Discrete Mathematics

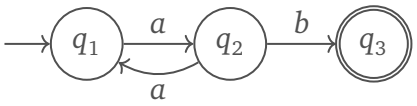
Exercises 10 – Solutions with Commentary

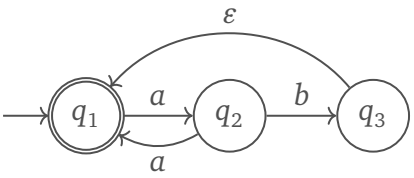
Marcelo Fiore Ohad Kammar Dima Szamozvancev

17. On regular languages

1. Why can't the automaton $Star(M)$ used in step (iv) of the proof of part (a) of Kleene's Theorem be constructed by simply taking M, making its start state the only accepting state and adding new ε-transitions back from each old accepting state to its start state?

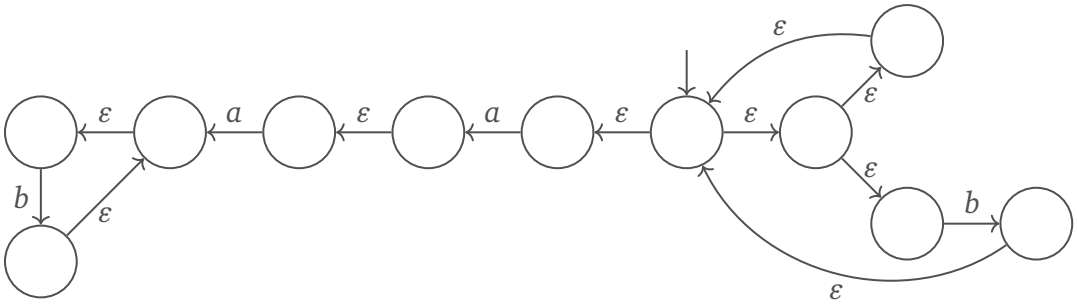
   The problem is that we would be meddling with the internals of the automaton in unexpected ways by turning a (potentially) non-accepting start state into an accepting one. If M has transitions looping back to the start state, we may be able to accept partially recognised strings prematurely. For example, the automaton on the left below recognises the language $L(a(aa)^*b)$, but naively adding an ε-transition from q_3 to q_1 and making q_1 accepting would result in a machine that accepts not only the expected $L((a(aa)^*b)^*)$, but also $(aa)^*$. By adding a new start state we ensure that the automaton "commits" to performing a full repetition by explicitly transitioning into the start state of M.

2. Construct an NFA$^\varepsilon$ M satisfying $L(M) = L((\epsilon|b)^*aab^*)$ using Kleene's construction.

   Using the entirely algorithmic construction we get the following automaton:

   ♪ Of course, there is a lot of redundancy, especially the large number of ε-transitions that could be safely collapsed. The regex is not in its simplest form either, since $(\epsilon|b)^*$ is equivalent to b^*. However, these are concerns of implementation efficiency, and Kleene's theorem is a result that the two formalisms are "in principle" equivalent. There are examples of languages that could be concisely expressed as regexes but the size of DFAs recognising the language is exponential in the length of the redex (such as the strings which have a specific symbol in the $\{k^{\text{th}}\}$ last position); conversely, some languages are simple to

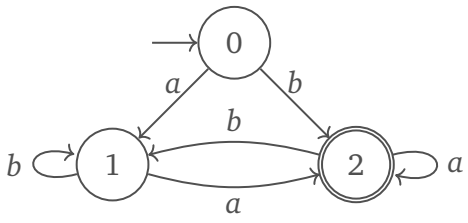
> recognise by a DFA, but the corresponding regexes are enormous (divisibility by 7 requires a DFA of 7 states, and converts to a regex of 10791 characters).

3. Show that any finite set of strings is a regular language.

> Let $L = \{u_1, u_2, \ldots, u_k\}$ be a finite set of strings. We construct the regular expression $u_1|u_2|\ldots|u_k$ which is clearly matched by all strings in L. Since such a regex exists, by Kleene's Theorem we conclude that L is regular.

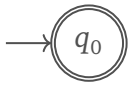
4. Use the construction given in the proof of part (b) of Kleene's Theorem to find a regular expression for the DFA M whose state set is $\{0, 1, 2\}$, whose start state is 0, whose only accepting state is 2, whose alphabet of input symbols is $\{a, b\}$, and whose next-state function is given by the following table.

δ	a	b
0	1	2
1	2	1
2	2	1

> The DFA specified in the question is as follows:
>
> 
>
>
> We apply Kleene's regex construction by first removing state 1, then state 2 – other orderings are possible. The recursive cases are not expanded further when the required redex is easily constructed by observation.
>
> $$
> \begin{aligned}
> r_{0,2}^{\{0,2\}} &= ba^* \\
> r_{0,1}^{\{0,2\}} &= r_{0,1}^{\{0\}} \mid r_{0,2}^{\{0\}}\left(r_{2,2}^{\{0\}}\right)^* r_{2,1}^{\{0\}} = a|ba^*b \\
> r_{1,1}^{\{0,2\}} &= r_{1,1}^{\{0\}} \mid r_{1,2}^{\{0\}}\left(r_{2,2}^{\{0\}}\right)^* r_{2,1}^{\{0\}} = b|aa^*b \\
> r_{1,2}^{\{0,2\}} &= r_{1,2}^{\{0\}} \mid r_{1,2}^{\{0\}}\left(r_{2,2}^{\{0\}}\right)^* r_{2,2}^{\{0\}} = a|aa^*a \\
> r_{0,2}^{\{0,1,2\}} &= r_{0,2}^{\{0,2\}} \mid r_{0,1}^{\{0,2\}}\left(r_{1,1}^{\{0,2\}}\right)^* r_{1,2}^{\{0,2\}} = ba^*|(a|ba^*b)(b|aa^*b)^*(a|aa^*a)
> \end{aligned}
> $$

5. If $M = (Q, \Sigma, \Delta, s, F)$ is an NFA, let $Not(M)$ be the NFA $(Q, \Sigma, \Delta, s, Q \setminus F)$ obtained from M by interchanging the role of accepting and nonaccepting states. Give an example of an alphabet Σ and an NFA M with set of input symbols Σ such that $\{u \in \Sigma^* \mid u \notin L(M)\}$ is *not* the same as $L(Not(M))$.

> A simple minimal example is the following automaton M with alphabet $\Sigma = \{a\}$:



We have that $L(M) = \{\,\varepsilon\,\}$, but interchanging the accepting and nonaccepting states would turn q_0 into a nonaccepting state so the language recognised is $\emptyset$. However, $\emptyset \neq \{\,a\,\}^* \setminus \{\,\varepsilon\,\}$.

6. Let $r = (a|b)^*ab(a|b)^*$. Find a regular expression that is equivalent to the complement for r over the alphabet $\{\,a, b\,\}$ with the property $L(\sim r) = \{\,u \in \{\,a, b\,\}^* \mid u \notin L(r)\,\}$.

   The language matching r consists of all strings that contain ab as a substring. Thus, the complement of $L(r)$ is the setof strings that do not contain ab as a substring, which is only possible if there are no occurrences of b after the first occurrence of a. The corresponding regular expression is thus simply b^*a^*.

7. Given DFAs $M_i = (Q_i, \Sigma, \delta_i, s_i, F_i)$ for $i = 1, 2$, let $And(M_1, M_2)$ be the DFA

$$(Q_1 \times Q_2, \Sigma, \delta, (s_1, s_2), F_1 \times F_2)$$

   where $\delta : (Q_1 \times Q_2) \times \Sigma \to (Q_1 \times Q_2)$ is given by

$$\delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a))$$

   for all $q_1 \in Q_1, q_2 \in Q_2$ and $a \in \Sigma$. Show that $L(And(M_1, M_2)) = L(M_1) \cap L(M_2)$.

   We prove the following lemma: for all strings $u \in \Sigma^*$ and states $q_1, q_2, q_1', q_2' \in Q$,

$$(q_1, q_2) \overset{u}{\Rightarrow} (q_1', q_2') \text{ in } And(M_1, M_2) \quad \iff \quad q_1 \overset{u}{\Rightarrow} q_1' \text{ in } M_1 \ \wedge\ q_2 \overset{u}{\Rightarrow} q_2' \text{ in } M_2$$

   This will directly imply the required result for q_1, q_2 the start states s_1, s_2 and q_1', q_2' a pair of accepting states in F_1, F_2. The lemma will be proved by induction on the length of the string u.

   Base case: $|u| = 0$, so $u = \varepsilon$. If $(q_1, q_2) \overset{\varepsilon}{\Rightarrow} (q_1', q_2')$ in $And(M_1, M_2)$, we must have that $q_1 = q_1'$ and $q_2 = q_2'$ because the automaton is deterministic and has no ε-transitions. But this is exactly the case when $q_1 \overset{\varepsilon}{\Rightarrow} q_1$ and $q_2 \overset{\varepsilon}{\Rightarrow} q_2$.

   Inductive case: $|u| = k + 1$, so $u = va$ with $|v| = k$ and $a \in \Sigma$. Assume the Ⓘ for the string v. If there is a path $(p_1, q_1) \overset{va}{\Rightarrow} (p_3, q_3)$ in $And(M_1, M_2)$, there must be two states p_2, q_2 such that $(p_1, q_1) \overset{v}{\Rightarrow} (p_2, q_2)$ and $(p_2, q_2) \overset{a}{\to} (p_3, q_3)$. By the Ⓘ, we have that $p_1 \overset{v}{\Rightarrow} p_2$ and $q_1 \overset{v}{\Rightarrow} q_2$, and $\delta((p_2, q_2), a) = (p_3, q_3)$ by definition holds if $\delta_1(p_2, a) = p_3$ and $\delta_2(q_2, a) = q_3$. Combining $p_1 \overset{v}{\Rightarrow} p_2$ with $p_2 \overset{a}{\to} p_3$ and $q_1 \overset{v}{\Rightarrow} q_2$ with $q_2 \overset{a}{\to} q_3$, we have a path $p_1 \overset{va}{\Rightarrow} p_3$ and $q_1 \overset{va}{\Rightarrow} q_3$ in M_1 and M_2 respectively, as required.

18. On the Pumping Lemma

1. Briefly summarise the proof of the Pumping Lemma in your own words.

♪ Bookwork exercise, mainly intended to get you to read and understood the proof in order to be able to reproduce it if needed. The core points to remember are:

- The pumping lemma property is a *necessary condition*: regularity of L implies PLP, but not the other way around.
- The statement of the PLP should be read as a kind of dialogue: what are the objects and constraints you are given (a regular language L, a word of an appropriate length, and the number of repetitions of the central string), and what are things you have control over (the minimum length of the string, and the decomposition).
- The negation of PLP (used in the contrapositive) is the same dialogue but flipped around. An important consequence that is easy to overlook in informal proofs of non-regularity is that the "opponent" chooses the decomposition: they will try their very best to "catch you out" so the proof must not make any assumptions on how the string is split (other than the constraints stated, which are there precisely to stop the opponent choosing a decomposition for which the expected reasoning doesn't work).

2. Consider the language $L \triangleq \{ c^m a^n b^n \mid m \geq 1 \wedge n \geq 0 \} \cup \{ a^m b^n \mid m, n \geq 0 \}$. The notes show that L has the pumping lemma property. Show that there is no DFA M which accepts L.

   Hint: argue by contradiction. If there were such an M, consider the DFA M' with the same states as M, with alphabet of input symbols just consisting of a and b, with transitions all those of M which are labelled by a or b, with start state $\delta_M(s_M, c)$ where s_M is the start state of M, and with the same accepting states as M. Show that the language accepted by M' has to be $\{ a^n b^n \mid n \geq 0 \}$ and deduce that no such M can exist.

   We follow the hint and take M and M' as given. We show that $L(M') = \{ a^n b^n \mid n \geq 0 \}$.

   ($\subseteq$) If $w \in \{ a, b \}^*$ is accepted by M', then $cw \in L(M)$ since the start state of M' is reached with a single c-transition from the start state of M. By definition of M, w must be of the form $a^n b^n$ for some $n \in \mathbb{N}$.

   ($\supseteq$) If $w = a^n b^n$, we have that $ca^n b^n \in L(M)$, and by the definition of M', we have that $a^n b^n \in L(M)$.

   Thus, from the assumption that the DFA M exists, we constructed a DFA M' such that $L(M') = \{ a^n b^n \mid n \geq 0 \}$; however, we know from the contrapositive statement of the Pumping Lemma that $\{ a^n b^n \mid n \geq 0 \}$ is not regular, so our assumption that M exists was wrong. Consequently, the language L is not regular.

   ♪ The proof presented here is done using a technique called *reduction*: we reduce the question of determining the membership of a string u in $\{ a^n b^n \mid n \geq 0 \}$ to the question of determining the membership of cu in L, so if the latter is answerable using a DFA, then so is the former (which leads to a contradiction). The central property of the mapping $u \mapsto cu$ is that $u \in \{ a^n b^n \mid n \geq 0 \}$ if and only if $cu \in L$. Reduction proofs will be discussed in more detail in the IB *Computation Theory* and *Complexity Theory* courses.