Regular grammars and tokenizing

- Regular grammars (and regular expressions) are powerful enough to describe the basic tokens of a language
- Tokens include the symbols, operators, keywords, identifiers, and literals of a language (e.g. rules for a valid integer, rules for a valid identifier, list of valid operators, etc)
- You have most likely encountered regular expressions, and different languages' syntax for them, multiple times already

Regular grammar rules

- Will leave the proofs/formality for CSCI 320, in 330 we're interested in the application
- First we need an alphabet: the basic list of characters or symbols that are valid in our language
- Then we need rules for combining those symbols into valid tokens
- We'll use lex-style syntax for our regular grammar rules, and simply assume our alphabet is the set of ascii characters

Grammar rules

- We can describe a keyword or operator made up of a fixed character sequence with a text string, e.g. “void”, “+”, “==”
- specify any one of a set of characters using square brackets, e.g [aeiou] matches any one of a, e, i, o, or u
- specify any character in a range is valid using square brackets, e.g. [a-f], [0-9], [a-zA-Z0-9_]
- Specify anything except a set or range of characters is valid by using ^ inside [ ], e.g. [^a-z], [^aeiou]

Pattern repetition

- We can specify a pattern can repeat a certain number of times

  (pattern)? means 0 or 1 times

  (pattern)* means 0 or more times

  (pattern)+ means 1 or more times

  (pattern){m,n} means m to n times, inclusive

- We can specify either of a choice of patterns is valid

  (pattern1) | (pattern2)

Examples

- A positive integer is one or more digits

  [0-9]+

- An identifier begins with either an underscore or an alphabetic character, and is then followed by any number of alphanumeric characters or underscores

  [a-zA-Z_][a-zA-Z0-9_]*

- The logical and operator is &&

  “&&”

Ambiguity

- What if our tokens overlap?
- e.g. if we specify “foo” is a keyword in the language but variable names can be anything alphabetic
- Either (a) we ensure our grammar rules are constructed so they don’t overlap, or (b) the tools have a fixed order to apply the grammar rules – e.g. check if it’s a keyword first, and if not then check the other possibilities