) (21 pts.)

Let $G$ be the context-free grammar

$$S \rightarrow AbS \mid BaS \mid A \mid B \mid \epsilon, \quad A \rightarrow aA \mid a, \quad B \rightarrow bB \mid b.$$ 

Prove that $L(G) = \Sigma^*$, where $\Sigma = \{a, b\}$. (Hint: Given $x \in \Sigma^*$, $x \neq \epsilon$, you may (or may not) need 3 or 4 cases, depending on how you word things. You may use separate “lemmas” for the languages of strings derived by $A$ and $B$ without involving them in the overall induction, and some other reasonable proof shortcuts are OK.)

Answer: First, $G$ derives all of $a^* \cup b^*$ via $S \rightarrow A \mid B$ directly, by the “lemmas” that $A \Rightarrow^* a^+$, $B \Rightarrow^* b^+$ and $S \Rightarrow^* \epsilon$. Any other string $x$ must have the form $uvw$ where $u \in a^+$ or $vaw$ where $v \in b^+$. Since in each case $|w| < |x|$, we have by induction hypothesis (on the length of strings) that $S \Rightarrow^* w$. Hence we can derive $x$ by whichever of these derivations applies: $S \Rightarrow^* AbS \Rightarrow^* ubS \Rightarrow^* ubw = x$ or $S \Rightarrow^* BaS \Rightarrow^* vaS \Rightarrow^* vaw = x$. Hence in all cases, $S \Rightarrow^* x$ and the induction goes through to make $L(G) = \Sigma^*$.

Extra Note about problem (2): In previous years I used a “proof script” for a more-formal approach to Structural Induction in which variables $A$ “personified” properties “$P_A$.” This year I felt that the value of the script was outweighed by the possibility of confusing $P_A$ as a soundness property or a comprehensiveness property (it can be both). Moreover, the English words “Every $x$ I derive . . .” speak soundness, but the similar words “I derive every $x$ that . . .” speak the opposite—comprehensiveness—and maybe this was confusing. So this year I re-cast my lectures to write $T_A$ when we care about a soundness target and $E_A$ when it is exact (i.e., when $A$ generates exactly the language specified by the property $E_A$). Anyway, here is how the proof would look using the script; this may help you review material from last year that has been left posted. The abbreviation “utrf” stands for “using this rule first”—structural induction corresponds to induction on the number $m$ of steps in a derivation where the induction hypothesis is applied to sub-derivations after the first step is taken. Again, the grammar is: $S \rightarrow \epsilon \mid b \mid BS \mid SA; A \rightarrow aS \mid AA; B \rightarrow a \mid bAaB$.

Answer: Define $P_S \equiv “Every x that I derive has no substring bb,” which is just the “vanilla” statement of membership in $E$. A top-down, goal-oriented way to get the stronger properties needed for $A$ and $B$ was to note that by the rule $S \rightarrow b$, one must allow that a substring $y$ derived from $S$ might both begin and end with $b$. Hence $S \rightarrow SA$ requires that strings $z$ such that $A \Rightarrow^* z$ cannot begin with $b$—which since $A$ is not nullable means $z$ must begin with $a$. Likewise $S \rightarrow BS$ mandates that any string $w$ derived from $B$ (which is not nullable either) must end in $a$. Thus the properties you want for $P_A$ and $P_B$ both include $P_S$, and add the clause “and if $A \Rightarrow^* x$ then $x$ begins with $a$” for $P_A$ and “if $B \Rightarrow^* x$ then $x$ ends in $a$” to $P_B$. The key further point required for full credit was that we need to uphold these extra clauses in the rules for $A$ and $B$ as well.
• $S \to \epsilon \mid b$: These rules uphold $P_S$ immediately.

• $S \to BS$: Suppose $S \Rightarrow^* x$ using this rule first. Then $x =: yz$ where $B \Rightarrow^* y$ and $S \Rightarrow^* z$. By IH $P_B, P_S$ on the right-hand side, neither $y$ nor $z$ has a bb substring and $y$ ends in $a$, so no bb appears where they concatenate either. So $x$ has no bb, which upholds $P_S$ on the left-hand side.

• $S \to SA$: Suppose $S \Rightarrow^* x$ utrf. Then $x =: yz$ where $S \Rightarrow^* y$ this time and $A \Rightarrow^* z$. By IH $P_S, P_A$ on RHS, $y$ and $z$ each have no internal bb and $z$ begins with $a$. So $x$ has no bb at the boundary either, which upholds $P_S$ on the LHS. (That finishes $S$, but we still need to maintain the stronger properties in the other rules.)

• $A \to aS$: Suppose $A \Rightarrow^* x$ utrf. Then $x =: ay$ where $S \Rightarrow^* y$. By IH $P_S$ on RHS, $y$ has no bb so neither does $x$, and the leading $a$ makes $ay = x$ immediately uphold the added clause of $P_A$. So $P_A$ is upheld on LHS.

• $A \to AA$: Suppose $A \Rightarrow^* x$ utrf. Then $x =: yz$ where $A \Rightarrow^* y$ and $A \Rightarrow^* z$. By IH $P_A$ on RHS (twice), neither $y$ nor $z$ has an internal bb, and $y$ begins with $a$ which implies the same for $x$. Is that all we need to say, i.e., is it immaterial that $z$ begins with $a$? No, that is needed too, to say that no bb occurs at the boundary between $y$ and $z$. So $P_A$ holds on the LHS.

• $B \to a$: Both clauses of $P_B$ are immediately upheld.

• $B \to bAaB$: Suppose $B \Rightarrow^* x$ utrf. Then $x =: byaz$ where $A \Rightarrow^* y$ and $B \Rightarrow^* z$. By IH $P_A, P_B$ on RHS, those strings have no internal bb, and they don’t touch and in fact surround an $a$, so the only bb danger can come from the leading $b$. That is averted, however, because $P_A$ on RHS implies that $y$ begins with $a$. We’re not done—we need to uphold “ends with $a$” as well to get $P_B$ on LHS, but this follows by “self-induction” since $z$ ends in $a$ by IH $P_B$ on RHS.

Thus the properties are upheld by all the rules, so $L(G) \subseteq E$ by structural induction.

Finally, for comprehensiveness, there is no “script”—and this year I’m just emphasizing conceptual insights. With this grammar the insight is conveyed by the derivations $S \Rightarrow BS \Rightarrow aS$ and $S \Rightarrow SA \Rightarrow bA \Rightarrow baS$. Thus we can iterate deriving $a$ or $ba$ at will and finish with either $\epsilon$ or $b$. The grammar thus derives all of $(a + ba)^*(\epsilon + b)$ which is the entire language $E$ of strings not having a bb substring. Hence $L(G) = E$, and moreover the rules $A \to AA$ and $B \to bAaB$ are unnecessary.