Formal Reasoning 2024
Solutions Test Block 2: Languages and Automata

(07/11/24)

Languages

1. Does the equation $(L_1L_2)^R = L_1^R L_2^R$ hold for all languages L_1 and L_2?

   (a) Yes, in both cases there is a combination of reversing and concatenating the languages.

   (b) Yes, when constructing these languages, it does not matter in which order you concatenate two words and reverse them.

   (c) is correct

   (c) No, but $(L_1L_2)^R = L_2^R L_1^R$ *does* hold, as $(uv)^R = v^R u^R$ for all words u and v.

   (d) No, $(L_1L_2)^R$ is not determined by L_1^R and L_2^R.

   Answer (c) is correct.

   Let us prove that the sets $(L_1L_2)^R$ and $L_2^R L_1^R$ are equal:

$$\begin{array}{rcl} (L_1L_2)^R & = & \text{by definition of concatenation} \\ & & \big\{(uv)^R \mid u \in L_1 \text{ and } v \in L_2\big\} \\ & = & \text{by definition of reverse} \\ & & \big\{v^R u^R \mid u \in L_1 \text{ and } v \in L_2\big\} \\ & = & \text{by definition of concatenation} \\ & & \big\{v^R \mid v \in L_2\big\}\big\{u^R \mid u \in L_1\big\} \\ & = & \text{by definition of complement} \\ & & L_2^R L_1^R \end{array}$$

   So $(L_1L_2)^R$ and $L_2^R L_1^R$ are indeed equal, but that doesn't prove that $(L_1L_2)^R$ and $L_1^R L_2^R$ are indeed not equal. However, the latter can easily be shown using $L_1 = \{a\}$ and $L_2 = \{b\}$. Obviously, $L_1^R = \{a\}$ and $L_2^R = \{b\}$. And now we have that

$$ba = (ab)^R \in (L_1L_2)^R$$

   but

$$ba \notin L_1^R L_2^R = \{ab\}$$

2. Which language is equal to $\overline{\mathcal{L}(a^*b^*)}$ over the alphabet $\{a, b\}$?

   (a) is correct

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

   (b) $\mathcal{L}(b^*a^*)$

   (c) $\mathcal{L}\big(\overline{a^*b^*}\big)$

   (d) none of the above

   Answer (a) is correct.

   Recall that $\mathcal{L}(a^*b^*)$ is the language of words consisting of zero or more a's followed by zero or more b's. So all a's must be in front of all b's.

   And hence the complement of this language is about words for which it is not the case that all a's are in front of all b's. Now this implies that there must be at least one b that is in front of an a. And this implies that there must be at least one sequence of ba inside the word. And it doesn't matter what happens before or after this ba. So this property is described by the regular expression $(a \cup b)^*ba(a \cup b)^*$ and hence the language $\mathcal{L}\big((a \cup b)^*ba(a \cup b)^*\big)$ is equal to $\overline{\mathcal{L}(a^*b^*)}$.

Note that the other options are indeed not equal to $\overline{\mathcal{L}(a^*b^*)}$:

- $\mathcal{L}(b^*a^*)$: this language doesn't contain the word aba which is an element of $\overline{\mathcal{L}(a^*b^*)}$.
- $\mathcal{L}\big(\overline{a^*b^*}\big)$: this language is not well defined, as we can't take the complement of the regular expression a^*b^*.

3. Consider the context-free grammar G_3:

$$S \to x \mid y \mid z \mid S + S \mid SS$$

This defines a language $\mathcal{L}(G_3)$ with alphabet $\Sigma = \{x, y, z, +\}$.

How many parse trees does the word

$$xy + z$$

have?

(a) 1

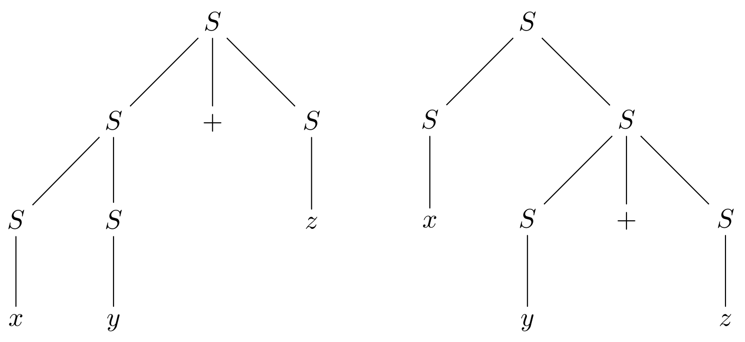
(b) is correct (b) 2

(c) 8

(d) 16

Answer (b) is correct.

Note that the question is about the number of different parse trees and not about the number of different productions! So the only choice we have is whether we first do the $+$ and then the concatenation of x and y, or first the concatenation of x and $y + z$ and then the $+$. So we get:



4. Consider the context-free grammar G_4:

$$S \to aS \mid Sb \mid \lambda$$

and the property

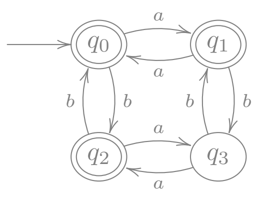
$$P(w) := \big[w \text{ does not contain } ba\big]$$

Is this property an invariant for G_4? Explain your answer.

No, this property is not an invariant. Take the word bS. Then certainly $P(bS)$ holds as it clearly doesn't contain ba. However, $bS \to baS$ by applying the rule $S \to aS$. But $P(baS)$ does not hold, as baS does contain ba.

Automata

5. Consider the deterministic finite automaton M_5:



We write $|w|_a$ for the number of a's in w, and likewise $|w|_b$ for the number of b's.

Which language is accepted by M_5?

(a) $\{w \in \{a,b\}^* \mid |w|_a + |w|_b \text{ is even}\}$
(b) $\{w \in \{a,b\}^* \mid |w|_a + |w|_b \text{ is odd}\}$
(c) is correct (c) $\{w \in \{a,b\}^* \mid |w|_a \times |w|_b \text{ is even}\}$
(d) $\{w \in \{a,b\}^* \mid |w|_a \times |w|_b \text{ is odd}\}$

Answer (c) is correct.

Note that q_0 represents the state where $|w|_a$ and $|w|_b$ are both even, q_1 represents the state where $|w|_a$ is odd and $|w|_b$ is even, q_2 represents the state where $|w|_a$ is even and $|w|_b$ is odd, and q_3 represents the state where $|w|_a$ and $|w|_b$ are both odd. Only q_3 is not a final state, so the situation that both $|w|_a$ and $|w|_b$ are odd is not accepted. Which means that at least one of $|w|_a$ and $|w|_b$ values must be even. Which is the same as stating that $|w|_a \times |w|_b$ must be even.

Let us also explain why the other options are wrong:

- $\{w \in \{a,b\}^* \mid |w|_a + |w|_b \text{ is even}\}$: this language doesn't contain a, but a is accepted in final state q_1.
- $\{w \in \{a,b\}^* \mid |w|_a + |w|_b \text{ is odd}\}$: this language doesn't contain λ, but λ is accepted in final state q_0.
- $\{w \in \{a,b\}^* \mid |w|_a \times |w|_b \text{ is odd}\}$: this language contains ab, but ab ends up in the non-final state q_3.

6. Consider the context-free grammar G_6:

$$S \to aS \mid Sb \mid \lambda$$

Is there a deterministic finite automaton that accepts $\mathcal{L}(G_6)$?

(a) is correct

(a) Yes, because this language is regular.
(b) Yes, all context-free languages are accepted by a deterministic finite automaton.
(c) No, the grammar is not right linear.
(d) No, a deterministic finite automaton for this language needs more than one state, and there is only one non-terminal in the grammar.

Answer (a) is correct.

Note that $\mathcal{L}(G_6) = \mathcal{L}(a^*b^*)$. So it is indeed regular.

Let us also explain why the other options are wrong:

- Yes, all context-free languages are accepted by a deterministic finite automaton. This is not true as the language $\{a^n b^n \mid n \in \mathbb{N}\}$ is context free because it is generated by the grammar

$$S \rightarrow aSb \mid \lambda$$

  but it is not regular, so there doesn't exist a DFA that accepts it.

- No, the grammar is not right linear. It is true that this grammar is not right linear, but there does exist a grammar that generates the same language which is right linear:

$$\begin{array}{lcl} S & \rightarrow & aS \mid B \\ B & \rightarrow & bB \mid \lambda \end{array}$$

  Obviously, since the answer is 'yes', the simple fact that this option states that the answer is 'no' already implies that it can't be a correct answer.

- No, a deterministic finite automaton for this language needs more than one state, and there is only one non-terminal in the grammar. It is true that a DFA for this language needs more than one state and it is true that this grammar has only one non-terminal, but there is no rule that the number of states should be equal to the number of non-terminals.

  And again, since the answer is 'yes', the simple fact that this option states that the answer is 'no' already implies that it can't be a correct answer.

7. Give a deterministic finite automaton $M_7 = \langle \Sigma, Q, q_0, F, \delta \rangle$ such that

$$\mathcal{L}(M_7) = \mathcal{L}(a^*ba^*)$$

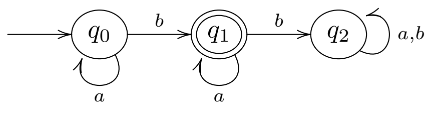
The definition of M_7 can be written in Ans as

```
M7 = <Sigma,Q,q0,F,delta>
```

followed by the definitions of all components of the tuple.

Write the states as `q0`, `q1`, etc., and give the function δ as a list of equations of the form `delta(q0,a) = ...`

This is a DFA that accepts $\mathcal{L}(a^*ba^*)$.



As a tuple it is defined as: `M7 = <Sigma,Q,q0,F,delta>` where

```
Sigma = {a,b}
Q = {q0,q1,q2}
F = {q1}
delta(q0,a) = q0      delta(q0,b) = q1
delta(q1,a) = q1      delta(q1,b) = q2
delta(q2,a) = q2      delta(q2,b) = q2
```

8. What is the type of the transition function δ of a non-deterministic finite automaton? (This type describes its input/output behavior.)

   (a) $\delta : Q \times \Sigma \to Q$

   (b) $\delta : Q \times \Sigma \to \mathcal{P}(Q)$

   (c) $\delta : Q \times (\Sigma \cup \{\lambda\}) \to Q$

   (d) is correct (d) $\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$

   Answer (d) is correct.

   As the transition function δ must also handle λ-transitions, it is clear that the solution must be $\delta : Q \times (\Sigma \cup \{\lambda\}) \to Q$ or $\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$. However, the first of these two options states that the result is a single state, but, as in an NFA there may be more than one outgoing arrow with the same symbol, the result should not be a single state, but a set of states. And to be precise, a subset of the set of states Q, which is by definition an element of the power set of Q, so the correct type is $\delta : Q \times (\Sigma \cup \{\lambda\}) \to \mathcal{P}(Q)$.