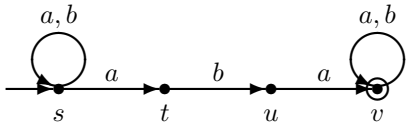
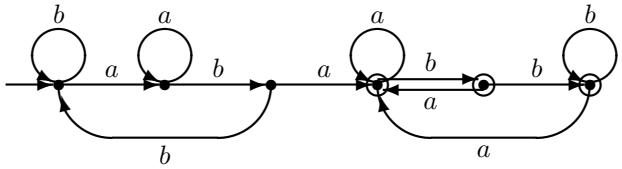
Lecture 13

DFA State Minimization

By now you have probably come across several situations in which you have observed that some automaton could be simplified either by deleting states inaccessible from the start state or by collapsing states that were equivalent in some sense. For example, if you were to apply the subset construction to the NFA

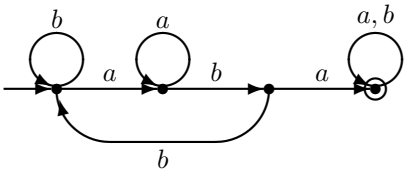


accepting the set of all strings containing the substring *aba*, you would obtain a DFA with $2^4 = 16$ states. However, all except six of these states are inaccessible. Deleting them, you would obtain the DFA



From left to right, the states of this DFA correspond to the subsets $\{s\}$, $\{s, t\}$, $\{s, u\}$, $\{s, t, v\}$, $\{s, u, v\}$, $\{s, v\}$.

Now, note that the rightmost three states of this DFA might as well be collapsed into a single state, since they are all accept states, and once the machine enters one of them it cannot escape. Thus this DFA is equivalent to



This is a simple example in which the equivalence of states is obvious, but sometimes it is not so obvious. In this and the next lecture we will develop a mechanical method to find all equivalent states of any given DFA and collapse them. This will give a DFA for any given regular set A that has as few states as possible. An amazing fact is that every regular set has a minimal DFA that is unique up to isomorphism, and there is a purely mechanical method for constructing it from any given DFA for A.

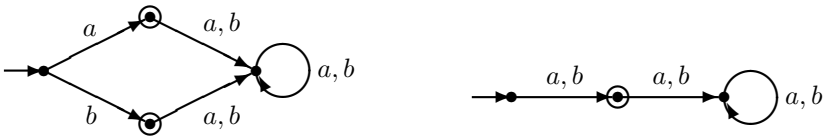
Say we are given a DFA $M = (Q, \Sigma, \delta, s, F)$ for A. The minimization process consists of two stages:

1. Get rid of inaccessible states; that is, states q for which there exists no string $x \in \Sigma^*$ such that $\widehat{\delta}(s, x) = q$.

2. Collapse "equivalent" states.

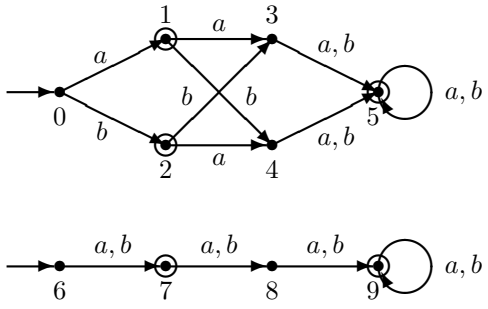
Removing inaccessible states surely does not change the set accepted. It is quite straightforward to see how to do this mechanically using depth-first search on the transition graph. Let us then assume that this has been done. For stage 2, we need to say what we mean by "equivalent" and how we do the collapsing. Let's look at some examples before giving a formal definition.

Example 13.1



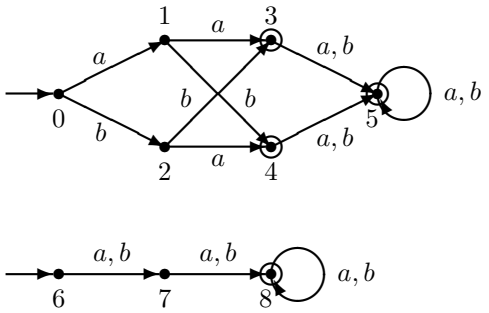
These automata both accept the set $\{a, b\}$. The automaton with four states goes to different states depending on the first input symbol, but there's really no reason for the states to be separate. They are equivalent and can be collapsed into one state, giving the automaton with three states. □

Example 13.2



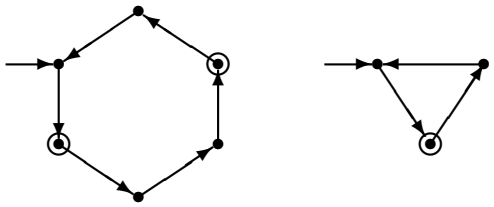
This example is a little more complicated. The automata both accept the set $\{a, b\} \cup \{\text{strings of length 3 or greater}\}$. In the first automaton, states 3 and 4 are equivalent, since they both go to state 5 under both input symbols, so there's no reason to keep them separate. Once we collapse them, we can collapse 1 and 2 for the same reason, giving the second automaton. State 0 becomes state 6; states 1 and 2 collapse to become state 7; states 3 and 4 collapse to become state 8; and state 5 becomes state 9. □

Example 13.3



Here we have modified the first automaton by making states 3, 4 accept states instead of 1, 2. Now states 3, 4, 5 are equivalent and can be collapsed. These become state 8 of the second automaton. The set accepted is the set of all strings of length at least two. □

Example 13.4



These automata both accept the set $\{a^m \mid m \equiv 1 \bmod 3\}$ (edge labels are omitted). In the left automaton, diametrically opposed states are equivalent and can be collapsed, giving the automaton on the right. □

The Quotient Construction

How do we know in general when two states can be collapsed safely without changing the set accepted? How do we do the collapsing formally? Is there a fast algorithm for doing it? How can we determine whether any further collapsing is possible?

Surely we never want to collapse an accept state p and a reject state q, because if $p = \widehat{\delta}(s, x) \in F$ and $q = \widehat{\delta}(s, y) \notin F$, then x must be accepted and y

must be rejected even after collapsing, so there is no way to declare the collapsed state to be an accept or reject state without error. Also, if we collapse p and q, then we had better also collapse $\delta(p, a)$ and $\delta(q, a)$ to maintain determinism. These two observations together imply inductively that we cannot collapse p and q if $\widehat{\delta}(p, x) \in F$ and $\widehat{\delta}(q, x) \notin F$ for some string x.

It turns out that this criterion is necessary and sufficient for deciding whether a pair of states can be collapsed. That is, if there exists a string x such that $\widehat{\delta}(p, x) \in F$ and $\widehat{\delta}(q, x) \notin F$ or vice versa, then p and q cannot be safely collapsed; and if no such x exists, then they can.

Here's how we show this formally. We first define an equivalence relation $\approx$ on Q by

$$p \approx q \quad \stackrel{\text{def}}{\iff} \quad \forall x \in \Sigma^* \ (\widehat{\delta}(p, x) \in F \iff \widehat{\delta}(q, x) \in F).$$

This definition is just a formal restatement of the collapsing criterion. It is not hard to argue that the relation $\approx$ is indeed an equivalence relation: it is

- *reflexive*: $p \approx p$ for all p;
- *symmetric*: if $p \approx q$, then $q \approx p$; and
- *transitive*: if $p \approx q$ and $q \approx r$, then $p \approx r$.

As with all equivalence relations, $\approx$ partitions the set on which it is defined into disjoint *equivalence classes*:

$$[p] \quad \stackrel{\text{def}}{=} \quad \{q \mid q \approx p\}.$$

Every element $p \in Q$ is contained in exactly one equivalence class $[p]$, and

$$p \approx q \quad \iff \quad [p] = [q].$$

We now define a DFA $M/\!\approx$ called the *quotient automaton*, whose states correspond to the equivalence classes of $\approx$. This construction is called a *quotient construction* and is quite common in algebra. We will see a more general account of it in Supplementary Lectures C and D.

There is one state of $M/\!\approx$ for each $\approx$-equivalence class. In fact, formally, the states of $M/\!\approx$ *are* the equivalence classes; this is the mathematical way of "collapsing" equivalent states.

Define

$$M/\!\approx \quad \stackrel{\text{def}}{=} \quad (Q', \Sigma, \delta', s', F'),$$

where

$$\begin{aligned} Q' &\stackrel{\text{def}}{=} \{[p] \mid p \in Q\}, \\ \delta'([p], a) &\stackrel{\text{def}}{=} [\delta(p, a)], \\ s' &\stackrel{\text{def}}{=} [s], \\ F' &\stackrel{\text{def}}{=} \{[p] \mid p \in F\}. \end{aligned} \tag{13.1}$$

There is a subtle but important point involving the definition of δ' in (13.1): we need to show that it is *well-defined*. Note that the action of δ' on the equivalence class $[p]$ is defined in terms of p. It is conceivable that a different choice of representative of the class $[p]$ (i.e., some q such that $q \approx p$) might lead to a different right-hand side in (13.1). Lemma 13.5 says exactly that this does not happen.

Lemma 13.5 *If $p \approx q$, then $\delta(p,a) \approx \delta(q,a)$. Equivalently, if $[p] = [q]$, then $[\delta(p,a)] = [\delta(q,a)]$.*

Proof. Suppose $p \approx q$. Let $a \in \Sigma$ and $y \in \Sigma^*$.

$$\begin{aligned} \widehat{\delta}(\delta(p,a),y) \in F &\iff \widehat{\delta}(p,ay) \in F \\ &\iff \widehat{\delta}(q,ay) \in F \qquad \text{since } p \approx q \\ &\iff \widehat{\delta}(\delta(q,a),y) \in F. \end{aligned}$$

Since y was arbitrary, $\delta(p,a) \approx \delta(q,a)$ by definition of $\approx$. □

Lemma 13.6 $p \in F \iff [p] \in F'$.

Proof. The direction $\Rightarrow$ is immediate from the definition of F'. For the direction $\Leftarrow$, we need to show that if $p \approx q$ and $p \in F$, then $q \in F$. In other words, every $\approx$-equivalence class is either a subset of F or disjoint from F. This follows immediately by taking $x = \epsilon$ in the definition of $p \approx q$. □

Lemma 13.7 *For all $x \in \Sigma^*$, $\widehat{\delta'}([p],x) = [\widehat{\delta}(p,x)]$.*

Proof. By induction on $|x|$.

Basis For $x = \epsilon$,

$$\begin{aligned} \widehat{\delta'}([p],\epsilon) &= [p] && \text{definition of } \widehat{\delta'} \\ &= [\widehat{\delta}(p,\epsilon)] && \text{definition of } \widehat{\delta}. \end{aligned}$$

Induction step Assume $\widehat{\delta'}([p],x) = [\widehat{\delta}(p,x)]$, and let $a \in \Sigma$.

$$\begin{aligned} \widehat{\delta'}([p],xa) &= \delta'(\widehat{\delta'}([p],x),a) && \text{definition of } \widehat{\delta'} \\ &= \delta'([\widehat{\delta}(p,x)],a) && \text{induction hypothesis} \\ &= [\delta(\widehat{\delta}(p,x),a)] && \text{definition of } \delta' \\ &= [\widehat{\delta}(p,xa)] && \text{definition of } \widehat{\delta}. \end{aligned}$$

□

Theorem 13.8 $L(M/\approx) = L(M)$.

Proof. For $x \in \Sigma^*$,

$$\begin{array}{rcll} x \in L(M/\approx) & \iff & \widehat{\delta}'(s', x) \in F' & \text{definition of acceptance} \\ & \iff & \widehat{\delta}'([s], x) \in F' & \text{definition of } s' \\ & \iff & [\widehat{\delta}(s, x)] \in F' & \text{Lemma 13.7} \\ & \iff & \widehat{\delta}(s, x) \in F & \text{Lemma 13.6} \\ & \iff & x \in L(M) & \text{definition of acceptance.} \end{array} \quad \square$$

$M/\approx$ Cannot Be Collapsed Further

It is conceivable that after doing the quotient construction once, we might be able to collapse even further by doing it again. It turns out that once is enough. To see this, let's do the quotient construction a second time. Define

$$[p] \sim [q] \quad \stackrel{\text{def}}{\iff} \quad \forall x \in \Sigma^* \ (\widehat{\delta}'([p], x) \in F' \iff \widehat{\delta}'([q], x) \in F').$$

This is exactly the same definition as $\approx$ above, only applied to the quotient automaton $M/\approx$. We use the notation $\sim$ for the equivalence relation on Q' to distinguish it from the relation $\approx$ on Q. Now

$$\begin{array}{rcll} & & [p] \sim [q] & \\ & \Rightarrow & \forall x \ (\widehat{\delta}'([p], x) \in F' \iff \widehat{\delta}'([q], x) \in F') & \text{definition of } \sim \\ & \Rightarrow & \forall x \ ([\widehat{\delta}(p, x)] \in F' \iff [\widehat{\delta}(q, x)] \in F') & \text{Lemma 13.7} \\ & \Rightarrow & \forall x \ (\widehat{\delta}(p, x) \in F \iff \widehat{\delta}(q, x) \in F) & \text{Lemma 13.6} \\ & \Rightarrow & p \approx q & \text{definition of } \approx \\ & \Rightarrow & [p] = [q]. & \end{array}$$

Thus any two equivalent states of $M/\approx$ are in fact equal, and the collapsing relation $\sim$ on Q' is just the identity relation $=$.

Lecture 14

A Minimization Algorithm

Here is an algorithm for computing the collapsing relation $\approx$ for a given DFA M with no inaccessible states. Our algorithm will mark (unordered) pairs of states $\{p, q\}$. A pair $\{p, q\}$ will be marked as soon as a reason is discovered why p and q are *not* equivalent.

1. Write down a table of all pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$ or vice versa.
3. Repeat the following until no more changes occur: if there exists an unmarked pair $\{p, q\}$ such that $\{\delta(p, a), \delta(q, a)\}$ is marked for some $a \in \Sigma$, then mark $\{p, q\}$.
4. When done, $p \approx q$ iff $\{p, q\}$ is not marked.

Here are some things to note about this algorithm:

- If $\{p, q\}$ is marked in step 2, then p and q are surely not equivalent: take $x = \epsilon$ in the definition of $\approx$.
- We may have to look at the same pair $\{p, q\}$ many times in step 3, since any change in the table may suddenly allow $\{p, q\}$ to be marked. We stop only after we make an entire pass through the table with no new marks.
- The algorithm runs for only a finite number of steps, since there are only $\binom{n}{2}$ possible marks that can be made,[1] and we have to make at least one new mark in each pass to keep going.

[1] $\binom{n}{k} \stackrel{\text{def}}{=} \frac{n!}{k!(n-k)!}$, the number of subsets of size k in a set of size n.

- Step 4 is really a statement of the theorem that the algorithm correctly computes $\approx$. This requires proof, which we defer until later.

Example 14.1 Let's minimize the automaton of Example 13.2 of Lecture 13.

		a	b
$\rightarrow$	0	1	2
	$1F$	3	4
	$2F$	4	3
	3	5	5
	4	5	5
	$5F$	5	5

Here is the table built in step 1. Initially all pairs are unmarked.

0					
–	1				
–	–	2			
–	–	–	3		
–	–	–	–	4	
–	–	–	–	–	5

After step 2, all pairs consisting of one accept state and one nonaccept state have been marked.

0					
✓	1				
✓	–	2			
–	✓	✓	3		
–	✓	✓	–	4	
✓	–	–	✓	✓	5

Now look at an unmarked pair, say $\{0, 3\}$. Under input a, 0 and 3 go to 1 and 5, respectively (write: $\{0, 3\} \rightarrow \{1, 5\}$). The pair $\{1, 5\}$ is not marked, so we don't mark $\{0, 3\}$, at least not yet. Under input b, $\{0, 3\} \rightarrow \{2, 5\}$, which is not marked, so we still don't mark $\{0, 3\}$. We then look at unmarked pairs $\{0, 4\}$ and $\{1, 2\}$ and find out we cannot mark them yet for the same reasons. But for $\{1, 5\}$, under input a, $\{1, 5\} \rightarrow \{3, 5\}$, and $\{3, 5\}$ is marked, so we mark $\{1, 5\}$. Similarly, under input a, $\{2, 5\} \rightarrow \{4, 5\}$ which is marked, so we mark $\{2, 5\}$. Under both inputs a and b, $\{3, 4\} \rightarrow \{5, 5\}$, which is never marked (it's not even in the table), so we do not mark $\{3, 4\}$. After the first pass of step 3, the table looks like

0					
✓	1				
✓	–	2			
–	✓	✓	3		
–	✓	✓	–	4	
✓	✓	✓	✓	✓	5

Now we make another pass through the table. As before, $\{0,3\} \to \{1,5\}$ under input a, but this time $\{1,5\}$ is marked, so we mark $\{0,3\}$. Similarly, $\{0,4\} \to \{2,5\}$ under input b, and $\{2,5\}$ is marked, so we mark $\{0,4\}$. This gives

$$\begin{array}{cccccc} 0 & & & & & \\ \checkmark & 1 & & & & \\ \checkmark & - & 2 & & & \\ \checkmark & \checkmark & \checkmark & 3 & & \\ \checkmark & \checkmark & \checkmark & - & 4 & \\ \checkmark & \checkmark & \checkmark & \checkmark & \checkmark & 5 \end{array}$$

Now we check the remaining unmarked pairs and find out that $\{1,2\} \to \{3,4\}$ and $\{3,4\} \to \{5,5\}$ under both a and b, and neither $\{3,4\}$ nor $\{5,5\}$ is marked, so there are no new marks. We are left with unmarked pairs $\{1,2\}$ and $\{3,4\}$, indicating that $1 \approx 2$ and $3 \approx 4$. □

Example 14.2 Now let's do Example 13.4 of Lecture 13.

		a
$\to$	0	1
	$1F$	2
	2	3
	3	4
	$4F$	5
	5	0

Here is the table after step 2.

$$\begin{array}{cccccc} 0 & & & & & \\ \checkmark & 1 & & & & \\ - & \checkmark & 2 & & & \\ - & \checkmark & - & 3 & & \\ \checkmark & - & \checkmark & \checkmark & 4 & \\ - & \checkmark & - & - & \checkmark & 5 \end{array}$$

Then:

- $\{0,2\} \to \{1,3\}$, which is marked, so mark $\{0,2\}$.
- $\{0,3\} \to \{1,4\}$, which is not marked, so do not mark $\{0,3\}$.
- $\{0,5\} \to \{0,1\}$, which is marked, so mark $\{0,5\}$.
- $\{1,4\} \to \{2,5\}$, which is not marked, so do not mark $\{1,4\}$.
- $\{2,3\} \to \{3,4\}$, which is marked, so mark $\{2,3\}$.
- $\{2,5\} \to \{0,3\}$, which is not marked, so do not mark $\{2,5\}$.
- $\{3,5\} \to \{0,4\}$, which is marked, so mark $\{3,5\}$.

After the first pass, the table looks like this:

$$\begin{array}{cccccc} 0 & & & & & \\ \checkmark & 1 & & & & \\ \checkmark & \checkmark & 2 & & & \\ - & \checkmark & \checkmark & 3 & & \\ \checkmark & - & \checkmark & \checkmark & 4 & \\ \checkmark & \checkmark & - & \checkmark & \checkmark & 5 \end{array}$$

Now do another pass. We discover that $\{0,3\} \to \{1,4\} \to \{2,5\} \to \{0,3\}$ and none of these are marked, so we are done. Thus $0 \approx 3$, $1 \approx 4$, and $2 \approx 5$. □

Correctness of the Collapsing Algorithm

Theorem 14.3 *The pair $\{p,q\}$ is marked by the above algorithm if and only if there exists $x \in \Sigma^*$ such that $\widehat{\delta}(p,x) \in F$ and $\widehat{\delta}(q,x) \notin F$ or vice versa; i.e., if and only if $p \not\approx q$.*

Proof. This is easily proved by induction. We leave the proof as an exercise (Miscellaneous Exercise 49). □

A nice way to look at the algorithm is as a finite automaton itself. Let

$$\mathcal{Q} \quad = \quad \{\{p,q\} \mid p,q \in Q,\ p \neq q\}.$$

There are $\binom{n}{2}$ elements of $\mathcal{Q}$, where n is the size of Q. Define a nondeterministic "transition function"

$$\Delta : \mathcal{Q} \quad \to \quad 2^{\mathcal{Q}}$$

on $\mathcal{Q}$ as follows:

$$\Delta(\{p,q\},a) \quad = \quad \{\{p',q'\} \mid p = \delta(p',a),\ q = \delta(q',a)\}.$$

Define a set of "start states" $\mathcal{S} \subseteq \mathcal{Q}$ as follows:

$$\mathcal{S} \quad = \quad \{\{p,q\} \mid p \in F,\ q \notin F\}.$$

(We don't need to write "... or vice versa" because $\{p,q\}$ is an unordered pair.) Step 2 of the algorithm marks the elements of $\mathcal{S}$, and step 3 marks pairs in $\Delta(\{p,q\},a)$ when $\{p,q\}$ is marked for any $a \in \Sigma$. In these terms, Theorem 14.3 says that $p \not\approx q$ iff $\{p,q\}$ is accessible in this automaton.