Lexing, RegEx, Automata Exercises
CS 364 — Spring 2022

These review exercises asks you to prepare answers to questions on regular languages and finite automata. Each of the questions has a short answer. You may discuss these exercises with other students and work on the problems together.

1 Definitions and Background

1. Define the following terms and give examples where appropriate.
   (a) lexeme:
   (b) token:
   (c) alphabet:
   (d) language over an alphabet:
   (e) regular language:
   (f) maximal munch rule:
   (g) lexical analyzer generator:
   (h) deterministic finite automaton:
   (i) nondeterministic finite automaton:
   (j) finite automaton acceptance:

2. What are the stages of an interpreter? What data types are passed between these stages?

3. What differences are there between a compiler and an interpreter?

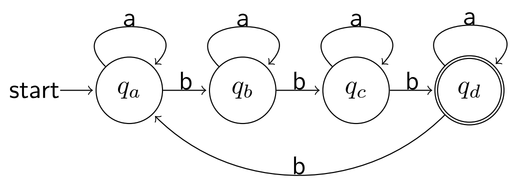
2 Regular Languages and Regular Expressions

1. Write a regular expression to match each of the following.
   - An RGB color: three comma-separated integers enclosed in parentheses
   - A Java variable name: a sequence of lowercase letters, upper case letters, numbers and underscores that does not begin with a number.

2. How can a character class be represented using only single match (a), empty match (ε), concatenation (`AB`), union (`A|B`), and Kleene star (`A*`)?

3. Determine whether or not the following languages are regular. Explain why in one or two sentences.
   - L_1 is all strings over the alphabet $\{(,)\}$ where the parentheses are balanced. For example, $(()(()))\in L_1$ but $(()\notin L_1$.
   - L_2 is all unique words that are printed in *Programming Language Pragmatics* by Michael L. Scott.
   - L_3 is all 10-digit numbers that are prime.
   - L_4 is the Reason language (as described in its reference manual). The alphabet is the set of all *tokens* and the language is the set of all valid Reason programs. **Hint:** Your answer should not be *YES*. can you think of two reasons why? **Aside:** This explains why we cannot use a lexer to *parse* languages like `snail` or Python or C.

4. Consider the following DFA over the alphabet $\Sigma = \{\mathsf{a}, \mathsf{b}\}$.



Give a one-sentence description of the language recognized by the DFA. Write a regular expression for the same language.

3 Finite Automata

1. Consider the following languages over the alphabet $\Sigma = \{a, b\}$.

   - L_1 : All strings that contain at least three a's.
   - L_2 : All strings that contain at most one b.
   - L_3 : All strings that contain at least three a's but at most one b.
   - L_4 : All strings that contain no b's.

   Aside: This example illustrates that regular languages are closed under intersection. Note that $L_3 = L_1 \cap L_2$.

   (a) For each of the languages L_1, L_2, L_3 and L_4, give a regular expression.

(b) For each of the languages L_1, L_2, L_3 and L_4, give a nondeterministic finite automaton (NFA). (You should thus give four separate NFAs.)

(c) For each of the languages L_1, L_2, L_3 and L_4, give a deterministic finite automaton (DFA). (You should thus give four separate DFAs.)

2. Consider the following languages:

   - L_1 is all strings over the alphabet $\Sigma = \{x, y\}$ where either x occurs an odd number of times or y occurs an odd number of times (or both).
   - L_2 is all strings over the alphabet $\Sigma = \{x, y, z\}$ where either x occurs an odd number of times or y occurs an odd number of times or z occurs an odd number of times (or both, or all three).

   Give a non-deterministic finite automaton (NFA) for the the languages L_1. Then give a separate NFA for L_2.

   Aside: Non-deterministic finite automata are no more powerful than DFAs in terms of the languages they can describe. They can be exponentially more succinct than DFAs, however.