1 Regular Expressions

1. (a) Write a regex that matches binary strings divisible by 8.
   Answer: \([01]*000\mid0\)

   (b) Negate the regex above. That is, write a regex that matches binary strings not divisible by 8.
   Answer: \([01]*10?0?\)

   (c) Given a number in unary, write a regex that matches only non-prime numbers. You may find back referencing helpful.
   Note: In unary, “2” is represented as “111”, “5” as “11111”.
   Answer: \((11+)\1+\1\)
   Note that a non-prime number can be decomposed as \(a \times b\), where \(a\) and \(b\) are greater than 1. Alternatively, we can think of this as \(a + a + \ldots + a\) (adding \(a\) to itself \(b\) times).
   In unary, addition is simply concatenation! Then, we just want to check if a unary number can be decomposed into \(b\) copies of length \(a\). For example, \(8 = 11111111 = 1111111 = 4 \times 2\), so 8 is not prime.

2 Finite State Automata

1. (a) Write the corresponding NFA for the regular expression \((0\mid1)^? (10\mid01)^+\)
   Answer:
3 Grammar Rewriting

Consider the simple ambiguous grammar we saw last time:

\[
\begin{align*}
S & : E \rightarrow \\
E & : E + E \\
& \quad | \quad E \ast E \\
& \quad | \quad \text{ID}
\end{align*}
\]

1. Show that the grammar is ambiguous with two different leftmost derivations of the string \(a + b \ast c\)

   Answer:

   2. \(S \rightarrow E \ast E \rightarrow E + E \ast E \rightarrow a + E \ast E \rightarrow a + b \ast E \rightarrow a + b \ast c\)

   3. \(S \rightarrow E + E \rightarrow a + E \rightarrow a + E \ast E \rightarrow a + b \ast E \rightarrow a + b \ast c\)

4. Rewrite this grammar so that it preserves the standard order of operations, is LL(1), and is unambiguous. Draw the resulting tree for the string \(a + b \ast c\).

   Answer:

   \[
   \begin{align*}
S & : E \rightarrow \\
E & : T E' \\
E' & : \epsilon \\
& \quad | \quad + E \\
T & : F T' \\
T' & : \epsilon \\
& \quad | \quad \ast T \\
F & : \text{ID}
\end{align*}
\]

5. Write down the equivalent unambiguous grammar that enforces both left associativity and correct precedence. Why can’t this be achieved with an LL(1) grammar?
Answer: This can’t be achieved with an LL(1) because it introduces left recursion

\[
\begin{align*}
S & : E \rightarrow \\
E & : E + T \\
& | T \\
T & : T * F \\
& | F \\
F & : ID
\end{align*}
\]

4 LR Parsing

1. Here are the item sets to the grammar from 3.3.
(Recall that an LR(1) item is of the form \( X : bc, a \), which says that we are trying to find \( X \) by production \( bc \) such that \( X \) is followed by nonterminal \( a \) and that we have accumulated \( b \) on the parsing stack)

\[
\begin{align*}
I_0 & \\
S & : \bullet E, \rightarrow \\
E & : \bullet E + T, \rightarrow/+
\end{align*}
\]

\[
\begin{align*}
I_2 & \\
E & : T \bullet, \rightarrow/+
\end{align*}
\]

\[
\begin{align*}
I_3 & \\
T & : \bullet T * F, \rightarrow/+/*
\end{align*}
\]

\[
\begin{align*}
I_4 & \\
F & : \bullet ID, \rightarrow/+/*
\end{align*}
\]

\[
\begin{align*}
I_5 & \\
E & : E + \bullet T, \rightarrow/+*
\end{align*}
\]

\[
\begin{align*}
I_6 & \\
T & : T * \bullet F, \rightarrow/+/*
\end{align*}
\]

\[
\begin{align*}
I_7 & \\
E & : E + T \bullet, \rightarrow/+*
\end{align*}
\]

\[
\begin{align*}
I_8 & \\
T & : T * \bullet F, \rightarrow/+/*
\end{align*}
\]

\[
\begin{align*}
I_9 & \\
F & : ID \bullet, \rightarrow/+/*
\end{align*}
\]

2. Fill the LR(1) parse table from the item sets.

Answer:
Production numbers for the DFA:

1. \( E : E + T \)
2. \( E : T \)
3. \( T : T \* F \)
4. \( T : F \)
5. \( F : ID \)

3. Use the LR(1) parse table to extract the reverse rightmost derivation of the string \( a + a \* a \).

**Answer:**

\[
\begin{align*}
& * \ a + a \* a \rightarrow \\
& a \ * \ a + a \* a \rightarrow \\
& F \ * \ a + a \* a \rightarrow \\
& T \ * \ a + a \* a \rightarrow \\
& E \ * \ a + a \* a \rightarrow \\
& E + \ * \ a + a \* a \rightarrow \\
& E + a \ * \ a \* a \rightarrow \\
& E + F \ * \ a \* a \rightarrow \\
& E + T \ * \ a \* a \rightarrow \\
& E + T \* a \* a \rightarrow \\
& E + T \* F \* a \* a \rightarrow \\
& E + T \* \rightarrow \\
& {E} \\
\end{align*}
\]