Exercises for chapter: Parsing

1. Show that the second condition in the definition of an NFA can be reduced to the first. Is a reduction the other way possible?
2. Show how this proof can be modified to use left-recursive grammars, that is, grammars that have productions of the form $N_i \to N_{i'}a$.
3. One could also define the output with a mapping $S \to O$. Show that the definitions are equivalent.
4. Write a DFA that can parse Fortran arithmetic expressions. In Fortran, exponentiation is written like `2**n`. It is also not allowed to have two operators in a row, so 2×-3 is notated `2*(-3)`.
5. Give the states and transitions for the grammar

   S $\longrightarrow$ x
   S $\longrightarrow$ (L)
   L $\longrightarrow$ S
   L $\longrightarrow$ L S

   Apply the above parsing algorithm to the string `(x,x,(x))`.
6. Rewrite the grammar of the second example to eliminate the dangling else problem.
7. In case of a shift-reduce conflict, yacc shifts. Write an example that proves this. Show what this strategy implies for the dangling else problem.