

CS273: Theory of Computation, Summer 2008

Homework 4

Due 3:00PM Monday, July 07, 2008 at SC 3303

Revision due 3:00PM Friday, July 11 2008 at SC 3303

1. [40 points; 10 per part] Give a grammar for each of the following languages. For each of your grammars, *explain in words* what each variable does.

(a) $\{x \in \{0, 1\}^* \mid x^R = \bar{x}\}$

Solution:

$$S \rightarrow 0S1 \mid 1S0 \mid \varepsilon$$

S generates all strings w with $w^R = \bar{x}$ by generating pairs of complementary bits from the center. ■

(b) $\{a^m b^n \mid n \neq m\}$

Solution:

$$S \rightarrow aSb \mid aA \mid bB$$

$$A \rightarrow aA \mid \varepsilon$$

$$B \rightarrow bB \mid \varepsilon$$

In this grammar, A generates a^* and B generates b^* . S generates $\{a^m b^n \mid n \neq m\}$ by first generating equal numbers of a 's and b 's from the center, and then forcing the production of one or more a 's or one or more b 's. ■

(c) $\{a^\ell b^m c^n \mid \ell, m, n \text{ are not all equal}\}$

Solution: First, note that if ℓ, m, n are not all equal, then either $\ell \neq m$ or $m \neq n$. (In fact, if you write down any list of numbers that are not all equal, some consecutive pair of numbers must be unequal.) We will design a grammar so that S_1 generates all strings $a^\ell b^m$ with $\ell \neq m$ and S_2 generates all strings $b^m c^n$ with $m \neq n$. Once again, A, B , and C generate a^*, b^* , and c^* respectively.

$$S \rightarrow S_1 C \mid A S_2$$

$$S_1 \rightarrow a S_1 b \mid a A \mid b B$$

$$S_2 \rightarrow b S_2 c \mid b B \mid c C$$

$$A \rightarrow a A \mid \varepsilon$$

$$B \rightarrow b B \mid \varepsilon$$

$$C \rightarrow c C \mid \varepsilon$$

■

(d) $\{a^m b^n \mid m \leq n \leq 2m\}$

Solution:

$$S \rightarrow aSb \mid aSbb \mid \varepsilon$$

In this grammar, S generates a 's and b 's from an interior point in the string. For each a generated, either one or two b 's are generated. Therefore the number of a 's will be at least the number of b 's, but not more than twice the number of b 's. ■

In the first language, \bar{x} is the string obtained by flipping every bit in x . For example $\overline{0010} = 1101$.

2. [15 points] Convert your grammar from problem (1b) into Chomsky normal form. Show all intermediate grammars.

Solution: (a) Add a new start variable.

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow aSb \mid aA \mid bB \\ A &\rightarrow aA \mid \varepsilon \\ B &\rightarrow bB \mid \varepsilon \end{aligned}$$

- (b) Remove ε production rules. The set of nullable variables is $\{A, B\}$.

$$\begin{aligned} S_0 &\rightarrow S \\ S &\rightarrow aSb \mid aA \mid a \mid bB \mid b \\ A &\rightarrow aA \mid a \mid \cancel{\varepsilon} \\ B &\rightarrow bB \mid b \mid \cancel{\varepsilon} \end{aligned}$$

- (c) Remove unit production rules. The only unit production rule is $S_0 \rightarrow S$.

$$\begin{aligned} S_0 &\rightarrow aSb \mid aA \mid a \mid bB \mid b \\ S &\rightarrow aSb \mid aA \mid a \mid bB \mid b \\ A &\rightarrow aA \mid a \\ B &\rightarrow bB \mid b \end{aligned}$$

- (d) Ensure each production's RHS has length at most 2. We introduce a new variable Z_1 with the rule $Z_1 \rightarrow Sb$.

$$\begin{aligned} S_0 &\rightarrow aZ_1 \mid aA \mid a \mid bB \mid b \\ S &\rightarrow aZ_1 \mid aA \mid a \mid bB \mid b \\ A &\rightarrow aA \mid a \\ B &\rightarrow bB \mid b \\ Z_1 &\rightarrow Sb \end{aligned}$$

(e) Add new dummy variables.

$$\begin{aligned}
 S_0 &\rightarrow U_a Z_1 \mid U_a A \mid a \mid U_b B \mid b \\
 S &\rightarrow U_a Z_1 \mid U_a A \mid a \mid U_b B \mid b \\
 A &\rightarrow U_a A \mid a \\
 B &\rightarrow U_b B \mid b \\
 Z_1 &\rightarrow S U_b \\
 U_a &\rightarrow a \\
 U_b &\rightarrow b
 \end{aligned}$$

■

3. [20 points; 10 per part] A grammar is *right-linear* if every production has the form $A \rightarrow xB$ or $A \rightarrow x$, where $x \in \Sigma^*$ and B is a variable. In other words, each production's right-hand side either contains zero variables, or contains *exactly one* variable as the last symbol.

A grammar G is *strongly right-linear* if every production has the form $A \rightarrow aB$ or $A \rightarrow \varepsilon$, where $a \in \Sigma$ is a single character, and B is a variable.

- (a) Show that any right-linear grammar can be converted to an equivalent *strongly* right-linear grammar.

Solution: Let G be a right-linear grammar. We convert G to an equivalent strongly right-linear grammar G' . First, we remove the unit productions $A \rightarrow B$ from G . We do this in the same way that the CNF conversion algorithm removes unit productions from a grammar. Because this step in the CNF conversion algorithm merely copies the RHS of already existing production rules to new production rules, we still have a right-linear grammar.

Note that at this point, every rule in the grammar is of the form $A \rightarrow x$ where $x \in \Sigma^*$ or $A \rightarrow xB$ where $x \in \Sigma^+$. For each rule of the form $A \rightarrow x_1 x_2 \cdots x_n$ with $n \geq 1$, we remove the rule and replace it with the rules

$$\begin{aligned}
 A &\rightarrow x_1 Z_1 \\
 Z_1 &\rightarrow x_2 Z_2 \\
 Z_2 &\rightarrow x_3 Z_3 \\
 &\vdots \\
 Z_{n-1} &\rightarrow x_n Z_n \\
 Z_n &\rightarrow \varepsilon
 \end{aligned}$$

where Z_1, \dots, Z_n are new variables. (Note that if $n = 0$, then the rule is of the form $A \rightarrow \varepsilon$, which is allowed in a strongly right-linear grammar.) Finally, for each rule of the

form $A \rightarrow x_1x_1 \cdots x_nB$ with $n \geq 1$, we remove the rule and replace it with the rules

$$\begin{aligned} A &\rightarrow x_1Z_1 \\ Z_1 &\rightarrow x_2Z_2 \\ Z_2 &\rightarrow x_3Z_3 \\ &\vdots \\ Z_{n-1} &\rightarrow x_nB \end{aligned}$$

where Z_1, \dots, Z_{n-1} are new variables. (Note that in this case $n = 0$ is not possible, as then the rule would be a unit production rule $A \rightarrow B$, and these were removed earlier.) The resulting grammar is strongly right-linear. ■

(b) Prove that if G is a strongly right-linear grammar, then $L(G)$ is regular.

Hint: See page 105 of Sipser.

Solution: Let $G = (V, \Sigma, R, S)$ be a strongly right-linear grammar. We show that $L(G)$ is regular by constructing an NFA M that recognizes $L(G)$. Note that because G is strongly right-linear, every intermediate form in the derivation consists of a string over Σ followed by a single variable in V . We construct M so that reading an input symbol corresponds to applying a production rule of the form $A \rightarrow xB$. Acceptance in the NFA corresponds to applying an ε production rule.

The NFA M maintains a state for each variable in G . The start state of M corresponds to the start variable of G . For each rule $A \rightarrow xB$ in G , the NFA includes a transition from A to B upon reading the input symbol x . The accept states of M are those that correspond to variables A with ε production rules ($A \rightarrow \varepsilon$). Formally, $M = (Q, \Sigma, \delta, q_0, F)$ where

- i. $Q = V$,
- ii. $\delta(A, x) = \{B \mid A \rightarrow xB \text{ is a rule in } R\}$,
- iii. $q_0 = S$, and
- iv. $F = \{A \mid A \rightarrow \varepsilon \text{ is a rule in } R\}$.

■

4. [30 points; 15 per part] Prove that the following languages are not context-free.

(a) $\{w \in \{a, b\}^* \mid w = w^R \text{ and } \#a(w) = \#b(w)\}$

Solution: Let $A = \{w \in \{a, b\}^* \mid w = w^R \text{ and } \#a(w) = \#b(w)\}$. We exhibit a winning strategy in the CFL pumping lemma game for A .

- i. The adversary picks an integer $p \geq 0$.
- ii. We choose the string $s = a^p b^p b^p a^p$. Note that $s \in A$ and $|s| \geq p$, so this is a legal choice for s .
- iii. The adversary splits s into 5 parts $s = wvxyz$ where $|vxy| \leq p$ and $|vy| \geq 1$. Note that because $|vxy| \leq p$, it is not possible that vxy overlaps both the first and last block of a 's.
- iv. For our next move, we will consider two cases.

- A. Suppose that vy consists entirely of a 's or entirely of b 's. In this case, we may choose i to be any integer except 1. (For concreteness, we might as well pick $i = 2$.) Note that because vy is not the empty string and consists entirely of one type of character, changing s to $uv^i xy^i z$ changes the number of occurrences of exactly one of the character types. Therefore $uv^i xy^i z$ cannot contain the same number of a 's and b 's, and so $uv^i xy^i z \notin A$. Thus, we win in this case.
- B. Suppose that vy contains both a 's and b 's. We choose $i = 0$. As we have observed, vxy cannot overlap both the first and last block of a 's. Therefore changing s to $uv^0 xy^0 z$ deletes some a 's from one end of s but leaves all p of the a 's at the other end in place. Also, because $|vxy| \leq p$, at least some of the b 's remain in $uv^0 xy^0 z$. Therefore $uv^0 xy^0 z$ either has the form $a^\ell b^m a^p$ or $a^p b^m a^\ell$ with $\ell < p$ and $m \geq 1$. In either case, $uv^0 xy^0 z$ is not a palindrome, and so it is not a string in A . Therefore we also win in this case.

Because we have a winning strategy in the CFL pumping lemma game for A , it follows that A is not a CFL. ■

- (b) $\{w \in \{a, b, c, d\}^* \mid \#a(w) = \#b(w) \text{ and } \#c(w) = \#d(w)\}$

Solution: Let $A = \{w \in \{a, b, c, d\}^* \mid \#a(w) = \#b(w) \text{ and } \#c(w) = \#d(w)\}$. We exhibit a winning strategy in the CFL pumping lemma game for A .

- i. The adversary picks an integer $p \geq 0$.
- ii. We choose the string $s = a^p c^p b^p d^p$. Note that $s \in A$ and $|s| \geq p$, so this is a legal choice for s .
- iii. The adversary splits s into 5 parts $s = uvxyz$ where $|vxy| \leq p$ and $|vy| \geq 1$. Note that because $|vxy| \leq p$, it is not possible that vxy contains both a 's and b 's. For the same reason, it is not possible that vxy contains both c 's and d 's.
- iv. We choose $i = 0$. Because vy is not the empty string, changing from s to $uv^0 xy^0 z$ removes some characters from s . However, if a 's are removed, then b 's cannot also be removed, and vice versa. Similarly, if c 's are removed, then d 's cannot also be removed, and vice versa. Therefore changing from s to $uv^0 xy^0 z$ disturbs at least one of the equations $\#a(w) = \#b(w)$, $\#c(w) = \#d(w)$. It follows that $uv^0 xy^0 z \notin A$, and so we win.

Because we have a winning strategy in the CFL pumping lemma game for A , it follows that A is not a CFL. ■

5. [Honors; 15 points] Give a grammar for $\{0, 1\}^* \setminus \{ww \mid w \in \{0, 1\}^*\}$. Explain how it works.

Solution: Let $A = \{0, 1\}^* \setminus \{ww \mid w \in \{0, 1\}^*\}$. Note that if z is not of the form $\{ww \mid w \in \{0, 1\}^*\}$, then either z has odd length, or $z = x_1 x_2 \cdots x_n y_1 y_2 \cdots y_n$ such that $x_j \neq y_j$ for some $1 \leq j \leq n$. That is, $A = A_1 \cup A_2$ where $A_1 = \{w \mid w \text{ has odd length}\}$ and $A_2 = \{x_1 x_2 \cdots x_n y_1 y_2 \cdots y_n \mid x_j \neq y_j \text{ for some } j\}$. We construct our grammar so that S_1 generates strings in A_1 and S_2 that generates all strings in A_2 . Generating strings in A_1 is fairly straightforward. Generating strings in A_2 requires a little trick.

Consider a string $s \in A_2$; a typical example is shown below.

$$s = \underbrace{\quad\quad\quad 0 \quad\quad\quad 1 \quad\quad\quad}_{\substack{|-j-1-| \quad |-\quad n-j \quad -||-j-1-| \quad |-\quad n-j \quad -|}}$$

When we view s as above, it seems difficult to design a grammar that generates such strings, because the blocks of length $j-1$ are interleaved with the blocks of length $n-j$. However, because the location of the center of s is not important, we are free to group the symbols of s as shown below.

$$s = \underbrace{\quad\quad\quad 0 \quad\quad\quad 1 \quad\quad\quad}_{\substack{|-j-1-| \quad |-\quad j-1 \quad -||-\quad n-j \quad -| \quad |-\quad n-j \quad -|}}$$

Thus, we can construct a grammar for A_2 by using two variables C and D . We will have C generate strings of the form $x0y$ with $|x| = |y|$ and D generate strings of the form $x1y$ with $|x| = |y|$. Our grammar is as follows.

$$\begin{aligned} S &\rightarrow S_1 \mid S_2 \\ S_1 &\rightarrow 00S_1 \mid 01S_1 \mid 10S_1 \mid 11S_1 \mid 0 \mid 1 \\ S_2 &\rightarrow CD \mid DC \\ C &\rightarrow 0C0 \mid 0C1 \mid 1C0 \mid 1C1 \mid 0 \\ D &\rightarrow 0D0 \mid 0D1 \mid 1D0 \mid 1D1 \mid 1 \end{aligned}$$

■