HW1CSE105W25: Sample Solutions

CSE105W25 Team

January 17, 2025

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.

Resources: To review the topics for this assignment, see the class material from Weeks 0 and 1 and Review Quiz 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. Chapter 0 exercises 0.1, 0.2, 0.3, 0.5, 0.6, 0.9. Chapter 1 exercises 1.19, 1.23.

You will submit this assignment via Gradescope (https://www.gradescope.com) in the assignment called "hw1CSE105W25".

Assigned questions

1. **Strings and languages: finding examples and edge cases** (12 points):

(a) (*Graded for completeness*) [1] Give five (different) example alphabets that are meaningful or useful to you in some way. Specify them formally, either with roster notation (which means listing all and only distict elements between { and } and separated by commas) or with another approach to precisely define all and only the elements of the alphabet.

Solution: Here are five different example alphabets:

- $\{C, S, E, 1, 0, 5\}$ is the collection of characters in the name of our class

[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.

- $\{0, 1\}$ is an alphabet we'll be seeing a lot of in this class
- $\{a, b\}$ is another alphabet we'll see quite often
- $\{\text{☺}, \text{☹}\}$ because an alphabet can contain any arbitrary symbol, not just numbers and letters
- The set of all positive integers less than 10 is an example of an alphabet we don't use the roster method to define

(b) (*Graded for completeness*) Give an example of a finite set over an alphabet and an infinite set over an alphabet. You get to choose the alphabet, and you get to choose the sets. The goal is to practice communicating your choices and definitions with clear and precise notation. One habit that will be useful (for this course, and beyond), is to think of your response for each question as a well-formed paragraph: include all the information that is relevant so that your solution is self-contained, and so that each sentence is grammatically constructed.

Solution: Let's define our alphabet as $\{0, 1\}$. An example of a finite set over this alphabet is the set $\{0, 1\}$. We know this set is finite because there are only two elements in it! An example of an infinite set would be $\{0, 1\}^*$, which contains all possible strings formed for our alphabet, or an infinite amount of strings.

(c) (*Graded for correctness*) [2] Define an alphabet Σ_1 and an alphabet Σ_2 and a language L_1 over Σ_1 that is also a language over Σ_2 and a language L_2 over Σ_2 that is **not** a language over Σ_1. 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 L_1 is a language over both Σ_1 and Σ_2 and a description of why the given example L_2 is a language over Σ_2 and not over Σ_1.

Solution: Let's define $\Sigma_1 = \{0, 1\}$ and $\Sigma_2 = \{1, 2\}$.
The language $L_1 = \{1\}$ is a language over both alphabets since both alphabets contain the character 1 and therefore we can form the string 1 using only characters that are among the characters in Σ_1 and also using only characters that are among the characters in Σ_2.
The language $L_2 = \{2\}$ is a language over Σ_2 because 2 is an element of Σ_2 so the length one string whose only character is 2 is a string over the alphabet. It is **not** a language over L_1 because we cannot form the string 2 out of any of the characters in L_1.

2. **Regular expressions** (20 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 those 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

[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.

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! See for example the regular expression ε^*. When you consider all possible strings formed using the empty string, you'll only ever be able to produce the empty string. That means that

$$L(\varepsilon^*) = \{\varepsilon\}^* = \{\varepsilon^i \mid i \geq 0\} = \{\varepsilon\}$$

This set has exactly one (distinct) elementis so it is not infinite. Hence, the regular expression ε^* does not describe an infinite language even though it contains a Kleene star symbol.

(c) (*Graded for correctness*) For this question, the alphabet is $\{a, b, c\}$. A friend is trying to design a regular expression that describes the set of all strings over this alphabet that end in c. Classify each of the following attempts as

- Correct. Explain why.
- Error Type 1: Incorrect, because (even though each string that is in the language described by the regular expression ends in c) there is a string that ends in c that is not in the language described by the regular expression. Give this example string and explain why it proves we're in this case.
- Error Type 2: Incorrect, because (even though each string that ends in c is in the language described by the regular expression), there is a string in the language described by the regular expression that does not end in c. Give this example string and explain why it proves we're in this case.
- Error Type 3: Incorrect, because there are two counterexample strings, one which is a string that ends in c that is not in the language described by the regular expression and one which is in the language described by the regular expression but does not end in c. Give both example strings and describe why each has the given property.

Worked example for reference: Consider the regular expression $(a \cup b \cup c)^*$. This regular expression has **Error Type 2** because it describes the set of all strings over $\{a, b, c\}$, so even though each string that ends in c is in this language, there is an example, say ab that is a string in the language described by the regular expression (because we consider the string formed as a result of the Kleene star operation which has 2 slots and where the first slot matches the a in $a \cup b \cup c$ and the second slot matches b in $a \cup b \cup c$) but does not end in c (it ends in b).

---

i. The regular expression is

$$(a \cup b)^* \circ c$$

Solution: This expression has **Error Type 1**. The strings in the language described by this regular expression will always end in c (this is guaranteed by having the $\circ c$ at the end of the regular expression). However, consider the string cc. This string ends in c but, if it were to be in $L(\ (a \cup b)^* \circ c\ )$, by definition of set-wise concatenation, we'd need $c \in L(a \cup b)^*$, which is not true because $L((a \cup b)^*)$ is the set of strings whose characters are a's or b's (and not c).

ii. The regular expression is

$$(a \circ b \circ c)^*$$

Solution: This regular expression has **Error Type 3**. Based on the definition of Kleene star, we know that ε is in the language described by this regular expression (corresponding to choosing $k = 0$ "slots" in the set-builder definition of Kleene star). Since ε does not end in c, it serves as a counterexaple string in the language described by the regular expression that does not end in c.
For the second counterexample, we notice that all strings in $L((a \circ b \circ c)^*)$ will be zero or more repetitions of abc. The same example from the previous part of the question cc is a string that ends in c that (again) won't be in the language described by the regular expression, this time because cc is not the result of repeating abc some number of times.

iii. The regular expression is

$$a^*c \ \cup \ b^*c \ \cup \ c^*c$$

Solution: Again, we have **Error Type 1**. The strings in the language described by this regular expression will always end in c: each of the subexpressions in the union have c concatenated on the end. Any string in $L(a^*c \ \cup \ b^*c \ \cup \ c^*c)$ will be described by at least one of a^*c or b^*c or c^*c, and in each of these cases, it's guaranteed that the string will end with the character c.
To find the counterexample string that ends in c but is not in the language described by $a^*c \ \cup \ b^*c \ \cup \ c^*c$, let's use the example abc. We notice that any string in $L(a^*c \ \cup \ b^*c \ \cup \ c^*c)$ has at most two distinct characters since there is no internal regular expression that combines more than two characters. However, abc uses all three possible characters from the alphabet, so it is not in this set.

3. **Functions over languages** (18 points):

For each language L over an alphabet Σ, we have the associated sets of strings (also over Σ)

$$L^* = \{w_1 \cdots w_k \mid k \geq 0 \text{ and each } w_i \in L\}$$

and

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

and

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

Also, recall the set operations union and intersection: for any sets X and Y

$$X \cup Y = \{w \mid w \in X \text{ or } w \in Y\}$$

$$X \cap Y = \{w \mid w \in X \text{ and } w \in Y\}$$

(a) (*Graded for completeness*) Specify an example language A over $\{0, 1\}$ 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:
Before we give the example, we're going to prove a few helper claims (lemmas).
Lemma 1: For any alphabet Σ, if a language L over Σ contains ε, then $EXTEND(L)$ is the set of all strings over Σ.
Proof of lemma 1: Let Σ be an alphabet and assume L is a language over Σ that has ε as an element. We need to prove that $EXTEND(L) = \Sigma^*$. Since $EXTEND(L)$ is a set of strings over Σ, it's enough to prove that $\Sigma^* \subseteq EXTEND(L)$. Let w be an arbitrary string over Σ. We need to show that $w \in EXTEND(L)$. To do that we need to write w as uv for some $u \in L$ and $v \in \Sigma^*$. Because of our assumption that the empty string is in L, we can let $u = \varepsilon$ and $v = w$. Then $w = \varepsilon w = uv$, as required to prove that $w \in EXTEND(L)$.
Lemma 2: For any alphabet Σ, for any language L over Σ, $L \subseteq SUBSTRING(L)$.
Proof of Lemma 2: Let Σ be an alphabet and assume L is a language over Σ. Let $w \in L$ be arbitrary. We need to prove that $w \in SUBSTRING(L)$. By definition of the $SUBSTRING$ function, this means finding $x, y \in \Sigma^*$ such that $xwy \in L$. Consider $x = y = \varepsilon$ (which is a string over any alphabet, and therefore over our alphabet Σ). Then we have $xwy = \varepsilon w \varepsilon = w$, which is an element of L by assumption. Thus, $w \in SUBSTRING(L)$.
Now we're ready for our example.
Let A be the set of all strings over $\Sigma_1 = \{0, 1\}$. That is

$$A = \{0, 1\}^*$$

Applying Lemma 1 to A, since $\varepsilon \in A$, we can see that

$$EXTEND(A) = \{0, 1\}^*$$

Applying Lemma 2 to A, we can see that

$$\{0,1\}^* \subseteq SUBSTRING(A)$$

Since $SUBSTRING(A)$ is a language, it is a subset of the set of all strings. With this, we proved subset inclusion in both directions, so

$$\{0,1\}^* = SUBSTRING(A)$$

Which now means we can conclude

$$SUBSTRING(A) = EXTEND(A)$$

(b) (*Graded for correctness*) Specify an example language B over $\{0,1\}$ such that

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

and

$$SUBSTRING(B) \cup EXTEND(B) = \{0,1\}^*$$

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 $\{0,1\}^*$ (respectively), or (2) a sufficiently general and correct argument why there is no such example, referring back to the relevant definitions.

Solution: Let B be the language that only contains the empty string. That is

$$B = \{\varepsilon\}$$

To prove that $SUBSTRING(B) = \{\varepsilon\}$ we need to show that the empty string is an element of $SUBSTRING(B)$, and that there are no other elements of this. Let $w = \varepsilon$, and $x = y = \varepsilon$. Then $xwy = w = \varepsilon \in B$, so $\varepsilon \in SUBSTRING(B)$. To show that there are no other strings in $SUBSTRING(B)$, consider an arbitrary string $w \neq \varepsilon$, so $|w| > 0$. Notice that no matter what x and y are, $|xwy| \geq |w| > 0$, so $xwy \neq \varepsilon$, so $xwy \notin B$, so $xwy \notin SUBSTRING(B)$. Thus

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

Using Lemma 1 from the solution to part (a), since ε is an element of B,

$$EXTEND(B) = \{0,1\}^*$$

By definition of intersection,

$$\begin{aligned} SUBSTRING(B) \cap EXTEND(B) &= \{\varepsilon\} \cap \{0,1\}^* \\ &= \{x | x = \varepsilon \text{ and } x \in \{0,1\}^*\} = \{\varepsilon\} \end{aligned}$$

and by definition of union

$$SUBSTRING(B) \cup EXTEND(B) = \{\varepsilon\} \cup \{0,1\}^* \\ = \{x | x = \varepsilon \text{ or } x \in \{0,1\}^*\} = \{0,1\}^*$$

(c) (*Graded for correctness*) Specify an example **infinite** language C over $\{0,1\}$ such that

$$SUBSTRING(C) \neq \{0,1\}^*$$

and

$$SUBSTRING(C) = C^*$$

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 $SUBSTRING(C)$, C^* (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: Let C be the language of strings formed using zero or more 1s (and no other characters). That is

$$C = \{1^k \mid k \geq 0\}$$

This is indeed an infinite language because it has elements of each possible length. However, C is not equal to the set of all strings because it doesn't include any strings that have the character 0. We will show that

$$SUBSTRING(C) = C = C^*$$

Lemma 2 from the solution to part (a) gives that $C \subseteq SUBSTRING(C)$. To get the first set equality we want, we need to show that $SUBSTRING(C) \subseteq C$. Let $w \in SUBSTRING(C)$ be arbitrary. Then there exists x, y such that $xwy \in C$. By the definition of C, $xwy = 1^k$ for some $k \geq 0$. In particular, there are nonnegative integers g, i, j so that $x = 1^g$ and $w = 1^i$ and $y = 1^j$ so that $g + i + j = k$. Bu that means that w is a string of all 1s, which is the exact condition to be in C, so $w \in C$. This proves $SUBSTRING(C) \subseteq C$.
We have shown subset inclusion in both direction, so

$$SUBSTRING(C) = C$$

Next we'll show that $C = C^*$. For the first subset inclusion, let $w \in C$ be arbitrary. Based on the definition of Kleene star operation, $C^* = \{w_1 ... w_k \mid k \geq 0$ and each $w_i \in C\}$. Then $w \in C^*$ by letting $k = 1$ and $w_1 = w \in C$. This proves $C \subseteq C^*$. For the second subset inclusion, let $w \in C^*$ be arbitrary. Then, $w = w_1 \dots w_k$ where $k \geq 0$, and $w_i \in C$. Notice that each of these $w_i = 1^{k_i}$. That is, they contain k_i amount of 1's. So the whole string w is just a whole bunch of 1's, more specifically $w = 1^K$ where $K = \sum_{i=1}^{k} k_i$. Since this is still some amount of 1's, $w \in C$. This proves $C^* \subseteq C$.

We have now shown subset inclusion in both direction, so

$$C = C^*$$

Which now gives us

$$SUBSTRING(C) = C = C^*$$

(d) (*Graded for correctness*) Specify an example **finite** language D over $\{0, 1\}$ such that

$$EXTEND(D) \neq \{0, 1\}^*$$

and

$$EXTEND(D) = D^*$$

or explain why there is no such example. A complete solution will include either (1) a precise and clear description of your example language D and a precise and clear description of the result of computing $EXTEND(D)$, D^* (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: There is no such example.
To prove this, we'll use the proof by contradiction strategy. Assume, towards a contradiction, that there is a finite language D such that $EXTEND(D) \neq \{0, 1\}^*$ and $EXTEND(D) = D^*$.
From Lemma 1 in the solution to part (a), we know that $\varepsilon \notin D$ (because if it were, then $EXTEND(D) = \{0, 1\}^*$). Now let's consider D^*. By definition of Kleene star (choosing $k = 0$ in the set-builder definition), the empty string is an element of D^*. By assumption that $EXTEND(D) = D^*$, $\varepsilon \in EXTEND(D)$. By definition of the $EXTEND$ function, this means that there is a string $u \in D$ and $v \in \Sigma^*$ such that $\varepsilon = uv$. The only strings that concatenate to the empty string are the empty string itself, i.e. $u = v = \varepsilon$ (we cannot "shorten" strings by concatenating them with other strings). Thus, it must be that $\varepsilon \in D$, a contradiction with our previous observation that $\varepsilon \notin D$. Since the existence of a set with these properties led to a contradiction, there cannot be such a set.