LANGAGES FORMELS 2025-26
CM 3

5. The minimal automaton

A smallest possible DFA for any regular language.

Let $(M, \cdot_M, 1_M)$ be a monoid, and S a set (not necessarily finite).

An **action** of M on S is a function $\bullet : S \times M \to S$ such that, for every $s \in S, u, v \in M$,

$$s \bullet 1_M = s \qquad \text{and} \qquad (s \bullet u) \bullet v = s \bullet (u \cdot v).$$

Fact There is a bijection between the set of actions of M on S and the set of monoid homomorphisms $M \to S^S$.

(S^S: monoid of functions $S \to S$ with composition and id_S as unit.)

Proof. Given an action $\bullet : S \times M \to S$, define $\varphi_\bullet : M \to S^S$ by $\varphi_\bullet(m) = \lambda s.\, m \bullet s$, for $m \in M$.

Given a homomorphism $\varphi : M \to S^S$, define, for $m \in M$, $s \in S$, $s \bullet_\varphi m := \varphi(m)(s)$.

Exercise: The assignments $\bullet \mapsto \varphi_\bullet$ and $\varphi \mapsto \bullet_\varphi$ give a well-defined bijection. □

In particular, a DFA $A = (Q, \Sigma, \delta, I, F)$ may be equivalently described as $A = (Q, \Sigma, \bullet, i, F)$ where $\bullet$ is an action of Σ^* on Q, $i \in Q$, $F \subseteq Q$. In this description, A accepts $w \in \Sigma^*$ if, and only if, $i \bullet w \in F$. We will often use the latter perspective on DFA's.

Let $\mathcal{P}(\Sigma^*)$ be the set of all languages in alphabet Σ.

For any $w \in \Sigma^*$ and $L \in \mathcal{P}(\Sigma^*)$, define

$$L \cdot w := w^{-1}L. \qquad (= \{x \in \Sigma^* \mid wx \in L\})$$

This is an action of Σ^* on $\mathcal{P}(\Sigma^*)$:

$$L \cdot \varepsilon = \varepsilon^{-1}L = L \quad \text{and} \quad L \cdot (uv) = (uv)^{-1}L = v^{-1}(u^{-1}L) = (L \cdot u) \cdot v.$$

Also define $F := \{L \in \mathcal{P}(\Sigma^*) \mid \varepsilon \in L\}$.

} the "universal automaton"

Note: no initial state chosen yet.

Let $L \in \mathcal{P}(\Sigma^*)$ be any language. For any $w \in \Sigma^*$, we have:

$$w \in L \iff \varepsilon \in w^{-1}L \iff L \cdot w \in F$$

⚠ ⇒ The set of all languages, equipped with the above action $\cdot$ and set of final states F, lets us define a deterministic automaton $\mathcal{U}_L := (\mathcal{P}(\Sigma^*), \Sigma, \cdot, L, F)$ with infinitely many states, which recognizes the language L.

Conclusion. Any language can be recognized by a deterministic infinite automaton.

We call an automaton A **accessible** if, for every state q, there exists a path from an initial state to q.

By a **subautomaton** of $A = (Q, \Sigma, \delta, I, F)$, we mean an automaton of the form

$A' = (Q', \Sigma, \delta \cap (Q' \times \Sigma \times Q'), I \cap Q', F \cap Q')$, where $Q' \subseteq Q$ is a subset.

Fact. The largest accessible subautomaton of a DA $A = (Q, \Sigma, \cdot, i, F)$, **Reach(A)**, has set of states $\{i \cdot w \mid w \in \Sigma^*\}$, and $\mathcal{L}(\text{Reach}(A)) = \mathcal{L}(A)$.

Let $L \subseteq \Sigma^*$. The set of reachable states in the universal automaton $\mathcal{U}_L$ is:

$$\{L \cdot w \mid w \in \Sigma^*\} = \{w^{-1}L \mid w \in \Sigma^*\} =: Q_L$$

This is the set of states of an accessible subautomaton of $\mathcal{U}_L$:

$A_L := \text{Reach}(\mathcal{U}_L) = (Q_L, \Sigma, \cdot, L, \overbrace{F \cap Q_L}^{F_L})$, the **Nerode automaton** of L.

Theorem. Let $L \subseteq \Sigma^*$. Then L is regular if, and only if, the set $\{w^{-1}L \mid w \in \Sigma^*\}$ is finite.

Proof. We proved "$\Rightarrow$" last lecture. "$\Leftarrow$": the Nerode automaton is then a DFA recognizing L, since the universal automaton recognizes L, and $A_L = \text{Reach}(\mathcal{U}_L)$. □

We will show that A_L is **minimal** among the DFA's recognizing L.

Let $A = (Q, \Sigma, \cdot, i, F)$ be any DFA. We define the function

$$\begin{aligned} \mathcal{L}_A : Q &\longrightarrow \mathrm{Rec}(\Sigma^*) \\ q &\longmapsto \mathcal{L}_A(q) := \{ w \in \Sigma^* \mid q \cdot w \in F \}. \end{aligned}$$

So $\mathcal{L}(A) = \mathcal{L}_A(i)$.

Lemma. For any $q \in Q$, $u \in \Sigma^*$, $\mathcal{L}_A(q \cdot u) = u^{-1} \mathcal{L}_A(q)$.

Proof. Let $w \in \Sigma^*$.

$$\begin{aligned} w \in \mathcal{L}_A(q \cdot u) &\iff (q \cdot u) \cdot w \in F && \text{(definition of } \mathcal{L}_A\text{)} \\ &\iff q \cdot (uw) \in F && \text{(action)} \\ &\iff uw \in \mathcal{L}_A(q) && \text{(definition of } \mathcal{L}_A\text{)} \\ &\iff w \in u^{-1} \mathcal{L}_A(q). && \text{(definition of } u^{-1}\text{)} \end{aligned}$$

□

Proposition. Let $A = (Q, \Sigma, \cdot, i, F)$ be an accessible DFA, and $L := \mathcal{L}(A) = \mathcal{L}_A(i)$.

The assignment $(q \in Q) \longmapsto \mathcal{L}_A(q)$ is a well-defined surjective function $Q \twoheadrightarrow Q_L$.

Proof. Let $q \in Q$. Pick $u \in \Sigma^*$ such that $i \cdot u = q$, by accessibility.

By the Lemma, $\mathcal{L}_A(q) = \mathcal{L}_A(i \cdot u) = u^{-1}\mathcal{L}_A(i) = u^{-1}L$, which is in Q_L.

For surjectivity, let $u^{-1}L \in Q_L$ for $u \in \Sigma^*$. Then $\mathcal{L}_A(i \cdot u) = u^{-1}L$ as above. □

A surjective function $f : Q \to R$ is a morphism from $A = (Q, \Sigma, \cdot, i, F)$ to $B = (R, \Sigma, \cdot, j, G)$ if

(1) $f(i) = j$, (2) for every $a \in \Sigma$, $q \in Q$: $f(q \cdot a) = f(q) \cdot a$; (3) $f^{-1}(G) = F$.

NB: There exist different definitions of "morphism of automata" in the literature. This one is from [Pin 2021].

Proposition. The surjective function $\mathcal{L}_A : Q \twoheadrightarrow Q_L$ is a morphism.

Proof. 1) $\mathcal{L}_A(i) = L$;

2) $\mathcal{L}_A(q \cdot a) = a^{-1}\mathcal{L}_A(q) = \mathcal{L}_A(q) \cdot a$;

3) $\varepsilon \in \mathcal{L}_A(q) \iff q \in F$. □

[Pin2021] J.-É. Pin, Ch. 1 in: Handbook of Automata Theory (2021)

https://fr.wikipedia.org/wiki/Jean-Éric_Pin

Theorem. Let $L \in \mathrm{Rec}(\Sigma^*)$. For any DFA A with $\mathcal{L}(A) = L$, there exist a subautomaton A' of A and a surjective morphism $A' \twoheadrightarrow A_L$.

In particular, $\#Q_L \leq \#Q_A$.

Proof. Let $A' := \mathrm{Reach}(A)$. Then $\mathcal{L}(A') = \mathcal{L}(A) = L$, and A' is accessible.

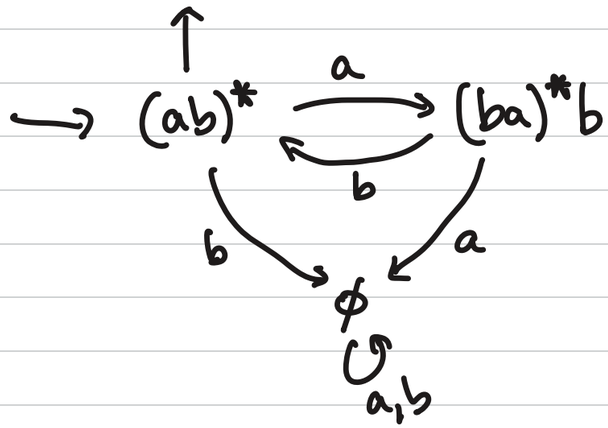
By the two preceding propositions, $\mathcal{L}_{A'} : A' \twoheadrightarrow A_L$ is a surjective morphism.

Also: $\#Q_L \leq \#Q_{A'} \leq \#Q_A$. □

Example. Let $L = (ab)^*$, $\Sigma = \{a, b\}$. We compute the Nerode automaton A_L.

$a^{-1}L = (ba)^*b$ $\qquad$ $a^{-1}(a^{-1}L) = \emptyset$ $\qquad$ $a^{-1}\emptyset = b^{-1}\emptyset = \emptyset$.

$b^{-1}L = \emptyset$ $\qquad$ $b^{-1}(a^{-1}L) = b^{-1}((ba)^*b) \overset{(*)}{=} (ab)^*$



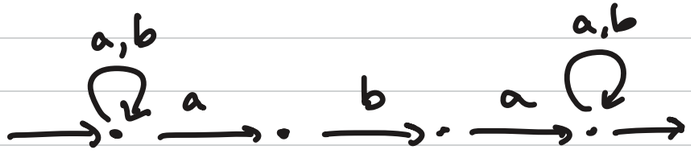
Here, we were "lucky" that we saw the equality $(*)$.

How to do this in general?

6. Minimization

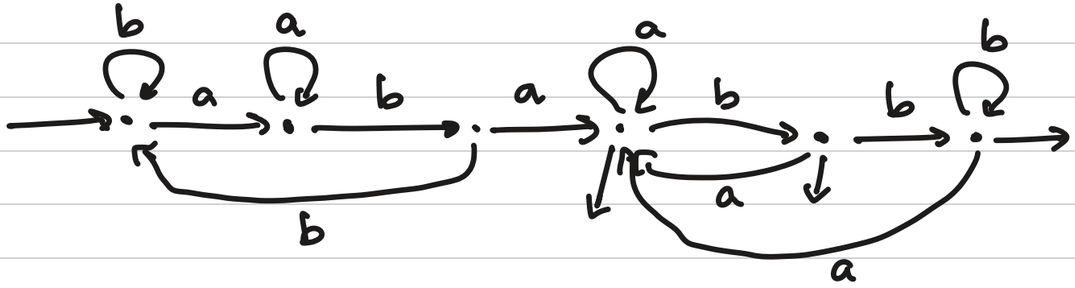
How to compute the minimal automaton, starting from a given DFA.

Example from last lecture: $L = \{w \in \Sigma^* \mid aba \text{ is a factor of } w\}$

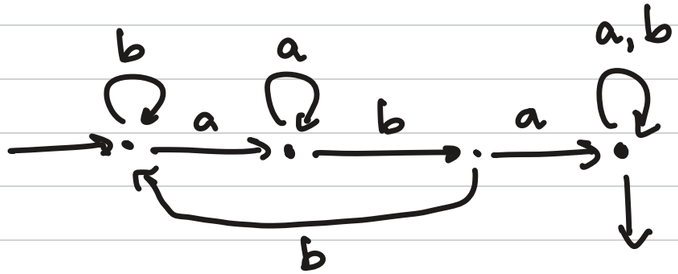


determinize $2^4 = 16$ states.

delete inaccessible states



collapse last 3 states



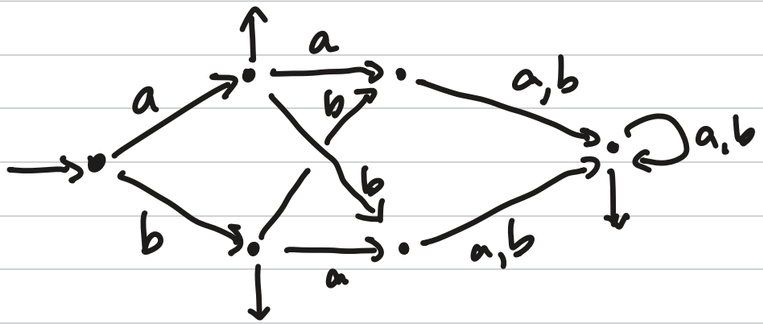
only 4 states!

For $L \in \mathrm{Rec}(\Sigma^*)$, a DFA A such that $\mathcal{L}(A) = L$ is **minimal** if, for every DFA B such that $\mathcal{L}(B) = L$, $\#Q_B \geq \#Q_A$.

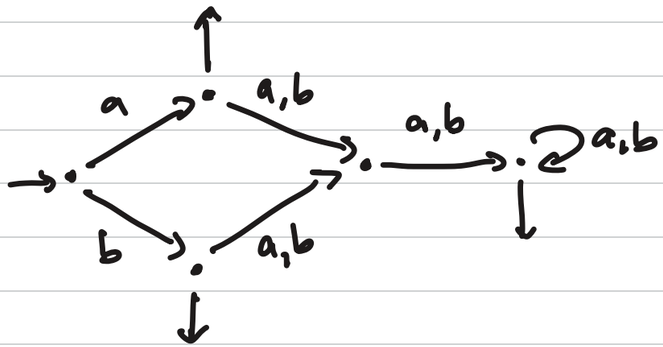
Lecture 13 in: D. Kozen, Automata and Computability, Springer-Verlag (1997)

Another example of collapsing, step-by-step.

Example.

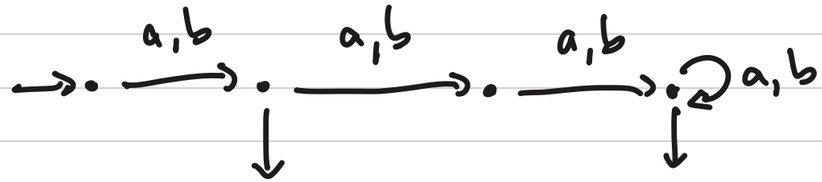


A_0



first collapse

A_1



second collapse

A_2 = minimal DFA for $\mathcal{L}(A)$

Q. When should we allow collapsing? How to formalize this process?

The formal notion used for "collapsing" is that of an **equivalence relation**, i.e.,
a binary relation E on a set S which is **reflexive** ($\forall s \in S,\ sEs$),
("sEt" is notation for: $(s,t) \in E$) **symmetric** ($\forall s,t \in S,\ sEt \Rightarrow tEs$), and
transitive ($\forall s,t,u \in S,\ sEt \,\&\, tEu \Rightarrow sEu$).

We can then form the **quotient** S/E, whose elements are **equivalence classes**.
We have the **quotient map** $\pi_E : S \twoheadrightarrow S/E$, sending $s \in S$ to its equivalence class $\pi_E(s) = [s]_E$.
("We often write "$[s]$", when E is fixed.")

When $f : S \to T$ is a function, we define the **kernel** of f to be the relation

$$\ker(f) := \{ (s,s') \in S^2 \mid f(s) = f(s') \}.$$

Fact. For any $f : S \to T$, $\ker(f)$ is an equivalence relation on S.

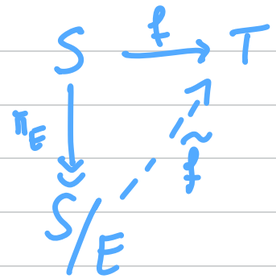
Proof. Exercise. □

Prop. Let E be an equivalence relation on S and $f: S \to T$ a function.

Then: there exists a function $\tilde{f}: S/E \to T$ such that $\tilde{f} \circ \pi_E = f$

if, and only if,

$E \subseteq \ker(f)$.



Moreover, at most one such $\tilde{f}$ exists.

Proof. Exercise. □

Prop. For any function $f: S \to T$, there exists a unique function $\tilde{f}: S/\ker f \to T$ such that $\tilde{f} \circ \pi_{\ker f} = f$. Also: (1) $\tilde{f}$ is injective; (2) $\tilde{f}$ is bijective if f is surjective.

Proof. The function exists by the previous proposition, applied in the case $E := \ker f$.

For (1), let $[s], [s'] \in S/\ker f$ be such that $\tilde{f}([s]) = \tilde{f}([s'])$.

Then $f(s) = f(s')$, i.e., $(s, s') \in \ker f$. Thus, $[s] = [s']$.

For (2), let $t \in T$. Pick $s \in S$ with $f(s) = t$. Then also $\tilde{f}([s]) = t$. □

Let $A = (Q, \Sigma, \cdot, i, F)$ be an accessible DFA and $L := \mathcal{L}(A)$.

The **Nerode equivalence** is the kernel of the function $\mathcal{L}_A : Q \twoheadrightarrow Q_L$

Explicitly, it is the equivalence relation $\equiv$ on Q defined by:

for $q, r \in Q$: $q \equiv r$ if, and only if, $\mathcal{L}_A(q) = \mathcal{L}_A(r)$.

Note: $q \equiv r$ if, and only if, for all $w \in \Sigma^*$, $q \cdot w \in F \iff r \cdot w \in F$.

By the preceding **Prop**, we have a bijection $Q/_{\equiv} \xrightarrow{\tilde{\mathcal{L}}_A} Q_L$.

Define the DFA $A/_{\equiv} := \left(Q/_{\equiv}, \Sigma, \cdot, [i], \{[f] \mid f \in F\}\right)$ by

$$[q] \cdot a := [q \cdot a].$$

$\mathcal{L}_A$ is a morphism!

Note that $\cdot$ is **well-defined**: if $q \equiv q'$, then $\mathcal{L}_A(q \cdot a) = \mathcal{L}_A(q) \cdot a = \mathcal{L}_A(q') \cdot a = \mathcal{L}_A(q' \cdot a)$.

We next prove that $A/_{\equiv}$ is in fact **isomorphic** to A_L.

Isomorphism Theorem. The function $\widetilde{\mathcal{L}_A}$ is an isomorphism of automata, i.e., a bijective morphism. (↳ for our notion of morphism! equivalent: ∃ inverse morphism.)

Proof. We already saw $\widetilde{\mathcal{L}_A}$ is a bijection.

1) $\mathcal{L}_A([i]) = \mathcal{L}_A(i)$, the initial state of A_L. (Since $\mathcal{L}_A$ is a morphism, property (1))

2) For any $q \in Q$, $a \in \Sigma$,
$$\begin{aligned}
\widetilde{\mathcal{L}_A}([q]\cdot a) &= \widetilde{\mathcal{L}_A}([q\cdot a]) && \text{(definition } \cdot\text{)}\\
&= \mathcal{L}_A(q\cdot a) && \text{(definition } \widetilde{\mathcal{L}_A}\text{)}\\
&= \mathcal{L}_A(q)\cdot a && (\mathcal{L}_A \text{ morphism 2})\\
&= \widetilde{\mathcal{L}_A}([q])\cdot a && \text{(definition of } \widetilde{\mathcal{L}_A}\text{)}
\end{aligned}$$

3) For any $q \in Q$, $\widetilde{\mathcal{L}_A}([q]) = \mathcal{L}_A(q)$ is final in $A_L \iff q$ is final in A ($\mathcal{L}_A$ morphism 3) □

Side note: The same proof shows: if $f: A \twoheadrightarrow B$ is an automata morphism then the automaton $A/_{\ker(f)}$, defined as above, is isomorphic to B.

Conclusion:

The minimal automaton for L can be obtained from any accessible DFA for L as its quotient under $\equiv$.

Let $A = (Q, \Sigma, \cdot, i, F)$ be an accessible automaton. We will compute the Nerode equivalence.

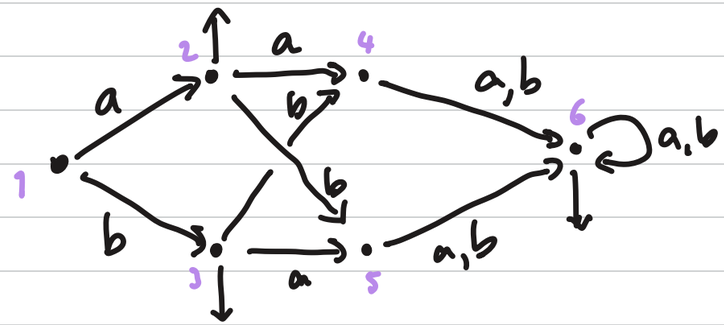
Moore's algorithm or **Table-filling**:

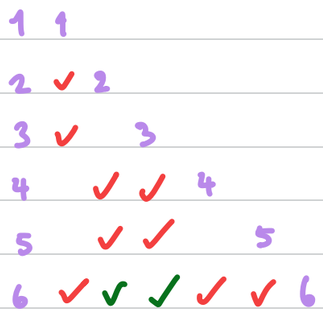
- Initialize an array T of Boolean values indexed by $\binom{Q}{2} := \{ P \subseteq Q \mid \#P = 2 \}$, with
$$T(\{p,q\}) := \quad p \in F \text{ XOR } q \in F .$$
- Repeat while there are changes in T:
$$(*) \left\{ \begin{array}{l} \text{for every pair } \{p,q\} \in \binom{Q}{2} \text{ such that NOT } T(\{p,q\}), \text{ for every } a \in \Sigma: \\ \quad \text{if } p \cdot a \neq q \cdot a \text{ and } T(\{p \cdot a, q \cdot a\}): \text{ set } T(\{p,q\}) \leftarrow \text{true}. \end{array} \right.$$

Proposition The algorithm terminates after $\leq \#\binom{Q}{2}$ repetitions of $(*)$, and takes time polynomial in $\#Q$.

Proof. Each repetition of $(*)$ takes time $\leq \#\binom{Q}{2} \cdot \#\Sigma$, and, if it is not the final repetition, then it must flip at least one value of T from false to true. Thus, there cannot be more than $\#\binom{Q}{2}$ repetitions, and the algorithm takes $\leq \#\binom{Q}{2}^2 \cdot \#\Sigma = O(n^4 k)$ where $n = \#Q$ and $k = \#\Sigma$. (Better bounds are possible.) □

Example.



# reps of (*)	0	1	2	3
T				

terminate.

We see that $q \equiv r$ iff ($q = r$ or NOT $T(\{q,r\})$), that is,

$q \not\equiv r$ iff ($q \neq r$ and $T(\{q,r\})$).

Here, $2 \equiv 3$ and $4 \equiv 5$, as expected.

Proposition. Let T_{out} be the array after Moore's algorithm terminates. Then, for any $p, q \in Q$,

$$p \equiv q \quad \text{if, and only if,} \quad p = q \text{ or NOT } T_{out}(\{p,q\}).$$

Proof. Let M be the number of repetitions of $(*)$ and T_m the array after m reps $(0 \le m \le M)$.

Claim. For any $0 \le m \le M$, any $p \neq q$ in Q, $T_m(\{p,q\})$ if, and only if, $\mathcal{L}_A(p) \cap \Sigma^{\le m} \neq \mathcal{L}_A(q) \cap \Sigma^{\le m}$.

(We will prove this on the next page.) By definition, $T_{out} = T_M$.

"$\Rightarrow$" Suppose $p \equiv q$. If $p \neq q$ and $T_M(\{p,q\})$, then the claim gives $\mathcal{L}_A(p) \neq \mathcal{L}_A(q)$, contradiction.

"$\Leftarrow$" We prove by induction on $w \in \Sigma^*$: for all p, q, if $p \cdot w \in F$ XOR $q \cdot w \in F$, then $T_M(\{p,q\})$. $(*)$

If $|w| \le M$, this follows from the claim. Suppose $w = av$ for $a \in \Sigma$ and $v \in \Sigma^*$, and that $p \cdot w \in F$ XOR $q \cdot w \in F$. Then $(p \cdot a) \cdot v \in F$ XOR $(q \cdot a) \cdot v \in F$. By IH, $T_M(\{p \cdot a, q \cdot a\})$. Since the algorithm terminated, we must also have $T_M(\{p,q\})$, since $p \cdot a \neq q \cdot a$.

Reading $(*)$ contrapositively, we get: if NOT $T_M(\{p,q\})$ then $p \equiv q$.

Clearly, if $p = q$ then $p \equiv q$. So we are done if we prove the claim.

Claim. For any $0 \le m \le M$, any $p \neq q$ in Q, $T_m(\{p,q\})$ if, and only if, $\mathcal{L}_A(p) \cap \Sigma^{\le m} \neq \mathcal{L}_A(q) \cap \Sigma^{\le m}$.

Proof of Claim. $m=0$: by initialization of T. $m \Rightarrow m+1$ $(m < M)$. Suppose $T_{m+1}(\{p,q\})$.

If we already had $T_m(\{p,q\})$, apply IH. Otherwise, pick $a \in \Sigma$ such that $p \cdot a \neq q \cdot a$ and $T_m(\{p \cdot a, q \cdot a\})$.

By IH, pick $v \in \Sigma^{\le m}$ with $(p \cdot a) \cdot v \in F$ XOR $(q \cdot a) \cdot v \in F$. Choose $w := av$.

Conversely, suppose $w \in \Sigma^{\le m+1}$ with $p \cdot w \in F$ XOR $q \cdot w \in F$. If $|w| \le m$, $T_m(\{p,q\})$ by IH, so $T_{m+1}(\{p,q\})$

If $|w| = m+1$, write $w = av$ with $a \in \Sigma$ and $v \in \Sigma^m$. Then $(p \cdot a) \cdot v = p \cdot w$ and $(q \cdot a) \cdot v = q \cdot w$.

In particular, $p \cdot a \neq q \cdot a$, and by IH, $T_m(\{p \cdot a, q \cdot a\})$. So $T_{m+1}(\{p,q\})$. □ □

Corollary. The minimal DFA is obtained by collapsing any pair of states p, q such that NOT $T(\{p,q\})$.

Proof. By the Proposition, this gives $A/\equiv$. By the Isomorphism Theorem, this is the minimal DFA. □

Note that the proof shows what Moore's algorithm is computing: successive refinements of eq. rel's:

Writing $p \equiv_m q \overset{\text{def}}{\iff} \mathcal{L}_A(p) \cap \Sigma^{\le m} = \mathcal{L}_A(q) \cap \Sigma^{\le m}$, the algo. computes until $\equiv_m = \equiv_{m+1}$.

- Moore's algorithm can also be used for other problems than minimization:
  - State equivalence: Given a DFA A and states p, q, does $\mathcal{L}_A(p)$ equal $\mathcal{L}_A(q)$?
  - Automaton equivalence: Given DFA's A and B, is $\mathcal{L}(A) = \mathcal{L}(B)$?

    → In the disjoint union $A \sqcup B$, check whether i_A and i_B are equivalent states.
- An entirely different algorithm is due to Brzozowski, based on the following:

Proposition For any automaton A, the automaton $\mathrm{Reach}(\mathrm{Det}(\mathrm{Reach}(\mathrm{Det}(A^R))^R))$ is minimal for $\mathcal{L}(A)$.

The reverse of an automaton $A = (Q, \Sigma, \delta, I, F)$ is $A^R := (Q, \Sigma, \delta^R, F, I)$ with $\delta^R := \{(u,a,v) \mid (v,a,u) \in \delta\}$.

Less efficient than Moore, since Det is costly: compare minimal automata for $\Sigma^+ a \Sigma^n$ and $\Sigma^n a \Sigma^+$.

However, works on NFA's, and generalizes to other settings. [Bonchi et al. 2014]

F. Bonchi et al. "Algebra-coalgebra duality in Brzozowski's minimization algorithm", TOCL 15(1), 1-29 (2014).

- Yet another algorithm, due to Hopcroft, performs minimization in $O(k \cdot n \log n)$, where $k = \#\Sigma$ and $n = \#Q$. See for example [Berstel et al. 2021].

J. Berstel, L. Boasson, O. Carton, I. Fagnot, "Minimisation of automata", Ch. 10 in Handbook of Automata Theory (2021)