Exercise 1: Regular expressions (3 points)

If you use a calculator, floating point numbers are commonly illustrated in the following form:

-1.23456789e-123

Show that the language of floating point numbers in calculator notation is regular, by giving a regular expression for it.

Solution

Let $D := \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ be the set of decimal digits and $\Sigma := \{-, , ., \} \cup D$. Then one possible representation of the above (intentionally ambiguously) defined language over $\Sigma$ is given by the following regular expression:

$$(\varepsilon \cup -)D^+ (\varepsilon \cup (, D^+))(\varepsilon \cup (E(\varepsilon \cup -)D^+)).$$

Exercise 2: Context-Free Grammars (3+3+0 points)

Show that the following languages over the alphabet $\Sigma = \{a, b\}$ are context-free by giving a corresponding context-free grammar (CFG).

(a) $L_1 := \{w \in \Sigma^* \mid$ the length of $w$ is odd$\}.$

(b) $L_2 := \{a^mb^na^n \mid m, n \geq 1\}.$

(c) $L_3 := \Sigma^* \setminus \{a^n b^n \mid n \geq 0\}$ (voluntary, no points!).

Solution

(a) $S \rightarrow aT \mid bT, \ T \rightarrow aU \mid bU \mid \varepsilon, \ U \rightarrow aT \mid bT.$

(b) $S \rightarrow TU, \ T \rightarrow aT \mid a, \ U \rightarrow bUa \mid ba.$

(c) $S \rightarrow 1A \mid A0 \mid 0S1, \ A \rightarrow 0A \mid 1A \mid \varepsilon.$

For a string $s$ to be in the language, it must either: (1) $s$ starts with 1; or (2) $s$ ends with 0; or (3) $s$ starts with 0 and ends with 1 but the intermediate part is in the language.

Exercise 3: Chomsky Normal Form (5 points)

Convert the following CFG into an equivalent CFG in Chomsky Normal Form (CNF). Write down the grammar you obtain after each step of the conversion algorithm.

$$A \rightarrow BAB \mid B \mid \varepsilon$$

$$B \rightarrow 00 \mid \varepsilon$$
Solution

Step one: add a new starting state.

\[
\begin{align*}
S & \rightarrow A \\
A & \rightarrow BAB | B | \epsilon \\
B & \rightarrow 00 | \epsilon
\end{align*}
\]

Step two: remove the \( \epsilon \) rule.

\[
\begin{align*}
S & \rightarrow A | \epsilon \\
A & \rightarrow BAB | B | AB | BA | BB \\
B & \rightarrow 00
\end{align*}
\]

Step three: remove unit rule.

\[
\begin{align*}
S & \rightarrow BAB | 00 | AB | BA | BB | \epsilon \\
A & \rightarrow BAB | 00 | AB | BA | BB
\end{align*}
\]

Step four: convert long rule.

\[
\begin{align*}
S & \rightarrow CB | 00 | AB | BA | BB | \epsilon \\
A & \rightarrow CB | 00 | AB | BA | BB \\
B & \rightarrow 00 \\
C & \rightarrow BA
\end{align*}
\]

Final step.

\[
\begin{align*}
S & \rightarrow CB | DD | AB | BA | BB | \epsilon \\
A & \rightarrow CB | DD | AB | BA | BB \\
B & \rightarrow DD \\
C & \rightarrow BA \\
D & \rightarrow 0
\end{align*}
\]

Exercise 4: Cocke-Younger-Kasami Algorithm (3+3 points)

Consider this CFG in CNF \( G := (V := \{S, A, B, C, X, Y, Z\}, \Sigma := \{a, b, c\}, R, S) \) with \( R \) given by:

\[
\begin{align*}
S & \rightarrow XY \\
A & \rightarrow a \\
B & \rightarrow b \\
C & \rightarrow c \\
X & \rightarrow AB | AZ \\
Y & \rightarrow c | CY \\
Z & \rightarrow XB
\end{align*}
\]

For \( w := x_1 \cdots x_n \) with \( x_1, \ldots, x_n \in \Sigma \) we define \( T_{i,j} \) with \( 1 \leq i \leq n, 1 \leq j \leq n - i + 1 \) as the set of those variables of \( V \) from which we can derive \( x_i \cdots x_{i+j-1} \) (a substring of \( w \) which starts at the \( i \)-th symbol of \( w \) and has length \( j \)). I.e. \( T_{i,j} := \{X \in V \mid X \rightarrow^* x_i \ldots x_{i+j-1}\} \).

It holds that \( w \in L(G) \) if and only if \( S \in T_{1,n} \). The Cocke-Younger-Kasami (CYK) algorithm computes the sets \( T_{i,j} \) (and stores them in a table \( T[1..n,1..n] \)) for a given CFG in CNF and input string \( w := x_1 \cdots x_n \). Its running time is polynomial in \( n \) (\( O(n^3) \) to be more specific).
Use the CYK-algorithm (manually) to test whether or not the following strings are in \( L(G) \):

\[ w_1 := aabbcc \text{ and } w_2 := aaabc. \]

*Hint: Draw the table \( T[1..n, 1..n] \) and note down your intermediate results. The entries \( T[i,j] \) with \( j > n - i + 1 \) are not required for this algorithm (you can mark them accordingly).*

**Algorithm 1 CYK-algorithm**

**Input:** String \( w = x_1 \cdots x_n \), CFG \( G := (V, \Sigma, R, S) \) in CNF.

1. for \( i := 1 \) to \( n \) do
2. \( T[i, 1] := \{ X \in V \mid X \rightarrow x_i \} \)
3. end for
4. for \( j := 2 \) to \( n \) do
5. for \( i := 1 \) to \( n - (j - 1) \) do
6. \( T[i,j] := \emptyset \)
7. for \( k := 1 \) to \( j - 1 \) do
8. \( T[i,j] := T[i,j] \cup \{ X \in V \mid X \rightarrow YZ \in R \text{ and } Y \in T[i,k] \text{ and } Z \in T[i+k,j-k] \} \)
9. end for
10. end for
11. end for

**Solution**

The tables \( T \) we obtain for the given input strings \( w_1, w_2 \) are shown in Table 1 and Table 2. According to this result \( w_1 \in L(G) \) since \( S \in T[1,6] \) and \( w_2 \notin L(G) \) since \( S \notin T[1,6] \).

![Table 1: Result of the CYK-Algorithm with input \( w_1 = aabbcc \).](image)

![Table 2: Result of the CYK-Algorithm with input \( w_2 = aaabc \).](image)