COMS W3261
Computer Science Theory
Homework #1

Alexander Roth

2014 – 09 – 15

Problems

1. Use a computer program that supports regular expressions to find the lexicographically first longest English word in the dictionary that can be made up using only letters in your first and last name. A letter in your name can be used zero or more times. Show the regular expression and program you used to find your answer.

Solution My name is Alexander Roth. After removing duplicate characters, we are left with the string `alexndroth`. Now, it must be a string that solely contains these characters; thus, we capture these characters with `[` and `]`, like so `[alexndroth]`. Now, since these strings cannot be substrings of other words, we must include `^` to signify the beginning of the string, and `$` to signify the end of the string. To find all matches, we would include `*` as the operator, which gives a large list of words when passed into egreps. Thus, our regular expression passed into egrep is:

```
egrep '^[alexndroth]*$' /usr/share/dict/words
```

which prints three words of length 15, the first being "hexatetrahedron". I found the longest first string by using a small program I wrote (shown below).

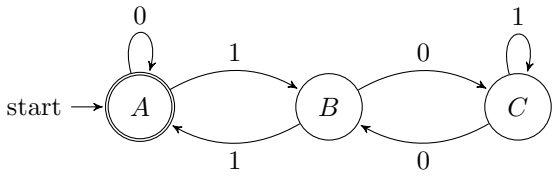
```
#!/usr/bin/env python
import sys

def main(filename):
        text_file = open(filename)
        text_list = text_file.readlines()
        text_file.close()
        length = 0
        answer = ""
```

```
        for line in text_list:
            if len(line) > length:
                length = len(line)
                answer = line.rstrip()
        print(answer + ", " + str(length))

if __name__ == '__main__':
        main(sys.argv[1])
```

2. In Lecture 2 (September 8, 2014) we presented a three-state DFA that recognizes exactly those binary strings representing integers divisible by 3. Using induction, prove that state A represents all and only the prefixes of binary strings that are numbers congruent to 0 mod 3, state B all and only the prefixes congruent to 1 mod 3, and state C all and only the prefixes congruent to 2 mod 3



Solution

- Consider the DFA $D = (\{A, B, C\}, \{0, 1\}, \delta, A, \{A\})$ where the transition function δ has the transition diagram shown above.
- Let L be the language consisting of all binary strings representing integers divisible by 3. We shall prove that $L(D) = L$.
- To begin, we shall show that every string accepted by D is in L, by proving the following inductive hypothesis by induction on n, the moves made by D accepting a binary string w, for $n \geq 0$:
  Inductive Hypothesis 1:
  (a) If $\delta^n(A, w) = A$, then w is a binary string whose value is equivalent to 0 mod 3. Here δ^n means n moves by D.
  (b) If $\delta^n(B, w) = B$, then w is a binary string whose value is equivalent to 1 mod 3.
  (c) If $\delta^n(C, w) = C$, then w is a binary string whose value is equivalent to 2 mod 3.
    - Basis. $n = 0$ which implies that $w = \epsilon$ Note that by appending a 1 to w, the value of w is $2x + 1$, where x is the original string. Similarly, appending 0 to w gives us the value $2x$ where x is the substring without 0.
    - Induction. Assume that IH1 is true for $0, 1, 2, \ldots, n$ moves.
      * Suppose D makes $n + 1$ moves on a string $w = x1$ and enters state A after reading x and then enters state B after reading the final 1. The value of w is $2x + 1$. From

the inductive hypothesis, we know that x must be a binary string with a value congruent to 0 mod 3. Therefore, $x1$ must be a binary string congruent to 1 mod 3.

* Suppose D makes $n+1$ moves on a string $w = x0$ and enters state A after reading x and then enters state A after reading the final 0. The value of w is 2x. From the inductive hypothesis, we know that x must be a binary string with a value congruent to 0 mod 3. Therefore, $x0$ must also be a binary string congruent to 0 mod 3.
* Suppose D makes $n+1$ moves on a string $w = x1$ and enters state B after reading x and then enters state A after reading the final 1. The value of w is $2x+1$, From the inductive hypothesis, we know that x must be a binary string with a value congruent to 1 mod 3. Thus, $x1$ must be a binary string congruent to 0 mod 3.
* Suppose D makes $n+1$ moves on a string $w = x0$ and enters state B after reading x and then enters state C after reading the final 0. The value of w is $2x$. From the inductive hypothesis, we know that x must be a binary string with a value congruent to 1 mod 3. Therefore, $x1$ must be a binary string congruent to 2 mod 3.
* Suppose D makes $n+1$ moves on a string $w = x1$ and enters state C after reading x and then enters state C after reading the final 1. The value of w is $2x+1$. From the inductive hypothesis, we know that x must be a binary string with a value congruent to 2 mod 3. Therefore, $x1$ must also be a binary string congruent to 2 mod 3.
* Suppose D makes $n+1$ moves on a string $w = x0$ and enters state C after reading x and the enters state B after reading the final 0. The value of w is $2x$. From the inductive hypothesis, we know that x must be a binary string a value congruent to 2 mod 3. Therefore, $x0$ must be a binary string congruent to 1 mod 3.

- We have now shown that IH1 is true for all sequences of n moves, where $n \geq 0$.
- We now need to show that every string in L is accepted by D. To do this, we shall prove the following inductive hypothesis by induction on n, the length of a string w for $n \geq 0$:
  Inductive Hypothesis IH2:
  (a) If w is a binary string that has a value congruent to 0 mod 3, then $\delta^*(A, w) = A$.
  (b) If w is a binary string that has a value congruent to 1 mod 3, then $\delta^*(B, w) = B$.

(c) If w is a binary string that has a value congruent to 2 mod 3, then $\delta^*(C, w) = C$.

- Basis. $n = 0$; that is $w = \epsilon$. D accepts the empty string since the start state is a final state.
- Induction. Assume IH2 is true for all string w of length $0, 1, 2, \ldots, n$.
  - Suppose w is now a string of length $n+1$ and $w = x0$ and x is a binary string that has a value congruent to 0 mod 3. From IH2, $\delta^*(A, x) = A$. Since $\delta(A, 0) = A$, $\delta^*(A, w) = A$. That is, the value of w is $2x$, which is equivalent to 0 mod 3.
  - Suppose w is now a string of length $n+1$ and $w = x1$ and x is a binary string that has a value congruent to 0 mod 3. From IH2, $\delta^*(A, x) = A$. Since $\delta(A, 1) = B$, $\delta^*(A, w) = B$. That is, the value of w is $2x + 1$, which is equivalent to 1 mod 3.
  - Suppose w is now a string of length $n + 1$ and $w = x0$ and x is a binary string that has a value congruent to 1 mod 3. From IH2, $\delta^*(B, x) = B$. Since $\delta(B, 0) = C$, $\delta^*(B, w) = C$. That is, w has a value of $2x$, which is equivalent to 2 mod 3.
  - Suppose w is now a string of length $n + 1$ and $w = x1$ and x is a binary string that has a value congruent to 1 mod 3. From IH2, $\delta^*(B, x) = B$. Since $\delta(B, 1) = A$, $\delta^*(B, w) = A$. That is, w has a value of $2x + 1$, which is equivalent to 0 mod 3.
  - Suppose w is now a string of length $n+1$ and $w = x0$ and x is a binary string that has a value congruent to 2 mod 3. From IH2, $\delta^*(C, x) = C$. Since $\delta(C, 0) = B$, $\delta^*(C, w) = B$. That is, w has a value of $2x$, which is equivalent to 1 mod 3
  - Suppose w is now a string of length $n+1$ and $w = x1$ and x is a binary string that has a value congruent to 2 mod 3. From IH2, $\delta^*(C, x) = C$. Since $\delta(C, 1) = C$, $\delta^*(C, w) = C$. That is, w has a value of $2x + 1$, which is equivalent to 2 mod 3.

- We have now shown that IH2 is true for all strings of length n, $n \geq 0$.
- From the two inductive hypotheses, we can conclude that w is accepted by D iff w is in L. In other words, $L(D) = L$.

3. Let L be the language generated by the regular expression

$$(a + b)^* a(a + b)(a + b)$$

(a) Construct a nondeterministic finite automaton (NFA) for L. (Identify the five components of your NFA. Use a transition diagram for the transition function.)

(b) Show how your NFA processes the input string `abaab`.

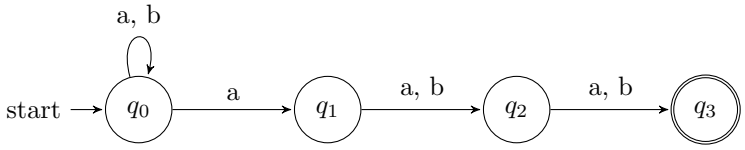
(c) Using the subset construction convert your NFA into an equivalent DFA.

(d) Show how your DFA processes the input string `abaab`.

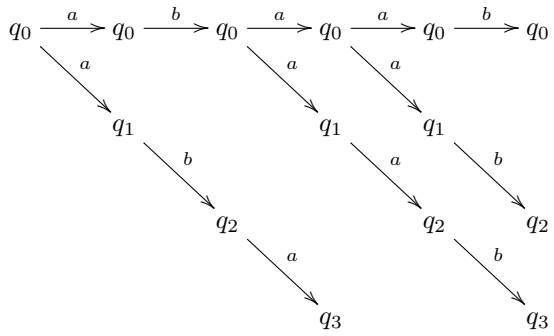
Solution (a) The five components are:

$$A = (\{q_0, q_1, q_2, q_3\}, \{\mathtt{a}, \mathtt{b}\}, \delta, q_0, q_3)$$

where δ is the transition diagram represented below.



(b)



where the initial q_3 becomes stuck, due to its inability to process the rest of the string.

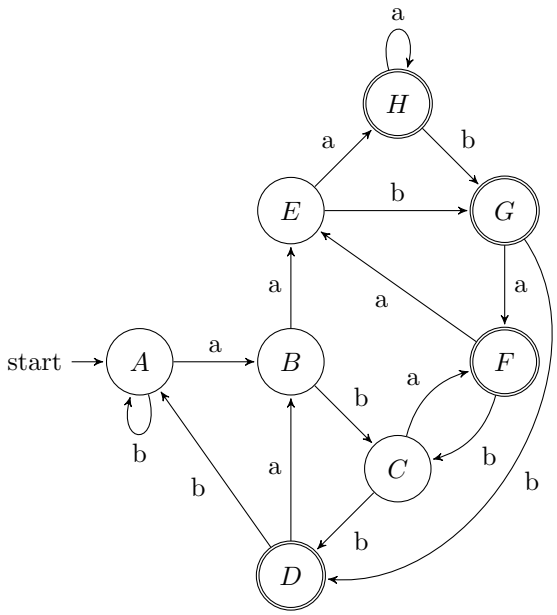
(c) We have this transition table formed through "lazy evaluation":

	a	b
$\rightarrow \{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$\{q_0, q_2\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$*\{q_0, q_3\}$	$\{q_0, q_1\}$	$\{q_0\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$
$*\{q_0, q_1, q_3\}$	$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$
$*\{q_0, q_2, q_3\}$	$\{q_0, q_1, q_3\}$	$\{q_0, q_3\}$
$*\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_1, q_2, q_3\}$	$\{q_0, q_2, q_3\}$

Let us rename the transition state table using states $A, \ldots$

	a	b
$\rightarrow A$	B	A
B	E	C
C	F	D
$*D$	B	A
E	H	G
$*F$	E	C
$*G$	F	D
$*H$	H	G

which gives us the transition diagram:



So now that we have this monstrosity, we have

$$A = (Q, \Sigma, \delta, q_0, F)$$

where

1. $Q = \{A, B, E, F, G, H\}$
2. $\Sigma = \{a, b\}$
3. δ is the transition diagram pictured above.
4. q_0 is state A.
5. $F = \{D, F, G, H\}$.

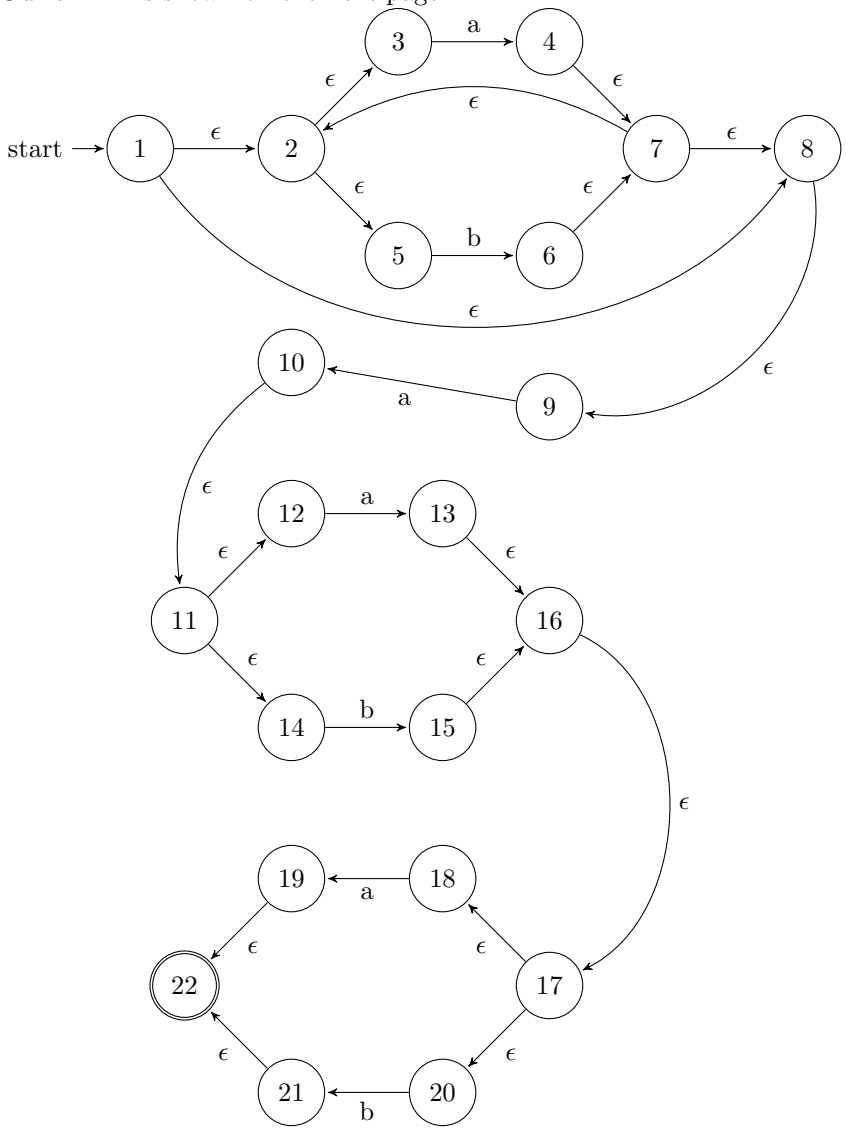
(d) We start on state A. The machine reads an a and transitions to state B, we then read a b, and move to state C. From there, we read an a and move to state F. The machine reads another a and moves to state E. Finally, the machine reads the last character, a b and halts on state G, accepting the string $abaab$.

4. Construct an epsilon-NFA from the regular expression in problem (3) using the McNaughton-Yamada-Thompson algorithm. Show how your epsilon-NFA processes the input string `abaab`.

Solution The regular expression in (3) is

$$(a+b)^*a(a+b)(a+b)$$

Our ϵ-NFA is shown on the next page:



Possible States	Next Input
$\rightarrow \{1, 2, 3, 5, 8, 9\}$	a
$\{4, 7, 2, 3, 5, 8, 9, 10, 11, 12, 14\}$	b
$\{6, 7, 2, 3, 5, 8, 9, 15, 16, 17, 18, 20\}$	a
$*\{4, 7, 2, 3, 5, 8, 9, 10, 11, 12, 14, 19, 22\}$	a
$\{4, 7, 2, 3, 5, 8, 9, 10, 11, 12, 14, 13, 16, 17, 18, 20\}$	b
$*\{6, 7, 2, 3, 5, 8, 9, 15, 16, 17, 18, 20, 21, 22\}$	$\emptyset$

Thus, the string `abaab` is accepted by this machine. The path is

$$1, 2, 3, 4, 7, 2, 5, 6, 7, 8, 9, 10, 11, 12, 13, 16, 17, 20, 21, 22$$

and this is found by following the epsilon closures after each input is read. The string will be accepted by this ϵ-NFA.

5. Let L be the language $\{w \mid w$ is any string of a's and b's where the number of a's is different from the number of b's $\}$. If L is regular, construct a DFA for L; if not, prove that L is not regular.

Solution: Let us assume that L is a regular language. Then $L = L(A)$ for some DFA A. Since L can be represented by a DFA, we can suppose that A has n states. Thus, we can apply the pumping lemma to DFA A, where A represents L, which is the language of any string of a's and b's where the number of a's is different from the number of b's. Using the Closure of Regular Languages under Complementation, we can find a complimentary language $\overline{L} = L(B)$ such that B is a DFA $(Q, \Sigma, \delta, q_0, Q - F)$, that is B is exactly like A, but the accepting states of A have become nonaccepting states of B. In our case, that means the only strings in this language are all strings of equal number of a's and b's (not in any particular order).

Now we use the pumping lemma to show that the complimentary language of L is not regular, via a proof by contradiction. Since $\overline{L}$ is a regular language, as stated above, we can apply the pumping lemma to it. Suppose n is the constant that must exist if $\overline{L}$ is regular. We then pick a string $w = a^n b^n$, that is, n a's followed by n b's, a string with equal a's and b's that must be in $\overline{L}$. Now, the string can be broken into three parts: x, y, and z. All we know is that $y \neq \epsilon$, and $|xy| \leq n$. Since $|xy| \leq n$, and xy comes at the front of W, we know that x and y only consist of a's. The pumping lemma tells us that xz is in $\overline{L}$, if $\overline{L}$ is regular, when $k = 0$. However, xz has n b's, since all the b's of w are in z. But xz also has fewer tha n a's, because we lost the a's of y. Since $y \neq \epsilon$, we know that there can be no more than $n - 1$ a's among x and z. Thus, after assuming $\overline{L}$ is a regular language, we have proved a fact known to be false, that xz is in $\overline{L}$, which is a clear violation of condition (3) of the pumping lemma. We have a proof by contradiction of the fact that $\overline{L}$ is not regular.

Using the closure under complementation for Regular Languages, we know that the complement of a regular language is another regular language.

Since we proved that $\overline{L}$ is not a regular language, we have also proved that L is not a regular language.