CHMC Advanced Group: Finite Automaton and Regular Languages

6/22/2019

1 Introduction

In the theory of computation, one considers questions concerning the notions of complexity and computability. To establish the foundation of these ideas and, more broadly, to understand what defines a computer, simple models are built at first. From these, more complex models and associated theories are constructed. In this worksheet, you will learn about one of the simplest models, the finite automaton, and regular languages which are closely related to finite automaton.

2 Finite Automaton

We begin by first defining what a finite automaton is and then explore some examples.

A *finite automaton* is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

1. Q is a finite set called the *states*.
2. Σ is a finite set call the *alphabet*.
3. $\delta : Q \times \Sigma \to Q$ is the *transition function*.
4. $q_0 \in Q$ is the *start state*
5. $F \subseteq Q$ is the set of *accept states*.

To unwrap the definition, we go step by step through the five points, with an example to illustrate these ideas.

Consider an elevator in a building with five floors. Each floor represents a state the elevator may occupy. The alphabet in this case is the set of inputs, namely the floor buttons $\{1, 2, 3, 4, 5\}$. The elevator has transition function that takes a floor number as an input and changes the floor the elevator is currently on to the input floor, and opens the door. If the elevator is already on the input floor, the elevator does not move and its doors open. For this example, we do not specify an accept state, and we specify a start state as the first floor. Explicitly, this data can be summarized as

1. $Q = \{\text{Floor } 1,\ \text{Floor } 2,\ \text{Floor } 3,\ \text{Floor } 4,\ \text{Floor } 5\}$.
2. $\Sigma = \{1, 2, 3, 4, 5\}$.
3. $\delta(\text{Floor } i, j) = \text{Floor } j$.
4. $q_0 = 1$.
5. $F = \emptyset$.

The finite set Q is a set of possible states the automaton can be in at any given point. The alphabet Σ can be considered as possible inputs to the finite automaton. The transition function δ is a function that takes in both a state and a letter from the alphabet and returns a state. Think of this as a list of instructions that the machines follows, telling it what to do when a given letter is input while the machine is in a given state. The start state q_0 tells the machine in which state it should start before it begins receiving input. Finally, the accept states F is a collection of states in which the finite automaton can stop taking inputs and can possibly be empty, in which case the automaton never stops. The finite automaton may be in an accept state and continue to take inputs, so it does not need to stop once it has hit an accept state.

Consider the following example of a finite automaton, denoted here by M_1.

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3.

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

4. q_1 is the start state.
5. $F = \{q_2\}$

Exercise 2.1 For each state in M_1, draw a small circle and label it. For the accept states, draw a second circle inside the circle associated to it to remind yourself which states are accept states. For each instruction given by the transition function, draw an arrow from the input state to the output state and label the arrow with the appropriate letter associated to it. For instance, since $\delta(q_1, 0) = q_1$, draw an arrow from q_1 to itself and label it with 0. Similarly, since $\delta(q_2, 0) = q_3$, draw an arrow from q_2 to q_3 labelled 0.

A word is built out of the letters of the alphabet. Given an input of a word, a finite automaton reads each letter in the word from left to right, evaluating the transition function on each letter of the word that it encounters.

Exercise 2.2 Try inputting various words into M_1 constructed from the alphabet $\Sigma = \{0, 1\}$. Which words put the machine into an accept state? Remember, a machine does not need to stop if it is in an accept state, but rather can continue on. Which words do not put the machine into an accept state? Consider what letter must be used to allow the machine into an accept state. Also, if a word does not end in a 1, what must it end in to be accepted?

If A denotes the set of strings that a machine M accepts, we say that M *accepts* A or M *recognizes* A and that A is the *language* of M. We denote this by $L(M) = A$. If there are no accept states, ($F = \emptyset$), then the machine does not recognize any language. In addition, if there are no paths consisting of states $q_{i_1}, q_{i_2}, \ldots, q_{i_k}$ and inputs $i_1, i_2, \ldots, i_{k-1}$ such that $\delta(q_{i_j}, i_j) = q_{i_{j+1}}$ where q_{i_1} is the start state and q_{i_k} is an accept state, then the finite automaton does not recognize any language. Finally, if the start state is an accept state, then the empty word is accepted by M, so it is in A.

Exercise 2.3 In the previous exercise, you identified words that the machine accepted. Try to generalize your results to determine the language A_1 that the machine M_1 recognizes.

Consider the finite automaton M_2 given by the following data,

1. $Q = \{q_1, q_2\}$
2. $\Sigma = \{0, 1\}$
3.

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

4. q_1 is the start state.
5. $F = \{q_2\}$

Exercise 2.4 Try out different words from the alphabet $\Sigma = \{0, 1\}$. Which words are not accepted? Which words are accepted? Can you determine what $L(M_2)$ is?

Exercise 2.5 What happens if you change the accept state in M_2 from $F = \{q_2\}$ to $F = \{q_1\}$? What is the language that is recognized? Denote this machine by M_2'.

Let M_3 denote the finite automaton given by the following data,

1. $Q = \{s, q_1, q_2, r_1, r_2\}$
2. $\Sigma = \{a, b\}$
3.

	a	b
s	q_1	r_1
q_1	q_1	q_2
q_2	q_1	q_2
r_1	r_2	r_1
r_2	r_2	r_1

4. s is the start state.
5. $F = \{q_1, r_1\}$

Exercise 2.6 Draw a diagram of the automaton M_3. Try out various words. Can you determine which words that start with an a are words M_3 accepts? How about the words that start with an a b that M_3 accepts. Once you have figured this, try to explicitly write down what $L(M_3)$ is. Hint: Once you have initially chosen a, does that side of the finite automaton look like one of the previous examples you have considered?

Define M_4 to be the finite automaton with the following data,

1. $Q = \{q_0, q_1, q_2\}$
2. $\Sigma = \{0, 1, 2, \text{reset}\}$
3.

	0	1	2	reset
q_0	q_0	q_1	q_2	q_0
q_1	q_1	q_2	q_0	q_0
q_2	q_2	q_0	q_1	q_0

4. q_0 is the start state.
5. $F = \{q_0\}$

In this automaton, the "reset" letter automatically sends the automaton back to the start state.

Exercise 2.7 Draw a diagram of M_4 and experiment with different words. Since "reset" always sends the machine back to the start state, try words that only consist of the characters 0, 1, and 2. Look at the sums of digits in words that are accepted and words that are not accepted. Do you see a pattern? Try to determine what $L(M_4)$ is.

Exercise 2.8 Try to generalize the construction of M_4 using the same alphabet, but with states $\{q_0, q_1, q_2, \ldots, q_{n-1}\}$ for $n \geq 4$. This will recognize a language similar to that of $L(M_4)$, but this time based off of n rather than 3.

3 Regular Language

In the previous section, we introduced the notion of a finite automaton and explored a few examples. In addition, we also explored languages that these particular automaton recognized.

We say that finite automaton M *recognizes language* A if $A = \{w \mid M \text{ accepts } w\}$. A language is called a *regular language* if there is some finite automaton that recognizes it.

In the previous section, the languages you determined from the different finite automaton were all languages accepted by a finite automaton, namely the automaton associated to it. Thus, each of these languages is a regular language. However, one may be given a regular language and try to build a finite automaton (which necessarily exists since the language is regular) that recognizes the language.

Consider the language E_1 consisting of strings of 0's and 1's such that each word has an odd number of 1's. To construct a finite automaton consider reading a string that is input one character at a time. The automaton does not need to keep track of the whole string, but rather whether the number of 1's read so far is even or odd. If a 0 is read, the parity (even or odd) of 1's does not change; if a 1 is read, the parity of 1's switches. This suggests there should be two states, q_{even} and q_{odd}. Next, we determine a transition function. As noted before, an input of 0 does not change the state, so $\delta(q_{\text{even}}, 0) = q_{\text{even}}$ and $\delta(q_{\text{odd}}, 0) = q_{\text{odd}}$. Finally, an input of a 1 changes the state to the other state. Hence, $\delta(q_{\text{even}}, 1) = q_{\text{odd}}$ and $\delta(q_{\text{odd}}, 1) = q_{\text{even}}$. To determine the start state, we must start in this state assuming the finite automaton has not seen any symbols so far, which means an even number of 1's and thus in q_{even}. Finally, the accept state will be q_{odd} since the finite automaton must accept strings with an odd number of 1's.

To summarize, the finite automaton constructed above that recognizes E_1 is given by,

1. $Q = \{q_{\text{even}}, q_{\text{odd}}\}$
2. $\Sigma = \{0, 1\}$
3.

	0	1
q_{even}	q_{even}	q_{odd}
q_{odd}	q_{odd}	q_{even}

4. q_{even} is the start state.
5. $F = \{q_{\text{odd}}\}$

Let E_2 be the language consisting of words that contain the string 11, where the alphabet is $\Sigma = \{0, 1\}$. As the automaton scans digits, if it sees a 0 it keeps scanning. However, if it sees a 1, it should move to a new state. In this new state, if it sees a 0 it should return back to the old state; if it sees a 1 it should move to an accept state. If the finite automaton is in the accept state, reading a 0 or 1 should not move it out of the accept state. Thus, there should be three states, q, q_1, q_{11}. If the automaton is in q and reads a 0, it stays put and if it reads a 1, it moves to q_1. If the automaton is in q_1 and reads a 0, it moves back to q and if it reads a 1, it moves to q_{11}. Finally, if it is in q_{11}, it should remain regardless of what it reads. The automaton should start in q and accept in q_{11}. Thus, it is given by

1. $Q = \{q, q_1, q_{11}\}$
2. $\Sigma = \{0, 1\}$
3.

	0	1
q	q	q_1
q_1	q	q_{11}
q_{11}	q_{11}	q_{11}

4. q is the start state.
5. $F = \{q_{11}\}$

Exercise 3.1 Let E_2' be the language consisting of word that end with the string 11, where the alphabet is $\Sigma = \{0, 1\}$. Adapt the finite automaton that recognizes E_2 to a finite automaton that recognizes E_2'. Think about what happens if the machine is in the accept state and reads a 0 or a 1?

Exercise 3.2 Let E_3 be the language that contains the string 101, where the alphabet is $\Sigma = \{0, 1\}$. Build a finite automaton that recognizes this language.

Suppose that a language E can be written as the union of two different regular languages, say $E = E' \cup E''$, where each word in E is either a word from E' or a word from E''. A natural question that arises is can a finite automaton be built that recognizes E, that is, is the union of two regular languages a regular language itself? In fact, this is possible.

The idea is that you run both finite automaton simultaneously, where states of this new automaton are pairs of states, one from each of the two finite automaton that recognize each language. As an input is read, the state in the first coordinate changes according to the first finite automaton and the state in the second coordinate changes according to the second finite automaton. The start state of the new automaton is the ordered pair of start states of the two finite automata. Finally, the new finite automaton is in an accept state if either coordinate is in an accept state. Note, if instead the new finite automaton were in an accept state only when both coordinates occupied an accept state, this would then only recognize words that were in both languages, the intersection of E' and E'', denoted by $E = E' \cap E''$.

Let $(Q_1, \Sigma, \delta_1, q_1, F_1)$ and $(Q_2, \Sigma, \delta_2, q_2, F_2)$ be two finite automata that recognize E' and E'' respectively. Define the following data,

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$.
2. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$.
3. $q = (q_1, q_2)$.
4. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$.

Then $(Q, \Sigma, \delta, q, F)$ is a finite automaton that recognizes $E' \cup E''$.

Similarly,

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1,\ r_2 \in Q_2\}$.
2. $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$.
3. $q = (q_1, q_2)$.
4. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}$.

describes a finite automaton that recognizes $E' \cap E''$.

Recall that E_1 consists of words constructed from the alphabet $\Sigma = \{0, 1\}$ that contain an odd number of 1's, and E_2 consists of words using the same alphabet that contain the string 11.

Exercise 3.3 Construct a finite automaton that recognizes $E = E_1 \cup E_2$ from above. Note, you should have 6 possible states and 4 accept states.

Exercise 3.4 Construct a finite automaton that recognizes $E = E_1 \cap E_2$ from above. Note, you should have 6 possible states and 1 accept state.

References

1. Sipser, Michael, *Introduction to the Theory of Computation,* Second Edition, Thomson Course Technology, Boston, 2006.