CSCI 2400, section 3: Models of computation

Solutions to Homework 5

Problem 1.

First, we need to remove the λ-productions. By removing $A \to \lambda$ we obtain the grammar:

$$
\begin{aligned}
&S \to abAB|abB \\
&A \to bAB|bB \\
&B \to BAa|Ba|A|\lambda
\end{aligned}
$$

By removing $B \to \lambda$ we obtain the grammar:

$$
\begin{aligned}
&S \to abAB|abB|abA|ab \\
&A \to bAB|bB|bA|b \\
&B \to BAa|Ba|A|Aa|a
\end{aligned}
$$

Next, we need to remove the unit-productions. By removing $B \to A$ we obtain:

$$
\begin{aligned}
&S \to abAB|abB|abA|ab|abAA \\
&A \to bAB|bB|bA|b|bAA \\
&B \to BAa|Ba|Aa|a|AAa
\end{aligned}
$$

Next, we need to remove useless productions. There are no useless productions.

Finally, transform the grammar to Chomsky normal form. First, we use new variables T_a and T_b for the terminals a and b.

$$
\begin{aligned}
&S \to T_aT_bAB|T_aT_bB|T_aT_bA|T_aT_b|T_aT_bAA \\
&A \to T_bAB|T_bB|T_bA|b|T_bAA \\
&B \to BAT_a|BT_a|AT_a|a|AAT_a \\
&T_a \to a \\
&T_b \to b
\end{aligned}
$$

Last, we break the productions of length more than 2 to smaller productions using new intermediate variables V_i.

$$
\begin{aligned}
&S \to T_aV_1|T_aV_3|T_aV_4|T_aT_b|T_aV_5 \\
&V_1 \to T_bV_2 \\
&V_2 \to AB \\
&V_3 \to T_bB \\
&V_4 \to T_bA
\end{aligned}
$$

$$V_5 \to T_b V_6$$
$$V_6 \to AA$$
$$A \to T_b V_2 | T_b B | T_b A | b | T_b V_6$$
$$B \to B V_7 | B T_a | A T_a | a | A V_7$$
$$V_7 \to A T_a$$
$$T_a \to a$$
$$T_b \to b$$

Problem 2.

(a) The NPDA for language $\{a^n b^{2n} : n \geq 0\}$ is shown in Figure 1. The initial stack symbol is \$. The construction is similar to the one in Example 7.2, on page 184. State q_0 reads the a's, and for each a it pushes two a's in the stack. State q_2 reads the b's, and for each b it pops an a from the stack. The automaton enters the final state q_2 only when the bottom stack symbol \$ is reached, which means that the a's have matched twice as many b's.

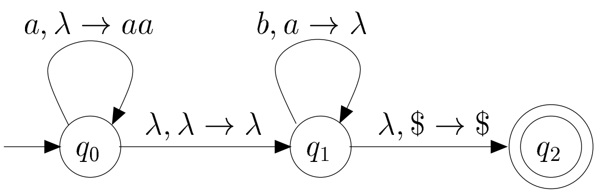

Figure 1: The NDPA for language $\{a^n b^{2n} : n \geq 0\}$

(c) The NPDA for language $\{a^n b^n c^{n+m} : n, m \geq 0\}$ is shown in Figure 2. The initial stack symbol is \$. State q_0 reads the a's and pushes them in the stack. State q_1 reads the b's and pushes them in the stack. State q_2 reads the c's and pops an a or an b from the stack for each c it reads from the input. The automaton enters the final state q_3 only when the bottom stack symbol \$ is reached, which means that the a's and b's have matched the c's.

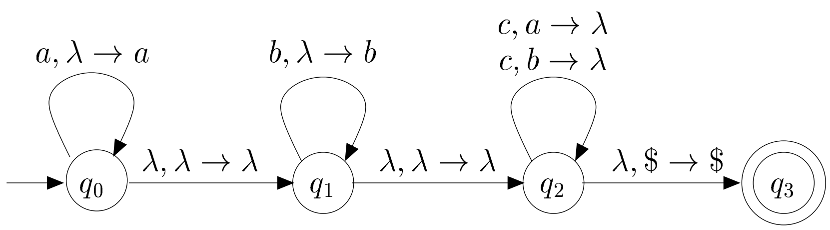

Figure 2: The NDPA for language $\{a^n b^n c^{n+m} : n, m \geq 0\}$

(h) The NPDA for language $\{w : n_a(w) = 2n_b(w)\}$ is shown in Figure 3. The construction is similar to the one in Example 7.3, page 186. For simplicity, we assume that λ is also accepted. The initial stack symbol is \$. State q_0 does all the computation. State q_0 tries to match each a with a b, and each b with an a. If it reads an a from the input and the top of the stack is b it pops the b, otherwise it pushes a on the stack. Similar when it reads b. To make sure that a's are twice as many as b's, whenever it needs to push a b in the stack it pushes two b's instead. When the bottom of the stack is reached it means that all the a's matched the b's and that the a's are twice as many than tha b's, in which case the machine accepts the input string in state q_1 (when all the input has been read). The transition for the c symbol simply consume the c's from the input.

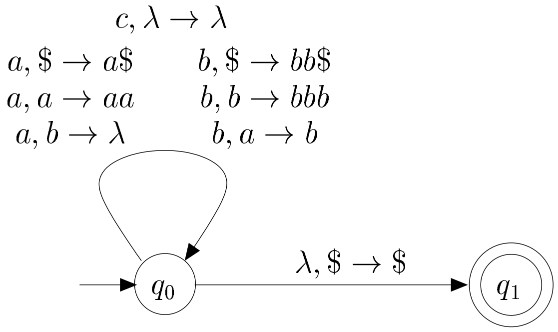

Figure 3: The NDPA for language $\{w : n_a(w) = 2n_b(w)\}$

(i) The NPDA for language $\{w : n_a(w) + n_b(w) = n_c(w)\}$ is shown in Figure 4. For simplicity, we assume that λ is also accepted. The initial stack symbol is \$. State q_0 does the most important computation. State q_0 tries to match each c with either an a or a b. If it reads a c from the input and the top of the stack is a (or b) it pops the a (or b), otherwise it pushes c on the stack. When it reads an a from the input it tries to match it only with a c in the stack, otherwise it pushes the a in the stack. Similar for b. When the bottom of the stack is reached it means that the c's matched the a's and the b's, in which case we can take the transition to final state q_1 and accept the input (if all the input is consumed).

Problem 3.

The NPDA for language $\{w_1 c w_2 : w_1, w_2 \in \{a,b\}^*, w_1 \neq w_2^R\}$ is shown in Figure 5. The initial stack symbol is \$. State q_0 pushes in the stack all the symbols of the string w_1. When the input symbol is c then the state changes to q_1. At this point the current contents of the stack is w_1^R (when we see the stack contents from top to bottom). State q_1 then reads string w_2 and tries to match the string w_2 with the string w_1^R which is stored in the stack. For the matching, when it reads an a from the input it pops and a from the stack, and

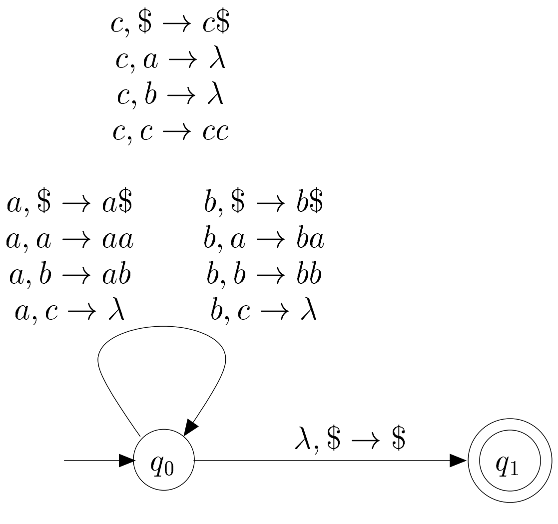

Figure 4: The NDPA for language $\{w : n_a(w) + n_b(w) = n_c(w)\}$

similar for b's. If q_1 finds a mismatch then the state changes to q_2. A mismatch means that $w_1^R \neq w_2$, or equivalently $w_1 \neq w_2^R$. To detect the mismatch, there are transitions from state q_1 to q_2 for each possible mismatch:

Input is a, top of stack is b or \$.

Input is b, top of stack is a or \$.

Input is c.

In state q_2 the rest of the input is consumed and the input is accepted.

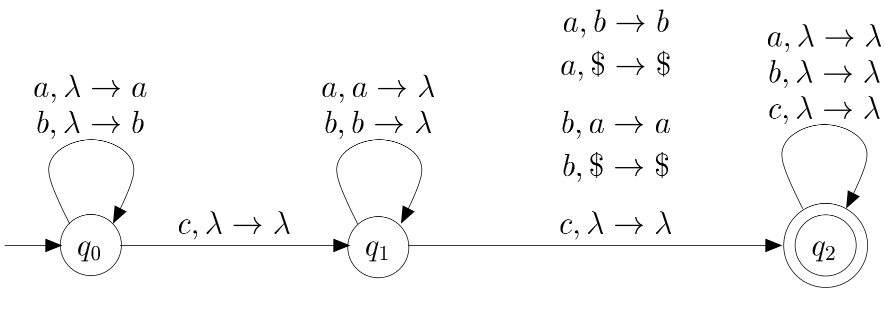

Figure 5: The NDPA for language $\{w_1 c w_2 : w_1, w_2 \in \{a, b\}^*, w_1 \neq w_2^R\}$

Problem 4.

The corresponding NPDA for the grammar is given in Figure 6. The way we obtained this automaton is as described in the class. For each terminal symbol, e.g. a, we add the transition $a, a \to \lambda$ in the loop of q_1. For each production $X \to y$, we add the transition $\lambda, X \to y$ in the loop of q_1.

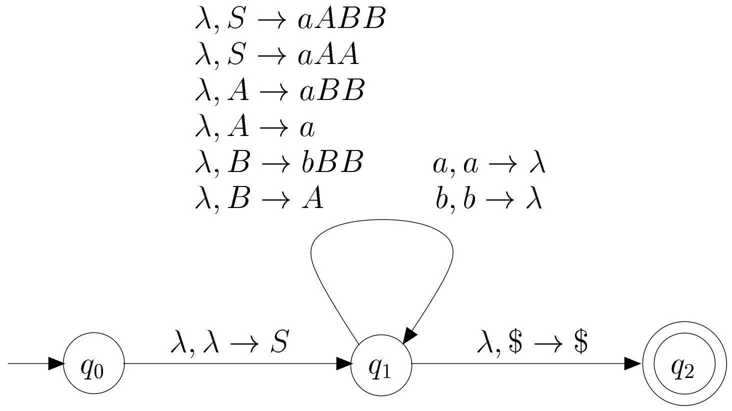

Figure 6: The corresponding NDPA for the grammar

Problem 5.

We will use the procedure described in the class for converting an NPDA to an equivalent grammar. First, we modify the NPDA to an equivalent one where the stack is emptied when the the input is accepted. In Figure 7 we see the original and the modified NPDA. In the resulting automaton all transitions have the desired form $a, A \to BC$ or $a, A \to \lambda$.

Next, we continue by building the grammar for each transition. For transition $a, z \to Az$ from state q_0 to q_0 we have the following productions:

$$(q_0 z q_0) \to a(q_0 A q_0)(q_0 z q_0) \mid a(q_0 A q_1)(q_1 z q_0) \mid a(q_0 A q_f)(q_f z q_0)$$
$$(q_0 z q_1) \to a(q_0 A q_0)(q_0 z q_1) \mid a(q_0 A q_1)(q_1 z q_1) \mid a(q_0 A q_f)(q_f z q_1)$$
$$(q_0 z q_f) \to a(q_0 A q_0)(q_0 z q_f) \mid a(q_0 A q_1)(q_1 z q_f) \mid a(q_0 A q_f)(q_f z q_f)$$

For transition $b, A \to AA$ from state q_0 to q_0 we have the following productions:

$$(q_0 A q_0) \to b(q_0 A q_0)(q_0 A q_0) \mid b(q_0 A q_1)(q_1 A q_0) \mid b(q_0 A q_f)(q_f A q_0)$$
$$(q_0 A q_1) \to b(q_0 A q_0)(q_0 A q_1) \mid b(q_0 A q_1)(q_1 A q_1) \mid b(q_0 A q_f)(q_f A q_1)$$
$$(q_0 A q_f) \to b(q_0 A q_0)(q_0 A q_f) \mid b(q_0 A q_1)(q_1 A q_f) \mid b(q_0 A q_f)(q_f A q_f)$$

For transition $a, A \to \lambda$ from state q_0 to state q_1, we have the following production:

$$(q_0 A q_1) \to a$$

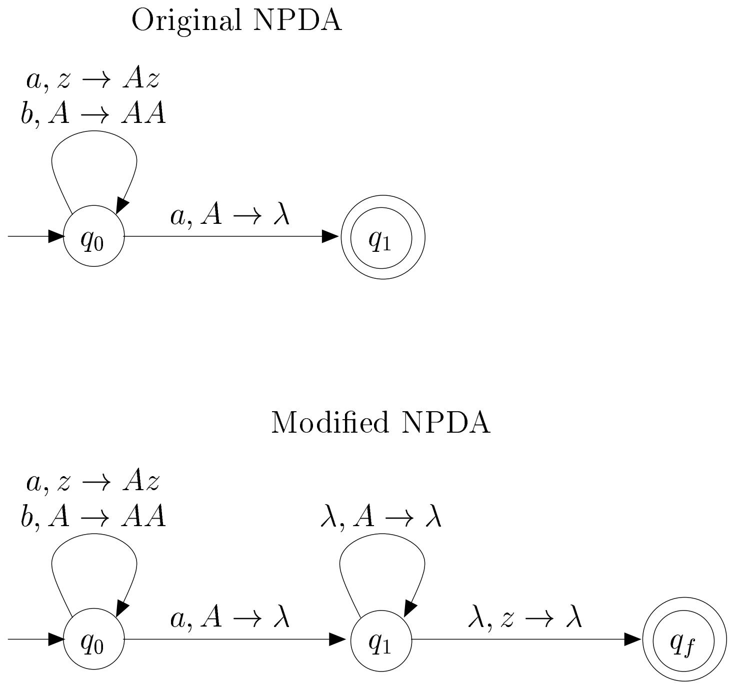

Figure 7: The original and the modified NPDA

For transition $\lambda, A \rightarrow \lambda$ from state q_1 to state q_1, we have the following production:

$$(q_1 A q_1) \rightarrow \lambda$$

Finally, for transition $\lambda, z \rightarrow \lambda$ from state q_1 to state q_f, we have the following production:

$$(q_1 z q_f) \rightarrow \lambda$$

The start variable of the grammar is:

$$(q_0 z q_f)$$

By examining the NPDA it is easy to see that the language accepted is ab^*a. Therefore a string accepted of length four is $abba$. The same string is also generated by the grammar with the following leftmost derivation:

$$\begin{aligned}
(q_0 z q_f) &\Rightarrow a(q_0 A q_1)(q_1 z q_f) \\
&\Rightarrow ab(q_0 A q_1)(q_1 A q_1)(q_1 z q_f) \\
&\Rightarrow abb(q_0 A q_1)(q_1 A q_1)(q_1 A q_1)(q_1 z q_f) \\
&\Rightarrow abba(q_1 A q_1)(q_1 A q_1)(q_1 z q_f) \\
&\Rightarrow abba(q_1 A q_1)(q_1 z q_f) \\
&\Rightarrow abba(q_1 z q_f) \\
&\Rightarrow abba
\end{aligned}$$