CSE 501 A. Compiler Construction
Homework #1.

Jae Dong Hwang(0726655)
Jaedong@uw.edu

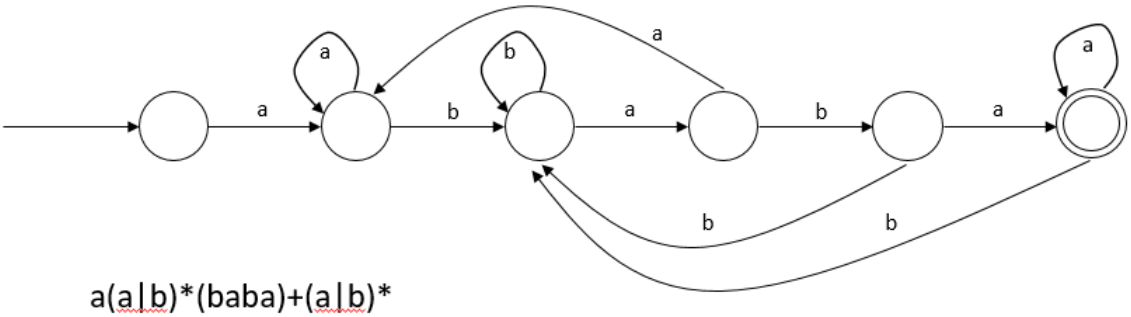
1) For each of the following regular expressions, (i) give an example of two strings that can be generated by the regular expression and two that use the same alphabet but cannot be generated, and (ii) give an English description of the set of strings generated (for example, "all strings consisting of the word 'cow' followed by 1 or more 'x's and 'o's in any order", not just a transliteration of the regular expression operations into English).
    a) (a|xy)*
        i) CAN: axy, xya, aa, or xyaaa, CANNOT: xay, ayx
        ii) All strings consisting of 0 or more of ‘a’s and/or ‘xy’s
    b) b(oz)+o
        i) CAN: bozo, bozozo, CANNOT: bo, bzoo
        ii) All strings starting with ‘b’ followed by one or more of ‘oz’s that followed by ‘o’ at the end.
    c) ((ε|0)1)*
        i) CAN: 01, 0101, 00101, 10111 CANNOT: 10, 001
        ii) Empty or a sequence of sequence digits with zero or more of leading 0s followed by 1.

2) Give regular expressions that generate the following sets of strings.
    a) All strings of a's and b's with at least 3 a's.
        i) (b*a+b*)(b*a+b*)(b*a+b*)+
    b) All strings of a's and b's where b's only appear in sequences whose length is a multiple of 2 (a few examples: abba, bbbbabbaaa, a and ε are in this set; aba, b, ababa, and abbab are not).
        i) (a*bba*)*
    c) All strings of lower-case letters that contain the 5 vowels (aeiou) exactly once and in that order, with all other possible sequences of lower-case letters before, after, or in between the individual vowels.
        i) ((allexcept_aeiou)*a(allexcept_aeiou)*e(allexcept_aeiou)*i(allexcept_aeiou)o*(allexcept_aeiou)*u(allexcept_aeiou)*)*

3) (Cooper & Torczon exercise 2 for section 2.2, parts (a) and (b) only. p. 80). Construct a DFA accepting each of the following languages:

a) {*w* in {a,b}* | *w* starts with 'a' and contains 'baba' as a substring}

Homework#1. Problem #3(1)

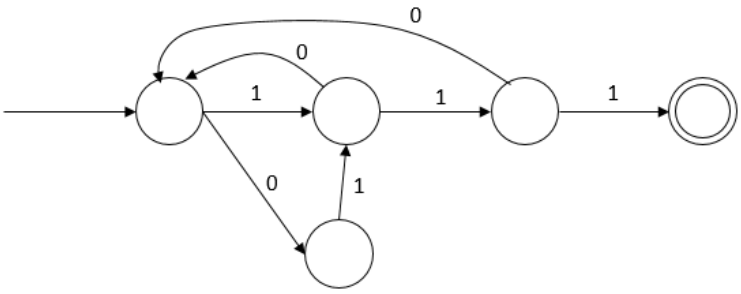


a(a|b)*(baba)+(a|b)*

b) {*w* in {0,1}* | *w* contains '111' as a substring and does not contain '00' as a substring }

You do not need to go through the full subset construction to produce this DFA from a NFA, although you can use some of those ideas to help you produce your answer.

Homework#1. Problem #3(2)



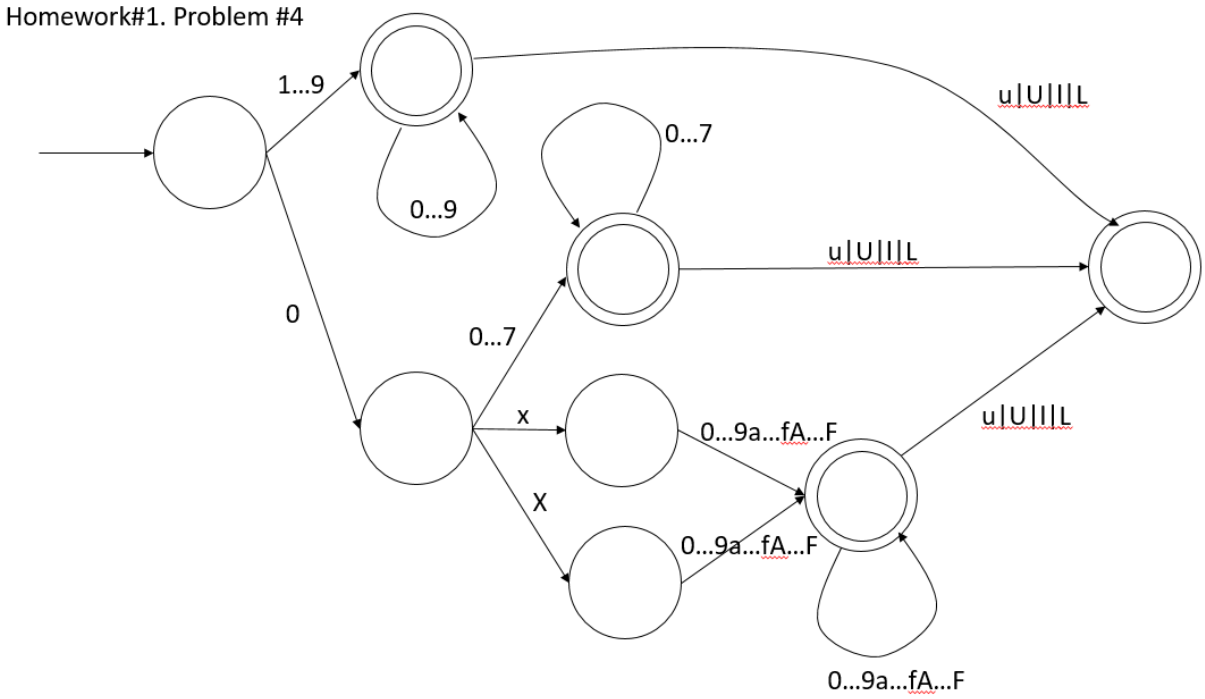
4) In *The C Programming Language* (Kernighan and Ritchie), an integer constant is defined as follows: An integer constant consisting of a sequence of digits is taken to be octal if it begins with `0` (digit zero), decimal otherwise. Octal constants do not contain the digits `8` or `9`. A sequence of digits preceded by `0x` or `0X` (digit zero) is taken to be a hexadecimal integer. The hexadecimal digits include `a` or `A` through `f` or `F` with values 10 through 15.

   An integer constant may be suffixed with the letter `u` or `U`, to specify that it is unsigned. It may also be suffixed by the letter `l` or `L` to specify that it is long.

   a) Write a regular expression that generate C integer constants as described above. (Of course you can break the solution down into a regular expression with several named parts if it makes things easier to write and read -- which it probably will.)
      i) ([1-9][0-9]*|0[0-7]*|(0x|0X)[0-9a-fA-F]*)(u|U|l|L)?

   b) Draw a DFA that recognizes integer constants as defined by your solution to part (a). You may draw this directly; you don't need to formally trace through an algorithm for converting a regular expression to a NFA and then constructing a DFA from that. However, you might find it useful to do so at least partially.

      Hint: You might find it helpful to alternate between designing the DFA and writing the regular expressions as you work on your solution.



([1-9][0-9]*|0[0-7]*|(0x|0X)[0-9a-fA-F]*)(u|U|l|L)?

5) A comment in C is a sequence of characters `/* ... */`.

(a) Write a set of regular expressions that generate C-style comments. You can restrict the alphabet to lower-case letters, digits, spaces, newlines (\n), carriage returns (\r) and the characters `*` and `/`. Also, remember that in C comments do not nest (i.e., a `*/` marks the end of a comment no matter how many times `/*` appears before it.)

i) Syntax for numeric constants:
digits ::= [0-9]
lower_letters ::= [a-z]
space ::= \w
newline::= \n
returns ::=\r
period ::= .
slash ::= /
star ::= *[^slash]
start ::= /*
end ::= *+/
allowed_characters ::= lower_letters|digits|space|newline|return|star|period|slash|starts
regular expressions: {start(allowed _characters|^end)*end,
start(allowed_characters|one_or_more_stars^slash)*end}

(b) Draw a DFA that recognizes C comments as defined by your solution to part (a). As with the previous problems, you may draw this directly without tracing through the steps of the regexp->NFA->DFA algorithms.

Hints: be careful about what is included in the `...` between `/*` and `*/`. Also, as with the previous problem, you may find it helpful to alternate between the regular expressions and DFA as you work on the problem.

Homework#1. Problem #5