CS 4114
Solutions to Midterm Exam
March 2, 2000

[30] 1. Consider the following language:

\[ L_1 = \{ w \in \{a, b\}^* \mid n_a(w) = 3 \text{ or } n_b(w) = 5 \}. \]

1. Give examples of 5 strings that are in \( L_1 \) and of 5 strings that are not in \( L_1 \).

2. Give a regular expression that represents \( L_1 \).

---

1. Examples of strings that are in \( L_1 \) include:

\[ aaa \ bbbbb \ abbbba \ abababab \ bababa. \]

Examples of strings that are not in \( L_1 \) include:

\[ \lambda \ aaaa \ bbaba \ abab \ baabaaaaa. \]

2. First rewrite \( L_1 \) as the obvious union:

\[ L_1 = \{ w \in \{a, b\}^* \mid n_a(w) = 3 \} \cup \{ w \in \{a, b\}^* \mid n_b(w) = 5 \}. \]

The first set in this union consists of three \( a \)'s with any number of \( b \)'s interspersed. Hence a regular expression for this set is \( b^*ab^*ab^*ab^* \). Similarly, a regular expression for the second set is \( a^*ba^*ba^*ba^*ba^*ba^* \). Therefore, the following regular expression represents \( L_1 \):

\[ b^*ab^*ab^*ab^* \cup a^*ba^*ba^*ba^*ba^*ba^*. \]

---

[30] 2. Consider the following language:

\[ L_2 = \{ a^ib^jd^k \mid i + j = k + \ell \}. \]

1. Give examples of 5 strings that are in \( L_2 \) and of 5 strings that are not in \( L_2 \).

2. Give a context-free grammar that generates \( L_2 \).
1. Examples of strings that are in $L_2$ include:

$$\lambda \ a\ b\ c\ d \ a\ a\ c\ a\ d\ d\ \ a\ b\ b\ d\ d\ d\ d\ \ a\ c.$$

Examples of strings that are not in $L_2$ include:

$$b\ d\ \ d\ a\ \ d\ e\ a\ c\ c\ d\ c\ \ b.$$

2. Start with the normal assumption that $S$ will be the start symbol of the grammar we construct.

Let $w \in L_2$ be a nonempty string. Clearly $w$ has even length $\geq 0$. The case $w = \lambda$ will be an easy base case, so assume that $|w| \geq 2$. Let $w = \sigma\tau$, where $\sigma, \tau \in \Sigma$ and $x \in \Sigma^*$. There are four cases for $\sigma$ and $\tau$.

(a) $\sigma = a$ and $\tau = d$. Then $x$ could be any element of $L_2$ and this case is handled by the rule $S \rightarrow aSd$.

(b) $\sigma \neq a$ and $\tau \neq d$. Then $x$ is in \{b^i e^k | j = k\} and this case is handled by the rules $C \rightarrow bC e | \lambda$.

(c) $\sigma = a$ and $\tau \neq d$. Then $x = a^i b^j e^k$, where $i + j = k$ and $i > 0$. This is taken care of by the rule $A \rightarrow aA c$.

(d) $\sigma \neq a$ and $\tau = d$. Then $x = b^j e^k d^\ell$, where $j = k + \ell$ and $\ell > 0$. This is taken care of by the rule $B \rightarrow bBd$.

Putting this all together with some chain rules to go from one case to another, we obtain this grammar:

$$S \rightarrow aSd | A | B | C$$

$$A \rightarrow aAc | C$$

$$B \rightarrow bBd | C$$

$$C \rightarrow bCe | \lambda.$$

---

[40] 3. Context-free grammar $G_3$ is given by the following productions:

1. $S \rightarrow ba$

2. $S \rightarrow SbSa$.

1. Show the portion of $g_L(G_3)$, the leftmost graph of $G_3$, searched by a breadth-first top-down parse of the string $u = babaa$. (See, for example, Figure 4.3.)
2. Discuss the appropriateness of $G_3$ (a) for top-down parsing and (b) for bottom-up parsing.

1. Of course, $g_L(G_3)$ corresponds to leftmost derivations in $G_3$. If we apply rule 1 before rule 2 in the breadth-first parse, then we obtain the following portion of $g_L(G_3)$ before successfully parsing $babb_a$.

![Diagram](image)

2. Since rule 2 of $G_3$ displays direct left recursion, $G_3$ is not suited to top-down parsing. For example, unbounded lookahead is needed to determine how many times rule 2 should be applied initially.

Since the right-hand sides of all the rules of $G_3$ have length $\geq 2$, grammar $G_3$ is well suited to bottom-up parsing. Any time a handle is reduced, a shorter string results. Hence bottom-up parsing will always terminate.

[50] 4. Context-free grammar $G_4$ is the following:

$$
S \rightarrow aSA \mid BA \\
A \rightarrow aA \mid a \\
B \rightarrow Bb \mid e.
$$

Convert $G_4$ into an equivalent context-free grammar in Chomsky normal form by following these steps:

1. **Eliminate recursion on the start symbol**;
2. **Eliminate $\lambda$ rules**;
3. **Eliminate chain rules**;
4. **Eliminate useless symbols**; and
5. Attain Chomsky normal form by eliminating non-binary rules.

1. Since S appears on the right-hand side of a rule, we need a new start symbol S' to eliminate recursion on the start symbol. The resulting grammar is

\[
G'_4: \quad S' \rightarrow S \\
S \rightarrow aSA | BA \\
A \rightarrow aA | a \\
B \rightarrow Bb | c.
\]

2. \(G'_4\) has no \(\lambda\) rules, so we continue to the next step with \(G'_4\).

3. \(G'_4\) has one chain rule, \(S' \rightarrow S\). Eliminating that results in the grammar:

\[
G''_4: \quad S' \rightarrow aSA | BA \\
S \rightarrow aSA | BA \\
A \rightarrow aA | a \\
B \rightarrow Bb | c.
\]

4. We use Algorithm 5.3.2. to compute

\[
\text{TERM} = \left\{ A \mid A \xrightarrow{G} w \text{ with } w \in \Sigma^* \right\} = \{S', S, A, B\}.
\]

Since all nonterminals can generate a string of terminals, we cannot eliminate any for that reason.

Then we use Algorithm 5.3.4. to compute

\[
\text{REACH} = \left\{ A \mid S \xrightarrow{G} uAv \right\} = \{S', S, A, B\}.
\]

Again we cannot eliminate any nonterminals, so we continue to the last step with \(G''_4\).

5. Adding nonterminals \(A', B',\) and \(T_1\) with appropriate modifications to the rules, we
obtain a Chomsky normal form grammar that is equivalent to $G_4$:

\[
G'_4 : \quad S' \rightarrow A'T_1 | BA \\
S \rightarrow A'T_1 | BA \\
A \rightarrow A'A | a \\
B \rightarrow BB' | c \\
A' \rightarrow a \\
B' \rightarrow b \\
T_1 \rightarrow SA.
\]