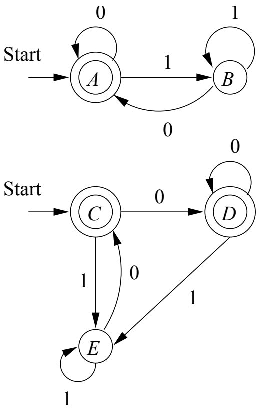
Regular language equivalence

Let L and M be regular languages (specified by means of some representation)

To test $L \stackrel{?}{=} M$:

- convert L and M representations into DFAs
- construct the union DFA (never mind if there are two start states)
- apply state equivalence algorithm
- if the two start states are distinguishable, then $L \neq M$, otherwise $L = M$

Example



This is the union
↓
Consider all states for the state equivalence algorithm

Example

The state equivalence algorithm produces the table

A and C are equivalent, so the two languages are the same

B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

remember the base case:
$q_1 \not\equiv q_2$ if $q_1 \in F$, $q_2 \notin F$
or otherwise

We have $A \equiv C$, thus the two DFAs are equivalent

Both DFAs recognize language $L(\epsilon + (\mathbf{0} + \mathbf{1})^* \mathbf{0})$

DFA minimization

Important application of the equivalence algorithm : given DFA as input, produces equivalent DFA with **minimum number of states**

Minimal DFA is **unique**, up to renaming of the states

Idea :

- eliminate states that are unreachable from the initial state
- merge equivalent states into an individual state

Example

B	x						
C	x	x					
D	x	x	x				
E		x	x	x			
F	x	x	x		x		
G	x	x	x	x	x	x	
H	x		x	x	x	x	x
	A	B	C	D	E	F	G

State partition based on the equivalence relation :
$\{\{A, E\}, \{B, H\}, \{C\}, \{D, F\}, \{G\}\}$

Example

B	x			
C		x		
D		x		
E	x		x	x
	A	B	C	D

$\{\{A, C, D\}, \{B, E\}\}$

State partition based on the equivalence relation :
$\{\{A, C, D\}, \{B, E\}\}$ → only two states instead of 5

Transitivity

Theorem If $p \equiv q$ and $q \equiv r$, then $p \equiv r$

Proof

Suppose to the contrary that $p \not\equiv r$

- Then $\exists w$ such that $\hat{\delta}(p, w) \in F$ and $\hat{\delta}(r, w) \notin F$ or the other way around
- *Case* 1 : $\hat{\delta}(q, w)$ is accepting. Then $q \not\equiv r$
- *Case* 2 : $\hat{\delta}(q, w)$ is not accepting. Then $p \not\equiv q$

Therefore it must be that $p \equiv r$ $\square$

Relation $\equiv$ is reflexive, symmetric and transitive : thus $\equiv$ is an **equivalence relation**

We can talk about equivalence classes

DFA minimization

To minimize DFA $A = (Q, \Sigma, \delta, q_0, F)$, construct DFA $B = (Q/_{\equiv}, \Sigma, \gamma, q_0/_{\equiv}, F/_{\equiv})$, where

Merge all states equivalent to a final state

merge states equivalent to the start state

merge equivalent states

- elements of $Q/_{\equiv}$ are the equivalence classes of $\equiv$
- elements of $F/_{\equiv}$ are the equivalence classes of $\equiv$ composed by states from F
- $q_0/_{\equiv}$ is the set of states that are equivalent to q_0
- $\gamma(p/_{\equiv}, a) = \delta(p, a)/_{\equiv}$

new transition function

unite all states equivalent to the $\delta(p,a)$ state

DFA minimization

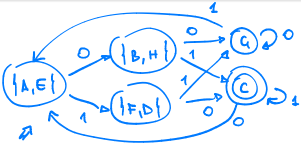
In order for B to be well defined we have to show that

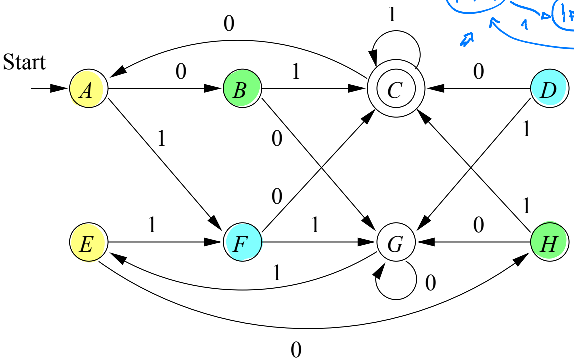
$$\text{If } p \equiv q \text{ then } \delta(p, a) \equiv \delta(q, a)$$

If $\delta(p, a) \not\equiv \delta(q, a)$, then the equivalence algorithm would conclude that $p \not\equiv q$. Thus B is well defined

Example

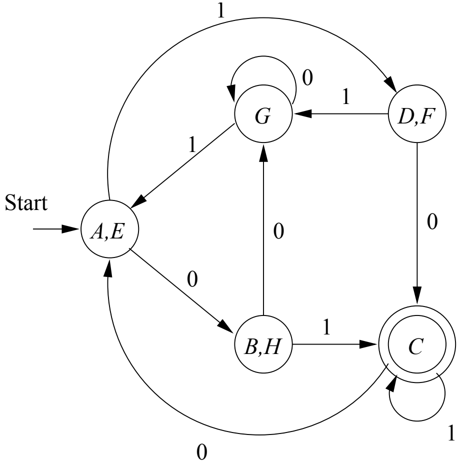
Minimize





Example

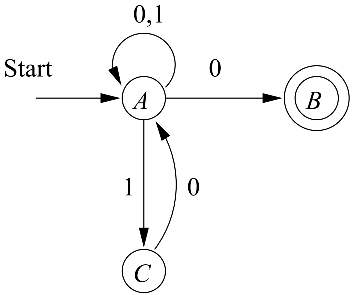
We obtain



Automata minimization

We **cannot** apply the algorithm to NFAs

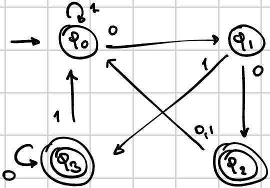
Example : To minimize



we simply remove state C. However, $A \not\equiv C$

Es 22/1/19

Given an FA, convert it to a RE

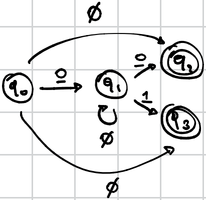


Remember!

Formula: $R_{i,j} + Q_{i,k} S_{k,k}^{*} Q_{k,j}$

$(R + SU^{*}T)^{*}SU^{*}$

1) remove q_1

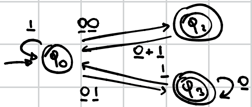


to reach q_2: $\emptyset + \underline{0}\,\emptyset^{*}\underline{0} = \underline{00}$

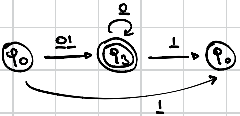
to reach q_3: $\emptyset + \underline{0}\,\emptyset^{*}\underline{1} = \underline{01}$

$\Downarrow$

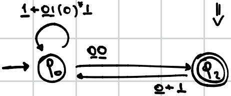
A



2.i) remove q_3 from A



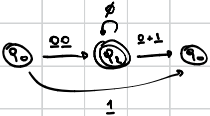
$1 + 01(0)^*1$



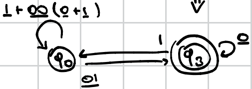
$= ((1 + 01(0)^*1) + 00\emptyset^*(0+1))^* 00\emptyset^*$

$= (1 + 01 0^*1 + 00(0+1))^* 00$

2.ii) remove q_2 from A



$1 + 00\emptyset^*(0+1)$



$= (1 + 00(0+1) + 01 0^*1)^* 01 0^*$

Final : $(1 + 01 0^*1 + 00(0+1))^* 00 + (1 + 00(0+1) + 01 0^*1)^* 01 0^*$

Es) $L_1 = \{ w \mid w \in \{ab\}^*, \#_b(w) \leq \#_a(w) \leq 2 * \#_b(w) \}$

Is L_1 regular?

Proof that it's not regular with the pumping lemma:

consider $a^{2n}b^n$: $\underbrace{\overset{n}{\overbrace{a \ldots}}}_{xy} \; a b \ldots b$

for $k > 1$, $\#_a(xy^kz) = \underbrace{\#_a(xyz)}_{=2n} + \underbrace{\#_a(y)}_{\geq 1,\text{ since } y \neq \varepsilon} \cdot \underbrace{(k-1)}_{>0} > 2n = 2\#_b(w)$

$\Downarrow$

$xy^kz \notin L_1 \Rightarrow$ not a regular language

$$L_2 = \{ w \mid w \in \{a,b\}^*, \ \#_a(w) \neq \#_b(w) \ \underline{\text{and}} \ \#_a(w) \neq 2\#_b(w) \}$$

$$\overline{L_2} = \{ w \mid w \in \{a,b\}^*, \ \#_a(w) = \#_b(w) \ \underline{\text{or}} \ \#_a(w) = 2\#_b(w) \}$$

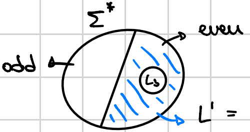
consider $w = a^n b^n = \underbrace{\overbrace{a \ldots a}^{n}}_{xy} b \ldots b$

for $k=0$, $\#_a(xy^kz) = \overset{n}{\#_a(xyz)} - \overset{1}{(k+1)}\overset{\geq 1}{\#_a(y)} < n = \#_b(w)$

$\Downarrow$

$xy^kz \notin \overline{L_2}$ so $\overline{L_2}$ is not regular $\Rightarrow$ L_2 is not regular

$L_3 = \{ ww' \mid w, w' \in \{a,b\}^*, \ |w| = |w'|, \ w \neq w' \}$



$L' = \{ ww \mid w \in \{a,b\} \}$

regular

$\Sigma^* \setminus \{odd\} = \{even\} = L''$

supposing L_3 is regular

$L'' \setminus L_3 = L'$ ⟵ should be regular, but L' is not

$w = a^n b \rightarrow ww = \underbrace{a^n b}_{w} \underbrace{a^n b}_{w} = \underbrace{\overbrace{a \ldots ab}^{n}}_{xy} a \ldots ab$

$\neq n$ since $y \neq \varepsilon$

if $k = 0$, $\quad xy^k z = a^{n-|y|} b a^n b \Rightarrow xy^k z \notin L'$

L' is not regular $\Rightarrow$ **L_3 is not regular**