Formal Reasoning 2023
Solutions Test Block 2: Languages and Automata
(09/11/23)

Languages

1. What is λ *not* used for in this course?

(a) is correct

   (a) a symbol from the alphabet Σ
   (b) a word from Σ^*
   (c) a regular expression
   (d) a label for a transition in a non-deterministic finite automaton

   Answer (a) is correct.

   The λ in this course occurs in several situations:

   - To indicate the empty word, which implies that $\lambda \in \Sigma^*$.
   - To indicate a regular expression r for which $\mathcal{L}(r) = \lambda$.
   - To indicate so-called λ-transitions in NFAs which allow transitions from one state to another without reading any symbols.

   However, it is never used as a symbol in the alphabet.

2. For which regular expression r_2 do we have that

$$\mathcal{L}(r_2) = \{w \in \{a, b\}^* \mid w \text{ contains an even number of } b\text{'s}\}$$

(a) is correct

   (a) $r_2 = a^*(ba^*ba^*)^*$
   (b) $r_2 = a^*(ba^*b)^*a^*$
   (c) $r_2 = (a \cup bb)^*$
   (d) all three of these possibilities

   Answer (a) is correct.

   Each word $w \in \mathcal{L}(r_2)$ can be split up into several parts:

   - The word w starts with zero or more a's until the first b (if there is any), which is realized by the expression a^*.
   - If there is no b at all, then all symbols in w are just a's.
   - After this initial series of a's, we can divide w in (possibly zero) blocks that start with a b, followed by any number of a's, followed by a second b to make sure that each block has an even number of b's, followed again by any number of a's, where all a's until the next b should be taken into account. Such a block is represented by the expression ba^*ba^*.
   - Repetition of these blocks makes sure that all words in this language can be created by the expression $a^*(ba^*ba^*)^*$.

   In addition, it is easy to see that the other candidates do not create all words with an even number of b's:

- The expression $(a \cup bb)^*$ prevents creating the word bab, as there is no way to create an a in between the two b's.
- The expression $a^*(ba^*b)a^*$ prevents creating the word $bbabb$, as there is no way to create an a in between two blocks with each two b's.
- And if these two expressions aren't correct, the answer 'all three of these possibilities' is obviously wrong.

3. Consider the following context-free grammar G_3 for a language with alphabet $\{\mathsf{a}, \mathsf{e}, \mathsf{h}, \mathsf{l}, \mathsf{n}, \mathsf{o}, \mathsf{r}, \mathsf{s}, \mathsf{t}, \mathsf{v}, \mathsf{y}, \mathsf{J}, \mathsf{M}, \sqcup\}$:

$$S \to N \sqcup V \mid N \sqcup V \sqcup N$$
$$N \to \mathsf{John} \mid \mathsf{Mary}$$
$$V \to \mathsf{loves} \mid \mathsf{hates}$$

How many different productions are there in G_3 for the word:

$$\mathsf{John} \sqcup \mathsf{loves} \sqcup \mathsf{Mary}$$

(a) 1
(b) 3
(c) is correct (c) 6
(d) none of the above

Answer (c) is correct.

Any production of the word $\mathsf{John} \sqcup \mathsf{loves} \sqcup \mathsf{Mary}$ starts with the step $S \to N \sqcup V \sqcup N$ and after this the two N's and the single V have to be replaced by two nouns and a verb. The order in which this is done is completely free. And this means that we have three options to choose the first non-terminal that is being replaced, then two options for the second non-terminal that is being replaced, and finally, there is no choice anymore for the last non-terminal to be replaced. So the number of different paths is $3 \times 2 \times 1 = 6$.

4. Consider the context-free grammar G_4:

$$S \to AB$$
$$A \to aA \mid \lambda$$
$$B \to bB \mid \lambda$$

Someone wants to show that $ba \notin \mathcal{L}(G_4)$ and considers the property:

$$P_4(w) := \big[\text{in } w \text{ there is no } b \text{ before an } a\big]$$

Is this a suitable invariant for this? Explain your answer.

No, this is not a suitable invariant for showing that $ba \notin \mathcal{L}(G_4)$. In fact, it isn't even an invariant! Take $v = Ba$. Then obviously $P(Ba)$ does hold. However, we also have the production $Ba \to bBa$, but $P(bBa)$ clearly doesn't hold. So the proposed property is not an invariant.

Automata

5. We define the language L_5 by:

$$L_5 := \mathcal{L}(a^*bb^*) = \{a^n b^m \mid n \in \mathbb{N}, n \geq 0, m \in \mathbb{N}, m > 0\}$$

Consider the following statement:

> Each deterministic finite automaton M with $\mathcal{L}(M) = L_5$ has to have at least three states, because there has to be a final state which will be different from the initial state, and there also has to be a non-final state that is different from the initial state.

Is this correct?

(a) Yes, this is correct. Each M with $\mathcal{L}(M) = L_5$ will have a sink state (which for example will be reached after processing ba), which is a non-final state that is different from the initial state.

(b) Yes, this is correct. However, a machine M with $\mathcal{L}(M) = L_5$ does not need to have a sink state. There can be more than one non-final state different from the initial state.

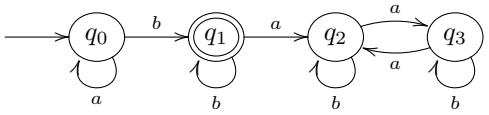
(b) is correct

(c) No, this is not correct. There is a DFA with less than three states for the language L_5.

(d) No, this is not correct. The minimal number of states needed for L_5 is indeed three, but the argumentation is not correct.

Answer (b) is correct.

As $\lambda \notin L_5$ it follows that the initial state is not a final state. And as $b \in L_5$ we know that any DFA accepting this language must have a final state. So each DFA for this language must have at least two states. However, note also that $ba \notin L_5$, so after reading b and getting in the end state, we need an a transition to a non-final state. This non-final state cannot be the initial state, as that would allow looping and the word bab would be accepted. So it must be a new non-final state. And hence we have at least three states. And although the simplest way would be to make this new non-final state a sink, it doesn't have to be a sink. Instead of a sink, we can add more non-final states and create loops within a set of non-final states that have no way to reach final states anymore. See for instance this DFA:



6. Consider the following statement:

> If a deterministic finite automaton M with alphabet $\{a, b\}$ has exactly three states, and there is a word $w \in \mathcal{L}(M)$ with $|w| = 3$, then the language $\mathcal{L}(M)$ will contain infinitely many different words.

Which of the following holds?

(a) is correct

(a) This is correct. When processing the word w, at least one state will be encountered twice, and by going around that loop, one can find arbitrary large words in $\mathcal{L}(M)$.

(b) This is correct. A language of a DFA always contains infinitely many different words.

(c) This is not correct. A word w with three symbols is too short to necessarily encounter the same state twice. There is a machine with these properties that only accepts finitely many different words.

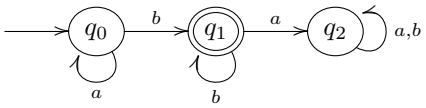
(d) This is not correct. If the automaton M has no final states, the language of M will be empty, and will certainly not contain infinitely many different words.

Answer (a) is correct.

For reading a word w with three symbols, we need exactly three steps in the automaton. And as we also have a starting state, it means that for reading such a word we need four states. Now because we only have three states, the pigeonhole principle tells us that there is at least one state that is visited at least twice. So there is a loop involved in the path of accepting w. Now this means that we can follow this loop arbitrarily often and still end up in an accepting state, which means that there are indeed infinitely many different words in the language.

Note that the other 'This is correct.' option makes no sense as a DFA does not need to have final states and in that situation the language accepted by the automaton is the empty language, which does not have infinitely many words, but zero words.

7. Consider the following deterministic finite automaton M_7:



Give the right linear context-free grammar that corresponds to this automaton. Do not simplify the grammar in any way, just give the result of the conversion from automata to grammars as it was described in the lectures and course notes.

Let us associate state q_0 with the non-terminal S, q_1 with A, and q_2 with B. Then the grammar corresponding to the automaton becomes

$$\begin{aligned} S &\to aS \mid bA \\ A &\to aB \mid bA \mid \lambda \\ B &\to aB \mid bB \end{aligned}$$

Note that this is indeed a right linear context-free grammar.

8. Consider the following quintuple:

$$M_8 = \langle \{a\}, \{q_0, q_1\}, q_0, \varnothing, \delta_8 \rangle$$

with

$$\delta_8(q_0, a) = \varnothing$$
$$\delta_8(q_0, \lambda) = \varnothing$$
$$\delta_8(q_1, a) = \varnothing$$
$$\delta_8(q_1, \lambda) = \{q_1\}$$

Is this a correct non-deterministic finite automaton?

(a) is correct

(a) Yes, it is.

(b) No, the machine does not have final states.

(c) No, the machine has no transitions for the symbol a.

(d) No, the state q_1 is not reachable from the initial state q_0.

Answer (a) is correct.

This is what the automaton looks like in a diagram:

The definition of an NFA allows an empty set of final states. It just means that the corresponding language will be empty, so it is not a very interesting automaton, but it is allowed.

In addition, one of the differences between an NFA and a DFA is that an NFA doesn't need to have exactly one outgoing transition for each symbol of the alphabet in each state. In particular, it is allowed that none of the states has an outgoing a transition.

And finally, as there are no restrictions on the transitions in an NFA, it is allowed to have non-reachable states. In fact, that is also allowed for DFAs. The fact is that these non-reachable states play no role in accepting words as they are not adding anything to the language.