Name:

Formal Languages & Automata

Summer 2017 Practice Midterm Key

Four basic concepts where presented in the introductory material: Two machines (DFAs and NFAs) and two language descriptors (regular expressions and regular grammars). DFAs and NFAs can be described by transition graphs or transition tables. An overarching theorem is that each of these representations define exactly the same language class: Regular languages. A basic skill is the ability to construct a reduction from any one of these representations to any other. In practice, the typical sequence of reductions is:

$$\text{Regular expression} \mapsto \text{NFA} \mapsto \text{DFA} \mapsto \text{minimal DFA}$$

Regular expression can describe numbers and names (identifiers, password, ...) and can be useful in many applications.

Another basic idea is that regular languages are closed under many common operations: Union, intersection, complement, concatenation, star, and reversal. And another basic idea: The pumping lemma can be used to show some languages are not regular. And, yet another basic idea: Decision problems about regular languages are usually decidable. That is, decision problems usually have terminating algorithms that always correctly answers the question.

1 Induction, Contradictions, & Relations

Score

1. (5 pts) In graph theory, a *complete graph* G has an (undirected) edge from every node to *every other* node. But, a transition graph TG is *complete* only if there is an (directed) edge from every node to *every* node. Let $|\mathbb{V}| = n$ be the number of nodes in a graph. Prove that $|\mathbb{E}| = \binom{n}{2}$ for a complete graph G. Prove that $|\mathbb{E}| = n^2$ for a complete transition graph TG.

   Answer: As a basis, if $n = 1$ a complete graph has no edges and $\binom{1}{2} = 0$. On the other-hand a complete transition graph has $1 = n^2$ edge (a loop).

   Now assume the statements are `True` for some $n \geq 1$. and consider what happens when another vertex is added to a graph. To make the (undirected) graph complete, $n = \binom{n}{1}$ (undirected) edges must be added. And, the number of edges in the new graph is the number of edges in the old plus the n new edges. That is,

   $$\binom{n}{2} + \binom{n}{1} = \binom{n+1}{2} \quad \text{by Pascal's identity.}$$

   When a new vertex is added to a transition graph, there are several new edges to count:

   - There are n new edges from the old nodes to the new node.
   - There are n new edges from the new node to the old nodes.
   - There is 1 from the new node to itself.

   An additional $2n + 1$ (directed) edges must be added to complete the new transition graph. And, $n^2 + (2n + 1) = (n + 1)^2$.

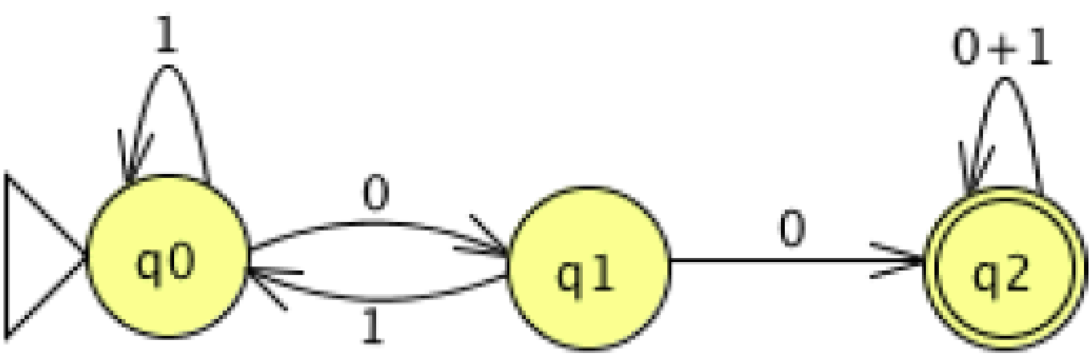
Score

2. (5 pts) One Gradiance question caused me to wonder: Is it possible to construct an NFA over $\Sigma = \{0, 1\}$ such that $n(k)$, the number of strings of length k, is F_{k+1}, a Fibonacci number? I think it is. Prove me right or wrong. Recall, the Fibonacci sequence is

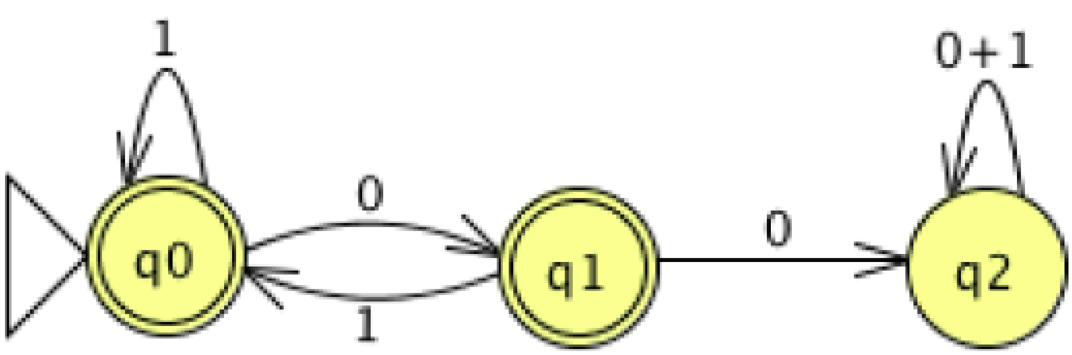
$$\langle F_0, F_1, F_2, F_3, F_5, F_6, \ldots\rangle = \langle 0, 1, 1, 2, 3, 5, \ldots\rangle$$

Answer: I recalled that certain binary strings are related to Fibonacci numbers. A quick search revealed the number binary strings without double zeros 00 is a forms the Fibonacci sequence.

A regular expression that describes all bit strings with a double zero is $(0+1)^*00(0+1)^*$. And a finite automata that accepts the double zero language is



The complement must be the strings without double zeros and its finite automata is



Score

3. (5 pts) Pretend that a string w of length n is accepted by DFA M. Pretend $n \geq m$ where m is the cardinality of states Q. Use the pigeonhole principle to prove that the transition graph for M has a cycle of length 1 or more.

Answer: A walk of length n in a graph transverse a sequence of states

$$\langle q_0, q_{\pi_1}, q_{\pi_2}, \ldots, q_{\pi_n} \rangle$$

This is a sequence of $n+1 > m$ states, and by the pigeonhole principle some state, call it q_k must occur twice. That is, there is some cycle of states $q_k, \cdots, q_k$ in the walk. The graph contains a cycle.

Score

4. (5 pts) Let M be a DFA. Say strings x and y in Σ^* are *indistinguishable*, written $x \equiv y$ if and only if

$$\delta^*(q_0, x) = \delta^*(q_0, y)$$

Prove that the *indistinguishable* relation is an equivalence, that is it is reflexive, symmetric, and transitive.

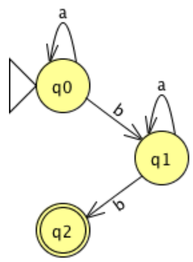
Answer: Since $\delta^*()$ is a function, the following hold

Reflexive	$\delta^*(q_0, x) = \delta^*(q_0, x)$
Symmetric	$(\delta^*(q_0, x) = \delta^*(q_0, y)) \Rightarrow (\delta^*(q_0, y) = \delta^*(q_0, x))$
Transitive	$((\delta^*(q_0, x) = \delta^*(q_0, y)) \wedge (\delta^*(q_0, y) = \delta^*(q_0, z))) \Rightarrow (\delta^*(q_0, x) = \delta^*(q_0, z))$

Therefore, *indistinguishable* is an equivalence relation on strings.

2 Machines and Languages

Consider the transition graph below for machine M (drawn using JFLAP).



Score

1. (5 pts) Does graph describe a DFA or an NFA?

   Answer: Machine M is a DFA: There is only one transition from each state on a given input and there are no λ transitions.

Score

2. (5 pts) Create a transition table for the machine

   Answer:

	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	$\emptyset$	$\emptyset$

Score

3. (5 pts) Is $L = L(M)$ regular? If so, answer the questions below. Otherwise, why is L not regular?

   (a) Write an English description of the strings w accepted by L.

   Answer: The string $w \in L$ if and only if it has exactly two b's and ends in a b. (It can have any number of a's.

   (b) Write a regular expression that describes strings w accepted by L.

   Answer: A regular expression for L is

$$r = a^*ba^*b$$

   (c) Construct a right-linear grammar for L.

   Answer: Use the state names as non-terminals (variables). The production rules are:

$$q_0 \to aq_0$$
$$q_0 \to bq_1$$
$$q_1 \to aq_1$$
$$q_1 \to bq_2$$
$$q_2 \to \lambda$$

3 Reduce NFA to DFA

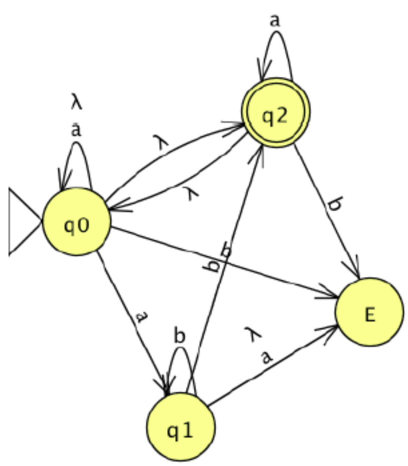
Consider the transition table for NFA N:

	a	b	λ
q_0	$\{q_0, q_1\}$	$\emptyset$	$\{q_2\}$
q_1	$\emptyset$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_2\}$	$\emptyset$	$\{q_1\}$

Score

1. (5 pts) Draw a transition graph for machine N. Consider using JFLAP unless you have another tool you know for the problem. (I guess by hand will do, but that is so nineteenth century)

   Answer: (drawn using JFLAP)



Score

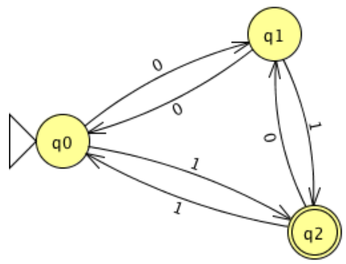
2. (5 pts) Convert N into a DFA.

   Answer: (drawn using JFLAP)

4 Reduce an NFA to a Regular Expression

Score

- (5 pts) Eliminate state q_1 from the transition graph below by:



  - Create an edge from q_0 to q_0 and appropriately label it with a regular expression r_{00} that describes all walks from q_0 to q_0.

$$r_{00} = 00$$

  - Create an edge from q_0 to q_2 and appropriately label it with a regular expression r_{02}.

$$r_{02} = 1 + 01$$

  - Create an edge from q_2 to q_0 and appropriately label it with a regular expression r_{20}.

$$r_{20} = 1 + 00$$

  - Create an edge from q_2 to q_2 and appropriately label it with a regular expression r_{22}.

$$r_{22} = 01$$

$$((00)^*(1+01)(01)^*(1+00))^*(00)^*(1+01)(01)^*$$

1. Why can you eliminate state q_1 and all edges from or to it to create a reduced 2-state transition graph?

   Answer: Every path from the initial state to the final state that passes through q_1 has been captured by these regular expressions.

2. Why does the regular expression

$$r_{00}^* r_{02}(r_{22} + r_{20} r_{00}^* r_{02})^*$$

   describe the regular expression of the automaton?

   Answer: Written out, the expression is

$$(00)^*(1+01)(01 + (1+00)(00)^*(1+01))^*$$

   To summarize, you can state in state q_0 reading double 0's for as long as you like. There are then two paths to the final state q_2: Directly by 1 and indirectly by 01 through q_1. Once in the final state you stay there reading 01 or go back to state q_0 via $1 + 00$, stay there for a while, and then come back to the final state q_2 via $1 + 01$. And you can do this any number of times.

5 Reduce Regular Expression to NFA

Score

(5 pts) Construct an NFA that accepts the language $L((bb)^*a^* + b^*(aa)^*)$.

Answer: The idea is to glue together simple parts to construct the machine. (drawn using JFLAP).

6 A NFA to a Regular Grammar

Score

1. (5 pts) Define: Right-Linear Grammar

   Answer: All productions have the form $A \to wB$ or $A \to w$ where $w \in \Sigma^*$.

Score

2. (15 pts) Consider the grammar $G = (S, T, V, P)$ with production rules

$$S \to aA$$
$$A \to aaA \mid \lambda$$

   (a) Construct a DFA that accepts/recognizes, strings derived from the grammar G.

   Answer:

   (b) What is the language $L = L(G)$ defined by the above grammar G?

   Answer:

   (c) What is a regular expression describes the grammar G?

   Answer:

7 The Pumping Lemma

Score

1. (5 pts) Let L be a regular language. What are the conclusions of the pumping lemma for regular languages? Give simple English conclusions and more precise mathematical formulations.

   Answer: In English, if $w \in L$ is sufficiently long, then a walk on w from the initial state to a final state must contain at least one cycle. Moreover, the walk from the initial state through the first cycle once can be be no longer than the number of states. Finally, the label on the cycle, call it y, can be repeated an arbitrary number of times.

   In mathematics, this translate into: There exists an m such that for every string $w \in L$ with $|w| \geq m$ can be written as

   $$w = xyz$$

   where

   $$|xy| \leq m \quad \text{and} \quad |y| \geq 1$$

   and

   $$xy^i z \in L \quad \text{for all } i \in \mathbb{N}$$

   $$(\exists m \in \mathbb{N})(\forall((|w| \geq m) \wedge (w \in L)))(\forall i \in \mathbb{N})((w = xyz) \wedge (|y| \geq 1) \wedge (w_i = xy^i z \in L))$$

Score

2. (5 pts) Use the pumping lemma to show that language

   $$L = \left\{ww : w \in \{a,\, b\}^* \quad \text{is not regular.}\right\}$$

   Answer: Consider the string $w = a^m b$ where m is the lower bound where pumping can begin. The string $ww = (a^m b)(a^m b)$ is in L. The pumping lemma says, if L is regular, then $(a^m b)(a^m b)$ can be written as $xyz \in L$ where $|xy| \leq m$, $|y| \geq 1$, and

   $$xy^i z = w'w' \in L \quad \forall i \in \mathbb{N}$$

   By construction of ww, $xy = a^k$ for some $k \leq m$. Let $|y| = j \geq 1$. Then, by the pumping lemma

   $$xz = a^{m-j} b a^m b \in L \quad \text{a contradiction!}$$

Score

3. (5 pts) Let L be a regular language with m states. Clearly, if its DFA M accepts some string of length less than m, then L is non-empty

   Prove that if L is non-empty, then M accepts a string w with $|w| < m$. (Hint: If L is non-empty, let $w \in L$ be as short as any word in L. If $|w| \geq m$, use the pumping lemma to derive a contradiction that w is among the shortest words accepted by M.

8 Applications

Score

1. (5 pts) Write a regular expression that describes fixed point rational numbers (an optionally signed arbitrarily long string of digits followed by an optional decimal point optionally followed by an arbitrarily lone string of digits). Use the productions below for atomic derivations.

$$\begin{aligned} \textbf{digit} &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9 \\ \textbf{sign} &\to + \mid - \mid \lambda \\ \textbf{point} &\to . \end{aligned}$$

Answer: It may be a good idea to review ways to write fixed-point numbers.

- No sign: 123.321
- Signs: ±123.321
- No point: 123
- No fractional part: 123.
- No integer part: .321
- Full: (Sign part)(Integer + Integer. + .Integer + Integer.Integer)

This suggests the expression

$$(\textbf{sign})(\textbf{digit}^+ + \textbf{digit}^+\textbf{point} + \textbf{point digit} + \textbf{digit}^+\textbf{point digit}^+)$$

The Hopcroft-Ullman book (Hopcroft and Ullman, 1979) gives this answer:

$$(\textbf{sign})(\textbf{digit})^+(\textbf{point})(\textbf{digit}^* + \lambda) + (\textbf{point})(\textbf{digit})^+$$

Score

2. (5 pts) Describe the following passwords rules using regular expressions.

   (a) A password must be 5 or more characters long.
   Answer: The set of characters is undefined. Based on the question, I assume it is the set of lower and upper case English letters and the digits. Let l and u, and d represent arbitrary lower and upper case letters, an digits. The regular expression is

   $$(l + u + d)^5(l + u + d)^*$$

   (b) A password must be contain at least one lower case and one upper case English letter.
   Answer: Either the first l is before the first u, or vice versa. Before these distinguished characters any number of d's can occur. So the regular expression is

   $$d^*l(d + l)^*u(d + l + u)^* + d^*u(d + u)^*l(d + l + u)^*$$

   (c) A password must contain at least one digit.
   Answer: The regular expression is

   $$(l + u)^*d(l + u + d)^*$$

Comment: Put these three rules (regular expressions) together as an *and* phrase?

References

Hopcroft, J. E. and Ullman, J. D. (1979). *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley. [page 9]