Programming Languages and Computation

Problem Sheet 4: Regular Languages

* 1. Construct the powerset automaton for the automaton over $\{0, 1\}$ that is drawn below:

$$\begin{tikzcd}
{} \arrow[r] & q_0 \arrow[r, "0"] & q_1 \arrow[loop above, "{0,1}"] \arrow[r, "0"] & q_2
\end{tikzcd}$$

Solution

As usual, we only write down the part of the automaton that is actually reachable from the initial state.

$$\begin{tikzcd}
{} \arrow[r] & \{q_0\} \arrow[r, "0"] \arrow[dr, "1"'] & \{q_1\} \arrow[loop above, "1"] \arrow[r, bend left, "0"] & \{q_1, q_2\} \arrow[l, bend left, "1"] \arrow[loop right, "0"] \\
 & & \emptyset \arrow[loop below, "{0,1}"] &
\end{tikzcd}$$

** 2. Construct finite automata for each of the following sets:

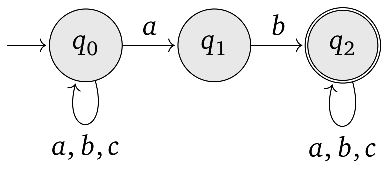
(a) The set of strings over $\{a, b, c\}$ containing the substring ab.

(b) The set $\{w \mid \text{some occurrence of } b \text{ in } w \text{ is } \textit{not} \text{ followed immediately by } c\}$ which is a subset of all strings over the alphabet $\{a, b, c, d\}$.

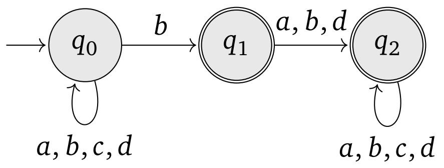
(c) The set of finite sequences of ternary (base-3) digits, i.e. $\{0, 1, 2\}$, that represent numbers *not* divisible by four. We assume that sequences are given to the automaton with most-significant digit first, e.g. the word 201 represents the number written 19 in decimal notation.

Solution

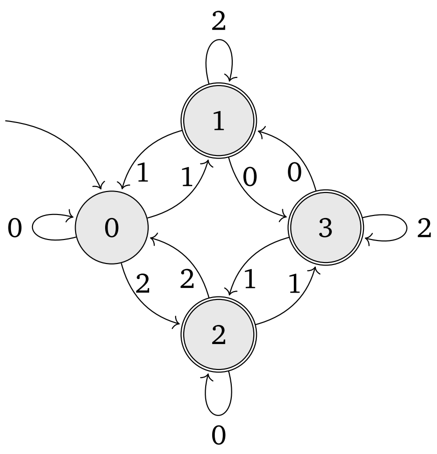
(a)



(b) The idea is for the machine to nondeterministically guess when it has found an occurrence of b that is *not* immediately followed by c and then check that the guess was correct.



(c) In this case, each word w is interpreted as a number and our automaton will keep track of the value of w mod 4 in its state space. To work out the transitions, think: "if I have seen a number that is i mod 4 and now I see the digit j, that means the extended number will be $(i * 3 + j)$ mod 4". After working out the transitions, all that remains is to distinguish 1, 2 and 3 as accepting.



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

Show that if L is a regular language, then so is $\{\text{rev}(w) \mid w \in L\}$.

Solution

Suppose L is regular. Then it is recognised by some deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$, i.e. $L(M) = L$.

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 nondeterministic automaton M' which, on each word $\mathsf{rev}(w)$, guesses which state q of M is it that w is accepted through and then works backwards through the transitions of M until it reaches the initial state.

Hence, let s be a new state not contained in Q and set $M' = (Q \cup \{s\}, \Sigma, \delta', s, \{q_0\})$, where:

$$\delta'(q,a) = \begin{cases} F & \text{if } q = s \\ \{p \mid \delta(p,a) = q\} & \text{otherwise} \end{cases}$$

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

> $$\mathsf{tail}(\epsilon) = \epsilon$$
> $$\mathsf{tail}(a \cdot w) = w$$
>
> Show 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 deterministic finite state automaton M of shape $(Q, \Sigma, q_0, \delta, F)$.

We will build a nondeterministic 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 $M' = (Q \cup \{s\}, \Sigma, q_0, \delta', F)$ where:

$$\delta'(q,a) = \begin{cases} \{p \mid \delta(q_0,a) = p\} & \text{if } q = s \text{ and } a = \epsilon \\ \{\delta(q,a)\} & \text{if } q \in Q \text{ and } a \neq \epsilon \\ \emptyset & \text{otherwise} \end{cases}$$

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

> $\boxed{\text{Hint}}$ $n^2 + m$ is not a square number whenever $0 < m \leq n$.

Solution

Call this language S. We give two solutions, the first is a proof from scratch, the second is using the pumping lemma. It is instructive to compare them.

To show that S is not regular, we will suppose that S is regular and aim to obtain a contradiction, so assume S is regular. Then, by the definition of regular language, there must be a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ that recognises S, i.e. $L(M) = S$. The set of states of a finite automaton is required to be finite, so Q must be finite, say of size k. Note, by definition of the language, whatever this particular number k is, we have:

$$1^{k^2} \in L(M)$$

i.e. 1^{k^2} is accepted by M. By definition of acceptance, necessarily there is a trace $(q_0, 1^{k^2}) \Rightarrow^* (q, \epsilon)$ for some $q \in F$. Since M is deterministic, the shape of such a trace must be as follows (note $1^{k^2} = 1^{k+k(k-1)} = 1^k 1^{k(k-1)}$):

$$(q_0, 1^k 1^{k(k-1)}) \Rightarrow (q_1, 1^{k-1} 1^{k(k-1)}) \Rightarrow (q_2, 1^{k-2} 1^{k(k-1)}) \Rightarrow^{k-3} (q_{k-1}, 1 1^{k(k-1)}) \Rightarrow (q_k, 1^{k(k-1)}) \Rightarrow^* (q, \epsilon)$$

Now, there are $k+1$ configurations in the first part of this trace, up to and including $(q_k, 1^{k(k-1)})$. This part of the trace contains $k+1$ states in the first component of each of the configurations. However, the automaton only has k states, so necessarily at least two of the configurations mention the same state.

Let's say two configurations from this part of the trace that mention the same state are $(q_i, 1^{k-i} 1^{k(k-1)})$ and $(q_j, 1^{k-j} 1^{k(k-1)})$, i.e. $q_i = q_j$. Let's assume $i < j$. Hence, for any word w, it follows that:

$$(q_i, 1^{j-i} w) \Rightarrow^{j-i} (q_i, w)$$

i.e. we can use the cycle on q_i to consume the first $j-i$ ones of any word, returning to the same state in which we started. So we can replay the same trace, but repeating the sequence of configurations that stretches from q_i to q_j, that is (since we now know that $q_i = q_j$) from q_i back to itself.

This allows us to build an accepting trace for $1^{k^2+(j-i)}$, i.e. $1^{k+(j-i)} 1^{k(k-1)}$:

$$(q_0, 1^{k+(j-i)} 1^{(k-1)\cdot k}) \Rightarrow^i (q_i, 1^{k+j-2i} 1^{(k-1)\cdot k}) \Rightarrow^{j-i} (q_i, 1^{k-i} 1^{(k-1)\cdot k}) \Rightarrow^{j-i} (q_j, 1^{k-(j-i)} 1^{(k-1)k}) \Rightarrow^* (q, \epsilon)$$

So $1^{k^2+(j-i)} \in L(M)$. However, following the hint, since $j - i < k$ we have that $k^2 + (j-i)$ is not itself square, thus contradicting the fact that $L(M) = S$. Thus we must withdraw our only assumption, namely that S is regular.

Now with the pumping lemma. 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 $|v| \leq p$. Hence, it follows from the hint that $p^2 + |v|$ is not square, contradicting the description of S. Hence, we must withdraw our only assumption, namely that S is regular.

*** 6. Using the closure properties of regular languages, prove that the following language is *not* regular:

$$\{a^i b^j c^k \mid i, j, k \in \mathbb{N} \text{ and, if } i = 1 \text{ then } j \neq k\}$$

Solution

To show that this language is not regular, we assume that it is and attempt to obtain a contradiction. If this language were regular, then by the closure of regular languages under intersection, it's intersection with ab^*c^*, i.e. the following language, is also regular:

$$\{ab^jc^k \mid j \neq k\}$$

But then, by the solution of Q4, so is the language $\{b^jc^k \mid j \neq k\}$ of its tails. But then, since regular language are closed under complement (Sheet 2, Q4), so is:

$$\{w \mid \text{if } w \text{ is of shape } b^jc^k \text{ then } j = k\}$$

(note that this language contains words like $bcbcb$ in which the b and c are mixed up). Finally, if we take the intersection of this language with b^*c^* then we find that:

$$\{b^jc^k \mid j = k\}$$

is guaranteed to be regular. However, we know that this language is not regular and thus we have our contradiction.

**** 7. (Optional) Assume that $M = (Q, \Sigma, \delta, q_0, F)$ is an automaton recognising the language L. Construct an automaton to recognise the language $\{v \mid \exists w.\ vw \in L \wedge |v| = |w|\}$.

Solution

For each word v in this language, we know that there is an extension w, of equal length, such that vw is accepted by M. Therefore, there is a run of M on input vw that ends in an accepting state. The idea of the following automaton is to start off by guessing which state M will be in after reading v, say $q \in Q$, and then check that:

(a) M reading input v starting from its initial state will reach q and

(b) M reading input w starting from q will reach a final state

In fact, it is not necessary to check that M reading in w from state q will reach a final state, but merely that there is *some* word of length $|w| = |v|$ which will take M from state q to an accepting state.

This is achieved by first guessing the "middle" state $q \in Q$ and then simulating M on v and simulating M on "any word" in parallel using a product construction. One additional complication is that, in addition to remembering the two states of M involved in this parallel simulation, the machine must also remember the state q that was guessed, because the condition that it must check before allowing acceptance is that it is this state that is reached after consuming v in the first part of the simulation.

Therefore, the states of the new machine that are responsible for carrying out this simulation are triples (q, q_2, q_3) where: q is the state that was guessed as the "middle", q_2 is the state currently reached by the simulation of M on v starting from its initial state and q_3 is the state of M currently reached by the simulation of M on "any word" starting from the "middle" state q.

Finally, to cover the special case when L contains ϵ and so v and w may be ϵ too, we add an additional accepting state p_1.

The new machine is $(Q', \Sigma, \delta', p_0, F')$ with:

- $Q' = \{p_0, p_1, \} \cup \{(q, q_1, q_2) \mid q \in Q,\ q_1 \in Q,\ q_2 \in Q\}$
- Transition function is defined by

$$\begin{array}{rcl} \delta'(p_0, \epsilon) & = & \{(q, q_0, q) \mid q \in Q\} \\ \delta'((q, q_1, q_2), a) & = & \{(q, q_1', q_2') \mid q_1' \in \delta(q_1, a) \wedge q_2' \in \bigcup_{b \in \Sigma} \delta(q_2, b)\} \end{array}$$

  If $\epsilon \in L$, then we add one further transition $\delta'(p_0, \epsilon) = \{p_1\}$. And on any other combination of state and letter the image of δ' is the empty set.

- $F = \{(q,\ q,\ q') \mid q \in Q \wedge q' \in F\} \cup \{p_1\}$