Homework 3

CptS 317, Spring 2021

Due Date: February 22, 2021 by 11:59pm Pacific.

To be submitted on Canvas.

1. Give English descriptions of the languages represented by the following regular expressions. Descriptions should be as simple as possible, but should avoid merely spelling out the conditions of the regular expression. As an example, if the regular expression is $0(0 \cup 1)^*$, an answer such as "The language of all binary strings starting with 0" would be ideal, while "The language of all binary strings where the first symbol is 0 and it is followed by an arbitrary number of 0s or 1s" might receive less points. Both are technically correct, but the former is easier to comprehend.

   a) $(0 \cup 1)^*1(0 \cup 1) \cup (0 \cup 1)^*1(0 \cup 1)(0 \cup 1)$

   b) $(c \cup e \cup g)^*e(c \cup e \cup g)^*g(c \cup e \cup g)^* \cup (c \cup e \cup g)^*g(c \cup e \cup g)^*e(c \cup e \cup g)^*$

2. Give regular expressions generating the following languages. In all cases, the alphabet is $\{0, 1\}$:

   a) $\{w | w$ begins with a 1 and ends with a $0\}$

   b) $\{w | w$ doesn't contain the substring $110\}$

   c) $\{w | w$ contains an even number of 0s, or exactly two 1s$\}$

   d) All strings except the empty string

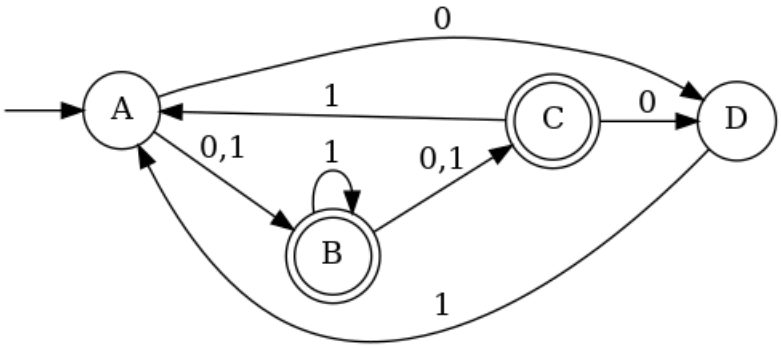
3. Let X, and Y be any two regular expressions. State true or false for the following. If your answer is false, give a counterexample.

   a) $(\epsilon \cup X)^* Y = X^* Y$

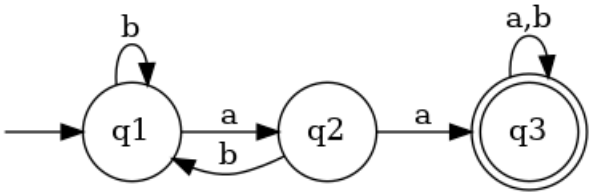
   b) $(X \cup Y)^* Y^* = (X^* Y)^*$

   c) $Y(XY \cup X)^* = (YX \cup X)^* X$

4. Convert the following NFA into an equivalent DFA:



5. Convert the following DFA into a 2-state GNFA using the iterative process discussed in class and in the textbook (i.e. first turn into a 5-state GNFA, then remove one state at a time)



6. JH-Lisp is a simple programming language for evaluating arithmetic expressions. It is described from the following components:

   (a) The alphabet of JH-Lisp consists of the open `'('` and close `')'` parentheses, the digits `0-9`, the space character `' '`, and the symbols `+ - * /`

   (b) An **operator** is a substring in JH-Lisp containing one of the symbols `+ - * /`.

(c) A **number** is a substring in JH-Lisp containing one or more of the digits `0-9`

(d) An **expression** is defined as either a **number** or a **list expression**

(e) A **list expression** is defined by the following, in order:

i. An open parenthesis: `'('`

ii. An **operator**

iii. A space: `' '`

iv. An **expression**

v. A space: `' '`

vi. An **expression**

vii. A close parenthesis: `')'`

(f) A string in the JH-Lisp language is a single **list expression**.

Examples of JH-Lisp strings may include:

```
(* 2 3)
```

```
(+ (* 5 3) (/ 8 4))
```

```
(- 25 (+ 4 2))
```

Is JH-Lisp a regular language? Why or why not? If you think it is, provide a regular expression or finite automaton which recognizes the language. If you think it is not, provide an explanation as to why no regular expression or finite automaton should be able to recognize it.