Theory cover letter
Your name: Joshua Hoshiko
Assignment name: Homework 2/Floating point FA
Date submitted: 9/2/2019
Time spent on assignment: 3 – 4 hours

"How'd it go?" Overall, the assignment wasn't too bad. The longest part of the homework was reading about how to do union, star, and concatenation NFAs.

Any remaining questions on the material? How to write regular expressions for strings that cannot contain a certain substring. An example could be the language of strings that do not contain the substring 'cab'.

Who you collaborated with or got help from (if anyone), and what references you consulted beyond the text and course notes. Nothing besides the textbook.

If this is an incomplete assignment, what is missing, or not working? Be specific. ~~Only one regular expression for number 15 from the previous homework.~~
none

Additional discussions specified for an individual assignment. Nothing I can think of.

Anything else? Nope! Thank you!

Theory of Computation
Fall 2019, August 27
Dr. Gurka

Regular Expressions & Non-deterministic Finite Automata

Due: Tuesday, September 3, beginning of class, on paper; also post to Moodle, time-stamped before class. Both packets should include a cover letter (template on Moodle). No late assignments accepted; partial submissions will receive partial credit. Read specifications carefully. All solutions must be readable.

For all problems, $\Sigma = \{a, b, c\}$. Test strings (good and bad) should be 'different,' in the sense that they are representative of different groups of strings.

---

1. Give regular expressions for the odd-numbered languages in homework #1. Remember to use only the regular expression syntax given in Sipser (star, union, parentheses).

2. Write NFA for the following languages. Include at least three good and three bad strings. Draw the NFA diagram, and give formal definitions for all your NFA. Evaluate whether each edge you have needs to be in the diagram. Do not show any trap states.

   "The language of all strings that ..."
   a. end in 'b' or 'c'
   b. do not have 'a' or 'b' as the second character from the end
   c. start with 'bb'
   d. do not contain 'c' and do not contain 'b'

3. Create NFA for the union (regular operation) of the following languages; use the construction given in class. For each, give at least four good strings (they should not all be from one language). Draw the diagrams, but do not give formal definitions. Do not reduce (remove states, etc.) the constructed NFA in any way. The underlying FA can be DFA or NFA.

   "The language of all strings that ..."
   a. [end with 'cc'] U [start with 'a']
   b. [start with 'c'] U [even number of 'a's]
   c. [less than 3 characters long] U [start and end with the same character]

4. Create NFA for the star (regular operation) of the following languages; use the construction given in Sipser. For each, give at least three good strings, underlining each substring from the language. For example, for the star of all strings that start with 'a', here are three strings (substrings separated by spaces for readability): string 1: a abc aabcba a, string 2: aaaa; string 3: ab acac. The first shows four substrings from the underlying language, the second shows one, and the third shows two. Although all starred languages contain epsilon, do not give it as one of your substrings (but do make sure it is accepted in your diagram). Draw the diagrams, but do not give formal definitions for your NFA. Do not reduce (remove states, etc.) the constructed NFA in any way. The underlying FA can be DFA or NFA.

   "The language of all strings that ..."
   a. contain 'cab'
   b. contain exactly one 'b'

c. are exactly two characters long
d. are at least length one and contain only one type of character (all 'a's or all 'b's or all 'c's)

5. Create NFA for the concatenation (regular operation) of the following languages; use the construction given in class. For each, give at least three good strings, using an underlining scheme similar to the above, that shows the substring from the first language and the substring from the second. Although epsilon may be a member of one or both of the languages, do not give it as one of your substrings. Do not give formal definitions for your NFA. Do not reduce (remove states, etc.) the constructed NFA in any way. The underlying FA can be DFA or NFA.

   "The language of all strings that ..."
   a. [start with 'a'] concatenated with [no more than 3 characters]
   b. [end with 'b'] concatenated with [end with 'b']
   c. [contain exactly one 'a'] concatenated with [contain 'ccc']
   d. [end with 'bb'] concatenated with [do not contain 'c']
   e. [do not contain 'c'] concatenated with [end with 'bb']

6. Included in this homework: floating point FA started in class.

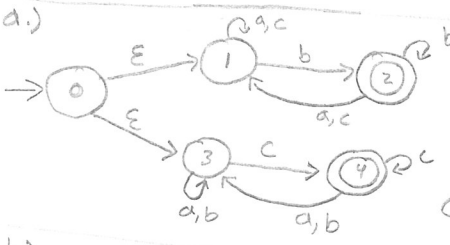
Joshua Hoshiko

① Regular expressions of FA's from Hw 1

1.) $\Sigma^* 0 \Sigma^*$ 3.) $000\Sigma^*$ 5.) $\Sigma\Sigma\Sigma\Sigma^*$ 7.) $1(\varepsilon \cup \Sigma\Sigma)$ 9.) $\Sigma^* abc$

11.) $\Sigma(\Sigma\Sigma)^*$ 13.) $\Sigma^* a \Sigma^* a \Sigma^*$ 15.) $(a \cup b)^*(c^*(bb \cup bc \cup aa \cup ab)^*)^*(cb \cup bc)^*$ 17.) $(b \cup c)^*$

② a.)



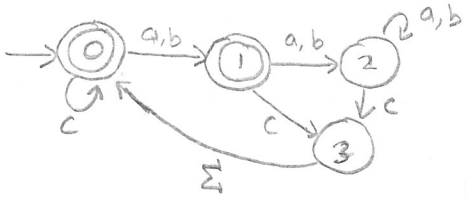
$\Sigma = \{a, b, c\}$
$S = 0$
$F = \{2, 4\}$
$Q = \{0, 1, 2, 3, 4\}$

$\delta =$

	a	b	c
0	1,3	2,3	1, 4
1	1	2	1
2	1	2	1
3	3	3	4
4	3	3	4

good	bad
abc	ε
ab	a
abcac	aba

b.)



$\Sigma = \{a, b, c\}$
$S = 0$
$F = \{0, 1\}$
$Q = \{0, 1, 2, 3\}$

	a	b	c
0	1	1	0
1	2	2	3
2	2	2	3
3	0	0	0

good	bad
abcc	ab
ca	abaac
ε	cba

c.)

$\Sigma = \{a, b, c\}$
$S = 0$
$F = \{2\}$
$Q = \{0, 1, 2\}$

	a	b	c
0	–	1	–
1	–	2	–
2	2	2	2

good	bad
bba	b
bbbc	ε
bb	abc

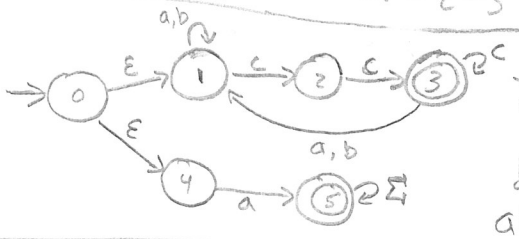
d.)

$\Sigma = \{a, b, c\}$
$S = 0$
$F = \{0\}$
$Q = \{0\}$

	a	b	c
0	0	–	–

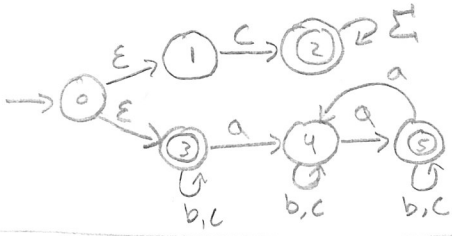
good	bad
a	b
ε	bc
aaa	cbb

③ a.)



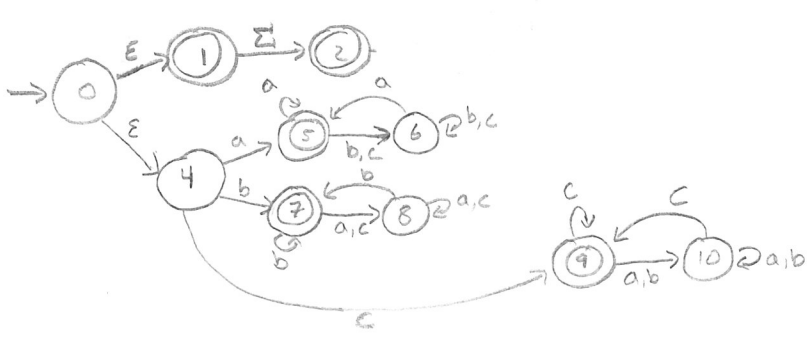
good	bad
abcc	ε
aba	bac
bacc	ca
abcca	cca

b.)



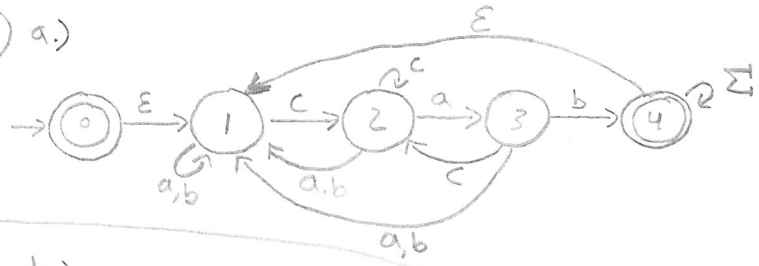
good	bad
ε	ac
aac	abaa
ca	bac
caba	a

c.)



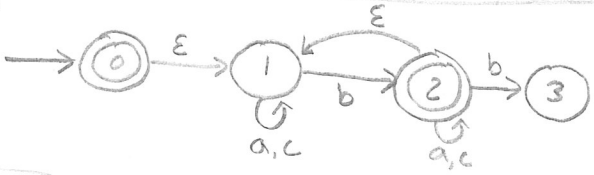
good	bad
aba	ε
abc	ab
c	bc
aa	ca

4. a.)



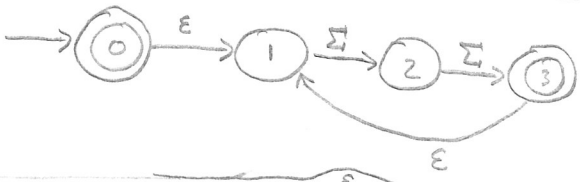
good
cab
abcabcab
cab abcabcab

b.)



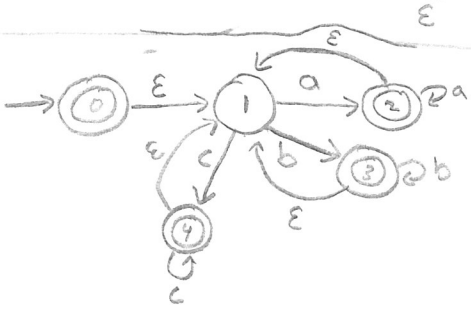
good
bb
abcba
bbabc

c.)



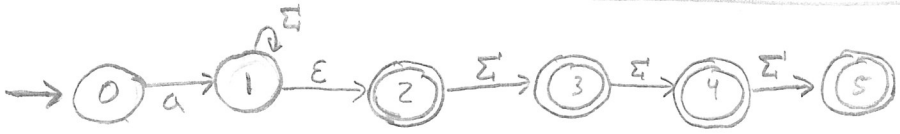
good
ab
abcc
abccabbb

d.)



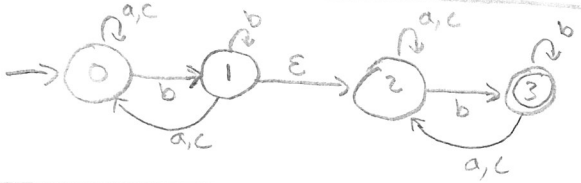
good
abc
aabccc
abaac

5. a.)



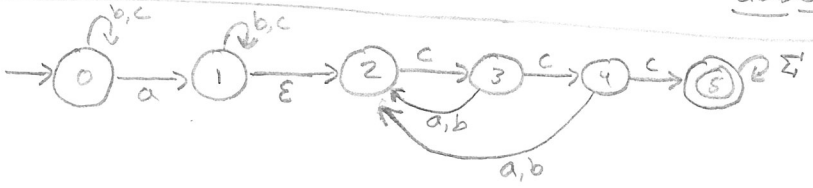
good
abcbc
abca
abcaabb

b.)



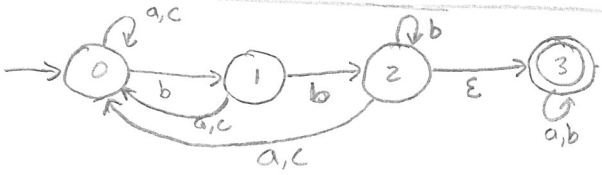
good
abbab
bb
abbacb

c.)



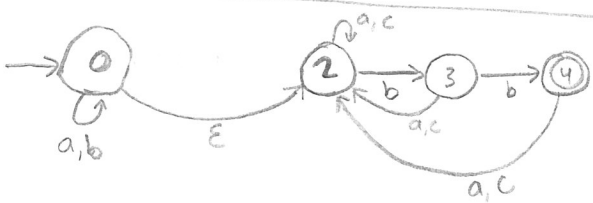
good
cabccc
abcaccc
abacccca

d.)



good
acbbab
bbbab
baccbba

e.)



good
abacabb
acbbabb
bb bb

Theory of Computation
Fall 2019, August 27
Dr. Gurka

Floating point FA
(exercise / homework)

Name: Joshua Hoshiko

Specifications and test plan consultants (classmates), if any: Daryl Lee, Peri Smith, Ryan Hiller, Joel Martinez

Due: Tuesday, September 3 with NFA hoework, paper copy beginning of class, Moodle copy posted before class (scan and post handwritten material); one cover letter for the packet; keep track of time spent outside class.

Directions: Write a DFA that recognizes a valid floating point number as described below.
in class, with classmates:

1. discuss the specification
2. create good and bad test strings (floating point numbers or close) – use the chart on the next page; note that you will be able to give only a small subset of all test cases that should be used;

at home, individually:

3. sketch out the DFA on scratch paper
4. test using your strings, correcting the draft DFA as necessary
    a. you may also use a classmate's test cases
5. write your final DFA on the third page
    a. use <digit> to label edges (do not list 0 through 9)
    b. everything must be clearly legible to receive credit

Work from the specification given below; do not look up any additional information on the definition of floating point; this is an exercise in translating to a DFA, not C correctness.

If you believe there are any ambiguities or omissions in the definition, explain them below, as well as stating what you are assuming in creating your DFA.

C Floating Point Specification (K & R)

assume no spaces

"A floating constant consists of an integer part, a decimal point, a fraction part, an e or E, an optionally signed integer exponent and an optional type suffix, one of f, F, l, or L. The integer and fraction parts both consist of a sequence of digits. Either the integer part or the fraction part (not both) may be missing; either the decimal point or the e and the exponent (not both) may be missing. The type is determined by the suffix; F or f makes it float, l or L makes it long double; otherwise it is double."

Further specifications or clarifications (continue on the back as needed)?

Test strings (see examples below)

- list 6-7 "different" valid floating point literals; each is an instance of a category of valid numbers
  - the description is what components you've used
- list 6-7 "different" invalid floating point literals, most of which should be "close" to a valid form
  - the description is what's wrong with the string (why it's invalid)

case	case description (each one should be different)	test string	expected result
1	integer part, decimal point, fractional part	3.62	accepted
2	Integer part only 36. ↔ 36	36	accepted
3	fraction part only 0.36 ↔ .36	.36	accepted
4	integer, decimal, fraction, exponent + digits	1.2e2	accepted
5	integer, decimal, fraction, Suffix 1.2F ↔ 1.2f	1.2f	accepted
6	integer, decimal, fraction, exponent + digits, Suffix	1.2e2f	accepted
7	integer, exponent + digits, Suffix	1E2L	accepted
8	decimal, fraction, exponent + digits, Suffix	.121	accepted
9	"e" without exponent digits	24e	rejected
10	only a decimal	.	rejected
11	multiple decimal points	.21.	rejected
12	two Suffixes	21.1fL	rejected
13	empty String is not accepted	ε	rejected
14	Suffix before an exponent	1.2fe22	rejected
15	no integer and fraction parts	.e22f	rejected
16	negative alone	-	rejected

all should work with

final DFA (better written sideways ...)

* This is an NFA *