CS143 Handout #5
Winter 2001

Review problems: FA and regex

To understand the innards of lex and use it effectively, you should be comfortable with regular expressions and finite automata, both deterministic and non. This material is introduced in CS103 or CS109 and explored in depth in CS154. CS143 expects students have covered the basics from the intro course. However, given the metamorphosing curriculum for those courses, some of you may feel less prepared on these topics than you'd like. I wrote up some review problems to provide additional practice for those who need it. For background, you may want to review chapter 10 of Ken Rosen's *Discrete Mathematics* textbook.

These problems are not to be handed in or graded. You can work them if you like, alone or in a group. You can also completely ignore them if you're already confident in your understanding. The answers are included so you can check your work. Feel free to come by our office hours with questions if you need help.

Regular expressions

1. Indicate which strings are in the language denoted by each regular expression.

a) `(ab|b)*c` matches which of these strings? `ababbc` `abab` `c` `babc` `aaabc`
b) `ab*c*(a|b)c` matches which of these strings? `acac` `acbbc` `abbcac` `abc` `acc`
c) `(a|b)a+(ba)*` matches which of these strings? `ba` `bba` `ababa` `aa` `baa`

2. Write regular expressions for each description. The alphabet Σ is the binary digits `{0, 1}`.

a) All strings which end in 01
b) All strings which contain exactly one 0
c) All strings which contain an even number of 1s and no 0s
d) All strings which contain an even number of 1s and any number of 0s
e) All strings which contain the substring 01
f) All strings which do not contain the substring 01

3. Describe the language denoted by the following regular expressions. The alphabet Σ is `{x, y}`.

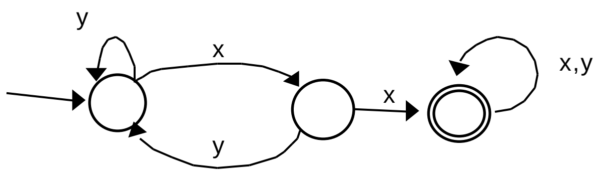
a) `x(x|y)*x`
b) `((x|y)(x|y))+`
c) `x*(yx+)*x*`
d) `(x|y)*(xx|yy)*y*`

One handy way to practice with regular expressions is to use the unix utility `grep` to search files for lines matching a given regular expression (see man page for usage). For example, you could use grep on the file /usr/dict/words (a big list of English words) to find those words that have all five vowels in order or end in "ion" or contain the letters 'j' and 'z' or can be spelled from your phone number. Let your imagination run wild!

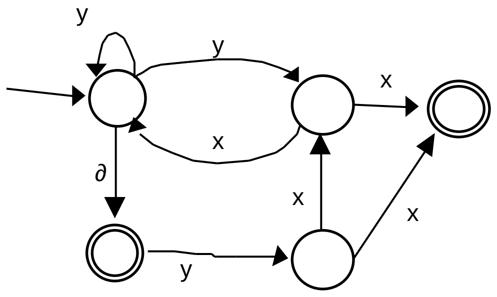
Finite automata

4. Indicate which strings are the language accepted by each automaton.

a) The DFA below accepts which of these strings? `xy   xyxxy   yyyx   xyyxyxyxxy`



b) The NFA below accepts which of these strings? `yyy   xx   ∂   yyyxy   yxxy   yx`



5. Construct the following automata.

a) Construct a DFA for the language in problem 2a.
b) Construct a DFA for the language in problem 2f.
c) Construct an NFA from the regular expression in problem 3a, and convert to an equivalent DFA.
d) Construct an NFA from the regular expression `(ab|a)*b+`, and convert to an equivalent DFA.

Answers

1. Those strings not in the language are crossed out.

a) `ababbc` ~~`abab`~~ `c` `babc` ~~`aaabc`~~
b) `acac` ~~`acbbc`~~ `abbcac` `abc` ~~`acc`~~
c) `ba` ~~`bba`~~ ~~`ababa`~~ `aa` `baa`

2. Note there can be many other equivalent regular expressions!

a)	`(0|1)*01`	(any number of binary digits can precede the ending 01)
b)	`1*01*`	(must be a zero somewhere, can be preceded or followed by 1s)
c)	`(11)*`	(ones must be added in pairs)
d)	`(0*10*10*)*`	(take answer to c and insert optional zeros in-between)
e)	`(0|1)*01(0|1)*`	(must be 01 somewhere, anything can come before or after)
f)	`1*0*`	(0 can only be followed by another 0)

3.

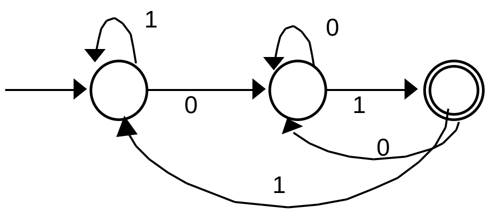
a) string must start and end with x
b) string must be of even length >= 2
c) every y is followed by at least one x (can't contain substring yy & can't end with y)
d) any string (i.e. regular expression matches Σ*)

4. Those strings not in the language are crossed out.

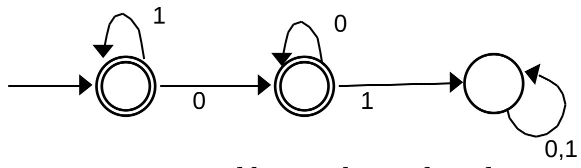
a) ~~xy~~ xyxxy ~~yyyx~~ xyyxyxyxxy
b) yyy ~~xx~~ ∂ yyyxy yxxy yx

5. Note there can be many equivalent automata. We tried to simplify to a fairly tidy version.

a)

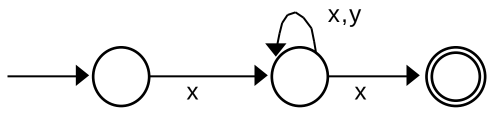


b) Start with the automata from above and send the path 01 to sinkhole, then the remaining states and transitions accept.

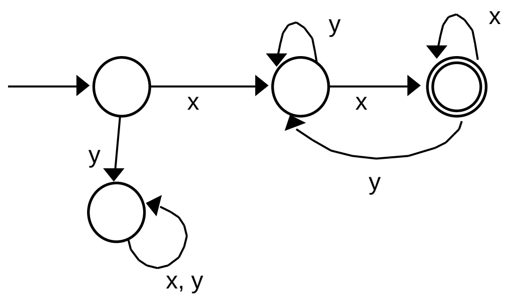


c) The NFA constructed by mechanical application of Thompson's algorithm would have ∂-transitions and be more complicated. This version has been simplified.

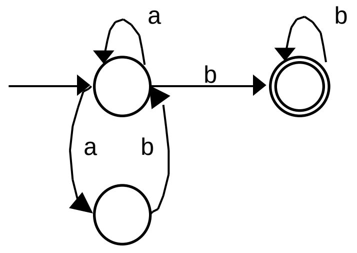
NFA:



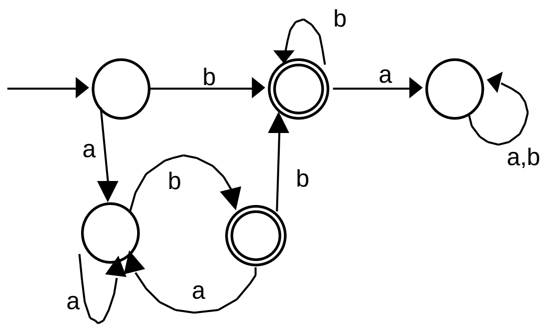
DFA



d) NFA:



DFA


b
b
a
a
b
b
a,b
a
a