Prof. Javier Esparza
Philipp Czerner

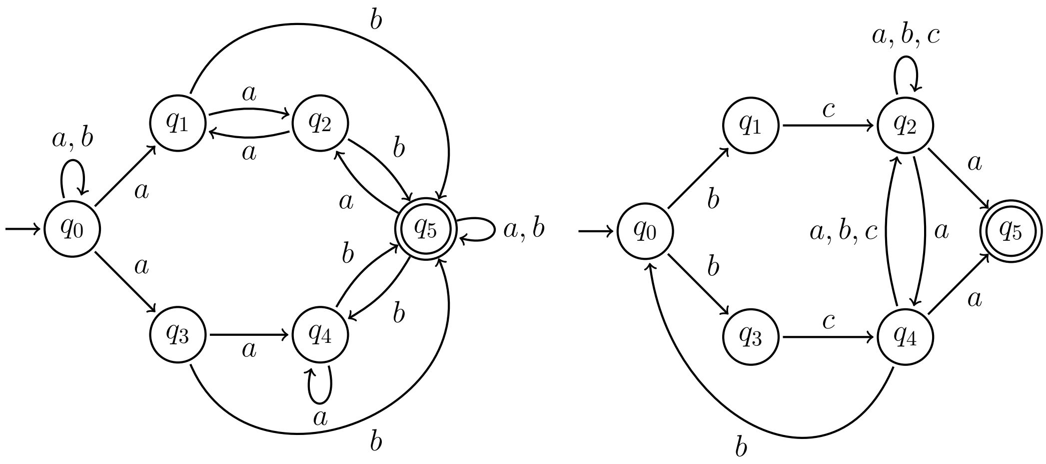
Technical University of Munich
Chair for Foundations of Software Reliability

Automata and Formal Languages

Winter Term 2023/24 – Exercise Sheet 4

Exercise 4.1.

Let A and B be respectively the following NFAs:



(a) Compute the coarsest stable refinements (CSR) of A and B.

(b) Construct the quotients of A and B with respect to their CSRs.

(c) Show that

$$L(A) = \{w \in \{a,b\}^* : w \text{ contains an occurrence of the subword } ab\}$$
$$L(B) = \{w \in \{a,b,c\}^* : w \text{ starts with } bc \text{ and ends with } a\}$$

(d) Are the automata obtained in (b) minimal?

Exercise 4.2.

Let $\Sigma = \{a,b\}$. For any $n \in \mathbb{N}$, let $L_n := \{ww^R : w \in \Sigma^n\}$, where w^R is the reverse of w, e.g. $(abc)^R = cba$. In Exercise 2.3, we have shown that every NFA (and hence also every DFA) recognizing L_n must have at least 2^n states. We refine this bound here for DFAs.

(a) Construct A_2, the minimal DFA for L_2.

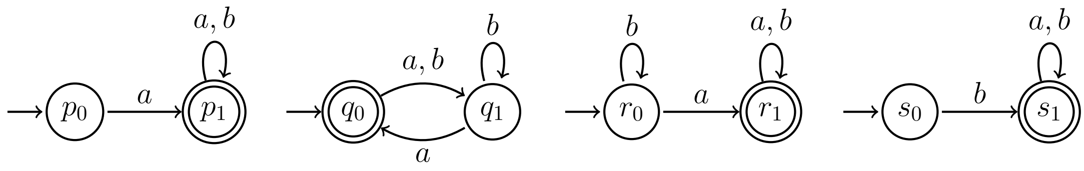
(b) What are the residuals of L_2? Assign them to the states of the DFA you gave for (a).

(c) Give a construction for a DFA that accepts L_n.

(d) How many states does the minimal DFA for L_n contain, for $n \geq 2$?

Exercise 4.3.

Consider the following DFAs A, B, C and D:



(a) Use pairings to decide *algorithmically* whether $L(A) \cap L(B) \subseteq L(C)$.

(b) Use pairings to decide *algorithmically* whether $L(D) \subseteq L(A) \cap L(B)$.

Exercise 4.4.

In this exercise we consider four (AI-generated) proof attempts. For each proof attempt, determine whether the proof is correct and mark the errors it contains (if any).

Claim 1. *Let X denote a language. Then X^* is regular.*

Proof attempt. To prove that the Kleene closure (star) of a language X, denoted as X^*, is regular, we can use a straightforward approach. If X is any language, and we take its Kleene closure X^*, then X^* is regular. This proof is based on the properties of regular languages and closure properties.

- **Basis:** The basis for the proof lies in the definition of regular languages. The empty set $\emptyset$ and the set containing the empty string $\{\varepsilon\}$ are both regular. Any single character from the alphabet, say $\{a\}$, is also regular.
- **Inductive Step:** Now, we can use closure properties of regular languages to show that if X is regular, then X^* is regular. The closure properties state that regular languages are closed under certain operations, including union, concatenation, and Kleene closure.
  - **Union:** If L_1 and L_2 are regular languages, then their union $L_1 \cup L_2$ is also regular.
  - **Concatenation:** If L_1 and L_2 are regular languages, then their concatenation $L_1 L_2$ is also regular.
  - **Kleene Closure:** If L is a regular language, then L^* is also regular.

Since X is any language, it can be empty or non-empty, finite or infinite, regular or non-regular. It doesn't matter because X^* is defined as the set of all possible strings that can be formed by concatenating zero or more strings from X.

So, by the closure property of the Kleene closure operation, X^* is regular.

In summary, regardless of whether X is regular or not, the Kleene closure X^* is always regular because regular languages are closed under the Kleene closure operation. □

Claim 2. *Let X denote a language. Then X^* is not regular in general. (In other words: There exists an X s.t. X^* is not regular.)*

Proof attempt. To disprove that the Kleene closure (star) of a language X, denoted as X^*, is not necessarily regular, we can use a counterexample. There exist languages for which their Kleene closure is not regular. One classic example is the language $L = \{a^n b^n : n \geq 0\}$, which is well-known to be non-regular. Let's show that L^* is also non-regular:

- **Base Case:** L contains the strings $\varepsilon, ab, aabb, aaabbb, \ldots$.
- **Inductive Step:** Now, let's consider L^*, the Kleene closure of L. It consists of all possible strings that can be formed by concatenating zero or more strings from L. L^* includes strings like $\varepsilon, ab, aaabbb, aabbabbb$, etc., which can have any number of 'a's and 'b's but must have an equal number of 'a's and 'b's. This is because L itself is the set of strings with an equal number of 'a's and 'b's.

L^* contains not only the strings in L but also strings that do not have an equal number of 'a's and 'b's, and this set of strings is not regular. For example, the string $a^p b^q$, where $p \neq q$, is not in L and is not regular. If you concatenate such strings in L^*, you will obtain non-regular strings.

Therefore, L^* is not regular because it contains non-regular strings. Since L^* is a subset of X^* (where X is any language), this demonstrates that X^* is not necessarily regular. □

Claim 3. *For a word w with length divisible by 3, let $f(w)$ denote the word you obtain by deleting the middle third. For example $f(aba) = aa$ and $f(abbabaaab) = abbaab$. Let L denote a regular language. Then $f(L)$ is regular.*

Proof attempt. To prove that the language $f(L)$ is regular for a given regular language L, we can construct a finite automaton that recognizes it. We can use the fact that regular languages are closed under certain operations, including the operation of deleting a contiguous portion of a string, as long as it is done consistently.

Let's define a finite automaton to recognize $f(L)$. This automaton will have states that represent the possible "states" of the string as we process it.

The automaton will work as follows:

(a) Start in the initial state.

(b) For each symbol in the input string, the automaton will transition to a new state. Initially, the automaton stores the characters in the first third of the string. As it reads more symbols, it continues to build the first third.

(c) When it reaches the last third of the input string, it transitions to an accepting state, indicating that the entire first two-thirds of the string have been processed.

(d) Any remaining characters in the last third of the input string are ignored, as they are not part of $f(w)$.

The automaton will have a state for each possible combination of characters in the first third of the string. Since L is regular, there is a finite number of such combinations, and the automaton can have a finite number of states.

Since we can construct a finite automaton to recognize $f(L)$, we can conclude that $f(L)$ is regular. This is based on the closure property of regular languages under the operation of deleting a contiguous portion of a string while processing it. □

Claim 4. *For a word w with length divisible by 3, let $f(w)$ denote the word you obtain by deleting the middle third. For example $f(aba) = aa$ and $f(abbabaaab) = abbaab$. Let L denote a regular language. Then $f(L)$ is not regular in general.*

Proof attempt. To disprove that the language $f(L)$ is necessarily regular for a regular language L, we can provide a counterexample. We will show that there exists a regular language L for which $f(L)$ is not regular.

Consider the regular language $L = \{a^{3n} : n \geq 0\}$, which consists of strings composed of a multiple of 3 'a' symbols. This language is regular because it can be recognized by a simple finite automaton that checks for strings of the form a^{3n}.

Now, let's look at the language $f(L)$. This language contains words obtained by removing the middle third of strings in L. The middle third contains 'n' 'a' symbols. Therefore, $f(L)$ consists of strings of the form a^{2n} for $n \geq 0$, which means it includes all strings with an even number of 'a' symbols.

However, the set of all strings with an even number of 'a' symbols is not regular. This can be shown using the pumping lemma for regular languages. If you assume that $f(L)$ is regular, then you would expect to be able to apply the pumping lemma to it, but it fails for such strings. You can't divide these strings into substrings to satisfy the pumping lemma's conditions because, after pumping, the number of 'a's in the string will still be even, which violates the requirement for the language to be regular.

Therefore, we've demonstrated that there exists a regular language L for which $f(L)$ is not regular, which disproves the claim that $f(L)$ is necessarily regular for all regular languages L. $\square$