

CMSC 330, Spring 2008, Midterm 1 Practice Problems (SOLUTIONS)

1. Programming languages
 - a. Explain how goals for programming languages have changed since the 1960's.
Shifted from efficiency to ease-of-programming
 - b. List 2 desirable attributes for a programming language where Ruby is better than C. Explain why.
Naturalness of application – Text processing is easier in Ruby
Cost of use – Small Ruby programs are simpler/quicker to write
 - c. List 2 methods for executing a program. Which method is used by Ruby?
Interpretation & compilation. Ruby is interpreted.
 - d. Explain why Ruby fits the definition of a scripting language.
Rich text processing (Regexp) and easy to use (implicit declarations)

2. Ruby basics
 - a. Write a Ruby method foo that takes an integer as a parameter. Call foo with 2 as its argument. Circle & label the formal and actual parameters in your code.
def foo(x) ... end ; foo(2) ; // x = formal param, 2 = actual parameter
 - b. Using different Ruby control statements, write 4 code fragments that iterate from i=1 to i=10.
1.upto(10) {|i| puts i; }
(1..10).each {|i| puts i; }
for i in (1..10) do puts i; end
i=1; while i<=10 do puts i; i+=1; end
i=1; do puts i; break if (i+=1)>10 end
 - c. Explain the difference between explicit and implicit variable declarations.
Explicit – declaration statements declare type of each variable used
Implicit – first use of a variable declares it and determines its type
 - d. List two advantages of static types.
Helps prevent subtle errors, catches more type errors at compile time
 - e. Using Ruby, write a class Teacher that contains an integer field students and an integer field totalStudents that is shared across all objects of class Teacher.
class Teacher
@@totalStudents = 0
def initialize
@students = 0
@@totalStudents += @students
end
end
 - f. Give an example of reference copy in Ruby.
x = "a" ; y = x
 - g. Give an example of testing for structural equality in Ruby.
x == y

3. Ruby advanced features

- a. Describe the string matched by the Ruby regular expression `/(\{2})/ ?`
\$1 = exactly 2 3's, i.e., "33"
- b. Describe the string matched by the Ruby regular expression `/([A-Z])/ ?`
\$1 = any single uppercase letter
- c. Describe the string matched by the Ruby regular expression `/([A-Z]*[0-9])/ ?`
\$1 = 0 or more uppercase letters followed by a single digit
- d. Describe the string matched by the Ruby regular expression `/(0$)/ ?`
\$1 = a 0 at the end of the line
- e. Describe the string matched by the Ruby regular expression `/(\.)/ ?`
\$1 = a single (literal) period
- f. What is the output of the following Ruby program?
`"CMSC 330" =~ /[0-9]+/`
`puts $1` // **330**
`puts $2` // **nil**
- g. What is the output of the following Ruby program?
`a = [4,5,6]`
`a[5] = 7`
`a.delete_at(1)`
`a.push(1)`
`puts a` // **4 6 nil nil 7 1**
`puts a.pop` // **1**
- h. What is the output of the following Ruby program?
`if "CMSC 330" =~ /1/ then`
`puts "t"`
`elsif "CMSC 330" !~ /1/ then`
`puts "f"` // **f**
`else`
`puts "n"`
`end`
- i. What is the output of the following Ruby program?
`a = ["c", "b", "a"]`
`puts a` // **c b a**
`b = a`
`a.sort!`
`puts b` // **a b c**
- j. What is the output of the following Ruby program?
`a = "CMSC 330 CMSC 351"`
`b = a.scan(/[A-Z]+/)`
`puts b` // **CMSC CMSC**
`a.scan(/[0-9]+ [A-Z]+/) { |x| puts x }` // **330 CMSC**
- k. What is the output of the following Ruby program?
`a = {4 => 6, 5 => 7}`
`puts a[4]` // **6**
`puts a[6]` // **nil**
`puts a.values` // **6 7 or 7 6**

1. What is the output of the following Ruby program?


```

h = Hash.new(0)
h["a"] = h["b"]
h["b"] = 7
h["c"] += 2
puts "#{h["a"]} #{h["b"]} #{h["c"]}" // 0 7 2

```
 - m. What is returned by “file = File.new(filename, "r"); lines = file.readlines();”?

Array of strings where each string is a line from the file <filename>
 - n. What is returned by “x = ARGV[0];”?

String for 1st command line parameter
 - o. Write a Ruby function foo that takes a code block and executes it twice.


```

def foo( ) 2.times{ yield } end
foo() { puts “Running” } // prints “Running Running”

```
4. Ruby programming
- a. Write a Ruby program that reads in lines of input from \$stdin and remembers all integers (consecutive digits) encountered. The program should stop and print out the list of integers in sorted order (from smallest to largest) when the word “Done!” is encountered.


```

a = Array.new
loop do
  line = $stdin.readline
  break if line =~ /Done\!/
  line.scan(/\d+/) { |x| a.push x.to_i }
end
a.sort!
a.each { |x| puts x }

```
 - b. Write a Ruby program that reads the name of a text file from the command line, opens the file, reads every line of text in the file, and prints only the lines that contain exclusively the following characters: uppercase and lowercase letters, digits, and underscore. For example, lines that contain space or punctuation should not be printed.


```

// Version that reads entire file into array
file = File.new(ARGV[0], "r")
lines = file.readlines
lines.each{ |line|
  line.chomp!
  if line !~/[^\A-Za-z0-9\_]+/ then
    puts line
  end
end
}

```

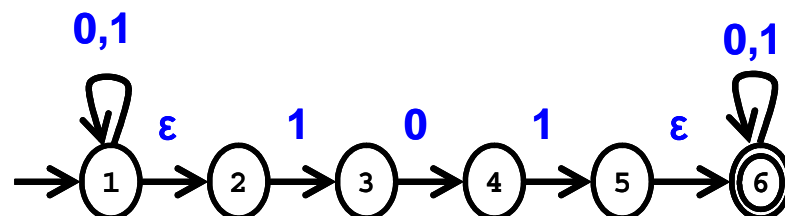
```
// Alternate version that reads file one line at a time
file = File.new(ARGV[0], "r")
until file.eof? do
  line = file.readline
  line.chomp!
  if line !~ /^[^A-Za-z0-9\_]+/ then
    puts line
  end
end
```

5. Regular expressions and languages

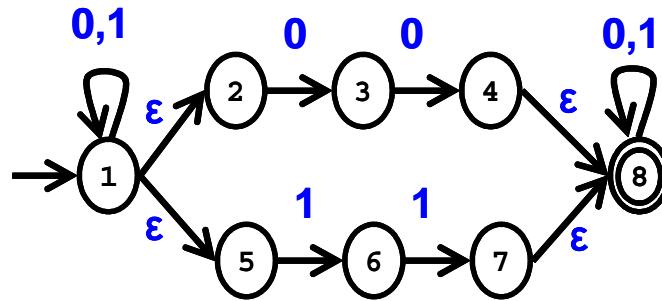
- From the perspective of formal language theory, what is a language?
Set of strings
- Given the language $A = \{\text{"aa"}, \text{"c"}\}$ and $B = \{\text{"b"}\}$, what is the language AB ?
\{"aab", "cb"\}
- Given the language $A = \{\text{"aa"}, \text{"c"}\}$, what is the language A^0 ?
\{\epsilon\}
- Given the language $A = \{\text{"aa"}, \text{"c"}\}$, what is the language A^2 ?
\{"aaaa", "cc", "aac", "caa"\}
- Given the language $A = \{\text{"aa"}, \text{"c"}\}$, what is the language A^* ?
\{\epsilon, "aa", "c", "aaaa", "cc", "aac", "caa", "aaaaaa" ... \}
- Give a regular expression for all binary numbers including the substring "101".
(0|1)*101(0|1)*
- Give a regular expression for all binary numbers with an even number of 1's.
0*(10*1)*0*
- Give a regular expression for all binary numbers that don't include "000".
(01 | 001 | 1)*(0 | 00 | \epsilon)

6. Finite automata

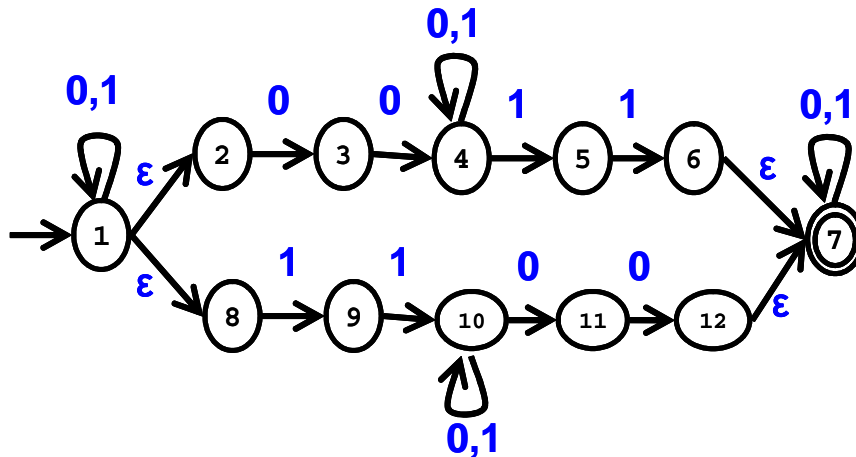
- When does a NFA accept a string?
If there any path for the string that ends at a final state for the NFA
- How long could it take to reduce a NFA with n states and t transitions to a DFA?
 2^n
- Give a NFA that only accepts binary numbers including the substring "101".



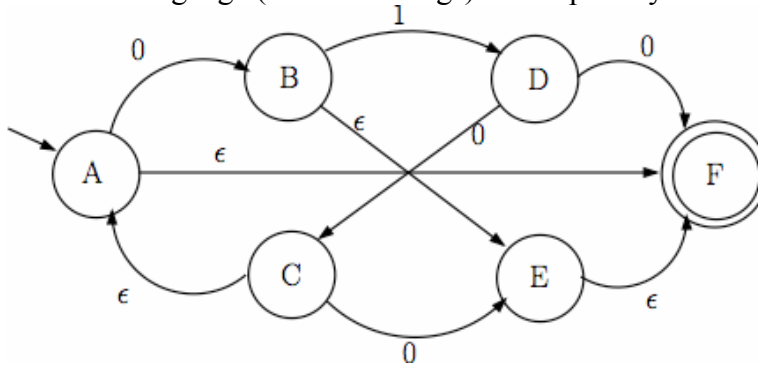
d. Give a NFA that only accepts binary numbers that include either “00” or “11”.



e. Give a NFA that only accepts binary numbers that include both “00” and “11”.



f. What language (or set of strings) is accepted by the following NFA?

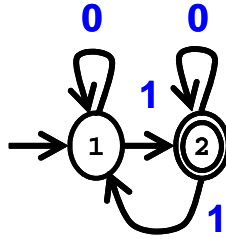


(010)^{*}(0| ϵ)

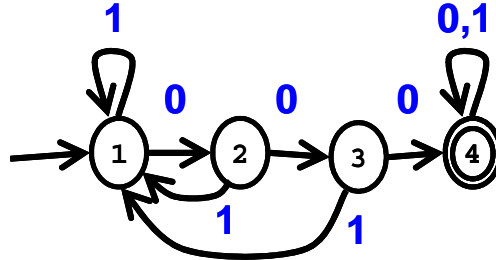
g. Compute the ϵ -closure of the start state for each of the NFA above.

- For NFA in (c) ϵ -closure(1) = {1,2}
- For NFA in (d) ϵ -closure(1) = {1,2,5}
- For NFA in (e) ϵ -closure(1) = {1,2,8}
- For NFA in (f) ϵ -closure(A) = {A,F}

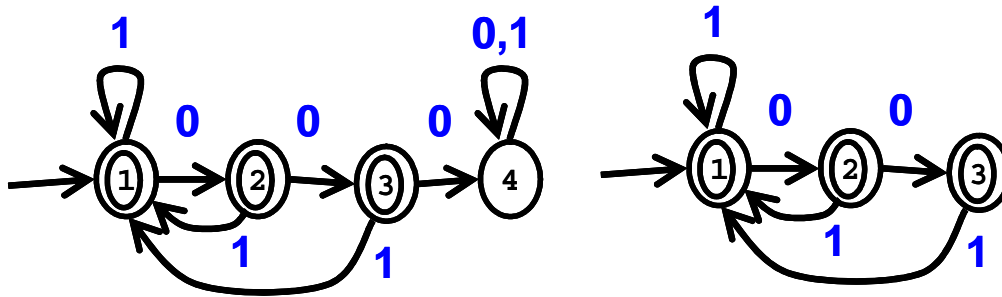
h. Give a DFA that only accepts binary numbers with an odd number of 1's.



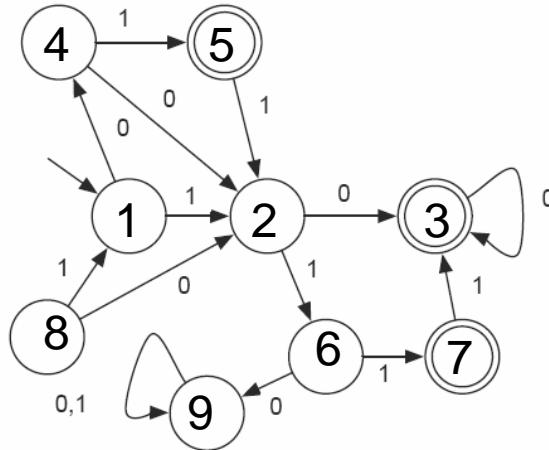
i. Give a DFA that only accepts binary numbers that include "000".



j. Give a DFA that only accepts binary numbers that don't include "000".



k. What language (or set of strings) is accepted by the following DFA?



Described as a list of strings:

{ "01", "111", "0011", "01111", "10", "000", "0110", "1111", "00111", "011111"... }

where all underlined strings may have any number of 0s appended

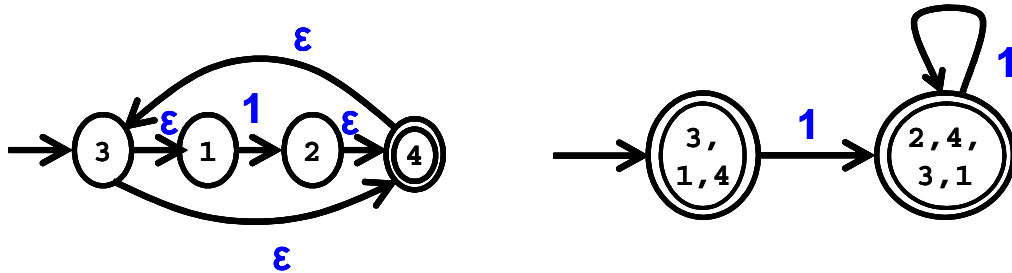
Described as a regular expression: $01 \mid (1 \mid 00 \mid 011)(11 \mid (0 \mid 111)0^*)$

Explanation (for each underlined portion of RE)

- 01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 1 to 5 and accepts
- 01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 1 to 2, then...
- 01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 2 to 7 and accepts
- 01 | (1 | 00 | 011)(11 | (0 | 111)0*) from state 2 to 3, then...
- 01 | (1 | 00 | 011)(11 | (0 | 111)0*) accepts w/ 0 or more 0's

1. For each regular expression: 1^* , $(0|01)^*0$, $(0^*|1)^*1$, $(10^*|01)^*$
- Reduce the RE to an NFA using the algorithm described in class.
 - Reduce the resulting NFA to a DFA using the subset algorithm.
 - Show whether the DFA accepts / rejects the strings "1", "11", "101"
 - Minimize the resulting DFA using Moore reduction
 - Are any 2 of the minimized DFA identical?

$1^* \rightarrow \text{NFA} \rightarrow \text{DFA}$



Accept / reject

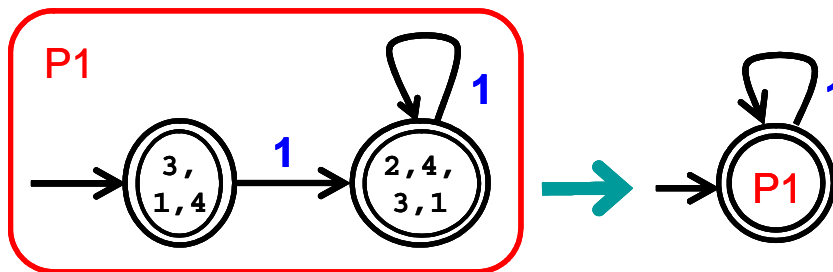
- "1" $\{3,1,4\} \rightarrow \{2,4,3,1\}$ accept
- "11" $\{3,1,4\} \rightarrow \{2,4,3,1\} \rightarrow \{2,4,3,1\}$ accept
- "101" $\{3,1,4\} \rightarrow \{2,4,3,1\} \rightarrow \text{reject}$

Minimized DFA

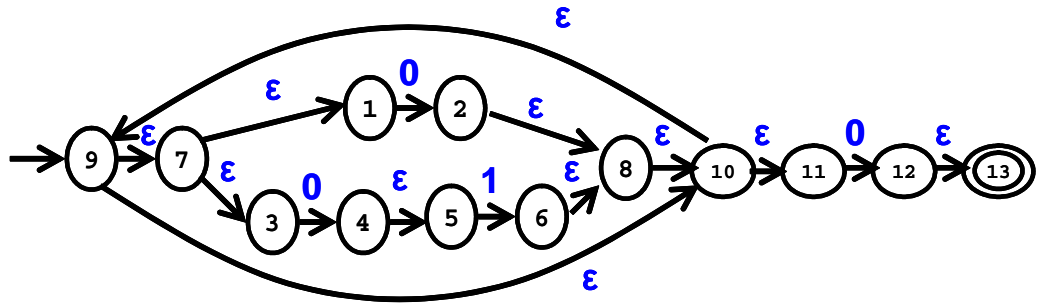
Initial partitions: $\text{accept} = \{ \{3,1,4\}, \{2,4,3,1\} \} = P1$,
 $\text{nonfinal} = \emptyset$

- $\text{move}(\{3,1,4\}, 1) \rightarrow P1$
- $\text{move}(\{2,4,3,1\}, 1) \rightarrow P1$

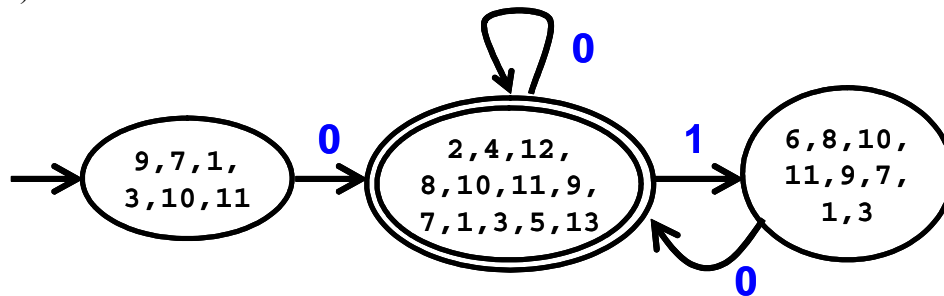
No need to split P1, minimization done. After cleanup, minimal DFA is



$(0|01)^*0 \rightarrow \text{NFA}$



$(0|01)^*0 \rightarrow \text{NFA} \rightarrow \text{DFA}$



Accept / reject

- “1” {9,7,1,3,10,11} → reject
- “11” {9,7,1,3,10,11} → reject
- “101” {9,7,1,3,10,11} → reject

Minimized DFA

Initial partitions: accept = { {2,4,...} } = P1,
nonfinal = { {9,7,...}, {6,8,...} } = P2

- move({9,7,...}, 0) → P1
- move({6,8,...}, 0) → P1
- move({9,7,...}, 1) → reject
- move({6,8,...}, 1) → reject

No need to split P2, minimization done. After cleanup, minimal DFA (different from previous minimal DFA) is

