HW1CSE105F24: Homework assignment 1 solution

CSE105F24 Team

Due: October 8th at 5pm, via Gradescope

In this assignment,

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

Resources: To review the topics for this assignment, see the class material from Weeks 0 and 1. We will post frequently asked questions and our answers to them in a pinned Piazza post.

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. **Finding examples and edge cases** (12 points):

With $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$ and $\Gamma = \{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$

(a) (*Graded for completeness*) [1] Give an example of a string over Σ that is meaningful to you in some way and whose length is between 5 and 20, and explain why this string is meaningful to you.

> **Solution:** An example of a string over Σ is **10824**, which is significant because this is the date that HW 1 is due!

[1]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 expect you to 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.

(b) (*Graded for completeness*) Calculate the number of distinct strings of length 3 over Σ and then explain your calculation.

> **Solution:** There are 10 distinct elements in Σ, the order of the elements matters in a string, and we can have duplicate elements in a string (ex: 100). This means that we will choose out of 10 elements for every character in the string, or the length of the string, for a total of $10^3 = \mathbf{1000}$ distinct strings of length 3 over Σ.

(c) (*Graded for completeness*) With the ordering $0 < 1 < 2 < 3 < 4 < 5 < 6 < 7 < 8 < 9 < A < B < C < D < E < F$, list the first 50 strings over Γ in string order. Explain how you constructed this list. *Note: you can write a program to generate this list if you'd like, and you may use any external tools to help you write this program. If you do use a program to generate the list, include it (and documentation for how it works) as part of your submission.*

> **Solution:** ε, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, 00, 01, 02, 03, 04, 05, 06, 07, 08, 09, 0A, 0B, 0C, 0D, 0E, 0F, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 1A, 1B, 1C, 1D, 1E, 1F, 20.

```
out = "e" # length 0
for i in range(16):
    out += ", " + hex(i)[2:].upper() # length 1
for i in range(16):
    out += ", 0" + hex(i)[2:].upper() # length 2
for i in range(16,33):
    out += ", " + hex(i)[2:].upper() # length 2 cont.
print(out)
```

(d) (*Graded for correctness*) [2] Give an example of a finite set that is a language over Σ and over Γ, or explain why there is no such set. A complete and correct answer will use clear and precise notation (consistent with the textbook and class notes) and will include a description of why the given example is a language over Σ and over Γ and is finite, or an explanation why there is no such example.

> **Solution:** Consider the set $\{00\}$. This set is a language that only contains the string 00. This is a language over Σ and over Γ, since the string 00 is only made up of the symbol 0, which is in both alphabets. The size of this set is 1, so it is finite.

(e) (*Graded for correctness*) Give an example of an infinite set that is a language over Σ and not over Γ, or explain why there is no such set. A complete and correct answer will use

[2]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.

clear and precise notation (consistent with the textbook and class notes) and will include a description of why the given example is a language over Σ and not over Γ and is infinite, or an explanation why there is no such example.

Solution: There is no such set. Notice that Σ is a subset of Γ, so any string over Σ will also be a string over Γ. It would follow that any language over Σ would also be a language over Γ, thus, there can be no set that is a language over Σ and not over Γ.

2. **Regular expressions** (10 points):

(a) (*Graded for completeness*) Give three regular expressions that all describe the set of all strings over $\{a, b\}$ that have odd length. Ungraded bonus challenge: Make the expressions as different as possible!

Solution: Let $\Sigma = \{a, b\}$. We use the shorthand (from the textbook) that Σ can also abbreviate the regular expression $(a \cup b)$.

1. $\Sigma \circ (\Sigma\Sigma)^*$
   Informally, this regular expression represents odd length strings as ones that have some character to start, and then are built up of pairs of characters (because each odd length is $1 + 2n$ for some nonnegative integer n).

2. $((\Sigma\Sigma)^* \circ \Sigma) \circ \varepsilon$
   Informally, the first part of this regular expression is similar to the one before, but uses that odd numbers can be expressed as $2n+1$ for a nonnegative integer n (instead of $1+2n$). Adding the $\circ\varepsilon$ at the end of the regular expression doesn't change the language described because it has the effect of extending each string in the original language by the empty string, which doesn't change the string.

3. $\Sigma \cup (\Sigma\Sigma\Sigma \circ (\Sigma\Sigma)^*)$
   Informally, in this regular expression, we're representing each odd length as either 1 (Σ is a regular expression that describes the collection of length 1 strings) or $3 + 2n$ for some nonnegative integer n.

(b) (*Graded for completeness*) A friend tells you that each regular expression that has a Kleene star (*) describes an infinite language. Are they right? Either help them justify their claim or give a counterexample to disprove it and explain your counterexample.

Solution: They are incorrect. Consider the regular expression $\emptyset^*$. This describes the language $\{\varepsilon\}$, which is a finite set. Another counterexample is the regular expression ε^*, which also describes the language $\{\varepsilon\}$.

3. **Functions over languages** (15 points):

For each language L over the alphabet $\Sigma_1 = \{0,1\}$, we have the associated sets of strings

$$SUBSTRING(L) = \{w \in \Sigma_1^* \mid \text{there exist } x, y \in \Sigma_1^* \text{ such that } xwy \in L\}$$

and

$$EXTEND(L) = \{w \in \Sigma_1^* \mid w = uv \text{ for some strings } u \in L \text{ and } v \in \Sigma_1^*\}$$

(a) (*Graded for completeness*) Specify an example language A over Σ_1 such that $SUBSTRING(A) = EXTEND(A)$, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language A and a precise and clear description of the result of computing $SUBSTRING(A)$, $EXTEND(A)$ (using the given definitions) to justify this description and to justify the set equality, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: An example is $A = \Sigma_1^*$.

First, we claim that $SUBSTRING(A) = \Sigma_1^*$. By definition of $SUBSTRING$, every element of $SUBSTRING(A)$ will also be an element of Σ_1^*, which shows that $SUBSTRING(A) \subseteq \Sigma_1^*$. On the other hand, for any string $w \in \Sigma_1^*$, since there exist $x = y = \varepsilon \in \Sigma_1^*$ such that $xwy = w \in A$, we get $w \in SUBSTRING(A)$. Therefore $\Sigma_1^* \subseteq SUBSTRING(A)$. Then we can conclude that $SUBSTRING(A) = \Sigma_1^*$.

We also want to show that $EXTEND(A) = \Sigma_1^*$. By definition of $EXTEND$, every element of $EXTEND(A)$ will also be an element of Σ_1^*, which shows that $EXTEND(A) \subseteq \Sigma_1^*$. On the other hand, for any string $w \in \Sigma_1^*$, since there exist $u = \varepsilon \in A$ and $v = w \in \Sigma_1^*$ such that $w = uv$, we get $w \in EXTEND(A)$. Therefore $\Sigma_1^* \subseteq EXTEND(A)$. Then we can conclude that $EXTEND(A) = \Sigma_1^*$.

Therefore, for $A = \Sigma_1^*$, we have $SUBSTRING(A) = EXTEND(A) = \Sigma_1^*$.

(b) (*Graded for correctness*) Specify an example language B over Σ_1 such that

$$SUBSTRING(B) = \{\varepsilon\}$$

and

$$EXTEND(B) = \Sigma_1^*$$

or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language B and a precise and clear description of the result of computing $SUBSTRING(B)$, $EXTEND(B)$ (using the given definitions) to justify this description and to justify the set equality with $\{\varepsilon\}$ and Σ_1^* (respectively), or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: Consider the language $B = \{\varepsilon\}$.

We claim that $SUBSTRING(B) = \{\varepsilon\}$. By definition of $SUBSTRING$, in order for some string $w \in \Sigma_1^*$ to be in $SUBSTRING(B)$, there has to exist strings x and y over Σ_1 such that xwy (the concatenation of x, w, and y) is in the language B. We can prove that $\varepsilon \in SUBSTRING(B)$ since there exists $x = y = \varepsilon \in \Sigma_1^*$ such that $x\varepsilon y = \varepsilon \in B$. Therefore, $\{\varepsilon\} \subseteq SUBSTRING(B)$. On the other hand, since ε is the only string in the language B, any string in $SUBSTRING(B)$ will have length of at most 0. Therefore, $SUBSTRING(B) \subseteq \{\varepsilon\}$. Then we can conclude that $SUBSTRING(B) = \{\varepsilon\}$.

We claim that $EXTEND(B) = \Sigma_1^*$. By definition of $EXTEND$, for some string $w \in \Sigma_1^*$ to be in $EXTEND(B)$, w must be able to be broken down into two strings u and v such that $w = uv$ such that $u \in B$ and $v \in \Sigma_1^*$. For any string $w \in \Sigma_1^*$, there exist $u = \varepsilon \in B$ and $v = w \in \Sigma_1^*$ such that $uv = \varepsilon w = w$, which means that $w \in EXTEND(B)$. Therefore, $\Sigma_1^* \subseteq EXTEND(B)$. By definition of $EXTEND$, every element of $EXTEND(B)$ is also an element of Σ_1^*, i.e. $EXTEND(B) \subseteq \Sigma_1^*$. Thus we can conclude that $EXTEND(B) = \Sigma_1^*$.

(c) (*Graded for correctness*) Specify an example **infinite** language C over Σ_1 such that

$$SUBSTRING(C) \neq \Sigma_1^*$$

and

$$EXTEND(C) \neq \Sigma_1^*$$

, or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language C and a precise and clear description of the result of computing $SUBSTRING(B)$, $EXTEND(B)$ (using the given definitions) to justify this description and to justify the set nonequality claims, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: Consider $C = \{1^k \mid k \geq 1, k \text{ an integer}\}$.

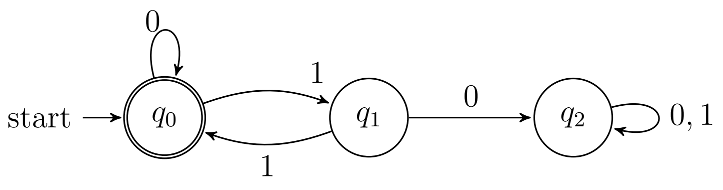
We can see that $SUBSTRING(C) = \{1^k \mid k \geq 0\}$, because by definition, $SUBSTRING(C)$ consists of all strings that are each a substring of some string in C. Substrings of strings with all 1s are also all 1s. We can also see that $EXTEND(C) = \{1^k w \mid k \geq 1 \text{ and } w \in \Sigma_1^*\}$, since by definition, $EXTEND(C)$ consists of all strings that are each the result of taking a string in C and concatenating it with a string over Σ_1. Since $\varepsilon \in C$ (setting $k = 0$ in 1^k), we can get any string in Σ_1 in $EXTEND(C)$.

We want to prove that $SUBSTRING(C) \neq \Sigma_1^*$. Consider string $00 \in \Sigma_1^*$. For any $x, y \in \Sigma_1^*$, $x00y \notin C$ since the strings in C do not contain the symbol 0. Therefore, $00 \notin SUBSTRING(C)$. We can conclude that $SUBSTRING(C) \neq \Sigma_1^*$.

Similarly, we want to prove that $EXTEND(C) \neq \Sigma_1^*$. Consider string $00 \in \Sigma_1^*$ again. For any string $u \in C$ and $v \in \Sigma_1^*$, $uv \neq 00$ since $u \in C$ has positive length and does not start with 0. Therefore, $00 \notin EXTEND(C)$. We can conclude that $EXTEND(C) \neq \Sigma_1^*$.

4. **Finite automata** (13 points):

Consider the finite automaton $(Q, \Sigma, \delta, q_0, F)$ whose state diagram is depicted below



where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, and $F = \{q_0\}$, and $\delta : Q \times \Sigma \to Q$ is specified by the look-up table

	0	1
q_0	q_0	q_1
q_1	q_2	q_0
q_2	q_2	q_2

(a) (*Graded for completeness*) A friend tries to summarize the transition function with the formula

$$\delta(q_i, x) = \begin{cases} q_0 & \text{when } i = 0 \text{ and } x = 0 \\ q_2 & \text{when } x < i \\ q_j & \text{when } j = (i+1) \mod 2 \text{ and } x = 1 \end{cases}$$

for $x \in \{0, 1\}$ and $i \in \{0, 1, 2\}$. Are they right? Either help them justify their claim or give a counterexample to disprove it and then fix their formula.

Solution: The proposed formula is incorrect. The counterexample is $\delta((q_2, 1))$, which is defined as q_2 following the second condition, but also defined as q_1 following the third condition. This makes the definition not a valid function.

One possible correct formula can be:

$$\delta((q_i, x)) = \begin{cases} q_0 & \text{when } x = i \\ q_2 & \text{when } x < i \\ q_1 & \text{when } x > i \end{cases}$$

(b) (*Graded for correctness*) Give a regular expression R so that $L(R)$ is the language recognized by this finite automaton. Justify your answer by referring to the definition of the semantics of regular expressions and computations of finite automata. Include an explanation for why each string in $L(R)$ is accepted by the finite automaton *and* for why each string not in $L(R)$ is rejected by the finite automaton.

Solution: $R = (0^*(11)^*)^*$

Justification: By the definition of regular expressions, the language described by R is

$$L(R) = \{w_1 ... w_k \mid k \geq 0 \text{ and each } w_i \in \{0^m 1^{2n} \mid m, n \geq 0\}\}$$

More intuitively, a string in $L(R)$ would have the pattern of alternating between arbitrary number of 0's and even number of 1's.

Take an arbitrary string in $L(R)$, the computation of the automaton on the string starts from q_0, and each time after we read a single 0 or an even number of 1's, we will end up back in q_0. Thus, the computation must end in q_0 for any string in $L(R)$, therefore each string in $L(R)$ is accepted by the finite automaton.

On the other hand, each string not in the language described by R will have a substring of consecutive odd number of 1's that is immediately after a 0 or the start of the string, and immediately before a 0 or the end of the string. After the computation on such a string starts, before reading the first 1 in the first consecutive odd number of 1's, we should be in q_0 as explained above. After reading the odd number of 1's, we will end up in state q_1. If the string ends after the 1's, the computation ends in q_1 which is not an accept state. If the string has a 0 followed by the 1's, the computation will end in q_2 which is also not an accept state. In either case, a string not in $L(R)$ will be rejected by the finite automaton.

In conclusion, $L(R)$ is the language recognized by the given finite automaton.

(c) (*Graded for correctness*) Keeping the same set of states $Q = \{q_0, q_1, q_2\}$, alphabet $\Sigma = \{0, 1\}$, same start state q_0, and same transition function δ, choose a new set of accepting states F_{new}

so that the new finite automaton that results accepts at least one string that the original one rejected **and** rejects at least one string that the original one accepted, or explain why there is no such choice of F_{new}. A complete solution will include either (1) a precise and clear description of your choice of F_{new} and a precise and clear the two example strings using relevant definitions to justify them, or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: Choose $F_{new} = \{q_1\}$. Before, string 1 was rejected because the computation of the finite automaton on it ended up in q_1 which was not an accept state, but now string 1 is accepted since q_1 becomes an accept state. Similarly, before, the empty string ε was accepted because q_0 was an accept state, but now ε is rejected since q_0 is no longer an accept state.