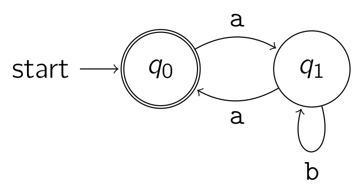
Today's Example

Let M be the automaton illustrated below.



(a) Formalise M as a quintuple.

(b) State the definition of the big-step transition function $\delta_*\colon Q \times \Sigma^* \rightharpoonup Q$, which accepts strings instead of symbols. What does $\mathcal{L}(M)$ mean?

(c) Work out $\delta_*(q_0, \texttt{abaa})$. What is $\delta_*(q_0, \texttt{aab})$?

(d) Prove that $\delta_*(q_0, \texttt{ab}^n\texttt{a}) = q_0$ for all $n \geqslant 0$.

(e) Prove that for any $\boldsymbol{s}, \boldsymbol{t} \in \Sigma^*$, $\delta_*(q, \boldsymbol{s} +\!\!+ \boldsymbol{t}) = \delta_*(\delta_*(q, \boldsymbol{s}), \boldsymbol{t})$.

Consider the language

$$L = \{(\texttt{ab}^n\texttt{a})^m : n, m \geqslant 0\}.$$

(f) Show that $L \subseteq \mathcal{L}(M)$. Do you think $\mathcal{L}(M) \subseteq L$?

Solution

(a) $M = (Q, \Sigma, \delta, q_0, F)$ where

- $Q = \{q_0, q_1\}$
- $\Sigma = \{\texttt{a}, \texttt{b}\}$
- $\delta\colon Q \times \Sigma \to Q$ is given by the transition table

δ	a	b
q_0	q_1	✗
q_1	q_0	q_1

  or more formally, $\delta = \{(q_0, \texttt{a}) \mapsto q_1, (q_1, \texttt{a}) \mapsto q_0, (q_1, \texttt{b}) \mapsto q_1\}$.

- $q_0 = q_0$
- $F = \{q_0\}$.

(b) δ_* is defined by

$$\delta_*(q, \boldsymbol{s}) = \begin{cases} q & \text{if } \boldsymbol{s} = \epsilon \\ \delta_*(\delta(q, \alpha), \boldsymbol{t}) & \text{if } \boldsymbol{s} = \alpha \,.\, \boldsymbol{t}. \end{cases}$$

$\mathcal{L}(M)$ denotes *the language accepted by the automaton M*, i.e., the set of all strings in Σ^* which, when traversed through the automaton, lead us to a final state, i.e.,

$$\mathcal{L}(M) = \{\boldsymbol{\sigma} \in \Sigma^* : \delta_*(q_0, \boldsymbol{\sigma}) \in F\}.$$

(c) Applying the definition from (b), we have

$$\begin{aligned} \delta_*(q_0, \texttt{abaa}) &= \delta_*(\delta(q_0, \texttt{a}), \texttt{baa}) \\ &= \delta_*(q_1, \texttt{baa}) \\ &= \delta_*(\delta(q_1, \texttt{b}), \texttt{aa}) \\ &= \delta_*(q_1, \texttt{aa}) \\ &= \delta_*(\delta(q_1, \texttt{a}), \texttt{a}) \\ &= \delta_*(q_0, \texttt{a}) \\ &= \delta_*(\delta(q_0, \texttt{a}), \epsilon) \\ &= \delta_*(q_1, \epsilon) = q_1, \end{aligned}$$

and

$$\begin{aligned} \delta_*(q_0, \texttt{aab}) &= \delta_*(\delta(q_0, \texttt{a}), \texttt{ab}) \\ &= \delta_*(q_1, \texttt{ab}) \\ &= \delta_*(\delta(q_1, \texttt{a}), \texttt{b}) \\ &= \delta_*(q_0, \texttt{b}) \\ &= \delta_*(\delta(q_0, \texttt{b}), \epsilon), \end{aligned}$$

at which point, we encounter $\delta(q_0, \texttt{b})$, which is not defined. Thus we conclude that $\delta_*(q_0, \texttt{aab})$ is undefined.

(d) By induction on n. For the base case with $n = 0$, we need to prove that $\delta_*(q_0, \texttt{ab}^0\texttt{a}) = q_0$, i.e., that $\delta_*(q_0, \texttt{aa}) = q_0$. Indeed,

$$\begin{aligned} \delta_*(q_0, \texttt{aa}) &= \delta_*(\delta(q_0, \texttt{a}), \texttt{a}) \\ &= \delta_*(q_1, \texttt{a}) \\ &= \delta_*(\delta(q_1, \texttt{a}), \epsilon) \\ &= \delta_*(q_0, \epsilon) = q_0, \end{aligned}$$

which completes the base case.

Now the inductive hypothesis with $n = k$ is that $\delta_*(q_0, \mathtt{ab}^k\mathtt{a}) = q_0$.

For the inductive step, we show that $\delta_*(q_0, \mathtt{ab}^{k+1}\mathtt{a}) = q_0$. Indeed,

$$\begin{aligned}
\delta_*(q_0, \mathtt{ab}^{k+1}\mathtt{a}) &= \delta_*(\delta(q_0, \mathtt{a}), \mathtt{b}^{k+1}\mathtt{a}) \\
&= \delta_*(q_1, \mathtt{b}^{k+1}\mathtt{a}) \\
&= \delta_*(q_1, \mathtt{bb}^k\mathtt{a}) \\
&= \delta_*(\delta(q_1, \mathtt{b}), \mathtt{b}^k\mathtt{a}) \\
&= \delta_*(q_1, \mathtt{b}^k\mathtt{a}) \qquad (*)
\end{aligned}$$

Now the inductive hypothesis tells us that

$$q_0 = \delta_*(q_0, \mathtt{ab}^k\mathtt{a}) = \delta_*(\delta(q_0, \mathtt{a}), \mathtt{b}^k\mathtt{a}) = \delta_*(q_1, \mathtt{b}^k\mathtt{a}).$$

Combining this fact with (*), we obtain that $\delta_*(q_0, \mathtt{ab}^{k+1}\mathtt{a}) = q_0$, which completes the proof. □

(e) By induction on the structure of $\boldsymbol{s}$. For the base case with $s = \epsilon$, we have

$$\begin{aligned}
\delta_*(q, \boldsymbol{s} +\!\!+ \boldsymbol{t}) &= \delta_*(q, \epsilon +\!\!+ \boldsymbol{t}) \\
&= \delta_*(q, \boldsymbol{t}) && (\text{since } \epsilon +\!\!+ \boldsymbol{\sigma} = \boldsymbol{\sigma} \text{ for any } \boldsymbol{\sigma} \in \Sigma^*) \\
&= \delta_*(\delta_*(q, \epsilon), \boldsymbol{t}), && (\text{by definition of } \delta_*)
\end{aligned}$$

as required. Now we proceed with the inductive step. Suppose the result holds for $\boldsymbol{s}' \in \Sigma^*$. We show it holds for $\boldsymbol{s} = \alpha\,.\,\boldsymbol{s}'$, where $\alpha \in \Sigma$.

We consider two different cases. First, suppose $\delta(q, \alpha)$ is defined. Then

$$\begin{aligned}
\delta_*(q, \boldsymbol{s} +\!\!+ \boldsymbol{t}) &= \delta_*(q, \alpha\,.\,\boldsymbol{s}' +\!\!+ \boldsymbol{t}) \\
&= \delta_*(\delta(q, \alpha), \boldsymbol{s}' +\!\!+ \boldsymbol{t}) && (\text{by definition of } \delta_*) \\
&= \delta_*(\delta_*(\delta(q, \alpha), \boldsymbol{s}'), \boldsymbol{t}) && (\text{by the hypothesis}) \\
&= \delta_*(\delta_*(q, \alpha\,.\,\boldsymbol{s}'), \boldsymbol{t}) && (\text{by definition of } \delta_*) \\
&= \delta_*(\delta_*(q, \boldsymbol{s}), \boldsymbol{t}),
\end{aligned}$$

as required. If, on the other hand, $\delta(q, \alpha)$ is undefined, then the left-hand side is

$$\begin{aligned}
\delta_*(q, \boldsymbol{s} +\!\!+ \boldsymbol{t}) &= \delta_*(q, \alpha\,.\,\boldsymbol{s}' +\!\!+ \boldsymbol{t}) \\
&= \delta_*(\delta(q, \alpha), \boldsymbol{s}' +\!\!+ \boldsymbol{t}), && (\text{by definition of } \delta_*)
\end{aligned}$$

which is undefined (since it features $\delta(q, \alpha)$), and the right-hand side is

$$\begin{aligned}\delta_*(\delta_*(q, \boldsymbol{s}), \boldsymbol{t}) &= \delta_*(\delta_*(q, \alpha \,.\, \boldsymbol{s}), \boldsymbol{t}) \\ &= \delta_*(\delta_*(\delta(q, \alpha), \boldsymbol{s}), \boldsymbol{t}), \qquad \text{(by definition of } \delta_*\text{)}\end{aligned}$$

which is also undefined. Thus if $\delta(q, \alpha)$ is undefined, then both sides of the result are undefined.

Therefore in both cases, the result holds. □

(f) To prove that $\mathcal{L}(M) = L$, we need to show both that $\mathcal{L}(M) \subseteq L$ and that $L \subseteq \mathcal{L}(M)$. The second one is easier, let's start form that.

Notice that what we want to prove is

$$\begin{gathered} L \subseteq \mathcal{L}(M) \\ \iff \{(\texttt{ab}^n\texttt{a})^m : n, m \geqslant 0\} \subseteq \{\boldsymbol{\sigma} \in \Sigma^* : \delta_*(q_0, \boldsymbol{\sigma}) \in F\} \\ \iff \text{for all } n, m \geqslant 0,\ (\texttt{ab}^n\texttt{a})^m = \boldsymbol{\sigma} \in \Sigma^* \text{ satisfies } \delta_*(q_0, \boldsymbol{\sigma}) \in \{q_0\} \\ \iff \text{for all } n, m \geqslant 0,\ \delta_*(q_0, (\texttt{ab}^n\texttt{a})^m) = q_0, \end{gathered}$$

so equivalently, we prove this last statement by induction on m.

For the base case with $m = 0$, we have

$$\delta_*(q_0, (\texttt{ab}^n\texttt{a})^0) = \delta_*(q_0, \epsilon) = q_0,$$

for all $n \geqslant 0$, which completes the base case.

The inductive hypothesis with $m = k$ is for all $n \geqslant 0$,

$$\delta_*(q_0, (\texttt{ab}^n\texttt{a})^k) = q_0.$$

For the inductive step, we prove the statement with $m = k + 1$. Indeed, for all $n \geqslant 0$, we have

$$\begin{aligned}\delta_*(q_0, (\texttt{ab}^n\texttt{a})^{k+1}) &= \delta_*(q_0, \texttt{ab}^n\texttt{a} \mathbin{+\!\!+} (\texttt{ab}^n\texttt{a})^k) \\ &= \delta_*(\delta_*(q_0, \texttt{ab}^n\texttt{a}), (\texttt{ab}^n\texttt{a})^k) && \text{(by part (e))} \\ &= \delta_*(q_0, (\texttt{ab}^n\texttt{a})^k) && \text{(by part (d))} \\ &= q_0, && \text{(by the hypothesis)}\end{aligned}$$

which completes the proof that $L \subseteq \mathcal{L}(M)$.

For the second part, we have $\mathcal{L}(M) \nsubseteq L$. Notice that something of the form $(\mathtt{ab}^n\mathtt{a})^m$ will always have the same number of `b`'s between the pairs of `a`'s, e.g.,

$$(\mathtt{ab}^3\mathtt{a})^4 = \mathtt{a}\underbrace{\mathtt{bbb}}_{3}\mathtt{aa}\underbrace{\mathtt{bbb}}_{3}\mathtt{aa}\underbrace{\mathtt{bbb}}_{3}\mathtt{aa}\underbrace{\mathtt{bbb}}_{3}\mathtt{a},$$

but the given DFSA M accepts strings with a varying number of `b`'s, such as

$$\mathtt{a}\underbrace{\ \mathtt{b}\ }_{1}\mathtt{aa}\underbrace{\mathtt{bbbbb}}_{5}\mathtt{aa}\underbrace{\mathtt{bbb}}_{3}\mathtt{aa}\underbrace{\qquad}_{0}\mathtt{a},$$

and strings like this are not in L.

Later on, we will see that it's impossible for any DFSA to have $\mathcal{L}(M) = L$ for a language like L (this is because L is not *regular*). Intuitively, such a DFSA would have to be able to "count" how many `b`'s it is producing, ensuring that it always inserts the same number. DFSAs are not capable of doing this. We need something more powerful than a DFSA, namely a pushdown automaton.