Convert the following CFG to a PDA

\[ S \rightarrow aAA \]
\[ A \rightarrow aS \mid bS \mid a \]

The PDA \( P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F) \) is defined as

\[ Q = \{ q \} \]
\[ \Sigma = \{ a, b \} \]
\[ \Gamma = \{ a, b, S, A \} \]
\[ q_0 = q \]
\[ Z_0 = S \]
\[ F = \{ \} \]

And the transition function is defined as:

\[ \delta(q, \epsilon, S) = \{(q, aAA)\} \]
\[ \delta(q, \epsilon, I) = \{(q, aS), (q, bS), (q, a)\} \]
\[ \delta(q, a, a) = \{(q, \epsilon)\} \]
\[ \delta(q, b, b) = \{(q, \epsilon)\} \]
Homework 9

Exercise 6.3.3 Solutions

In the following, $S$ is the start symbol, $e$ stands for the empty string, and $Z$ is used in place of $Z_0$.

1. $S \rightarrow [qZq] \mid [qZp]$

   The following four productions come from rule (1).

2. $[qZq] \rightarrow 1[qXq][qZq]$
3. $[qZq] \rightarrow 1[qXp][pZq]$
4. $[qZp] \rightarrow 1[qXq][qZp]$
5. $[qZp] \rightarrow 1[qXp][pZp]$

   The following four productions come from rule (2).

6. $[qXq] \rightarrow 1[qXq][qXq]$
7. $[qXq] \rightarrow 1[qXp][pXq]$
8. $[qXp] \rightarrow 1[qXq][qXp]$
9. $[qXp] \rightarrow 1[qXp][pXp]$

   The following two productions come from rule (3).

10. $[qXq] \rightarrow 0[pXq]$
11. $[qXp] \rightarrow 0[pXp]$

   The following production comes from rule (4).

12. $[qXq] \rightarrow e$

   The following production comes from rule (5).

13. $[pXp] \rightarrow 1$

   The following two productions come from rule (6).

14. $[pZq] \rightarrow 0[qZq]$
15. $[pZp] \rightarrow 0[qZp]$
Exercise 7.2.1(b)

We will use L to denote the language \( \{a^n b^n c^i \mid i \leq n\} \). For any constant \( n > 0 \), take a string to be \( z = a^n b^n c^n \). Clearly \( z \in L \). Now the string will be decomposed into \( z = uvwxy \), with \( vwx \neq \varepsilon \) and \( |vwx| \leq n \). We then have several cases to consider:

- \( vwx \in a^+ \)
  Pump up, and we will have more a’s than b’s. It does not belong to L.
- \( vwx \in b^+ \)
  Pump up, and we will have more b’s than a’s. It does not belong to L.
- \( vwx \in c^+ \)
  Pump up, and we will have more c’s than a’s and b’s. It does not belong to L.
- \( vwx \in a^+ b^+ \)
  Pump down, and we will have less a’s and b’s than c’s. It does not belong to L.
- \( vwx \in b^+ c^+ \)
  Pump up, and we will have more c’s than a’s. It does not belong to L.

Note that it is impossible to have \( vwx \in a^+ b^+ c^+ \), since \( |vwx| \leq n \). So we have finished the proof that L is not a CFL.

Exercise 7.2.1(d)

Let \( n \) be the pumping-lemma constant and consider \( z = 0^n 1^{n^2} \). We break \( Z = uvwxy \) according to the pumping lemma. If \( vwx \) consists only of 0's, then \( uwy \) has \( n^2 \) 1's and fewer than \( n \) 0's; it is not in the language. If \( vwx \) has only 1's, then we derive a contradiction similarly. If either \( v \) or \( x \) has both 0's and 1's, then \( uv^iwx^{i+1}y \) is not in \( 0^*1^* \), and thus could not be in the language.

Finally, consider the case where \( v \) consists of 0's only, say \( k \) 0's, and \( x \) consists of \( m \) 1's only, where \( k \) and \( m \) are both positive. Then for all \( i \), \( uv^{i+1}wx^{i+1}y \) consists of \( n + ik \) 0's and \( n^2 + im \) 1's. If the number of 1's is always to be the square of the number of 0's, we must have, for some positive \( k \) and \( m \): \( (n+ik)^2 = n^2 + im \), or \( 2ink + i^2k^2 = im \). But the left side grows quadratically in \( i \), while the right side grows linearly, and so this equality for all \( i \) is impossible. We conclude that for at least some \( i \), \( uv^{i+1}wx^{i+1}y \) is not in the language and have thus derived a contradiction in all cases.
Exercise 7.2.1(e)

We will use \( L \) to denote the language \( \{ a^n b^n c^i \mid n \leq i \leq 2n \} \). For any constant \( n > 0 \), take a string to be \( z = a^n b^n c^{2n} \). Clearly \( z \in L \). Now the string will be decomposed into \( z = uvwxy \), with \( vwx \neq \varepsilon \) and \( |vwx| \leq n \). We then have several cases to consider:

- \( vwx \in a^+ \)
  Pump up, and we will have more a’s than b’s. It does not belong to \( L \).

- \( vwx \in b^+ \)
  Pump up, and we will have more b’s than a’s. It does not belong to \( L \).

- \( vwx \in c^+ \)
  Pump up, and we will have more \( 2n \) c’s. It does not belong to \( L \).

- \( vwx \in a^+ b^+ \)
  Pump down, and we will have \( n-1 \) a’s and \( n-1 \) b’s but still \( 2n \) c’s which is not in the range. It does not belong to \( L \).

- \( vwx \in b^+ c^+ \)
  Pump up, and we will have more b’s and a’s. It does not belong to \( L \).

Note that it is impossible to have \( vwx \in a^+ b^+ c^+ \), since \( |vwx| \leq n \). So we have finished the proof that \( L \) is not a CFL.
CS 381 Homework #9 Problem 4

Question 7.4.3

a)

\[
\begin{array}{c|c|c|c}
{S, A, C} & {S, A} & {S, C} & {S, A, C} \\
\{B\} & {B} & {S, C} & {A, C} \\
\{B\} & {S, C} & {B} & {B} \\
\{S, C\} & {S, A} & {S, C} & {S, A} \\
\{A, C\} & {B} & {A, C} & {B} \\
\{A, C\} & {B} & {A, C} & {B} \\
\end{array}
\]

Since S is in the top left box, \textit{ababa} is in the language.

b)

\[
\begin{array}{c|c|c|c}
\{S, C\} & \{S, C\} & \{S, A, C\} & \{S, C\} \\
\{A, S, C\} & \{A, S\} & {B} & {B} \\
\emptyset & \{A, S, C\} & \{A, S\} & {B} \\
\{A, S\} & {B} & {B} & {S, C} \\
\{B\} & {A, C} & {A, C} & {A, C} \\
\end{array}
\]

Since S is in the top left box, \textit{baaab} is in the language.

c)

\[
\begin{array}{c|c|c|c}
\{S, A, C\} & \{S, A\} & \{S, A, C\} & \{S, A, C\} \\
\{S, A, C\} & \{S, A\} & \{S, A, C\} & \{S, A\} \\
\{B\} & {B} & \{S, A, C\} & \{S, A, C\} \\
\{B\} & {S, C} & \{S, A\} & \{S, A\} \\
\{A, C\} & {A, C} & \{B\} & \{A, C\} \\
\end{array}
\]

Since S is in the top left box, \textit{aabab} is in the language.
Let $N_{ijA}$ denote the number of distinct parse trees for substring $a_i \ldots a_j$ of the input $w$, starting from variable $A$ (i.e., with $A$ as the root of the parse tree). Note that we are using $A$ here as a metavariable, not any particular variable in $G$ that might have been named $A$. $N_{1nS}$, where $n = |w|$ and $S$ the starting variable of $G$, is the value we are interested in. We can augment the CYK algorithm to compute each $N_{ijA}$ as we compute the corresponding $X_{ij}$. That is, after computing $X_{ij}$ in CYK, we proceed to compute $N_{ijA}$ for each variable $A$.

Initially, we set all $N_{ijA}$ to 0.

For the base case, we can compute the first row of $N$ as follows. $N_{iiA}$ is 1 if $A \rightarrow a_i$ is a production of $G$. Otherwise, $N_{iiA}$ remains 0.

To compute $N_{ijA}$, $j - i > 0$, we look at each of the pairs $(X_{ii}, X_{i+1,j}), \ldots, (X_{i,j-1}, X_{jj})$ the same way plain CYK did. For each pair, we look at each element of the cross product of that pair. That is, for $(X_{ik}, X_{k+1,j})$, we consider all pairs $(B, C)$ such that $B \in X_{ik}$ and $C \in X_{k+1,j}$. If $A \rightarrow BC$ is a production, we increment $N_{ijA}$ by $N_{ikB} \times N_{k+1,jC}$.

When the algorithm completes, $N_{1nS}$ would contain the solution.

For the special case when $w = \varepsilon$, this algorithm won’t work, but the answer is easy. It’s 1 if $S \rightarrow \varepsilon$ is a production, 0 otherwise.