Formal Reasoning 2022
Solutions Test Block 2: Languages and Automata
(10/11/22)

Languages

1. Does the equality
$$(LL)^* = L^*L^*$$
hold for every language L?

   (a) Yes, and both languages are always equal to L^*.

   (b) Yes, but they are not always equal to L^*.

   (c) No, if you take $L = \{a, b\}$ then the word ab is only in one of the two languages $(LL)^*$ and L^*L^*.

   (d) No, if you take $L = \{a, b\}$ then the word a is only in one of the two languages $(LL)^*$ and L^*L^*.

(d) is correct

Answer (d) is correct.

If $L = \{a, b\}$ then $a \in L^*$ and $\lambda \in L^*$. Hence $a \in L^*L^*$. However, the language $(LL)^*$ only has words of even length, as $LL = \{aa, ab, ba, bb\}$ and the length of the word a is odd.

2.
$$\mathcal{L}((ab)^*(ba)^*) \cap \mathcal{L}((ba)^*(ab)^*) = \ldots$$

(a) is correct

   (a) $\mathcal{L}((ab)^* \cup (ba)^*)$

   (b) $\mathcal{L}((ab)^* \cap (ba)^*)$

   (c) $\mathcal{L}(\lambda)$

   (d) None of the above.

Answer (a) is correct.

First let us write down some words in $\mathcal{L}((ab)^*(ba)^*)$:

$$\lambda, \quad ab, ba, \quad abab, abba, baba, \quad ababab, ababba, abbaba, bababa, \quad \ldots$$

And let us also write down some words in $\mathcal{L}((ba)^*(ab)^*)$:

$$\lambda, \quad ba, ab, \quad baba, baab, abab, \quad bababa, babaab, baabab, ababab, \quad \ldots$$

So if we take the intersection of these two languages we get words like

$$\lambda, \quad ab, ba, \quad abab, baba, \quad ababab, bababa, \quad \ldots$$

Now let us see what we can say about the given options:

(a) $\mathcal{L}((ab)^* \cup (ba)^*)$: Maybe a bit surprising due to the change of $\cap$ on the level of languages to $\cup$ on the level of regular expressions, this is indeed correct. The regular expression $(ab)^*$ produces the words $\lambda, ab, abab, ababab, \ldots$ and the regular expression $(ba)^*$ produces the words $\lambda, ba, baba, bababa, \ldots$ and taking the union of these languages gives the same set of words that we described above.

(b) $\mathcal{L}((ab)^* \cap (ba)^*)$: This is not even a language as the $\cap$ can not be used in regular expressions.

(c) $\mathcal{L}(\lambda)$: This language does not include the word ab, so it isn't right.

(d) None of the above: This is obviously not correct.

3. Is the language

$$\{a^n b^n c^m \mid n, m \in \mathbb{N}\}$$

context-free? Explain your answer.

A context-free language is a language that is generated by a context-free grammar. This language is generated by the following context-free grammar, so it is indeed context-free.

$$\begin{aligned} S &\to AC \\ A &\to aAb \mid \lambda \\ C &\to cC \mid \lambda \end{aligned}$$

The productions work as follows (it is not needed to provide this part of the explanation):

- The start symbol S is always being replaced by the non-terminals A and B.
- The non-terminal A is used to build $a^n b^n$.
- The non-terminal C is used to build c^m.

Some other correct solutions:

$$\begin{aligned} S &\to A \mid Sc \\ A &\to aAb \mid \lambda \end{aligned}$$

$$\begin{aligned} S &\to A \\ A &\to BC \\ B &\to aBb \mid \lambda \\ C &\to cC \mid \lambda \end{aligned}$$

4. Consider the context-free grammar G:

$$\begin{aligned} S &\to aA \\ A &\to aA \mid bA \mid \lambda \end{aligned}$$

and the property

$$P(w) := \big(w \text{ starts with } a\big)$$

Is this property an invariant for this grammar?

(a) Yes, because the property holds for all words in $\mathcal{L}(G)$.

(b) Yes, because once a word in a production starts with a this will not change anymore.

(c) is correct

(c) No, the property does not hold for S.

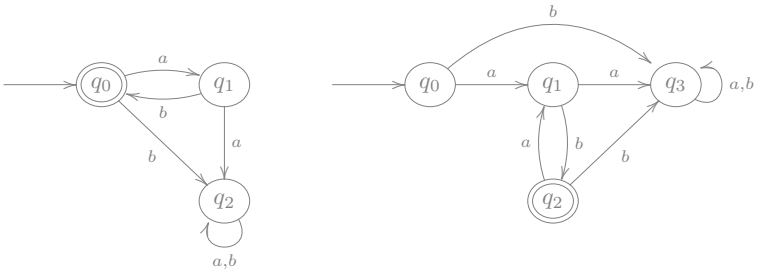
(d) No, $A \to aA$ is a production step, and the property holds for aA but not for A.

Answer (c) is correct.

If $P(w)$ would be an invariant then $P(S)$ should hold. But it does not, as the word S does not start with the letter a.

Automata

5. Consider the two deterministic finite automata M_1 and M_2 given by the state diagrams:



What is the relation between the two languages $\mathcal{L}(M_1)$ and $\mathcal{L}(M_2)$?

(a) $\mathcal{L}(M_1) = \mathcal{L}(M_2)$.

(b) $\mathcal{L}(M_1) \subset \mathcal{L}(M_2)$.

(c) is correct

(c) $\mathcal{L}(M_2) \subset \mathcal{L}(M_1)$.

(d) None of the above.

Answer (c) is correct.

Let us start by writing down some words in $\mathcal{L}(M_1)$:

$$\lambda, \quad ab, \quad abab, \quad ababab, \quad \ldots$$

And now let us write down some words in $\mathcal{L}(M_2)$:

$$ab, \quad abab, \quad ababab, \quad \ldots$$

So the languages are not the same and the only difference is that $\lambda \in \mathcal{L}(M_1)$ but not $\lambda \in \mathcal{L}(M_2)$.

Now let us see what we can say about the given options:

(a) $\mathcal{L}(M_1) = \mathcal{L}(M_2)$. This can't be correct as $\lambda \in \mathcal{L}(M_1)$ but not $\lambda \in \mathcal{L}(M_2)$.

(b) $\mathcal{L}(M_1) \subset \mathcal{L}(M_2)$. For basically the same reason this can't be correct: $\mathcal{L}(M_1) \subset \mathcal{L}(M_2)$ does not hold as $\lambda \in \mathcal{L}(M_1)$ but not $\lambda \in \mathcal{L}(M_2)$.

(c) $\mathcal{L}(M_2) \subset \mathcal{L}(M_1)$. This is correct: all words accepted by M_2 are also accepted by M_1, but, in addition, M_1 also accepts λ, so $\mathcal{L}(M_2)$ is indeed a strict subset of $\mathcal{L}(M_1)$.

(d) None of the above. This is obviously not correct.

Note that if q_0 in M_2 would have been a final state then $\mathcal{L}(M_1) = \mathcal{L}(M_2)$ would have been true!

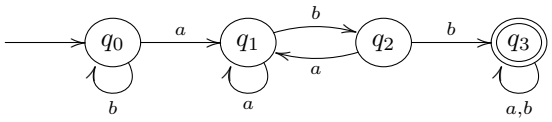
6. Give a right-linear context-free grammar with at most three non-terminals for the language:

$$\{w \in \{a, b\}^* \mid w \text{ does } \textit{not} \text{ contain } abb\}$$

Hint: Make a deterministic finite automaton for this language first.

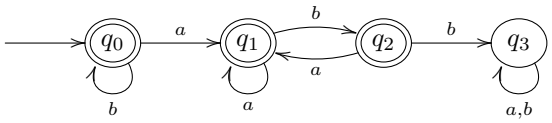
We start by making a DFA for the language

$$\{w \in \{a,b\}^* \mid w \textit{ does } \text{contain } abb\}$$



Then, by swapping final and non-final states, we transform this into a DFA that accepts the requested language

$$\{w \in \{a,b\}^* \mid w \text{ does } \textit{not} \text{ contain } abb\}$$



And by identifying state q_0 with non-terminal S, q_1 with A, q_2 with B and q_3 with C, we get the right-linear context-free grammar:

$$\begin{aligned} S &\to aA \mid bS \mid \lambda \\ A &\to aA \mid bB \mid \lambda \\ B &\to aA \mid bC \mid \lambda \\ C &\to aC \mid bC \end{aligned}$$

This is a correct grammar for the language, but it has too many non-terminals. However, as C doesn't add anything (as q_3 is a sink), we can simplify the grammar to:

$$\begin{aligned} S &\to aA \mid bS \mid \lambda \\ A &\to aA \mid bB \mid \lambda \\ B &\to aA \mid \lambda \end{aligned}$$

Instead of removing C we can also substitute B in A and end up with three non-terminals:

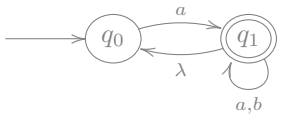
$$\begin{aligned} S &\to aA \mid bS \mid \lambda \\ A &\to aA \mid baA \mid bbC \mid b \mid \lambda \\ C &\to aC \mid bC \end{aligned}$$

If we do both reductions we get:

$$\begin{aligned} S &\to aA \mid bS \mid \lambda \\ A &\to aA \mid baA \mid b \mid \lambda \end{aligned}$$

All of these equivalent grammars are right-linear as there is no rule where there is a non-terminal on the right hand side that isn't at the far right.

7. Consider the non-deterministic finite automaton $\langle \Sigma, Q, q_0, F, \delta \rangle$ with the state diagram:



What is the value of $\delta(q_1, a)$?

(a) $\delta(q_1, a) = q_1$

(b) is correct (b) $\delta(q_1, a) = \{q_1\}$

(c) $\delta(q_1, a) = \{q_0, q_1\}$

(d) None of the above.

Answer (b) is correct.

Let us review the given options:

(a) $\delta(q_1, a) = q_1$: This can't be correct as the result of δ should be a set.

(b) $\delta(q_1, a) = \{q_1\}$: This is correct; the only state you can go from q_1 with an a is q_1 itself.

(c) $\delta(q_1, a) = \{q_0, q_1\}$: This is not correct as there is no a-arrow from q_1 going to q_0.

(d) None of the above: This is obviously not correct.

8. Let be given a deterministic finite automaton M with exactly three states, for which it holds that $aaaaabbbbb = a^5b^5 \in \mathcal{L}(M)$. Is it then always the case that also $a^nb^5 \in \mathcal{L}(M)$ for some $n > 5$?

(a) No, this is never the case.

(b) No, this holds for *some* automata with this property, but not for all.

(c) Yes, because when processing the first five a's, the automaton has to go through a loop (because there are not enough different states), which can be repeated.

(c) is correct

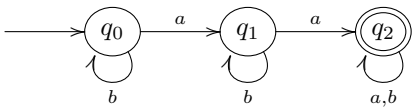
(d) Yes, because any machine with this property will accept all words in $\mathcal{L}(a^*b^*)$.

Answer (c) is correct.

If there are exactly three states q_0, q_1, and q_2, then it is possible to deal with the string aa without loops: start in the initial state q_0, read the first a and go to a second state q_1, read the second a and go to a third state q_2.

However, for the third a one needs to reuse one of the previously used states q_0, q_1, or q_2, as there simply is no fourth unused state anymore. But this means indeed that there must be a loop somewhere in reading the five a's, which implies that this loop can be traversed several times to accept a word a^nb^5 for some $n > 5$. This is known as the Pumping Lemma.

Note that there really exist DFAs with exactly three states that accept the word *aaaaabbbbb*, for instance



However, this automaton does not accept *abb*, so it does not accept $\mathcal{L}(a^*b^*)$, hence the last option is indeed incorrect.