

CS 475 Fall 2004 Exam 2

SOLUTIONS and GRADING KEY

INSTRUCTIONS (read carefully)

- Fill in the following information giving name and ID.

NAME:

NET-ID:

- Put your name and ID on every page.
- There are 6 problems, on 6 numbered pages. Make sure you have a complete exam.
- The point value of each problem is indicated next to the problem
- The exam is designed for 60 minutes, although you have 120 minutes.
- This is a closed book exam. No notes of any kind are allowed. Do all work in the space provided, using the backs of sheets if necessary. See the proctor if you need more paper.

Problem	Possible	Score
1	10	
2	10	
3	12	
4	10	
5	10	
6	8	
Total	60	

Problem 1 (10 points)

This problem is T/F. Completely write out “True” if the statement is necessarily true. Completely write out “False” if the statement is not necessarily true. Each correct answer is worth one point. Ambiguous answers will not receive credit.

1. CFLs are closed under concatenation with nonregular languages.

FALSE: Let $L_1 = \{\epsilon\}$ (a CFL), and L_2 be any non-CFL (hence nonregular). Then $L_1L_2 = L_2$, which is not a CFL.

2. If L is regular, then there is a deterministic PDA M such that $L(M) = L$.

TRUE: If L is regular, then there is a *DFA* accepting L .

3. If G is a grammar with $L(G)$ nonempty, then the length of the smallest string in $L(G)$ is linear in the size of G .

FALSE: We can easily get sentential forms to grow exponentially in length even with a CNF grammar. For example, $S \rightarrow A_1A_1$, $A_1 \rightarrow A_2A_2$, $A_2 \rightarrow A_3A_3 \dots A_{n-1} \rightarrow A_nA_n$, and $A_n \rightarrow a$. Then the shortest (in fact the only) string generated has length 2^n .

4. If $L \subseteq \{0, 1\}^*$ is a CFL, then the language $\{w_1\#w_2 : w_1 \in L^n \text{ and } w_2 \in L^n\}$ is a CFL.

TRUE: This problem should have read “... $w_2 \in L^n$ for some n ” but this is the natural interpretation anyway. If L is generated by grammar G with start symbol S , then the language above is generated by the grammar with new start symbol S' and new productions $S' \rightarrow SS'S \mid \#$

5. If L_1 is a CFL and L_2 is finite, then $L_1 - L_2$ is a CFL.

TRUE: Since L_2 is finite, it is regular, and so is its complement $\overline{L_2}$. But $L_1 - L_2 = L_1 \cap \overline{L_2}$, which is a CFL because CFLs are closed under intersection with regular sets.

6. Every language accepted by a PDA in the final-state acceptance model, is accepted by some PDA in the empty stack acceptance model.

TRUE: This is directly out of the book, and lecture.

7. If L is a CFL and R and S are regular, then the language $\text{Majority}(L, R, S) = \{w : w \text{ is in at least two of } L, R, S\}$ is a CFL.

TRUE: A PDA M_L (accepting by final state) for L can have its state set augmented to simultaneously simulate DFAs M_R and M_S for R and S while processing the input, and can then accept iff at least two of M_L, M_R, M_S are accepting.

8. Every CFL can be accepted by null stack by a PDA which has exactly 3 states.

TRUE: (Every CFL can be accepted by null stack by a PDA with exactly 1 state via the construction of a PDA from a CFG. We can add two useless states without changing the language.)

9. Every CFL not containing the empty string has a grammar where all productions are of one of the forms $A \rightarrow BCD$, $A \rightarrow ab$, or $A \rightarrow a$.

TRUE/FALSE: Online students received an earlier version of this problem, where the phrase “not containing the empty string” was not included. In that case, the answer is “False” since $\{\epsilon\}$ is a CFL and has no grammar of the above form. As written above (for in-class test takers), the correct

answer is “True”. While we wouldn’t expect you to think this through completely for a T/F question, your familiarity with normal forms and grammatical tricks should have allowed you to guess that one could modify a CNF grammar to get it in the above form.

10. $L = \{a^p b^q c^r d^s : p + r \neq q + s\}$ is a CFL.

TRUE: Notice that we are not matching a’s and c’s or b’s and d’s, but rather summing the number of a’s and c’s, and matching that with the total number of b’s and d’s. PDA M2 in problem 3 of this exam accepts strings with an equal number of a’s and b’s. We leave it as an exercise to modify it so that it accepts strings of a’s, b’s, c’s and d’s, matching the a’s plus c’s with b’s plus d’s in a similar way, while checking that the string is of the form $a^* b^* c^* d^*$.

Problem 2 (10 points)

Each of the following defines a language L . For each, answer “Regular” if L is necessarily regular, “CFL” if L is necessarily context-free but not necessarily regular, or “Neither” if L is not necessarily a context-free language. Your score is your number correct. Ambiguous answers will not receive credit.

1. $L = \{0^n 1^m 0^{m+1} : n \geq 0\}$

CFL (Easy to create a grammar for this.)

2. $L = \{0^n w 1^n : w \in \{0, 1\}^*\}$

REGULAR (w can absorb the entire string, so this is just $\{0, 1\}^*$.)

3. $L = \{0^n 1^m : \text{for some } k \leq 37, m = kn\}$

CFL (For each i between 0 and 37, let S_i be the start symbol of a grammar for $L_i = \{0^n 1^{in} : n \geq 0\}$ which generates i ones for each zero. Then create a new start symbol S and productions $S \rightarrow S_i$ for each i .)

4. $L = \{0^n w 0^n : w \in \{0, 1\}^*\}$

REGULAR (same as number 2 above)

5. $L = \{0^n 1^m : \text{for some } k \geq 37, m = kn\}$

NEITHER Intuitively, no way to check whether the number of 1s is some multiple of the number of 0s. In problem 3 above there were only a finite number of possible multiples to check.

6. $L = \{a^i b^j c^k : 0 < i < j < k\}$

NEITHER (pumping lemma, or other techniques, would show this easily).

7. $L = L_1 \cdot \overline{L_1}$, where L_1 is a CFL.

NEITHER (Let L_1 be a CFL whose complement is a nonCFL, and perhaps separate the two with a special symbol: $L_1 = \{x\# : x \text{ is not of the form } ww\}$. Then $L_1 \cdot \overline{L_1} = \{x\#y\# : x \text{ is not of the form } ww, \text{ but } y \text{ is of the form } ww\}$, which is clearly not a CFL.)

8. $L = L_1 \cap \overline{L_2}$, where L_1 and L_2 are CFLs

NEITHER (Let $L_1 = \Sigma^*$, and L_2 be any CFL whose complement is not a CFL.)

9. $L = \{ww^R : w \in L_1\}$, where L_1 is a CFL.

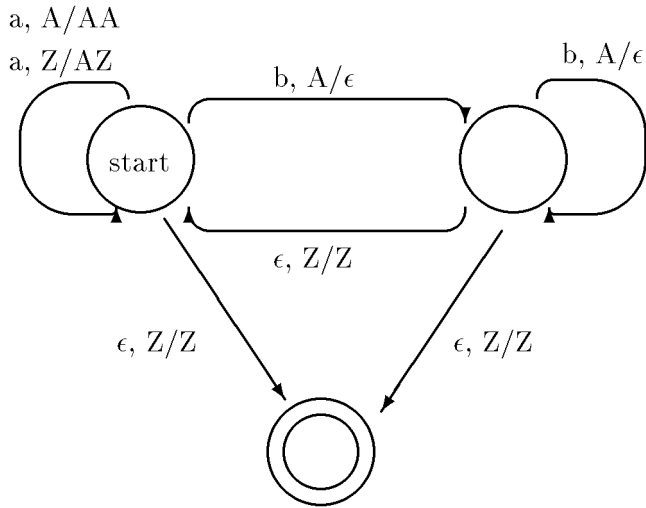
NEITHER (Let $L_1 = \{0^n 1^n : n \geq 0\}$, and then $L = \{0^n 1^{2^n} 0^n : n \geq 0\}$.)

10. L , where for some PDA M , $L = L(M)$ and for all $w \in L(M)$, there is an accepting computation of M on w in which M pops its stack at most 37 times. (M pops its stack if it executes a transition while removing the top of stack symbol and not replacing it with anything.)

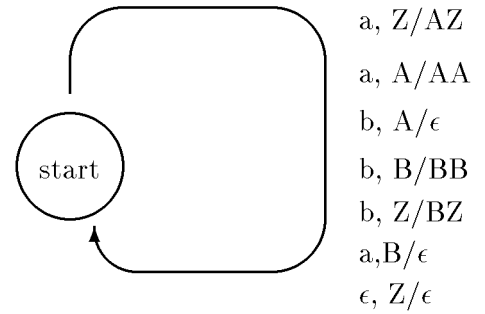
REGULAR (Intuitively, this means the PDA cannot take advantage of its unbounded stack, since after the 37th pop it can't go back and see what has been put there. An NFA which keeps the topmost 37 symbols of the stack in its finite memory should be sufficient to simulate the PDA, since, by the pop limitation, no symbol below the 37th can every be seen again.)

Problem 3 (12 points)

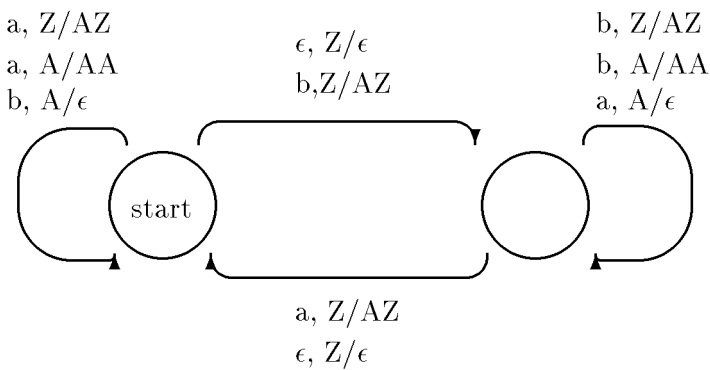
Exactly describe the languages accepted by the following PDAs. “Z” denotes the initial stack symbol. (Each is worth 3 points; don’t spend too much time on any one machine if you find it confusing.)



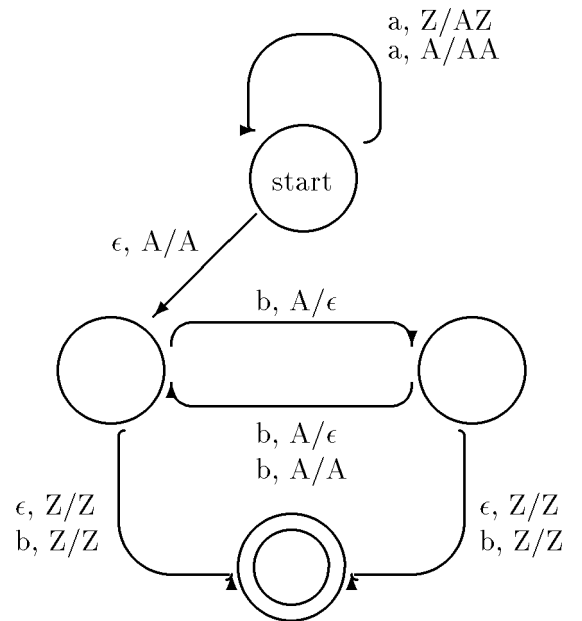
M1: ACCEPTS BY FINAL STATE



M2: ACCEPTS BY NULL STACK



M3: ACCEPTS BY NULL STACK



M4: ACCEPTS BY FINAL STATE

Answers to problem 3:

M1 accepts $\{a^n b^n : n \geq 0\}^*$. Note that this is different than $\{(a^n b^n)^m : n, m \geq 0\}$, which is not context free because it requires that each block of matching a's and b's must have the same number of a's and b's as all other blocks.

M2 accepts $\{w \in \{a, b\}^* : w \text{ has the same number of } a\text{'s and } b\text{'s.}\}$

M3 accepts the same language as M2.

M4 accepts $\{a^n b^m : 1 \leq n \leq m \leq 2n\}$

Grading Rubric (3 points possible for each part:

3 points for totally correct. Answer must include boundary condition. Technically, if your answer didn't preclude languages that contain other symbols (e.g., you might have said "strings with the same number of a's and b's") then your answer was incorrect, but we didn't deduct for this.

2 points for correct, except for failing to mention boundary condition (e.g. $n \geq 1$), or including an incorrect boundary condition, or some other minor error that results in a language that is *slightly* different.

1 point for an answer that correctly captures some significant constraint that describes the language, but fails to capture at least one additional significant constraint.

0 points for answers that do not fall in the above categories.

Shame on you if you didn't describe the language as a set, but instead gave a verbal description (Example: "some number of a's followed by an equal number of b's followed by some (perhaps other) number of a's followed by the same (perhaps other) number of b's... any number of times". If your description was completely correct, then no points were deducted. But if your description was a grammar, or a restatement of the running of the machine, or any other non-static description, then you probably received 0 points.

Problem 4 (10 points)

Construct a context-free grammar G such that $L(G) = \{a^i b^j c^k : 0 \leq i \leq j + k \leq 2i\}$.

Explain what each nonterminal does, and how your grammar works.

There are an infinity of ways to do this, but among the most straightforward and intuitive is the following.

$$\begin{aligned} S &\rightarrow aSc \mid aScc \mid T \mid aTbc \\ T &\rightarrow aTb \mid aTbb \mid \epsilon \end{aligned}$$

The key observations are:

- We should generate one or two b 's for each a . We should also generate one or two c 's for each a .
- We should generate the a 's and c 's *first*, and then generate the a 's and b 's, because when you generate matching characters on either side of a nonterminal, the first characters generated will be on the outsides of the string.

The nonterminal S generates “matching” a’s and c’s, finally leaving a T in the center. Thus, S derives sentential forms of the form $a^n T c^m$ where $0 \leq n \leq m \leq 2n$.

Then the nonterminal T generates “matching” a’s and b’s, and ultimately disappears, so T generates sentential forms of the form $a^n T b^m$ or $a^n b^m$ where $0 \leq n \leq 2m$.

Putting these two together, we see that S generates strings of the correct form. However, it doesn’t generate *all* of the strings of the correct form. Without the additional production $S \rightarrow aTbc$, the grammar would miss strings of the form: $a^n b^{2n-1} c$. That is, strings where the number of b’s and c’s are at the maximum allowed for the given number of a’s, and which also have a single c . In order to get $2n$ b’s and c’s, we cannot generate a single character with an a - we need to generate two for each. And, if there is to be only one c , we will need to generate a b with it. (The professor neglected to include this transition, and thus did not receive full credit.)

8 points were given to the construction, and 2 points for the explanation. However, if the construction could not be understood because what explanation offered, if any, was insufficient for the complexity of the solution, then up to 8 points were deducted.

Typical mistakes:

- Failing to enforce that $i \leq j + k \leq 2i$. (-4 points)
- Not enforcing the required order relationship (strings must be in form $a^* b^* c^*$). (-4 points)
- Not including the empty string. (-2 points)
- Minor other errors - including the failure to account for the special case above $a^n b^{2n-1} c$. (-1 point)
- Not explaining how your grammar works, as requested. (-2 points)

Problem 5 (10 points)

Let G be the grammar with start symbol S , nonterminals $\{A, B, C, D, S\}$, terminal symbols $\{a, b\}$, and productions as follows:

$$\begin{aligned} S &\rightarrow aAa \mid bBb \\ A &\rightarrow C \mid a \\ B &\rightarrow C \mid b \\ C &\rightarrow CDA \mid \epsilon \\ D &\rightarrow A \mid B \mid ab \end{aligned}$$

1. List all and only the nullable nonterminals.

$$\{A, B, C, D\}$$

2. Give a grammar G_2 without ϵ -productions such that $L(G_2) = L(G)$.

$$\begin{aligned} S &\rightarrow aAa \mid aa \mid bBb \mid bb \\ A &\rightarrow C \mid a \\ B &\rightarrow C \mid b \\ C &\rightarrow CDA \mid CD \mid CA \mid DA \mid D \mid A \\ D &\rightarrow A \mid B \mid ab \end{aligned}$$

3. Give a grammar G_3 without unit productions such that $L(G_3) = L(G_2)$.

The easiest thing to do is to recognize that $A, B, C,$ and D can all reach each other via unit productions, hence there is no need to distinguish between these at all. We can simply replace them all with A , for example, and obtain:

$$\begin{aligned} S &\rightarrow aAa \mid aa \mid bAb \mid bb \\ A &\rightarrow AAA \mid AA \mid a \mid b \mid ab \end{aligned}$$

and then continue on. But if we didn't simplify as above at this point, we'd have:

$$\begin{aligned} S &\rightarrow aAa \mid aa \mid bBb \mid bb \\ A &\rightarrow CDA \mid CD \mid CA \mid DA \mid a \mid b \mid ab \\ B &\rightarrow (\text{same as } A\text{'s right sides}) \\ C &\rightarrow (\text{same as } A\text{'s right sides}) \\ D &\rightarrow (\text{same as } A\text{'s right sides}) \end{aligned}$$

4. Give a grammar G_4 without useless symbols such that $L(G_4) = L(G_3)$.

Regardless of whether we simplified to the single nonterminal A , or kept all of $A, B, C,$ and D , the above grammar has no useless symbols, so our answer here is identical to the one above. You could simplify here instead of above. (You wouldn't be removing useless symbols - but the problem didn't ask you to remove useless symbols. It asked you to give a grammar without useless symbols that generates the same language. So, even if you incorrectly thought that B, C, D were useless, you would not be penalized because the simplified grammar you get by tossing out these redundant (but not technically useless) nonterminals still satisfies the specification of the problem.)

5. Give a grammar G_5 in Chomsky Normal Form such that $L(G_5) = L(G_4)$.

We first do it for the simplified (only A) grammar.

$$\begin{aligned} X_a &\rightarrow a \\ X_b &\rightarrow b \end{aligned}$$

$$\begin{aligned} S &\rightarrow X_aX \mid X_aX_a \mid X_bY \mid X_bX_b \\ X &\rightarrow AX_a \\ Y &\rightarrow AX_b \end{aligned}$$

$$\begin{aligned} A &\rightarrow AZ \mid AA \mid a \mid b \mid X_aX_b \\ Z &\rightarrow AA \end{aligned}$$

Each part was worth 2 points. As noted on the exam, each grammar was judged with reference to the preceding one. Answers that had minor mistakes received 1 point.

The correct answer if you kept the redundant nonterminals:

$$\begin{aligned} X_a &\rightarrow a \\ X_b &\rightarrow b \end{aligned}$$

$$\begin{aligned} S &\rightarrow X_aX \mid X_aX_a \mid X_bY \mid X_bX_b \\ X &\rightarrow AX_a \end{aligned}$$

$$Y \rightarrow BX_b$$

$$A \rightarrow CZ \mid CD \mid CA \mid DA \mid a \mid b \mid X_a X_b$$

$$Z \rightarrow DA$$

Productions for $B, C,$ and D are the same as for A .

Problem 6 (8 points)

If $a_1 a_2 \cdots a_n$ is a string, then define

$$\text{star}(a_1 a_2 \cdots a_n) = L(a_1^* a_2^* \cdots a_n^*).$$

If L is a language, define

$$\text{star}(L) = \bigcup_{w \in L} \text{star}(w).$$

Prove or disprove: If L is a CFL, then $\text{star}(L)$ is a CFL. (Note that $\text{star}(L)$ is not the same as L^* .)

3 points were awarded for correctly indicating that CFLs are closed under the star operation. For solutions with a correct answer, an additional 5 points could be earned with a proof.

There were a variety of ways to prove this.

Substitution The easiest solution is to note that CFLs are closed under substitution, and $\text{star}(L)$ is exactly what you get when you substitute for each character a , the CFL (in fact regular) language denoted by the regular expression a^* .

Grammar modification If G generates L , you can modify G by replacing each terminal symbol a with A (assume this is a new nonterminal not appearing in G), and adding the productions $A \rightarrow aA \mid \epsilon$. This in fact is the technique used to prove that CFLs are closed under substitution. Recall that in the general case, we'd let A be the start symbol of a grammar for the language associated with a .

PDA modification This was trickier than the above two methods. The basic idea is to insert some "self-loops" at each state allowing the PDA after reading a character to continue reading any number of the same characters, without modifying the stack or changing state.

The problem with just adding such a loop is that it doesn't ensure that the PDA only uses the loop when it should. That is, suppose that $(q, \alpha) \in \delta(p, a, X)$. Thus, a possible action of the machine on reading a with X on the top of the stack and in state p is to go to state q and replace X with α .

If we add the self-loop transition $(q, Y) \in \delta(q, a, Y)$ for any Y , allowing the machine to eat up as many a 's as it likes, then a word w that uses the transition $\delta(p, a, X) = (q, \alpha)$ that is accepted would still be accepted if we replicate the a any number of times. *However* there may be other transitions that go to state q just after reading *some other* character, say b , of w . Then the machine would allow the insertion of any number of a 's where it should not. A similar problem would occur if we introduced the self-loop at the state p .

The trick is to divert the a -transition from p to q to a *new* intermediate state s . Then s should have a self-loop allowing the machine to eat up as many additional a 's as it likes, without manipulating the stack, and finally an ϵ -transition from s to q .

Finally, note the above doesn't quite work: we need to allow a transition from p to q on ϵ , since a is to be replaced with a^* . The final construction then adds an additional ϵ -transition from p to s , *which manipulates the stack the same way a would have*.

Because there were many ways to go wrong with a machine modification, there was ample opportunity to lose points from a correct solution.

Grading Rubric

- Answers that said “not a CFL” received 0 points. (Kevin - if you see anything that changes your mind about this, let me know.)
- Answers using the “by substitution” approach typically received full credit. A point was deducted for failure to either assert, or show, that the languages \mathbf{a}^* were CFLs (Commenting that they’re regular was sufficient.)
- Answers that said something like: “for each word w , the star operation applied to w gives a CFL (in fact, a regular language), and then when we take the union of these for all w , we get a CFL because CFLs are closed under union” received 0 points (from the 5) because CFLs are not closed under the infinite union that is suggested.
- Answers using the grammar approach lost a point for failure to allow each symbol to be replaced with ϵ . (Note that adding $S \rightarrow \epsilon$ does not solve this problem.) No points were deducted if you started with a CNF grammar and failed to point out that your construction still works if the given L contained the empty string (which was lost when going to CNF).
- Answers using the PDA approach received at least 2 out of the 5 points. An additional 1 point was given if the construction correctly handled the “ ϵ ”-case, and an additional 2 points were given if it correctly handled the non-zero repetition of each character.