CS310M: Automata Theory (Minor)

Topic 5: Minimizing DFA

Paritosh Pandya

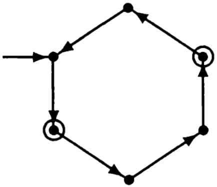
Indian Institute of Technology, Bombay

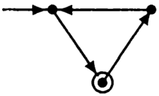
Course URL: https://cse.iitb.ac.in/∼pandya58/CS310M/automata.html

Autumn, 2021

Motivation

There can be multiple DFAs recognizing the same language.



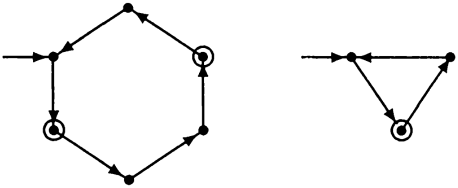


$|x| = 1 \bmod 3$

$(aaa)^*a$

Motivation

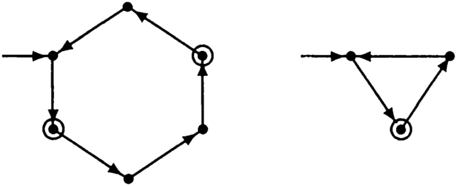
There can be multiple DFAs recognizing the same language.



- Which one is preferable? Why?

Motivation

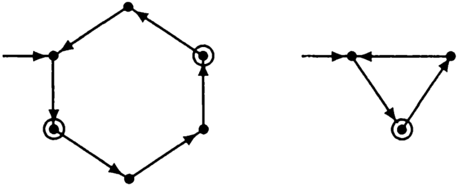
There can be multiple DFAs recognizing the same language.



- Which one is preferable? Why?
- ϵ-NFA to DFA conversion often gives rise to large automaton which may not be "optimal" in size.

Motivation

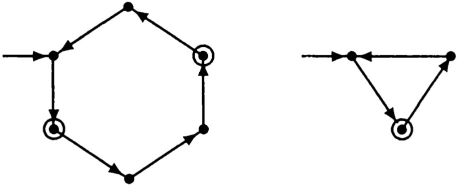
There can be multiple DFAs recognizing the same language.



- Which one is preferable? Why?
- ϵ-NFA to DFA conversion often gives rise to large automaton which may not be "optimal" in size.
- Conversion of Regular Expression to DFA gives rise to large automata which may not be "optimal" in size.

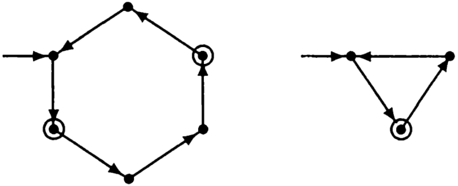
Motivation

There can be multiple DFAs recognizing the same language.



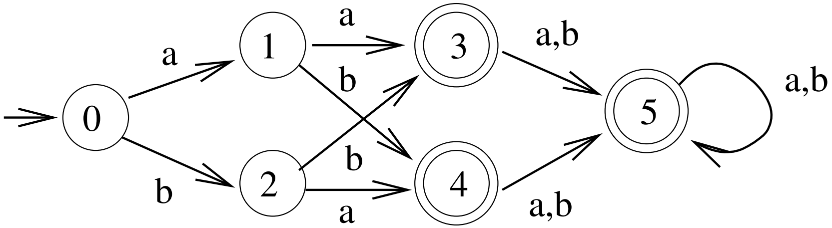
- Which one is preferable? Why?
- ϵ-NFA to DFA conversion often gives rise to large automaton which may not be "optimal" in size.
- Conversion of Regular Expression to DFA gives rise to large automata which may not be "optimal" in size.
- Optimal : smallest in size possible for the same language – minimal sized DFA

There can be multiple DFAs recognizing the same language.

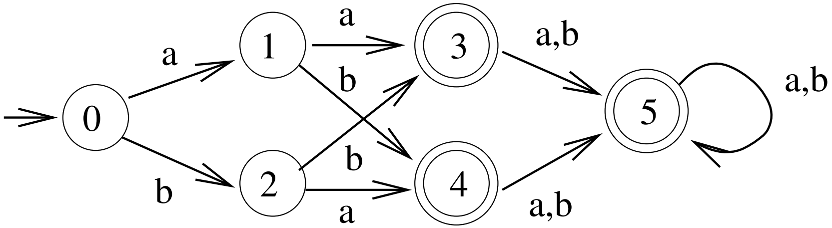


- Which one is preferable? Why?
- ϵ-NFA to DFA conversion often gives rise to large automaton which may not be "optimal" in size.
- Conversion of Regular Expression to DFA gives rise to large automata which may not be "optimal" in size.
- Optimal : smallest in size possible for the same language – minimal sized DFA
- Is there a unique minimal sized DFA? Minimal DFA

DFA Minimization



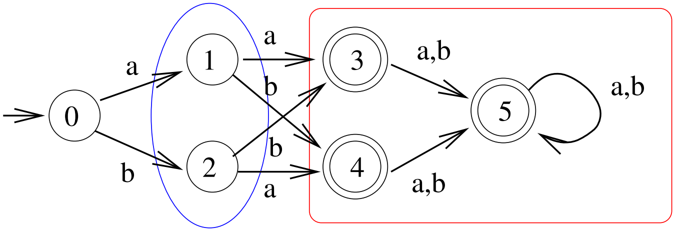
DFA Minimization

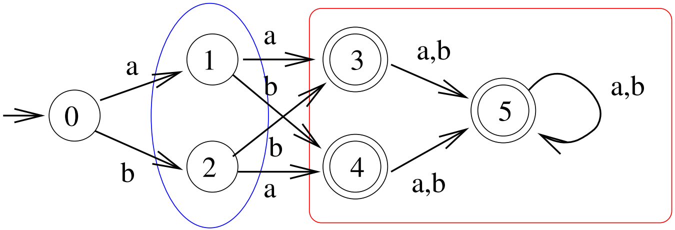


word x separates state p from state q iff

$\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.

DFA Minimization

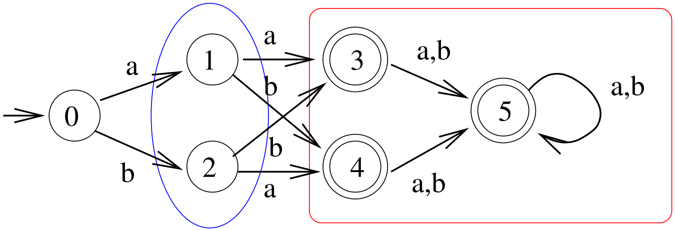




Equivalent States in a DFA

$$p \approx q \stackrel{\text{def}}{=} \forall x \in \Sigma^*. \; (\hat{\delta}(p,x) \in F \Leftrightarrow \hat{\delta}(q,x) \in F)$$

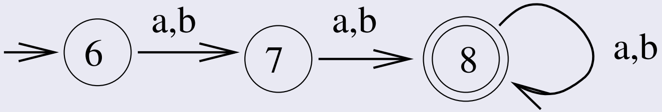
DFA Minimization



Equivalent States in a DFA

$$p \approx q \stackrel{\text{def}}{=} \forall x \in \Sigma^*. \ (\hat{\delta}(p,x) \in F \Leftrightarrow \hat{\delta}(q,x) \in F)$$

Equivalent Automaton



DFA Minimization (2)

$$p \approx q \quad \stackrel{\text{def}}{=} \quad \forall x \in \Sigma^*. \;\; (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$$

DFA Minimization (2)

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

Proposition $\approx$ is an equivalence relation, i.e.

(a) $\forall p.\ p \approx p$, (b) $\forall p, q.\ p \approx q \ \Rightarrow\ q \approx p$

(c) $\forall p, q, r.\ p \approx q \wedge q \approx r \ \Rightarrow\ p \approx r$

(Exercise: Check that above properties are true.)

(S, R) S - set $R \subseteq S \times S$

R is equivalence relation if refl., transitive, symm.

$(\mathbb{N}, R)$ xRy iff $x \bmod 3 = y \bmod 3$

$\{0, 3, 6, 9, \dots\}$ $\{1, 4, 7, \dots\}$ $[1]$

$\{2, 5, 8, \dots\}$ $[2]$

$[0] = [3]$

DFA Minimization (2)

$p \approx q \stackrel{\text{def}}{=} \forall x \in \Sigma^*. \; (\hat{\delta}(p, x) \in F \Leftrightarrow \hat{\delta}(q, x) \in F)$

Proposition $\approx$ is an equivalence relation, i.e.
(a) $\forall p.\ p \approx p$, (b) $\forall p, q.\ p \approx q \Rightarrow q \approx p$
(c) $\forall p, q, r.\ p \approx q \wedge q \approx r \Rightarrow p \approx r$
(Exercise: Check that above properties are true.)

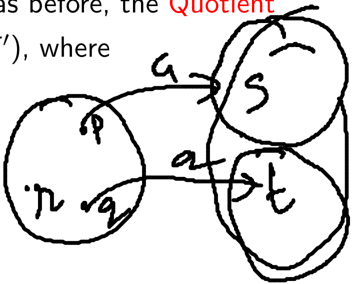
$\approx$ partitions Q into equivalence classes.
Let $[p]$ denote the equivalence class of p.

Example The classes are $\{0\}$, $\{1, 2\}$ and $\{3, 4, 5\}$.

Quotient Automaton

Given DFA $M = (Q, \Sigma, \delta, [q_0], F)$ and $\approx$ as before, the Quotient automaton is $M/\approx \stackrel{\text{def}}{=} (Q', \Sigma, \delta', [q_0], F')$, where

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



Given DFA $M = (Q, \Sigma, \delta, [q_0], F)$ and $\approx$ as before, the Quotient automaton is $M/\approx \stackrel{\text{def}}{=} (Q', \Sigma, \delta', [q_0], F')$, where

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

Well-formedness

Lemma(Congruence) $p \approx q \Rightarrow \forall a \in \Sigma.\ \delta(p, a) \approx \delta(q, a)$.

Lemma $p \in F \Leftrightarrow [p] \in F'$

$p \in F$ and $p \approx q \Rightarrow q \in F$

$p \approx q \Rightarrow \forall a.\ \delta(p,a) \approx \delta(q,a)$

Dow:

$p \approx q \Rightarrow \forall a, \forall y.$

$\hat{\delta}(p,ay) \in F \Leftrightarrow \hat{\delta}(q,ay) \in F$

$\Leftrightarrow \hat{\delta}(\delta(p,a),y) \in F \Leftrightarrow \hat{\delta}(\delta(q,a),y) \in F$

$\Leftrightarrow \delta(p,a) \approx \delta(q,a)$

Quotient Automaton

Given DFA $M = (Q, \Sigma, \delta, [q_0], F)$ and $\approx$ as before, the Quotient automaton is $M/\approx \; \stackrel{\text{def}}{=} \; (Q', \Sigma, \delta', [q_0], F')$, where

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

Well-formedness

Lemma(Congruence) $p \approx q \quad \Rightarrow \quad \forall a \in \Sigma.\; \delta(p, a) \approx \delta(q, a)$.

Lemma $p \in F \quad \Leftrightarrow \quad [p] \in F'$

Lemma $\hat{\delta}'([p], x) \quad = \quad [\hat{\delta}(p, x)]$.

Proof: By structural induction on x,

Base $\hat{\delta}([p], \epsilon) = [p] = [\hat{\delta}(p, \epsilon)]$

Correctness of Quotient Construction

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

Proof:

$x \in L(M/\approx)$ iff

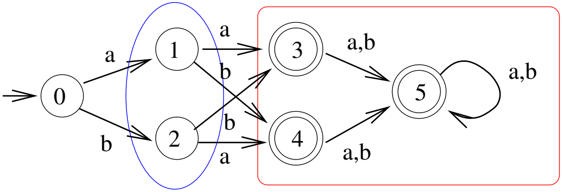
$$\hat{\delta}'([q_0], x) \in F'$$

$$\iff [\hat{\delta}(q_0, x)] \in F'$$

$$\iff \hat{\delta}(q_0, x) \in F$$

$$\iff x \in L(M)$$

Data Structure for ≈ relation



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

Minimization algorithm

Algorithm [Hopcroft 1971]

1. Make pairs table with $(p, q) \in Q \times Q$ and $p \leq q$.
2. Mark (p, q) if $p \in F \wedge q \notin F$ or vice versa.
3. Repeat following steps until no change occurs.
   1. Pick each unmarked state (p, q).
   2. If $(\delta(p, a), \delta(q, a))$ is marked for some $a \in \Sigma$ then mark (p, q).
4. For each pair, $p \approx q$ **iff** (p, q) is unmarked.

Minimization algorithm

Algorithm [Hopcroft 1971]

1. Make pairs table with $(p, q) \in Q \times Q$ and $p \leq q$.
2. Mark (p, q) if $p \in F \wedge q \notin F$ or vice versa.
3. Repeat following steps until no change occurs.
   1. Pick each unmarked state (p, q).
   2. If $(\delta(p, a), \delta(q, a))$ is marked for some $a \in \Sigma$ then mark (p, q).
4. For each pair, $p \approx q$ **iff** (p, q) is unmarked.

Termination: In each pass at least one new pair must get marked.

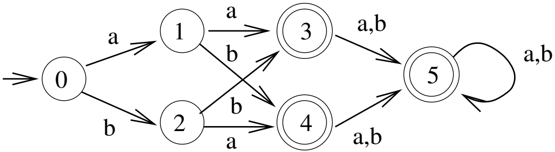
Theorem (p, q) is marked
iff $\exists x \in \Sigma^*.\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.
iff $p \not\approx q$.

Example

Example

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

Example



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

Proof of Theorem (1)

We prove (p, q) is marked implies
$\exists x \in \Sigma^*.\hat{\delta}(p, x) \in F \wedge \hat{\delta}(q, x) \notin F$ or vice versa.

Proof of Theorem (2)

We prove $\exists x \in \Sigma^*.\hat{\delta}(p,x) \in F \wedge \hat{\delta}(q,x) \notin F$ or vice versa.
implies (p,q) is marked