CSCE 355
2/25/2026

DFA minimization examples

Recall: A DFA is <u>sane</u> if every state is reachable from the start state.

Minimization algo:

Given a DFA $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ as input, output the equivalent DFA with the fewest possible states (unique minimum equivalent DFA).

1. Remove any unreachable states from A (A is sane now)
   [BFS from start state to mark reachable states; remove any unmarked states left.]
2. Merge sets of indistinguishable states (into single states).
   [Mark pairs of states that ~~and~~ <u>are</u> distinguishable, then any pairs left over are indistinguishable.]

Recall: states $p, q \in Q$ are <u>dist.</u> if $\exists w \in \Sigma^*$ such that

$$p \xrightarrow{w} p' \qquad q \xrightarrow{w} q'$$

one of p' & q' is accepting and the other is rejecting.

[w <u>distinguishes</u> p from q.]

Recursive rules for dist'ability:

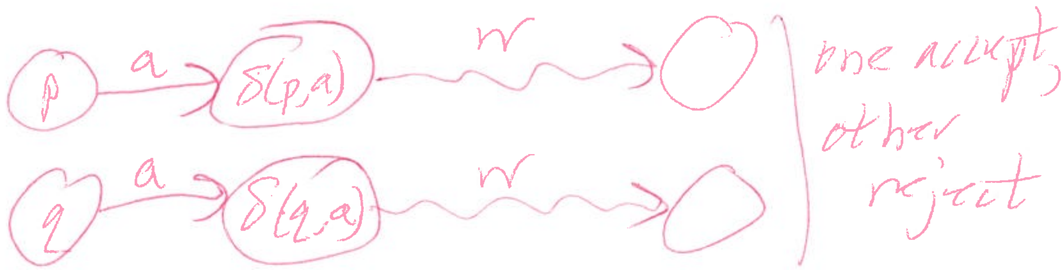
States p & q are distinguishable if either

Base case → 1. One of p & q is accepting & the other is rejecting ($w := \varepsilon$ distinguishes p from q)

inductive case → 2. If there exists a symbole $a \in \Sigma$ such that $\delta(p,a)$ and $\delta(q,a)$ are distinguishable, then p & q are distinguishable.

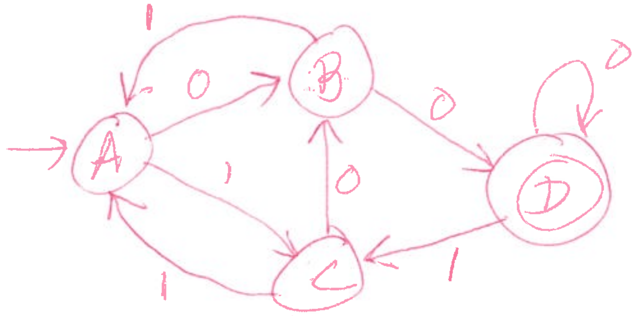
Apply (1) first, then apply (2.) repeatedly until can't mark any additional pairs as distinguishable.

For (2):



If w dist. $\delta(p,a)$ from $\delta(q,a)$, then aw dist. p from q.

Example:



B			
C			
D			
	A	B	C

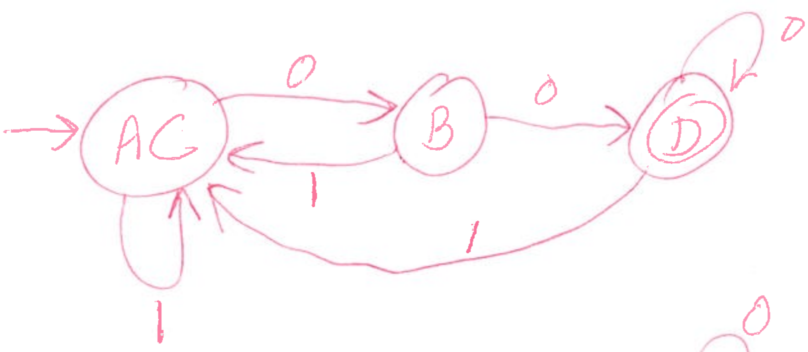
Step 1:

B			
C			
D	X	X	X
	A	B	C

Step 2:

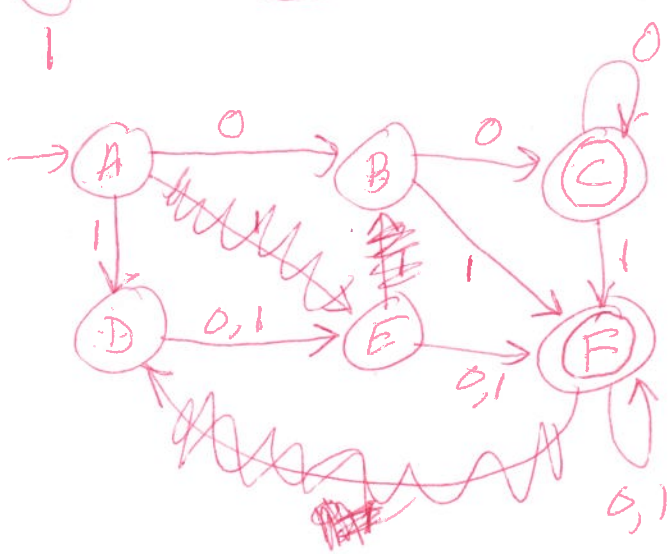
B	X		
C		X	
D	X	X	X
	A	B	C

Can't mark (A, C), so A & C are indist. (only pair). Now merge them:



The min equiv DFA

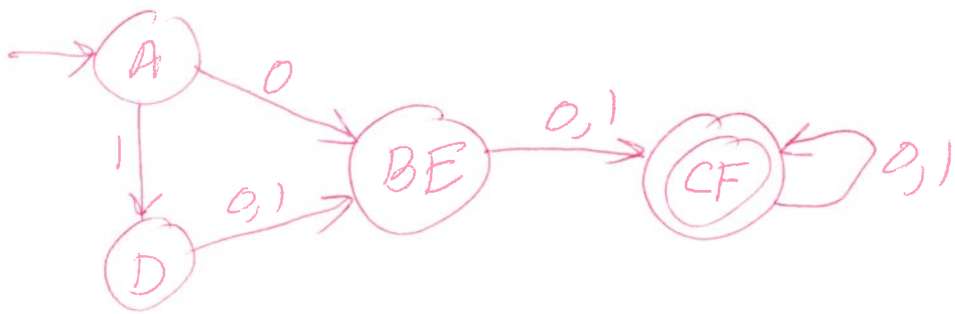
Example:



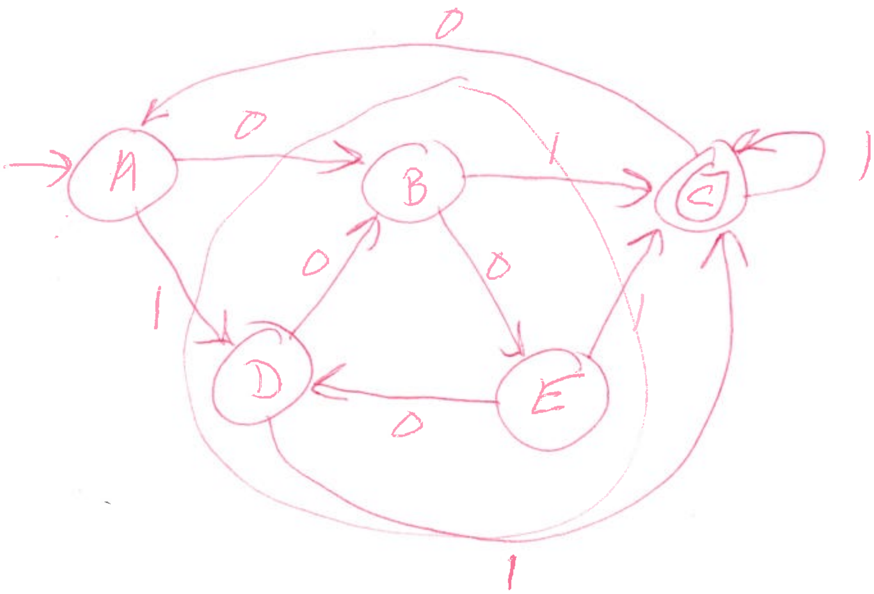
B	X				
C	X	X			
D	X	X	X		
E	X		X	X	
F	X	X		X	X
	A	B	C	D	E

Done

Merge B&E and C&F:

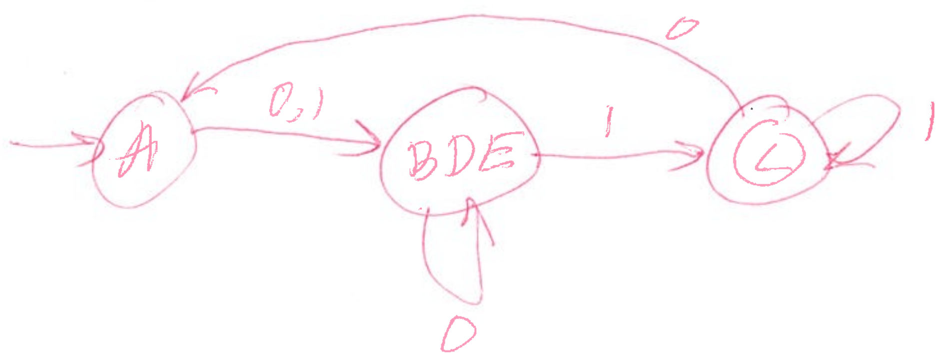


Example



	A	B	C	D
B	X			
C	X	X		
D	X		X	
E	X		X	

Merge B, D, & E into 1 state:



Context-free languages (CFLs)

All regular langs are CFLs, but not conversely.

[good at describing syntax with nested structures]

Def: A context-free grammar ~~(CFG)~~ (CFG) is a tuple $\langle V, \Sigma, S, P \rangle$ where

V & Σ are finite (disjoint) sets

$V \cap \Sigma = \emptyset$

~~S~~ ~~V~~ elements of V are ~~are~~ called variables, nonterminals, or syntactic categories

elements of Σ are called terminals or tokens

$S \in V$ called the start symbol

P is a finite set of productions (or rules) of the form,

$$A \to \alpha$$

where $A \in V$ and $\alpha \in \underbrace{(V \cup \Sigma)}_{\text{grammar symbols}}^*$

A is called the head of the production

and α is the body " " "

Def: Given $G = \langle V, \Sigma, S, P \rangle$ CFG,

A <u>derivation of G</u> is a sequence

$$\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_n \quad (\text{some } n \geq 0)$$

where α_i is a string of grammar symbols for every i;

- ~~$\alpha_0 = S$~~
- $\alpha_0 = S$
- $\alpha_n \in \Sigma^*$ (only terminal symbols)
- for every $0 \leq i < n$

  α_{i+1} results from α_i by replacing a variable occurrence (say A) in α_i by the body of some production whose head is A.

Say that α_n is <u>derivable</u> by this derivation.

Example: Productions are

$$P = \begin{cases} S \to 0S1 \\ S \to \varepsilon \end{cases} \qquad \begin{aligned} \Sigma &= \{0,1\} \\ V &= \{S\} \end{aligned}$$

A derivation:

$$\underline{S} \Rightarrow 0\underline{S}1 \Rightarrow 00\underline{S}11 \Rightarrow 000\underline{S}111 \Rightarrow 000111$$

<u>Def</u>: $L(G) = \{ w \in \Sigma^* : w \text{ is derivable from } G \}$

In this case, $L(G) = \{ 0^n 1^n : n \geq 0 \}$