# CS 320
# Computer Language Processing
# Exercise Set 3

March 19, 2025

**Exercise 1** If $L$ is a regular language, then the set of prefixes of words in $L$ is also a regular language. Given this fact, from a regular expression for $L$, we should be able to obtain a regular expression for the set of all prefixes of words in $L$ as well.

We want to do this with a function prefixes that is recursive over the structure of the regular expression for $L$, i.e. of the form:

$$\begin{aligned}
\text{prefixes}(\epsilon) &= \epsilon \\
\text{prefixes}(a) &= a \mid \epsilon \\
\text{prefixes}(r \mid s) &= \text{prefixes}(r) \mid \text{prefixes}(s) \\
\text{prefixes}(r \cdot s) &= \ldots \\
\text{prefixes}(r^*) &= \ldots \\
\text{prefixes}(r^+) &= \ldots
\end{aligned}$$

1. Complete the definition of prefixes above by filling in the missing cases.
2. Use this definition to find:
   (a) prefixes($ab^*c$)
   (b) prefixes($(a \mid bc)^*$)

**Exercise 2** Compute nullable, first, and follow for the non-terminals $A$ and $B$ in the following grammar:

$$\begin{aligned}
A &::= BAa \\
A &::= \\
B &::= bBc \\
B &::= AA
\end{aligned}$$

Remember to extend the language with an extra start production for the computation of follow.

**Exercise 3** Given the following grammar for arithmetic expressions:

$$
\begin{aligned}
S &::= Exp\ \textbf{EOF} \\
Exp &::= Term\ Add \\
Add &::= +\ Term\ Add \\
Add &::= -\ Term\ Add \\
Add &::= \\
Term &::= Factor\ Mul \\
Mul &::= *\ Factor\ Mul \\
Mul &::= /\ Factor\ Mul \\
Mul &::= \\
Factor &::= \textbf{num} \\
Factor &::= (Exp)
\end{aligned}
$$

1. Compute nullable, first, follow for each of the non-terminals in the grammar.
2. Check if the grammar is LL(1). If not, modify the grammar to make it so.
3. Build the LL(1) parsing table for the grammar.
4. Using your parsing table, parse or attempt to parse (till error) the following strings, assuming that **num** matches any natural number:
   (a) $(3 + 4) * 5$ **EOF**
   (b) $2 + +$ **EOF**
   (c) $2$ **EOF**
   (d) $2 * 3 + 4$ **EOF**
   (e) $2 + 3 * 4$ **EOF**

**Exercise 4** Argue that the following grammar is *not* LL(1). Produce an equivalent LL(1) grammar.

$$E ::= \textbf{num} + E \mid \textbf{num} - E \mid \textbf{num}$$

**Exercise 5** Consider the following grammar:

$$S ::= S(S) \mid S[S] \mid () \mid [\,]$$

Check whether the same transformation as the previous case can be applied to produce an LL(1) grammar. If not, argue why, and suggest a different transformation.