COL 352 Introduction to Automata and Theory of Computation

Nikhil Balaji

Bharti 420
Indian Institute of Technology, Delhi
nbalaji@cse.iitd.ac.in

February 15, 2023

Lecture 12: DFA Minimization

DFA Minimization

- Given a DFA when can we say there are redundant states?

DFA Minimization

- Given a DFA when can we say there are redundant states?
- Every regular language has a unique minimal DFA (upto isomorphism)

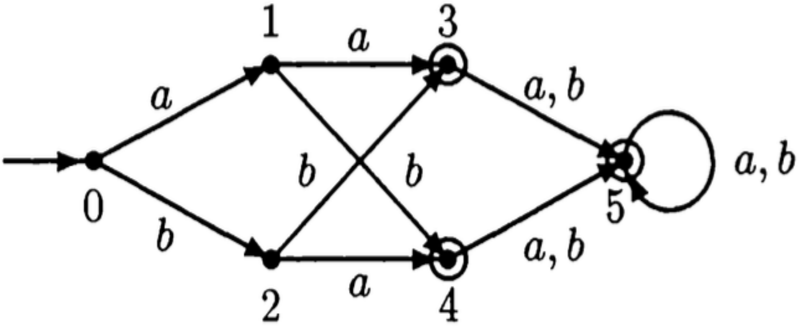
DFA Minimization

- Given a DFA when can we say there are redundant states?
- Every regular language has a unique minimal DFA (upto isomorphism)
- Rough idea: Given $M = (Q, \Sigma, q_0, \delta, F)$
  - Get rid of inaccessible states.

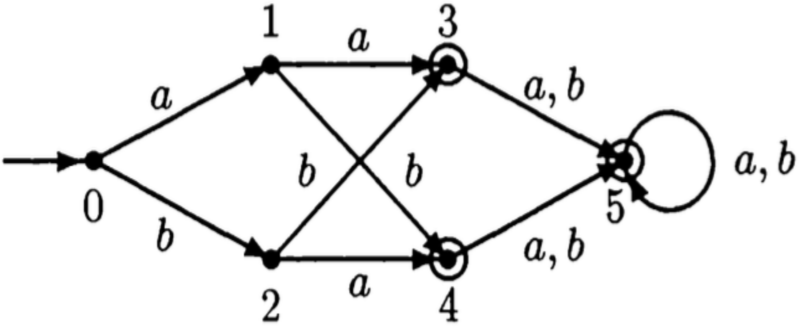
DFA Minimization

- Given a DFA when can we say there are redundant states?
- Every regular language has a unique minimal DFA (upto isomorphism)
- Rough idea: Given $M = (Q, \Sigma, q_0, \delta, F)$
  - Get rid of inaccessible states.
  - Collapse "equivalent" states.

Minimization : Example 1

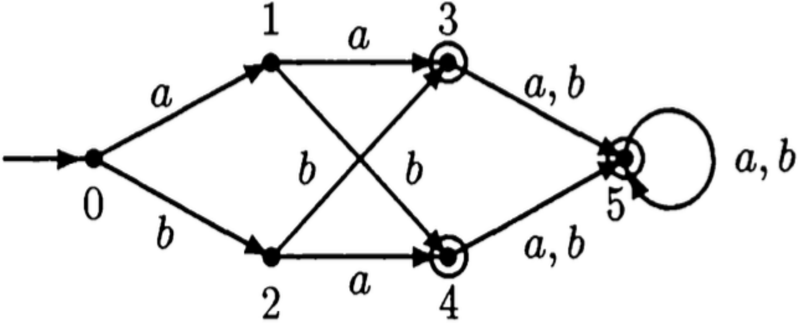


Minimization : Example 1

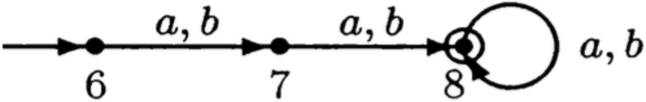


$L = \{\text{Strings of length } \geq 2\}$

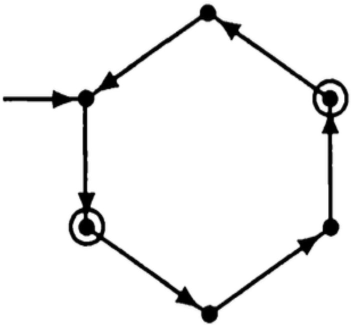
Minimization : Example 1

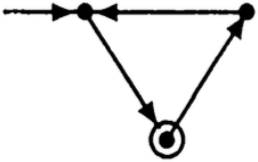


$L = \{\text{Strings of length } \geq 2\}$

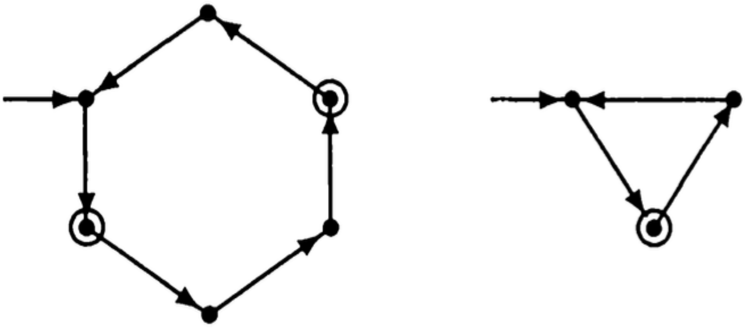


Minimization : Example 2





Minimization : Example 2



$$L = \{a^m \mid m \equiv 1 \pmod 3\}$$

A few pressing questions

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!
   If $\hat{\delta}(s,x) = p \in F$ and $\hat{\delta}(s,y) = q \notin F$, so cannot collapse p and q!
2. If we are collapsing p and q, better also collapse $\delta(p,a)$ and $\delta(q,a)$ for all $a \in \Sigma$.

Inductively, these two imply that we cannot collapse p and q if $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ for some string x.

A few pressing questions

- How do we know two states can be collapsed without changing the language of the DFA?
- Can we do this formally?
- Is there an efficient algorithm for doing this?
- How do we know we can't collapse further?

1. Should not collapse an accept and a reject state!
   If $\hat{\delta}(s,x) = p \in F$ and $\hat{\delta}(s,y) = q \notin F$, so cannot collapse p and q!
2. If we are collapsing p and q, better also collapse $\delta(p,a)$ and $\delta(q,a)$ for all $a \in \Sigma$.

Inductively, these two imply that we cannot collapse p and q if $\hat{\delta}(p,x) \in F$ and $\hat{\delta}(q,x) \notin F$ for some string x. Turns out this is necessary and sufficient to decide if a pair of states can be collapsed or not!

Quotient Construction

Quotient Construction

Define a relation $\equiv$ on the set of states:

Quotient Construction

Define a relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F)$$

Exericse: Show that $\equiv$ is an equivalence relation.

Quotient Construction

Define a relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^* (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F)$$

Exericse: Show that $\equiv$ is an equivalence relation.

$$[p] := \{q \mid q \equiv p\}$$ Equivalence classes

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

$$p \equiv q \iff [p] = [q]$$

Quotient Construction

Define a relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^*(\hat{\delta}(p,x) \in F \iff \hat{\delta}(q,x) \in F)$$

Exericse: Show that $\equiv$ is an equivalence relation.

$$[p] := \{q \mid q \equiv p\}$$ Equivalence classes

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

$$p \equiv q \iff [p] = [q]$$

Why not define an automaton whose states are just these equivalence classes?

Quotient Construction

Define a relation $\equiv$ on the set of states:

$$p \equiv q \iff \forall x \in \Sigma^* (\hat{\delta}(p,x) \in F \iff \hat{\delta}(q,x) \in F)$$

Exericse: Show that $\equiv$ is an equivalence relation.

$$[p] := \{q \mid q \equiv p\} \quad \text{Equivalence classes}$$

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

$$p \equiv q \iff [p] = [q]$$

Why not define an automaton whose states are just these equivalence classes? (This is exactly the "collapsing states" we wanted!)

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

Is the definition of δ' well-defined?

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

Is the definition of δ' well-defined?

Claim: If $p \equiv q$ then is $[\delta(p, a)] = [\delta(q, a)]$

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

Is the definition of δ' well-defined?

Claim: If $p \equiv q$ then is $[\delta(p,a)] = [\delta(q,a)]$ Let $p \equiv q$, $a \in \Sigma$ and $y \in \Sigma^*$

$$\hat{\delta}(\delta(p,a), y) \in F \iff \hat{\delta}(p, ay) \in F$$

Quotient Construction

Quotient Automaton

$M/\equiv := (Q', \Sigma, \delta', s', F')$, where

- $Q' := \{[p] \mid p \in Q\}$
- $\delta'([p], a) := [\delta(p, a)]$
- $s' = [s]$
- $F' = \{[p] \mid p \in F\}$

Is the definition of δ' well-defined?

Claim: If $p \equiv q$ then is $[\delta(p, a)] = [\delta(q, a)]$ Let $p \equiv q$, $a \in \Sigma$ and $y \in \Sigma^*$

$$\begin{aligned}
\hat{\delta}(\delta(p, a), y) \in F &\iff \hat{\delta}(p, ay) \in F \\
&\iff \hat{\delta}(q, ay) \in F \\
&\iff \hat{\delta}(\delta(q, a), y) \in F \quad \square
\end{aligned}$$

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$

Proof. By induction on $|x|$.

Basis: For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\ &= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)} \end{aligned}$$

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

$$\hat{\delta}'([p], xa) = \hat{\delta}'(\hat{\delta}'([p], x), a) \quad \text{(by definition of } \hat{\delta}'\text{)}$$

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$
Proof. By induction on $|x|$.
Basis: For $x = \varepsilon$,

$$\begin{aligned}
\hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\
&= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)}
\end{aligned}$$

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

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

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$
Proof. By induction on $|x|$.
Basis: For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\ &= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)} \end{aligned}$$

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

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

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$
Proof. By induction on $|x|$.
Basis: For $x = \varepsilon$,

$$\begin{aligned}
\hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\
&= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)}
\end{aligned}$$

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

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

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$
Proof. By induction on $|x|$.
Basis: For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\ &= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)} \end{aligned}$$

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

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

Claim: $L(M/\equiv) = L(M)$

Quotient Construction: Correctness

Claim: For all $x \in \Sigma^*$, $\hat{\delta}'([p], x) = [\hat{\delta}(p, x)]$
Proof. By induction on $|x|$.
Basis: For $x = \varepsilon$,

$$\begin{aligned} \hat{\delta}'([p], \varepsilon) &= [p] && \text{(by definition of } \hat{\delta}'\text{)} \\ &= [\hat{\delta}(p, \varepsilon)] && \text{(by definition of } \hat{\delta}\text{)} \end{aligned}$$

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

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

Claim: $L(M/\equiv) = L(M)$ Exercise!

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{aligned} [p] \quad &\approx \quad [q] \\ &\implies \quad \forall x (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \end{aligned}$$

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{aligned} [p] \quad \approx \quad & [q] \\ \implies \quad & \forall x(\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \\ \implies \quad & \forall x([\hat{\delta}(p, x)] \in F' \iff [\hat{\delta}(q, x)] \in F') \end{aligned}$$

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{array}{rcl} [p] & \approx & [q] \\ & \implies & \forall x(\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \\ & \implies & \forall x([\hat{\delta}(p, x)] \in F' \iff [\hat{\delta}(q, x)] \in F') \\ & \implies & \forall x(\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F) \end{array}$$

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{aligned}
[p] \quad &\approx \quad [q] \\
&\implies \quad \forall x(\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \\
&\implies \quad \forall x([\hat{\delta}(p, x)] \in F' \iff [\hat{\delta}(q, x)] \in F') \\
&\implies \quad \forall x(\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F) \\
&\implies \quad p \equiv q
\end{aligned}$$

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{aligned}
[p] &\approx [q] \\
&\implies \forall x (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \\
&\implies \forall x ([\hat{\delta}(p, x)] \in F' \iff [\hat{\delta}(q, x)] \in F') \\
&\implies \forall x (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F) \\
&\implies p \equiv q \\
&\implies [p] = [q]
\end{aligned}$$

Cannot Collapse Further

Define

$$[p] \approx [q] \iff \forall x \in \Sigma^* (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F')$$

Apply this relation on $M/\equiv$.

$$\begin{aligned}
[p] \quad &\approx \quad [q] \\
&\implies \quad \forall x (\hat{\delta}'([p], x) \in F' \iff \hat{\delta}'([q], x) \in F') \\
&\implies \quad \forall x ([\hat{\delta}(p, x)] \in F' \iff [\hat{\delta}(q, x)] \in F') \\
&\implies \quad \forall x (\hat{\delta}(p, x) \in F \iff \hat{\delta}(q, x) \in F) \\
&\implies \quad p \equiv q \\
&\implies \quad [p] = [q]
\end{aligned}$$

The relation $\approx$ is just equality $(=)$!

An algorithm for DFA minimization

Let M be a DFA with no inaccessible states. We will mark (unordered) pairs of states $\{p, q\}$ if we discover a reason why they are not equivalent.

1. Write down a table of pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$, or vice-versa.
3. Repeat until no change occurs: 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, $p \equiv q$ iff $\{p, q\}$ is not marked.

An algorithm for DFA minimization

Let M be a DFA with no inaccessible states. We will mark (unordered) pairs of states $\{p, q\}$ if we discover a reason why they are not equivalent.

1. Write down a table of pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$, or vice-versa.
3. Repeat until no change occurs: 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, $p \equiv q$ iff $\{p, q\}$ is not marked.

Remarks:

- If p and q are marked in Step 2 then definitely they are not equivalent as witnessed by the empty string!

An algorithm for DFA minimization

Let M be a DFA with no inaccessible states. We will mark (unordered) pairs of states $\{p, q\}$ if we discover a reason why they are not equivalent.

1. Write down a table of pairs $\{p, q\}$, initially unmarked.
2. Mark $\{p, q\}$ if $p \in F$ and $q \notin F$, or vice-versa.
3. Repeat until no change occurs: 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, $p \equiv q$ iff $\{p, q\}$ is not marked.

Remarks:

- If p and q are marked in Step 2 then definitely they are not equivalent as witnessed by the empty string!
- Same pair $\{p, q\}$ has to be visited by the algorithm multiple times (status might change because of other ✓ filled in the table)

Finding equivalent states

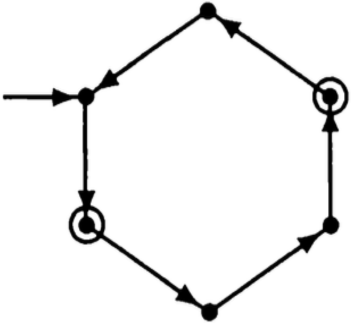
Finding equivalent states

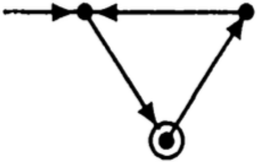
Given: DFA A

Finding equivalent states

Given: DFA A

Output: sets of states of A equivalent to each other





Finding equivalent states

Given: DFA A

Output: sets of states of A equivalent to each other

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Example

	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Example

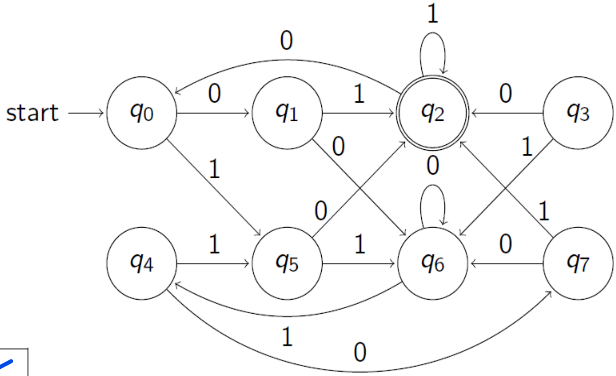
	0	1	2	3	4	5
a	1	2	3	4	5	0

(Red color indicates final states.)

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

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

Example



q_1	✓						
q_2	✓	✓					
q_3	✓	✓	✓				
q_4		✓	✓	✓			
q_5	✓	✓	✓		✓		
q_6	✓	✓	✓	✓	✓	✓	
q_7	✓		✓	✓	✓	✓	✓
D	q_0	q_1	q_2	q_3	q_4	q_5	q_6

Minimization problem

Minimization problem (for fixed Σ)

Given:	DFA A
Output:	sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

Minimization problem

Minimization problem (for fixed Σ)

Given:	DFA A
Output:	sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i, j) = --$
2. For each $1 \leq i < j \leq n$

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$
2. For each $1 \leq i < j \leq n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$

   $T(i,j) \leftarrow \checkmark$
3. Repeat

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$
2. For each $1 \leq i < j \leq n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$
   $T(i,j) \leftarrow \checkmark$
3. Repeat

   $\{$ For each $1 \leq i < j \leq n$

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: sets of states of A equivalent to each other

Algorithm

Let $Q = \{q_1, \ldots, q_n\}$.

1. For each $1 \leq i < j \leq n$, initialize $T(i,j) = --$
2. For each $1 \leq i < j \leq n$

   If $(q_i \in F$ AND $q_j \notin F)$ OR $(q_i \in F$ AND $q_j \notin F)$
   $T(i,j) \leftarrow \checkmark$
3. Repeat

   $\{$ For each $1 \leq i < j \leq n$
   If $\exists a \in \Sigma, T(\delta(q_i,a), \delta(q_j,a)) = \checkmark$
   then $T(i,j) \leftarrow \checkmark$
   $\}$

   Untill T stays unchanged.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

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

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

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

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

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

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

$\mathcal{S} \coloneqq \{\{p, q\} \mid p \in F, q \notin F\}$

- Step 2 marks elements of $\mathcal{S}$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

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

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

- Step 2 marks elements of $\mathcal{S}$.
- Step 3 marks pairs in $\Delta(\{p, q\}, a)$ when $\{p, q\}$ is marked for some $a \in \Sigma$.

Proof of Correctness

Claim: The pair $\{p, q\}$ is not marked by the algorithm if and only if there exists $x \in \Sigma^*$ such that $\hat{\delta}(p, x) \in F$ and $\hat{\delta}(q, x) \notin F$ or vice-versa, i.e., if and only if $p \not\equiv q$.

Proof. By induction (Exercise!).

An automaton view of the algorithm:

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

$\Delta : \mathcal{Q} \to 2^{\mathcal{Q}}$ defined as

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

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

- Step 2 marks elements of $\mathcal{S}$.
- Step 3 marks pairs in $\Delta(\{p, q\}, a)$ when $\{p, q\}$ is marked for some $a \in \Sigma$.
- Claim above says $p \not\equiv q \iff \{p, q\}$ if and only if $\{p, q\}$ is reachable from $\mathcal{S}$.

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: DFA B s.t. $L(A) = L(B)$ and B has the smallest number of states possible for recognizing $L(A)$

Example

	0	1	2	3	4	5
a	1	3	4	5	5	5
b	2	4	3	5	5	5

(Red color indicates final states.)

Minimization problem

Minimization problem (for fixed Σ)

Given: DFA A

Output: DFA B s.t. $L(A) = L(B)$ and B has the smallest number of states possible for recognizing $L(A)$

Example

	0	1	2	3	4	5
a	1	3	4	5	5	5
b	2	4	3	5	5	5

(Red color indicates final states.)

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

DIY!