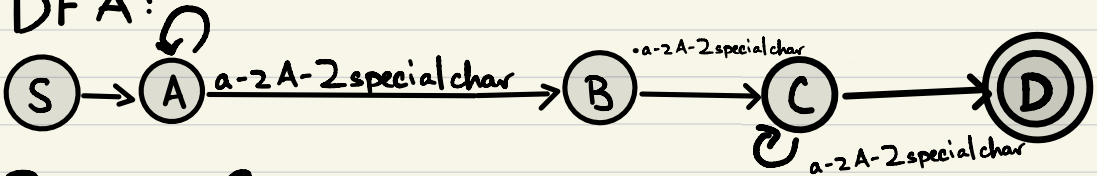
CSC 4330 Test 1

Schaum

1. Regex: [a-zA-Z0-9#!%.$'&+*\-/=?^_`{|}~]+(\.[a-zA-Z0-9#!%.$'&+*\-/=?^_`{|}~]+)*

DFA:

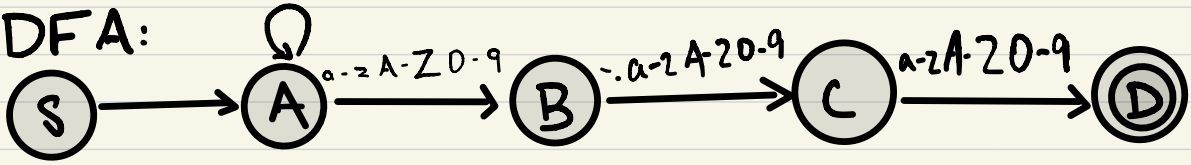


Regular Grammar:

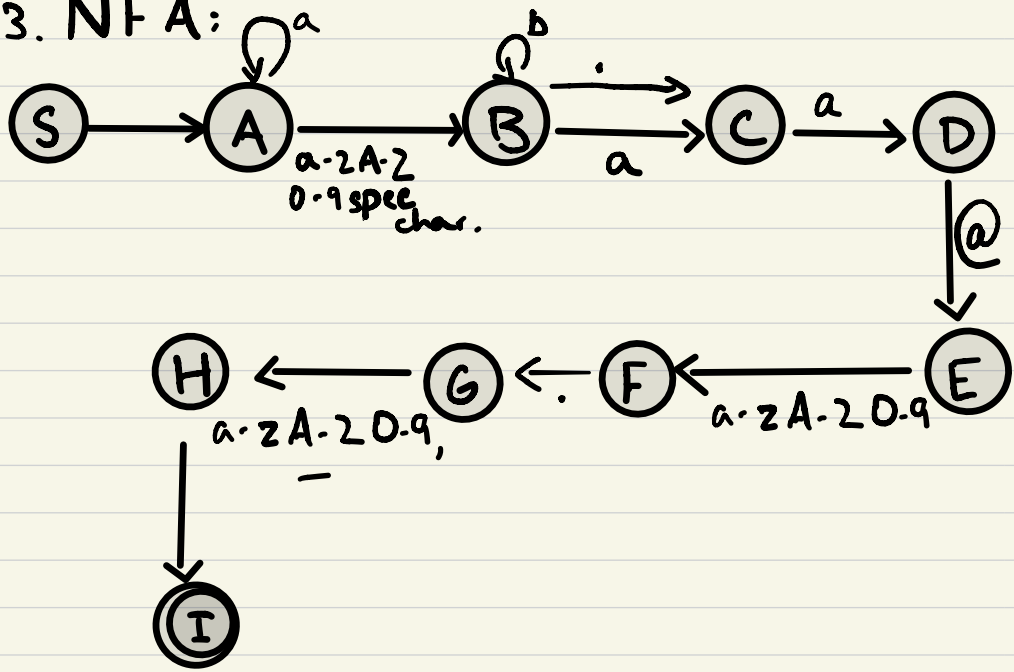
S → A
A → aA | aB
B → bC
C → cC | D

2. Regex: [a-zA-Z0-9]+([-.][a-zA-Z0-9]+)*.)+[a-zA-Z]{2,}$

DFA:

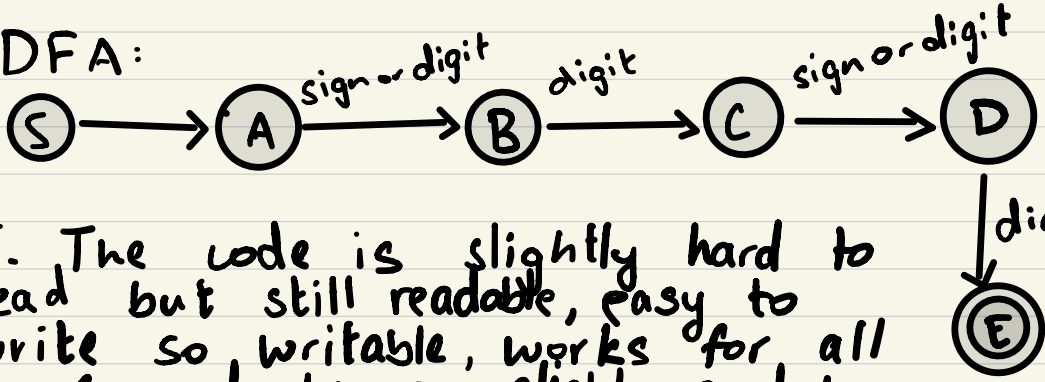


3. NFA:



4. Regex: `[+-]?([0-9]*\.[0-9]+|[0-9]+\.[0-9]*)([eE][+-]?[0-9]+)?`

DFA:



5. The code is slightly hard to read but still readable, easy to write so writable, works for all cases and hence reliable, and low cost.

Regular Grammar:

S → +S | -S | A
A → B | Bex
B → C | C.C | C.Cex | .C | .Cex
C → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
ex → E+ | E- | E
E → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

6. Regex: -?(0|[1-9][0-9]* | 0[xX][0-9a-fA-F]+ | 0[0-7]*)

Regular Grammar:

S → -N | N
N → D | H | O
D → [1-9] D | [0-9]
H → 0x [0-9a-fA-F]+
O → 0 | 0 O | 0 [1-7] O

7. The code is easy to understand hence readable, easily written and hence writable, works for all decimal, octal, and hexadecimal constants hence reliable, and has no cost to code or run so low cost.

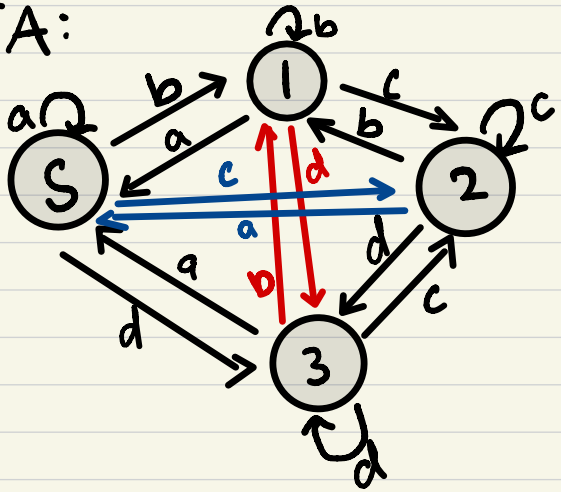
8. def is_email_address(string):
    regex = r'^[a-zA-Z0-9#!%$'*+"\-/=?^_`{|}~]+(\.[a-zA-Z0-9#!%$'*+"\-/=?^_`{|}~]+)*@[a-zA-Z0-9]+([-.][a-zA-Z0-9]+).[a-zA-Z]{2,}$'
    return re.match(regex, string)

9. def is_float(string):
    regex = r'^[+-]?([0-9]*\.[0-9]+|[0-9]+\.[0-9]*)([eE][+-]?[0-9]+)?$'
    return re.match(regex, string)

10. def is_integer(string):
    regex = r'^-?(0|[1-9][0-9]*|0[xX][0-9a-fA-F]+|0[0-7]*)$'
    return re.match(regex, string)

11. Regex: ^(aa)*(bb)*(c|d)*|(cbad){2}(\w)*$

DFA:



Regular Grammar:

$S \rightarrow A \mid B$
$A \rightarrow aBc \mid aBd$
$B \rightarrow bBc \mid bBd \mid cbadC$
$C \rightarrow cbadC \mid cbad$

12. The regex does not work for patterns with a single b, and it does not account for even occurences of cbad.
Examples:

bcd: Works even though there is no a.
cbadcbad: Would not match as there is no definition for string "cbad".
baaccdd: Would not work because there is no definition for c's and d's.

13. `^(?=(aa)*a(bb)*a(bb)*$)(b(cc|dd)*)|(cbad){2}*`

14.
```
import re
def valid_java_id(s):
    pattern = r'^[a-zA-Z_$][a-zA-Z0-9_$]*$'
    return re.match(pattern, s) is not None
```

16.
```
import re
def is_valid_C_comment(s)
multi_comments = r'^[/*a-zA-Z0-9_$][$*\]$'
    return re.match(multi_comments) is not None
```

17. For Q16, there are 36 lexemes.