CSCI 2400 – Models of Computation, Section 3

Solutions to Homework 3

Problem 1.

- (b). All strings not ending in 01:

\[ \lambda + 0 + 1 + (0 + 1)^*(00 + 10 + 11). \]

The expression \( \lambda + 0 + 1 \) describes the strings with length zero or one, and the expression \((0 + 1)^*(00 + 10 + 11)\) describes the strings with length two or more.

- (c). All strings containing an even number of 0's:

\[ 1^* + (1^*01^*)^*1^*. \]

The first expression \(1^*\) describes the strings with no 0's. The expression \((1^*01^*)^*1^*\) describes the strings with at least two 0's. You need to notice that any 0 must be followed by a matching 0 and between them there could be zero or more occurrences of 1's.

- (d). All strings having at least two occurrences of the substring 00:

\[ (1 + 0)^*00(1 + 0)^*00(1 + 0)^* + (1 + 0)^*000(1 + 0)^*. \]

The expression \((1 + 0)^*00(1 + 0)^*00(1 + 0)^*\) describes the strings with two separate occurrences of the substring 00. The expression \((1 + 0)^*000(1 + 0)^*\) describes the strings where two 00's appear in the substring 000.

- (f). All strings not containing the substring 101:

\[ 0^*(1^*000^*)^*1^*0^*. \]

Notice that a 1 may be followed by either a 1 or by a 00, and this pattern can be repeated as many times as we want. This pattern is expressed in \((1^*000^*)^*\). The extreme cases where a string can start or end with 0's or contain only 1's are covered by the expressions left and right from the pattern \((1^*000^*)^*\).

Problem 2.

We want to show that the family of regular languages is closed under symmetric difference. All we need to show is that for any two regular languages \(L_1\) and \(L_2\), the language \(L_1 \triangle L_2\) is regular. From the definition of the symmetric difference, (using set diagrams) we observe that:
\[ L_1 \circ L_2 = (L_1 \cup L_2) \cap (L_1 \cap L_2). \]

From Theorem 4.1, we know that the regular languages are closed under union, intersection, and complement. Therefore, we have that the language \( L_1 \circ L_2 \) is regular, as needed.

**Problem 3.**

\[
\begin{align*}
S & \to aaB|\lambda \\
B & \to BB \\
B & \to abS
\end{align*}
\]

The production \( S \to aaB \) corresponds to the first substring \( aa \) in the expression \((aab^*ab)^*\). The variable \( B \) generates the middle \( b^* \) and the last \( ab \). The production \( B \to abS \) implements the outermost star operation.

**Problem 4.**

\[
\begin{align*}
S & \to A_\epsilon|A_o \\
(\text{Both } n \text{ and } m \text{ are even}) \\
A_\epsilon & \to aaA_\epsilon|B_\epsilon \\
B_\epsilon & \to bbB_\epsilon|\lambda \\
(\text{Both } n \text{ and } m \text{ are odd}) \\
A_o & \to aaA_o|aB_o \\
B_o & \to bbB_o|b
\end{align*}
\]

Notice that \( n + m \) is even if either

- both \( n \) and \( m \) are even, or
- both \( n \) and \( m \) are odd.

The strings where both \( n \) and \( m \) are even are generated by the variables \( A_\epsilon \) and \( B_\epsilon \). Here, the production \( A_\epsilon \) generates an even number of \( a \)'s and the production \( B_\epsilon \) generates an even number of \( b \)'s. The strings where both \( n \) and \( m \) are odd are generated in a similar way by the productions \( A_o \) and \( B_o \).

**Problem 5.**

Consider a regular language \( L \). From Theorem 3.4, we know there exists a right-linear grammar \( G \) with \( L(G) = L \). In general, the productions of a right linear grammar have the form

\[ A \to a_1a_2\ldots a_nB \]
We need to transform such kind of productions to productions of the form
$A \rightarrow aB$. To do this we introduce new intermediate variables $B_1, B_2, \ldots$, and we
rewrite the production $A \rightarrow a_1a_2 \ldots a_nB$ as

\[
\begin{align*}
A & \rightarrow a_1B_1 \\
B_1 & \rightarrow a_2B_2 \\
B_2 & \rightarrow a_3B_3 \\
\vdots \\
B_{n-1} & \rightarrow a_nB
\end{align*}
\]

In a similar way we transform productions of the form $A \rightarrow a_1a_2 \ldots a_n$ to
productions of the form $A \rightarrow aA$ and $A \rightarrow a$.

We still need to take care of the extreme cases where in the grammar $G$ there
are rules of the form $A \rightarrow B$ or $A \rightarrow \lambda$. For the case $A \rightarrow B$ we look at all the
productions whose right-hand side end with the variable $A$ and we substitute this
with the variable $B$, then we remove the production $A \rightarrow B$ from the grammar.
We repeat this process until no more rules of this form appear in the grammar.
For the case $A \rightarrow \lambda$ we look at all the productions whose right-hand side end
with the variable $A$ and we substitute this with $\lambda$. We repeat this process until
no more rules of this form appear in the grammar.

**Problem 6.**

We are given two regular grammars $G_1$ and $G_2$. Let’s assume that these are
right-linear grammars. Let $S_1$ be the start variable of $G_1$, and $S_2$ be the start
variable of $G_2$.

For the union, we construct a new grammar $G$ such that $G$ contains all the
productions from $G_1$ and $G_2$ and it has two additional rules $S \rightarrow S_1|S_2$, where
$S$ is the new start variable of $G$. It is easy to see that $G$ will generate all the
strings of grammars $G_1$ and $G_2$, and therefore $L(G) = L(G_1) \cup L(G_2)$.

For the concatenation, we construct a new grammar $G$ from the grammars
$G_1$ and $G_2$ as follows. Find all the productions of $G_1$ that have the form
$A \rightarrow a_1a_2 \ldots a_n$ (these are the productions that produce only terminals). Add
to the right end of the right-hand side of each such production the variable $S_2$, so that these rules are transformed to $A \rightarrow a_1a_2 \ldots a_nS_2$. Now, the grammar
$G$ will consist from the productions of the transformed grammar $G_1$ and the
productions of the grammar $G_2$. The start variable of $G$ is $S_1$. It is easy to see
that using $G$ we can generate strings such that: at the point where the generation
of a substring from grammar $G_1$ finishes, the generation of a substring of $G_2$
starts. Therefore, grammar $G$ generates the language $L(G_1)L(G_2)$.

For the star operation, the construction is similar with the concatenation.
The difference now is that we only have grammar $G_1$, and the transformed
productions are of the form $A \rightarrow a_1a_2 \ldots a_nS_1$. The start variable is $S_1$. We
also add the production $S_1 \rightarrow \lambda$.

The constructions above are for the case where both grammars are right-
linear. The case where the grammars are left-linear is similar. In case where
one grammar is left-linear and the other is right-linear we need to convert the
left-linear grammar to a right-linear. We can do this by applying techniques
similar to Theorems 3.3, 3.4, 3.5 and Exercise 12, Section 2.3.