More on automata

Michael George

March 24 — April 7, 2014

1 Automata constructions

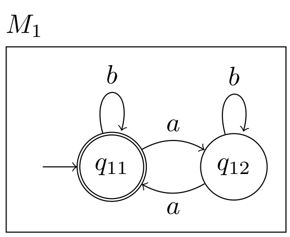
Now that we have a formal model of a machine, it is useful to make some general constructions.

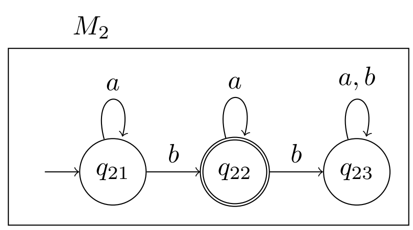
1.1 DFA Union / Product construction

Suppose we have two machines M_1 and M_2, and we wish to construct a machine M that recognizes $L(M_1) \cup L(M_2)$.

To be more specific, let's let Q_1 be the set of states of M_1, q_{01} be the starting state of M_1, A_1 be the accepting states of M_1, and δ_1 be the transition function of M_1. Similarly for M_2.

For example, we may want to accept the strings that have an even number of as and and only one b using machines M_1 that recognizes strings with an even number of as and M_2 that recognizes strings with exactly one b:



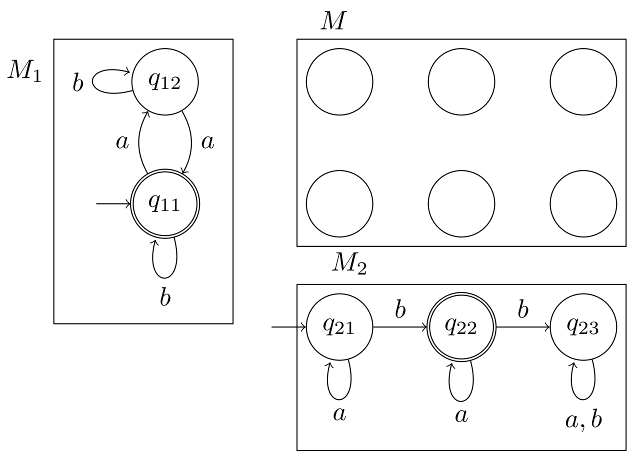


What information does M need to know while processing a string? If we knew what states M_1 and M_2 would be in while processing the string, we could decide whether to accept (accept if either one says "accept").

So, we create a state of M for each pair of states from M_1 and M_2. The intent is that if, after parsing the string x, M_1 ends in state q_1 and M_2 ends in state q_2, then M should end in the state (q_1, q_2). Thus the set of states of M is

$$Q_M = Q_{M_1} \times Q_{M_2}$$

We can draw these states in a grid: the states of M_1 form the x-axis and the states of M_2 form the y-axis:



To figure out the starting state, we ask what we know about the empty string. We know M_1 and M_2 would both be in their starting states (let's call them q_{01} and q_{02}). So M should be in the state (q_{01}, q_{02}).

$$q_{0M} = (q_{01}, q_{02})$$

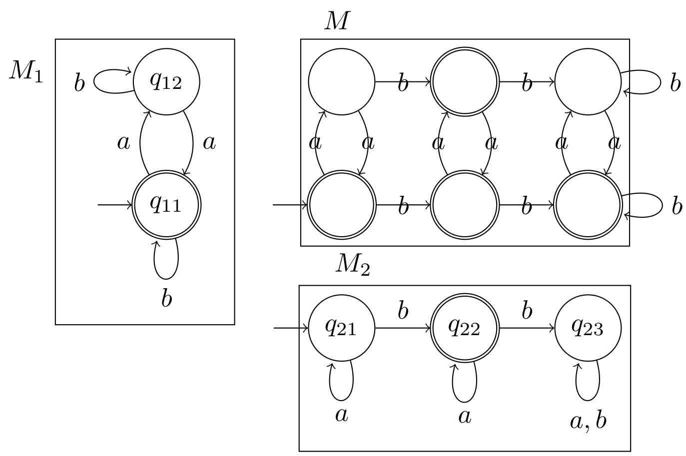
If M is in any state (q_1, q_2), and it sees some character a, where should it transition? Well, we know M_1 would have been in state q_1, and would thus have transitioned on a to state $\delta_1(q_1, a)$ (where δ_1 is the transition function for M_1). Similarly, M_2 would transition to $\delta_2(q_2, a)$. Thus M should transition to the state $(\delta_1(q_1, a), \delta_2(q_2, a))$.

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

Finally, which states of M should be accepting states? Since we are trying to accept the union of $L(M_1)$ and $L(M_2)$, we should accept the string x if either M_1 or M_2 would. So the state (q_1, q_2) should be an accept state if *either* q_1 is an accept state of M_1 or q_2 is an accept state of M_2:

$$A_M = \{(q_1, q_2) \mid q_1 \in A_1 \text{ or } q_2 \in A_2\}$$

Here is the complete picture of M:



We've built a machine. Can we prove that it accepts the correct language? Our intent is that if x causes M to transition to the state (q_1, q_2), then M_1 would be in q_1 and M_2 would be in q_2. Formally, we could prove

$$\widehat{\delta}_M(q_{0M}, x) = \widehat{\delta}_1(q_{01}, x), \widehat{\delta}_2(q_{02}, x)$$

using a straightforward inductive proof (you will work out the details for a closely related problem in the homework).

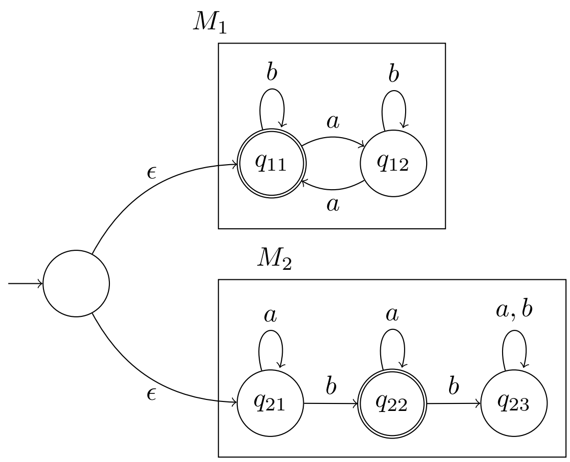
Using this, we can calculate the language of M:

$$\begin{aligned} L(M) &= \{x \in \Sigma^* \mid \widehat{\delta}_M(q_{0M}, x) \in A_M\} \\ &= \cdots \text{ expand using definitions, details in the homework } \cdots \\ &= \{x \mid x \in L(M_1) \text{ or } x \in L(M_2)\} \\ &= L(M_1) \cup L(M_2) \end{aligned}$$

which was our goal.

1.2 NFA Union

Constructing an NFA to recognise the union is much easier: we can simply create a new start state with epsilon transitions to the start states of the two original machines:



1.3 NFA to DFA conversion

It seems as though NFAs are "more powerful" than DFAs: we have more choices when constructing NFAs. It turns out however that NFAs and DFAs accept the same set of languages. That is, if a language L is recognized by an NFA N, then there is a DFA M that also recognizes L.

Given a string x, M should accept x if any of the states that N could reach while processing x are accepting states. We can use this idea to construct M: a state of M will correspond to a *set* of states of N.

$$Q_M = \mathbb{P}(Q_N)$$

Our intent is that if M is in the M-state S (which is a set of states of N), then N could be in any of the states of S.

M should accept in state S if any of the N-states $q \in S$ are themselves accept states:

$$A_M = \{S \in Q_M \mid \exists q \in S \text{ such that } q \in A_N\}$$

N starts in its start state q_{0N}, but it can immediately perform an epsilon transition. Thus after processing no input, M should be in the state corresponding to the set of states reachable from q_{0N} using epsilon transitions:

$$q_{0M} = \widehat{\epsilon}_N(q_{0N})$$

Finally, we need to construct the transition function δ_M to match our intended interpretation of the states of M. If M is in a state $S \in Q_M$ (which is a set of states of N), then we know N could have been in any of the states $q \in S$. That means that after processing an input a, N could be in any of the states reachable from q. This yields the following definition:

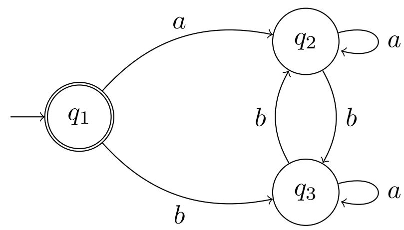
$$\delta_M(S, a) = \bigcup_{q \in S} \widehat{\delta}_N(q, a)$$

The key property of this construction is that $\widehat{\delta}_M(q_{0M}, x) = \widehat{\delta}_N(q_{0N}, x)$ (by an easy inductive proof on x). From this, we can expand the definitions of $L(M)$ and $L(N)$ to show that they are the same.

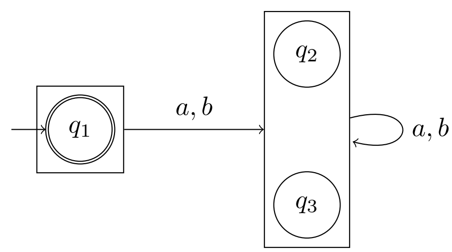
1.4 DFA minimization

One advantage of having a clear machine model is that we can reason about optimizations. One optimization we could do for DFAs is to reduce the number of states.

For example, the following DFA clearly recognizes the language $\{\epsilon\}$:



In a sense, the states q_2 and q_3 are equivalent: if we start processing a string x in either of them, we will always get the same answer. So we can lump them together into a single big "metastate":



We can generalize this idea. Let $\sim$ be the equivalence relation on Q defined by

$$q_1 \sim q_2 \text{ iff } \forall x \in \Sigma^*, \widehat{\delta}(q_1, x) \in A \iff \widehat{\delta}(q_2, x) \in A$$

This formalizes the idea that if we start processing x in q_1 or in q_2, we will always get the same answer. If we know $\sim$, we can construct an equivalent machine M_{min} as follows:

- The states Q_{min} are equivalence classes of states of M: $Q_{min} = Q_M / \sim$
- The accepting states of Q_{min} are the equivalence classes of accepting states of M. Note that if $q_1 \in A_M$ and $q_2 \sim q_1$ then $q_2 \in A_M$ (plug ϵ into the definition of $\sim$).
- The initial state of Q_{min} is just $[q_{0M}]$.
- The transition function δ_{min} is given by $\delta_{min}([q], a) = [\delta_M(q, a)]$. This is well-defined (proof by contradiction).