HW1 : Regular Expressions and Deterministic Finite Automata

CSE105Sp22 Solutions

In this assignment,

You practiced reading and applying the definitions of alphabets, strings, languages, Kleene star, and regular expressions. You used regular expressions and relate them to languages and automata. You used precise notation to formally define the state diagram of DFA, and you used clear English to describe computations of DFA informally.

Reading and extra practice problems: Sipser Section 0, 1.3, 1.1. Chapter 1 exercises 1.1, 1.2, 1.3, 1.18, 1.23.

Assigned questions

1. (*Graded for correctness*[1]) For L a set of strings over the alphabet $\{0,1\}$, we can define the following associated sets

$$LZ(L) = \{0^k w \mid w \in L, k \in \mathbb{Z}, k \geq 0\}$$

$$TZ(L) = \{w0^k \mid w \in L, k \in \mathbb{Z}, k \geq 0\}$$

Note: the commas in the set-builder definition indicate "and".

Note: 0^k is the result of concatenating 0 with itself k times; it is the string of k 0s.

Note: Formally, LZ and TZ are each functions, with domain the set of languages over $\{0,1\}$ and with codomain the set of languages over $\{0,1\}$.

(a) Specify an example language L_1 over $\{0,1\}$ where $LZ(L_1) = \Sigma^*$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language L_1 and a precise and clear description of the result of computing $LZ(L_1)$ using the definitions to justify this description and

[1]This means your solution will be evaluated not only on the correctness of your answers, but on your ability to present your ideas clearly and logically. You should explain how you arrived at your conclusions, using mathematically sound reasoning. Whether you use formal proof techniques or write a more informal argument for why something is true, your answers should always be well-supported. Your goal should be to convince the reader that your results and methods are sound.

justifying the set equality with Σ^*, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Sample solution: Consider $L_1 = \{0,1\}^*$, the set of all strings over $\{0,1\}$. Applying the definition

$$LZ(L_1) = \{0^k w \mid w \in \{0,1\}^*, k \in \mathbb{Z}, k \geq 0\}$$

To show that this set is the set of all strings over $\{0,1\}$, we need to show the set equality

$$LZ(L_1) = \{0,1\}^*$$

and we proceed by showing two subset inclusions:

- To prove that $LZ(L_1) \subseteq \{0,1\}^*$, consider an arbitrary element, call it x, of $LZ(L_1)$ and we work to show that it is an element of $\{0,1\}^*$. By definition of $LZ(L_1)$, x can be written as the concatenation of a string of 0s with a string over $\{0,1\}$. This means that x is a string of 0s and 1s, so is an element of $\{0,1,\}^*$, as required.
- To prove that $\{0,1\}^* \subseteq LZ(L_1)$, consider an arbitrary string y over $\{0,1\}$, and we need to show that it's in $LZ(L_1)$. By properties of the empty string, $y = \varepsilon y$, and we can rewrite the empty string as $\varepsilon = 0^0$. Thus, since 0 is an integer and $0 \geq 0$, we have written $y = 0^k w$ for $k = 0$ (a nonnegative integer) and $w = y$ (a string over $\{0,1\}$) and so y satisfies the condition in the set builder definition of $LZ(L_1)$, as required.

(b) Specify an example language L_2 over $\{0,1\}$ where $LZ(L_2)$ is a finite set, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language L_2 and a precise and clear description of the result of computing $LZ(L_2)$ using the definitions to justify this description and justifying why it is finite, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Sample solution: Consider $L_2 = \emptyset$, the empty set. In this case, applying the definition gives

$$LZ(L_2) = \{0^k w \mid w \in \emptyset, k \in \mathbb{Z}, k \geq 0\} = \emptyset$$

because there is no value for w that will make the condition in the set builder definition true, so there are no elements in this set. Since the empty set is finite (it has zero many elements, a nonnegative integer), $L_2 = \emptyset$ is the example we were looking for.

(c) Specify an example language L_3 over $\{0,1\}$ where $LZ(L_3) = TZ(L_3)$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language L_3 and a precise and clear description of the results of computing $LZ(L_3)$ and $TZ(L_3)$ using the definitions to justify this description and justifying the set equality, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Sample solution: Consider $L_3 = \{0\}$, the set whose only element is the string 0. In this case, applying the definitions give

$$LZ(L_3) = \{0^k w \mid w \in \{0\}, k \in \mathbb{Z}, k \geq 0\} = \{0^k \circ 0 \mid k \in \mathbb{Z}, k \geq 0\} = \{0^{k+1} \mid k \in \mathbb{Z}, k \geq 0\}$$
$$TZ(L_3) = \{w 0^k \mid w \in \{0\}, k \in \mathbb{Z}, k \geq 0\} = \{0 \circ 0^k \mid k \in \mathbb{Z}, k \geq 0\} = \{0^{k+1} \mid k \in \mathbb{Z}, k \geq 0\}$$

We see that these two sets are equal, as required. Notice that these sets are each described by the regular expression 0^+.

2. (*Graded for correctness*) Consider the two regular expressions over $\Sigma = \{0, 1\}$

$$R_1 = (\ (000 \cup 111)^* \ \cup\ (01)^*\ ) \qquad\qquad R_2 = (\ (000)^*(111)^*(\varepsilon \cup 0 \cup 1))$$

You will prove that

$$L(R_1) \not\subseteq L(R_2) \text{ and } L(R_2) \not\subseteq L(R_1) \text{ and } L(R_1) \cap L(R_2) \neq \emptyset \text{ and } L(R_1) \cup L(R_2) \neq \Sigma^*$$

by giving four example strings that witness these properties.

(a) Specify an example string w_1 such that $w_1 \in L(R_1) \cap L(R_2)$. Briefly justify your choice, referring to the definitions of the regular expressions and their semantics.

Sample solution: Consider $w_1 = \varepsilon$. To (informally) see that $w_1 \in L(R_1)$ we see that it can be written a 0 many "slots" in either of the Kleene star expressions, and since the two Kleene star expressions are combined with the union symbol, matching either one is sufficient for membership in the set being described. To (informally) see that $w_1 \in L(R_2)$ we notice that

$$\varepsilon = \varepsilon \circ \varepsilon \circ \varepsilon$$

so w_1 can be expressed as the concatenation of three parts, the first of which is in $L((000)^*)$ (using 0 slots), the second of which is in $L((111)^*)$ (again using 0 slots), and the third of which is in $L((\varepsilon \cup 0 \cup 1)) = \{\varepsilon\} \cup \{0\} \cup \{1\} = \{\varepsilon, 0, 1\}$. By definition of intersection, since $w_1 \in L(R_1)$ and $w_1 \in L(R_2)$, $w_1 \in L(R_1) \cap L(R_2)$.

(b) Specify an example string w_2 such that $w_2 \in L(R_1) \cap \overline{L(R_2)}$. Briefly justify your choice, referring to the definitions of the regular expressions and their semantics.

Sample solution: Consider $w_2 = 01$. To (informally) see that $w_1 \in L(R_1)$ we write $w_2 = 01 = (01)^1$ so $w_2 \in \{(01)^k \mid k \geq 0, k \in \mathbb{Z}\} = L((01)^*)$ and since $L(R_1)$ is the union of some set with $L((01)^*)$, membership in $L((01)^*)$ is sufficient to guarantee membership in $L(R_1)$. To (informally) see that $w_2 \notin L(R_2)$ we notice that there's no way to express it as a concatenation of some number of triples of 0s followed by some number of triples of 1s followed by at most one character, because w_2 has length 2 so there can be no 000s or 111s and the max length of the allowed suffix is 1, which is not enough for w_2.

(c) Specify an example string w_3 such that $w_3 \in \overline{L(R_1)} \cap L(R_2)$. Briefly justify your choice, referring to the definitions of the regular expressions and their semantics.

Sample solution: Consider $w_3 = 0$. To (informally) see that $w_3 \notin L(R_1)$ we notice that its length (1) is not an integer multiple of 3 so $w_3 \notin L(\ (000 \cup 111)^*)$ and its length is also not an integer multiple of 2 so $w_3 \notin L((01)^*)$. Since $L(R_1)$ is the union of these two sets, when w_3 is not in either of the sets, it's not in the union. To (informally) see that $w_3 \in L(R_2)$ we write $w_3 = 0 = \varepsilon \circ \varepsilon \circ 0$ so w_3 can be expressed as the concatenation of three parts, the first of which is in $L((000)^*)$ (using 0 slots), the second of which is in $L((111)^*)$ (again using 0 slots), and the third of which is in $L((\varepsilon \cup 0 \cup 1)) = \{\varepsilon\} \cup \{0\} \cup \{1\} = \{\varepsilon, 0, 1\}$.

(d) Specify an example string w_4 such that $w_4 \in \overline{L(R_1)} \cap \overline{L(R_2)}$. Briefly justify your choice, referring to the definitions of the regular expressions and their semantics.

Sample solution: Consider $w_4 = 00$. To (informally) see that $w_4 \notin L(R_1)$ we notice that its length (2) is not an integer multiple of 3 so $w_4 \notin L(\ (000 \cup 111)^*)$ but it's also not an element of $L((01)^*)$ because it's a nonempty string that has no 1s. Since $L(R_1)$ is the union of these two sets, when w_4 is not in either of the sets, it's not in the union. To (informally) see that $w_4 \notin L(R_2)$ we notice that there's no way to express it as a concatenation of some number of triples of 0s followed by some number of triples of 1s followed by at most one character, because w_4 has length 2 so there can be no 000s or 111s and the max length of the allowed suffix is 1, which is not enough for w_4.

3. (*Graded for fair effort completeness*[2])

(a) Pick a four letter alphabet (a nonempty finite set), and specify it, e.g. by filling in the blank $\Sigma =$ fill in your alphabet here.

Then, pick a language of cardinality (size) 2 over this alphabet, and specify it, e.g. by filling in the blank

$$L = \underline{\text{fill in your language here}}$$

Note: we encourage you to pay attention to syntax here. There are many correct answers; please be precise in how you present the sets you choose.

Sample solution: We define

$$\Sigma = \{1, 2, 3, 4\}$$

and

$$L = \{123, 4\}$$

(b) Give a regular expression that describes the language L you defined in part (a). Briefly justify why your regular expression works.

Sample solution: A regular expression that describes L is $123 \cup 4$, which is shorthand for

$$((1 \circ (2 \circ 3)) \cup 4)$$

[2]This means you will get full credit so long as your submission demonstrates honest effort to answer the question. You will not be penalized for incorrect answers. To demonstrate your honest effort in answering the question, we ask that you include your attempt to answer *each* part of the question. If you get stuck with your attempt, you can still demonstrate your effort by explaining where you got stuck and what you did to try to get unstuck.

Applying the definition of the language described by a regular expression:

$$\begin{aligned} L((1 \circ (2 \circ 3)) \cup 4) &= L(1 \circ (2 \circ 3)) \cup L(4) \\ &= L(1 \circ (2 \circ 3)) \cup \{4\} \\ &= L(1) \circ L((2 \circ 3)) \cup \{4\} \\ &= \{1\} \circ L((2 \circ 3)) \cup \{4\} \\ &= \{1\} \circ L(2) \circ L(3) \cup \{4\} \\ &= \{1\} \circ \{2\} \circ \{3\} \cup \{4\} \\ &= \{123\} \cup \{4\} = \{123, 4\} \end{aligned}$$

(c) Give a DFA that recognizes your language L you defined in part (a). Specify your DFA **both** using a formal definition **and** a state diagram. Briefly justify why your DFA works.

Sample solution: Informally, we'll build a DFA that has only two paths from the start state to the accept state, with each path labelled by one of the strings in the language L. The computation of the DFA on any other string should end in a state that is not in the set of accepting states.

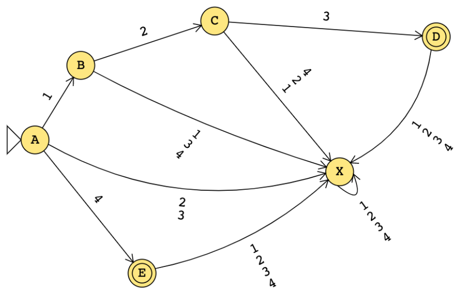
The formal definition of the DFA is

$$(\{A, B, C, D, E, X\}, \{1, 2, 3, 4\}, \delta, A, \{E, D\})$$

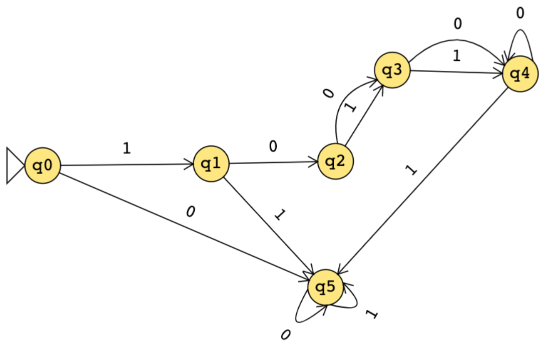
where $\delta : \{A, B, C, D, E, X\} \times \{1, 2, 3, 4\} \to \{A, B, C, D, E, X\}$ is given by the table of values:

	1	2	3	4
A	B	X	X	E
B	X	C	X	E
C	X	X	D	E
D	X	X	X	X
E	X	X	X	X
X	X	X	X	X

The state diagram of this DFA is:



4. (*Graded for correctness*) Consider the DFA C given by the state diagram below.



State diagram for DFA C

Suppose someone tells you that the formal definition of this DFA is

$$(Q, \Sigma, \delta, q_0, F) = (\{q0, q1, q2, q3, q4, q5\}, \{0, 1, 2\}, \delta, q0, q0)$$

where $\delta : Q \times \Sigma \to Q$ is given by

$$\delta((q, 0)) = \begin{cases} q5 & \text{if } q = q0 \\ qj & \text{if } q = qi \text{ and } i \in \{1, 2, 3\} \text{ and } j = i + 1 \\ q & \text{if } q \in \{q4, q5\} \end{cases} \qquad \delta((q, 1)) = \begin{cases} q1 & \text{if } q = q0 \\ q5 & \text{if } q \in \{q1, q4, q5\} \\ \delta((q, 0)) & \text{if } q = q2 \text{ or } q = q3 \end{cases}$$

(a) Confirm that this formal description is correct (in that it is consistent with the state diagram), or fix any and all mistakes in it. In your solution, explicitly address whether the description of the set of states is correct, whether the description of the alphabet is correct, whether the description of the transition function is correct, whether the description of the start state is correct, and whether the description of the accept states is correct, and why.

Sample solution: Let's go through each component of the formal definition in turn:

- The set of states, $Q = \{q0, q1, q2, q3, q4, q5\}$ is consistent with the state diagram.
- The alphabet $\Sigma = \{0, 1, 2\}$ is **not** consistent with the state diagram because the state diagram has no arrows labelled with 2 and a DFA state diagram needs to represent what the computation of the DFA will do on *any* input string over the specified alphabet. In particular, this means that every state in the DFA will have an outgoing arrow labelled with each symbol from the alphabet of the DFA.
- The transition function δ as specified is consistent with the state diagram, which

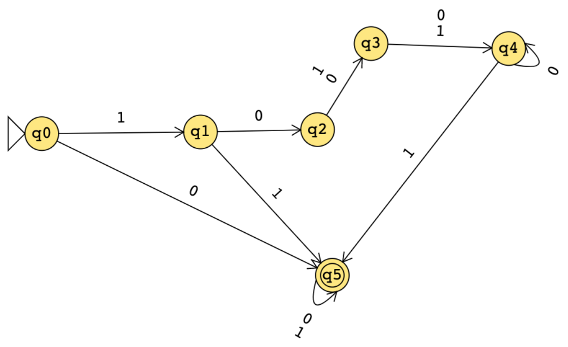
we can see by looking at each value it specifies:

$\delta((q0,0)) = q5$ gives the arrow labelled 0 from $q0$ to $q5$
$\delta((qi,0)) = qj$ if $q = qi$ and $i \in \{1,2,3\}$ and $j = i+1$ gives the arrows labelled 0 that go from $q1$ to $q2$, from $q2$ to $q3$, and from $q3$ to $q4$
$\delta((q,0)) = q$ if $q \in \{q4, q5\}$ gives the self-loops labelled 0 at $q4$ and $q5$
$\delta((q0,1)) = q1$ gives the arrow labelled 1 from $q0$ to $q1$
$\delta((q,1)) = q5$ if $q \in \{q1, q4, q5\}$ gives the arrows labelled 1 from $q1$, $q4$, and $q5$ to $q5$
$\delta((q,1)) = \delta((q,0))$ if $q = q2$ or $q = q3$ because the transitions from $q2$ and $q3$ are the same regardless of which input symbol is read

- The start state is consistent with the state diagram because $q0$ is drawn with a triangle next to it.
- The set of accept states is **not** consistent with the state diagram because it is of the wrong type and has the wrong value: the set of accept states needs to be a **set** (not an element of the set of states). In the state diagram we see that no states are drawn with double circles so the set of accept states is empty.

(b) Modify the set of accept states of this state diagram to get a different DFA (with the same set of states, alphabet, start state, and transition function) that recognizes an infinite language. Your solution should include the diagram of this new DFA and an explanation of why the language it recognizes is infinite.

Sample solution: Consider the DFA that with the same set of states, alphabet, start state, and transition function as the DFA in the state diagram of part (a), and has $\{q5\}$ as the set of accept states. Its state diagram is



To show that the language that this DFA recognizes is infinite, we will show that the DFA accepts each string of the form 0^k for $k \in \mathbb{Z}$, $k \geq 1$. There are infinitely many strings so the language of the DFA (which is a superset of the set of these nonempty strings of 0s), must also be infinite. Let k be an arbitrary positive integer. The computation of the DFA on 0^k starts at $q0$ and then transitions to $q5$ on reading the first 0, and then stays at $q5$ while the remaining $k-1$ 0s are read. At the end of processing the input string, the computation is therefore at $q5$, which is in the set of accepting states so the DFA accepts the string 0^k. *Note that it was important that $k > 0$ so that the first transition brings us to $q0$ rather than staying at $q0$.*

5. (*Graded for fair effort completeness*) Which of the following are valid descriptions using the terminology we have used in class and in the book so far? For those that aren't, explain what's wrong. For those that are, give an example of what's being described.

   (a) A finite automaton accepts a regular expression.

   Sample solution: This is not valid. Finite automata accept or reject **strings**, not regular expressions.

   (b) The language described by a regular expression is a finite automaton.

   Sample solution: This is also not valid. The language described by a regular expression is a set of strings whereas a finite automaton is a machine (not a set of strings).

   A valid statement that one could say is "The language described by a regular expression is a language recognized by a finite automaton."

   (c) The empty string is the language of some regular expression.

   Sample solution: This is not valid. The empy string is a string, not a set of strings. A language is a set of strings. The language of a regular expression is the language described by a regular expression and is the set of strings that is defined based on the semantics of the constituent pieces of the regular expression.

   (d) A string of length one uses one symbol from the alphabet.

   Sample solution: This statement is valid and correct. An example is that for the alphabet $\{0,1\}$, the string 1 is a string of length 1 and it uses one symbol (the symbol 1) from the alphabet.

   (e) The input string runs a finite automaton.

   Sample solution: This statement is not valid. String do not run machines.

   A valid statement that one could say is "The finite automaton runs (a computation) on an input string."