Programming Languages and Computation

Week 4: Regular Languages

In the problems this week you will need to make use of the formal definition of a finite state automaton, given at https://uob-coms20007.github.io/reference/regular/automata.html#finite-state-automaton.

* 1. Draw the diagram of the following automata:

(a) $(\{e, o\}, \{0, 1\}, \{(e, 0, o), (e, 1, o), (o, 0, e), (o, 1, e)\}, e, \{e\})$

(b) $(Q, \{0, 1\}, \Delta, q_0, Q)$ where $Q = \{q_0, q_1, q_2, q_3\}$ and Δ is:

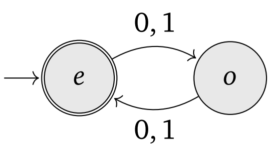
$$\{(q_0, 0, q_0), (q_0, 1, q_1), (q_1, 0, q_2), (q_1, 1, q_3), (q_2, 0, q_1), (q_3, 0, q_3)\}$$

(c) $(Q, \Sigma, \Delta, q_0, Q)$ where:

- $Q = \{1, 2, 3, 4, 5\}$
- $\Sigma = \{a, b\}$
- $\Delta = \{(i, a, i + 1) \mid 1 \leq i \leq 5\} \cup \{(j, b, j) \mid j \text{ is even}\}$
- $q_0 = 1$
- $F = \{1, 3, 5\}$

Solution

(a) Diagram:



(b) Diagram:

```
procedure cl(X):
  X' := ∅
  while X ≠ X' do
    X' := X
    for each q ∈ X'
      if (q, ϵ, q') ∈ Δ
        X := X ∪ {q'}
  return X
```

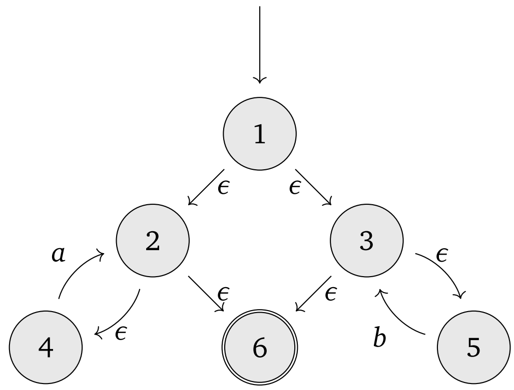
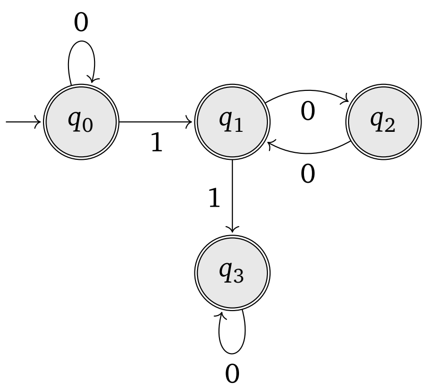
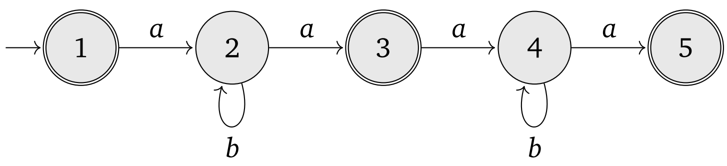


Figure 1: ϵ-closure of $X \subseteq Q$ wrt transitions Δ, and the automaton from Week 3 Q4(b)



(c) Diagram:



* 2. Suppose M is a finite automaton with states Q. The ϵ-closure of a set of states $X \subseteq Q$ in M, written $\mathsf{cl}(X)$, is the set of all states that can be reached from any state in X using only ϵ-transitions. It can be computed using the algorithm in Figure 1.

(a) Construct a table with two columns. Each row of the table should contain a state of the automaton from Figure 1 in the first column and the ϵ-closure of that state in the second column.

(b) Let the automaton in Figure 1 be $(Q, \{a, b\}, \Delta, 1, \{6\})$. Draw the diagram for the automaton $(Q', \{a, b\}, \Delta', \mathsf{cl}(1), Q')$ where $Q' = \{\mathsf{cl}(1), \mathsf{cl}(2), \mathsf{cl}(3)\}$ and:

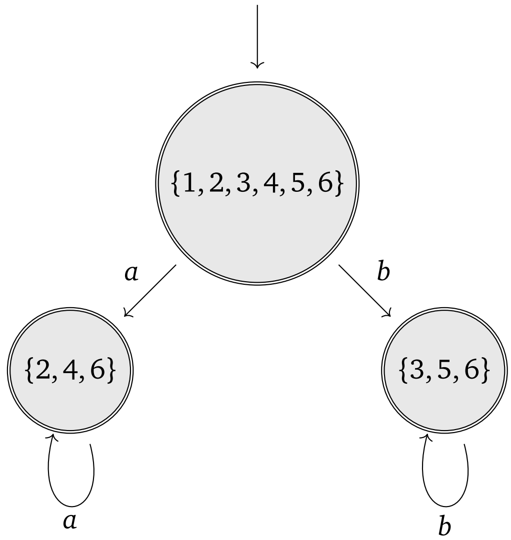
$$\Delta' = \{(X, \ell, \mathsf{cl}(j)) \mid \ell \in \{a, b\} \text{ and there is some } i \in X \text{ such that } (i, \ell, j) \in \Delta\}$$

Solution

(a) Table:

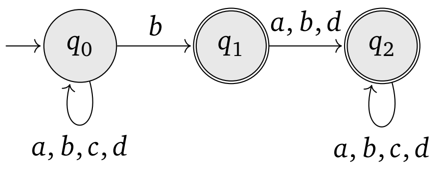
State	Closure
1	$\{1,2,3,4,5,6\}$
2	$\{2,4,6\}$
3	$\{3,5,6\}$
4	$\{4\}$
5	$\{5\}$
6	$\{6\}$

(b) Automaton:



** 3. Let $\mathsf{rev}(w)$ be the reverse of the word w, e.g. $\mathsf{rev}(abccd) = dccba$ and $\mathsf{rev}(\epsilon) = \epsilon$.

Let P be the following automaton:



(a) Construct another automaton that recognises $\{\mathsf{rev}(w) \mid w \in L(P)\}$. Try not to think about what this language actually looks like, instead try to think how you could "reverse" the diagram, because, in the next part, you will not have a specific language.

(b) Suppose $M = (Q, \Sigma, \Delta, q_0, F)$ is a finite automaton. By filling out (i)–(iii), complete the following definition of a finite automaton N in such a way that $L(N) = \{\mathsf{rev}(w) \mid w \in L(P)\}$.

Let s be a new state not in Q. Then finite automaton N is $(Q', \Sigma, \Delta', q_0', F')$ where:

- $Q' = Q \cup \{s\}$
- Δ' = (i)

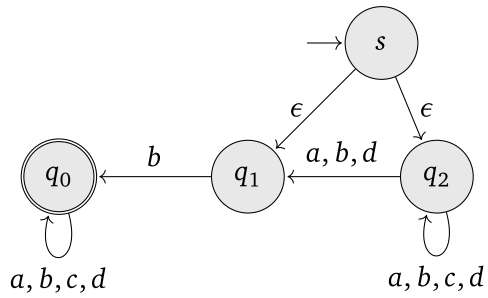
- $q_0' =$ (ii)
- $F' =$ (ii)

(c) Argue that if A is a regular language, then so is $\{\mathsf{rev}(w) \mid w \in A\}$.

Solution

To recognise the reverse of L, we just need to run the automaton "in reverse". Each of the reversed words can be obtained by starting from an accepting state of M and working backwards to the initial state. So we will construct an automaton M' which, on each word $\mathsf{rev}(w)$, guesses which state q of F that M ends in after an accepting run over w and then works backwards through the transitions of M until it reaches the initial state.

(a) Diagram:



To understand why it works, recall that accepting a word w corresponds to drawing a path from the initial state to an accepting state that is labelled by w. The reverse of every path from the initial state to an accepting state in the original automaton can be drawn in this automaton starting at s and ending in q_0. Conversely, every accepting run of this automaton is clearly in 1-1 correspondence with the reversal of an accepting run of the original.

(b) In the general case, we need to add a new initial state s, add ϵ-transitions from s to each of the original accepting states, reverse all the transitions of the original and set the original initial state as the only accepting state.

(i) $\{(s, \epsilon, q) \mid q \in F\} \cup \{(q, \ell, p) \mid p \in Q,\ \ell \in \Sigma,\ q \in Q,\ (p, \ell, q) \in \Delta\}$

(ii) s

(iii) $\{q_0\}$

(c) Suppose A is regular. Then it is recognised by some finite automaton M, i.e. such that $L(M) = A$. We can construct a new automaton N that recognises the reverse of the language of M as in the previous part. Therefore, $\{\mathsf{rev}(w) \mid w \in L(M)\}$ is regular, but this is just $\{\mathsf{rev}(w) \mid w \in A\}$ so this language is regular, as was required.

** 4. Let $\mathsf{tail}(w)$ be the tail of the word w, i.e:

$$\mathsf{tail}(\epsilon) = \epsilon$$
$$\mathsf{tail}(a \cdot w) = w$$

By following a similar approach to parts (b) and (c) of the previous question, argue that if S is regular, then so is $\{\mathsf{tail}(w) \mid w \in S\}$.

Solution

Suppose S is regular, so it is recognised by some finite state automaton M of shape $(Q, \Sigma, q_0, \delta, F)$.

We will build a finite state automaton M' that behaves as follows on each word $\mathsf{tail}(w)$ for $w \in S$: (i) use an ϵ-transition to guess which state M arrives at after reading in the first letter of w and then (ii) proceed as M would have done from there in order to accept. In this way, M' behaves like M except it does an epsilon transition at first whilst M would consume the first letter.

Let s be a state not in Q and then define $M' = (Q \cup \{s\}, \Sigma, \Delta', s, F)$ where Δ' is:

$$\Delta \cup \{(s, \epsilon, q) \mid M \text{ can reach } q \text{ from } q_0 \text{ by reading a single letter}\}$$

We could write "can reach q from q_0 by reading a single letter" using our traces notation: "$\exists a \in \Sigma.\ q_0 \xrightarrow{a}{}^* q$".

** 5. Show that language $S = \{w \in \{a, b\}^* \mid w = \mathsf{rev}(w)\}$ is not regular.

Solution

Suppose S is regular. Then it follows from the pumping lemma that there is some length $p > 0$ such that, for any string s of length at least p, we can split it into three pieces the middle of which can be pumped. So let us consider $a^p b^p b^p a^p$. It follows that $a^p b^p b^p a^p$ can be split as uvw with:

1. v not empty
2. $|uv| \leq p$
3. for all $i \in \mathbb{N}$: $uv^i w \in S$

By (2), we know that $uv = a^m$ for some $m \leq p$. By (3), we know that, for example $uvvw \in S$. Let's remember this fact as (*). Since v is not empty, it must be that $v = a^k$ for some $1 \leq k \leq m$. Hence, we have that $uvvw = a^{m-k} a^{2k} a^{p-m} b^p b^p a^p$, but $(m-k)+2k+(p-m) = p+k$ and $p+k > p$ (since $k \geq 1$). Hence, $uvvw \notin S$, contradicting (*) that we earlier deduced. Hence, we must withdraw our only supposition, which was that S is regular. Therefore, S is not regular.

*** 6. Prove that the language of squares (written in unary), $\{1^{n^2} \mid n \in \mathbb{N}\}$, is not regular.

Solution

Suppose S were regular, then it follows from the pumping lemma that there is a certain length of strings from S, say $p > 0$, at and beyond which we can guarantee repetition. So, let's consider 1^{p^2}, which is indeed a string of S with length at least p. The pumping lemma gives us that for this string (and others like it) the string can be split into three pieces: $1^{p^2} = uvw$. We don't know the exact split, but we are guaranteed that:

1. v is non-empty
2. $|uv| \leq p$
3. for all $i \in \mathbb{N}$: $uv^i w \in S$.

By (3), we know that, for example, $uvvw \in S$. But v is a nonempty string of length at most p. This means the word $uvvw = 1^{(p^2+|v|)}$ and $|v| \leq p$ (recall that $|v|$ is the length of v). However, $p^2 + |v|$ is not square, since it sits strictly between p^2 and $(p+1)^2$ (using $p \geq 1$ and $1 \leq |v| \leq p$):

$$p^2 < p^2 + |v| \leq p^2 + p < p^2 + 2p + 1 = (p+1)^2$$

Hence, we must withdraw our only assumption, namely that S is regular.