Theory of Computation

L.EIC, 2nd Year

João M. P. Cardoso
Email: jmpc@acm.org

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

DEI DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Outline

- Regular Languages and Non-Regular Languages
- Pumping Lemma for Regular Languages
- Properties of the Regular Languages
- Equivalence of States in DFAs and DFA minimization

Regular Languages (RLs)

- The Regular Languages are the ones that can be expressed by DFAs, NFAs, ε-NFAs or REs
- Examples of RLs:
  - L1=L(0*1*) is a regular language
  - L2=$\{0^n1^m \mid n,m \geq 0\}$ (note that both L1 and L2 represent the same language)
- What about L3=$\{0^n1^n \mid n \geq 0\}$? Is L3 a regular language? *(i.e., could you propose a DFA, NFA, ε-NFA or RE for L3?)*

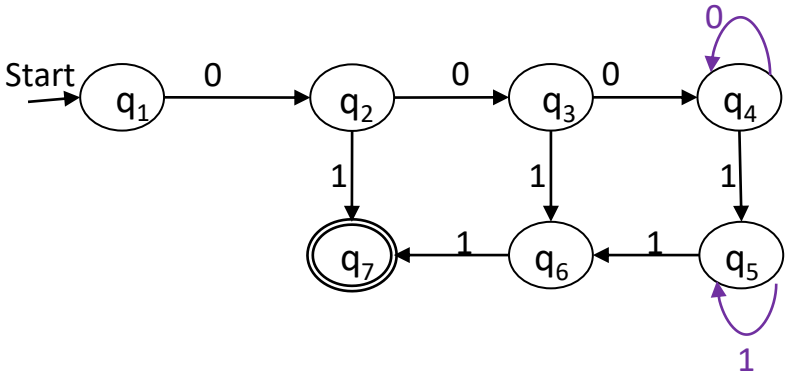
Non-Regular Languages

- $L_{01} = \{0^n 1^n \mid n \geq 1\}$
  - Sequences of 0s followed by sequences of 1s with the same number of 0s and 1s
- This language is not regular
  - If it was, there would exist a DFA for L_{01}
  - Supposing that the DFA has k states; after consuming k+1 prefixes in the input ε, 0, 00, …, 0^k, the DFA is in any state
- **Pigeonhole Principle**: if there are k+1 pigeons and k pigeonholes, there must be 2 pigeons in the same pigeonhole
  - Thus, after k+1 state transitions, it must have passed twice in the same state

Non-Regular Languages (cont.)

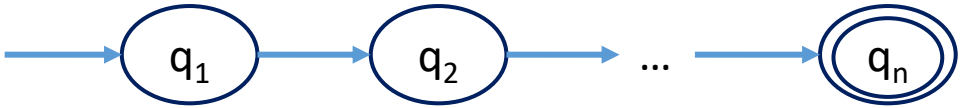
$L_{01} = \{0^n 1^n \mid n \geq 1\}$

- If after reading i or j 0s, start reading 1s then it must accept a string that finishes after i 1s in the first case, but not in the second and vice-versa
- There is not such a DFA for L_{01} and thus the language is not regular



Regular Languages

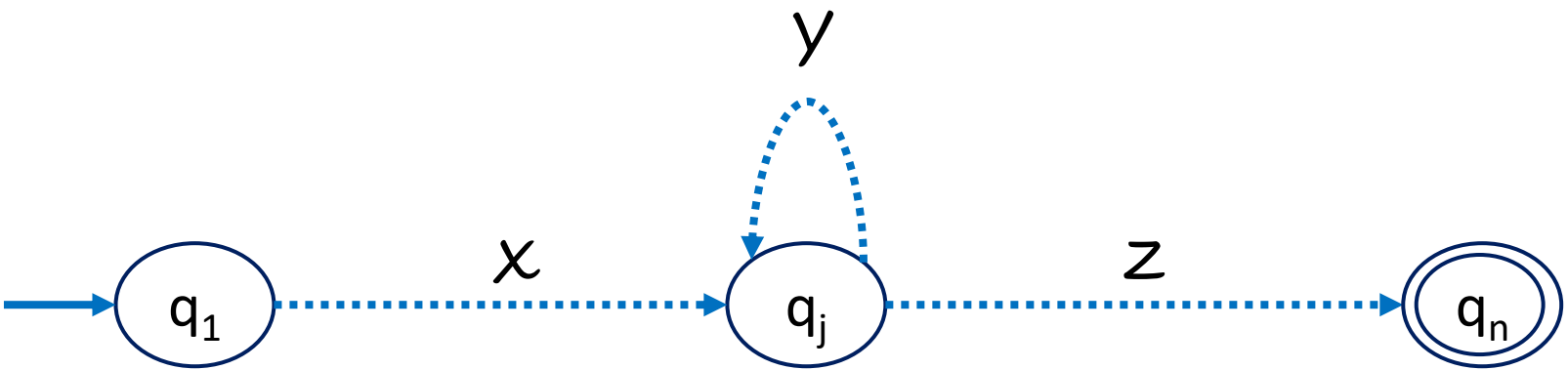
- Consider a DFA with *n* states
  - passing only 1× in each state, the length of any string *w* recognized by the DFA is: $|w| \leq n-1$
  - the maximum length passing only 1× in each state is *n-1* and corresponds to:



  - To recognize strings *w* with $|w| \geq n$, at least one state must be passed more than 1× => RL is an infinite language

Regular Languages

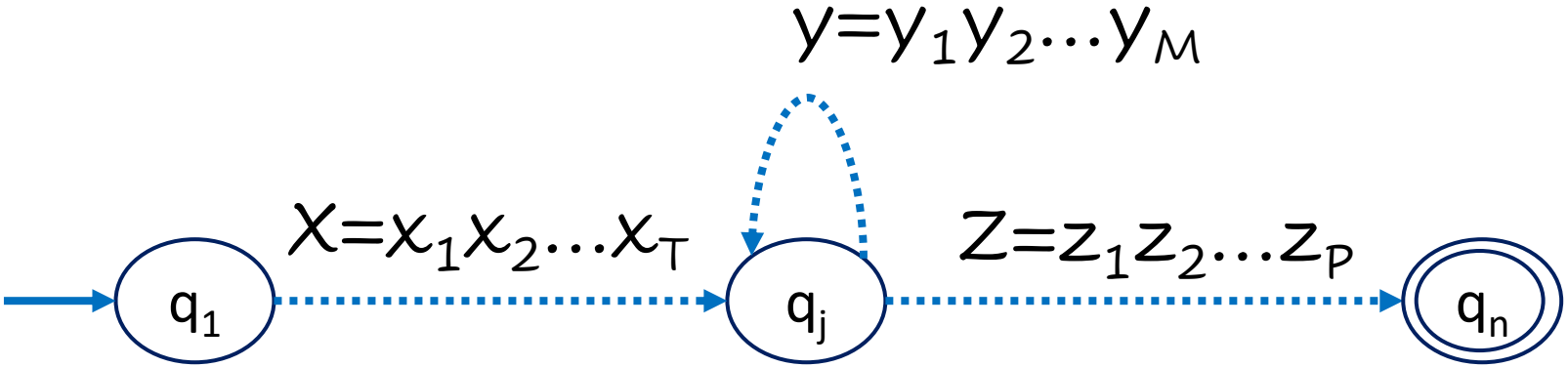
- Infinite RLs must have at least one state repeating:



Any $|w| \geq n$, $w=xy^*z$
$|x|$ and $|z|$ can be 0
$|y| \neq 0$

Regular Languages

- Infinite RLs must have at least one state repeating



$|xy| \leq n$

$xy^k z$ and $k \geq 0$

The Pumping Lemma

- Given an infinite regular language L
- there exists an integer n (critical length)
- for any string $w \in L$ with length $|w| \geq n$
- we can write $w = x\ y\ z$
- with $|xy| \leq n$ and $|y| \geq 1$ $y \neq \varepsilon$
- such that: $xy^k z \in L$ $k = 0, 1, 2, \ldots$ $k \geq 0$

The Pumping Lemma

- Given an infinite regular language L

I.e., we can always find a non-empty string y, near the beginning of w, that can be repeated (pumped) an arbitrary number of times or removed, producing strings that must belong to the language

- such that: $xy^kz \in L \quad k = 0, 1, 2, \ldots$ $k \geq 0$

Facts

- Finite languages are all regular languages (RLs)
- The *Pumping Lemma* can be applied to show that certain infinite languages are not regular
- But it cannot be used to show that a language is regular!
- The *Pumping Lemma* is a necessary but not sufficient condition for a language to be regular! (i.e., there are non-regular languages satisfying the Pumping Lemma)

Proofs Using the Pumping Lemma

- Proof by contradiction
- Hypothesis: we assume that the language is regular
- Then, if it does not meet the *Pumping Lemma,* we conclude that the language is not regular

Proof Example 1

- Prove that $L_{01} = \{0^m 1^m \mid m \geq 1\}$ is not a regular language
- Let's assume that L_{01} is a regular language. If it is a regular language, then it must satisfy the Pumping Lemma for regular languages
- …

The Pumping Lemma

- Given an infinite regular language L
- there exists an integer n (critical length)
- for any string $w \in L$ with length $|w| \geq n$
- we can write $w = x\ y\ z$
- with $|xy| \leq n$ and $|y| \geq 1$ $y \neq \varepsilon$
- such that: $xy^k z \in L$ $k = 0, 1, 2, \ldots$ $k \geq 0$

Proof Example 1

- Prove that $L_{01} = \{0^m 1^m \mid m \geq 1\}$ is not a regular language
- Let's assume that L_{01} is a regular language. If it is a regular language, then it must satisfy the Pumping Lemma for regular languages
- Let's pick $w = 0^n 1^n$
  - $|w| = 2n$ which is $\geq n$
  - $w = xyz$
    - Since $|xy| \leq n$ then xy must only consist of 0's and as $|y| \geq 1$ then y consist of 1 or more 0's
  - For k=0, $xy^k z = xz$ which consists of a number of 0's lower than the number of 1's, and so it does not $\in L$
  - L_{01} does not satisfy the Pumping Lemma for regular languages (contradicting the hypothesis)
  - Thus, L_{01} is not a regular language. *qed*

Proof Example 2

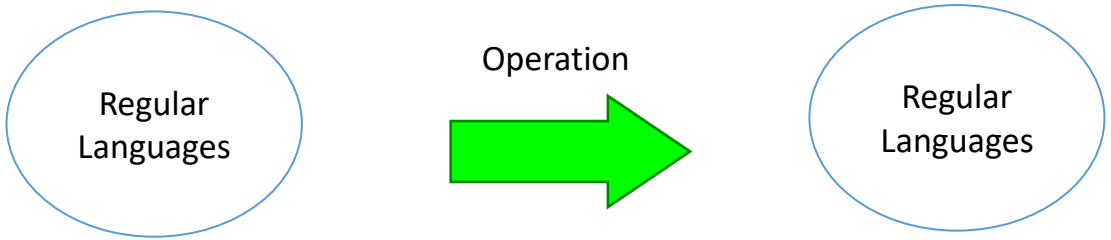
- Prove that L_{pr}, the language of all the chains of 1s such that their length is a prime number, is not a regular language
  - If L_{pr} was a regular language there would be an *n* meeting the conditions of the pumping lemma.
  - Let's select a prime number $p \geq n+2$ (there is an infinity of prime numbers) and $w=1^p$
  - By the pumping lemma, $w=xyz$ with $y \neq \varepsilon$ and $|xy| \leq n$
  - Given $|y| = m$, $|xz| = p-m$. The string $xy^{p-m}z$ must belong to L_{pr}, if the language is regular
  - But $|xy^{p-m}z| = |xz| + (p-m)|y| = p-m + (p-m)m = (m+1)(p-m)$
  - $|xy^{p-m}z|$ includes two factors $\neq 1$, thus it is not a prime number and is not in L_{pr} contradicting the hypothesis
    - $m>0$, thus $m+1>1$;
    - $p-m > 1$, because $p \geq n+2$ and $m=|y| \leq |xy| \leq n$
  - Thus, L_{pr} is not a regular language. *qed*

Exercise 1

- Using the pumping lemma for regular languages, show that $\{0^i10^i \mid i \geq 1\}$ is not a regular language.

Properties of Regular Languages

Closure Properties



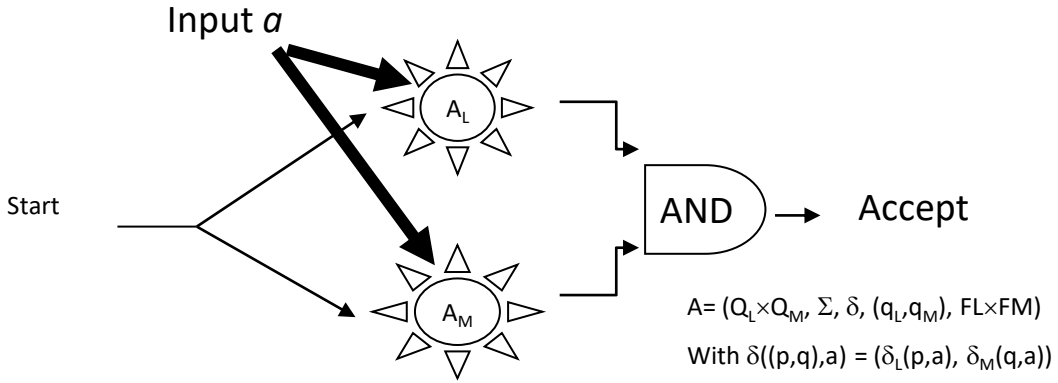
- The class of regular languages is closed to the operations:
  - Union, Intersection, Complement
  - Difference
  - Reverse
  - Closure (*) and Concatenation
  - Homomorphism and Inverse Homomorphism

Boolean Operations

- Theorem: if L and M are regular languages, $L \cup M$ is regular as well
  - If R and S are REs such that L=L(R) and M=L(S), then, by the definition of +, L(R+S) is $L \cup M$
- Theorem: if L is an RL, then the complement of L for Σ^* is a regular language as well
  1. From the RE of L, build an ε-NFA
  2. Convert it to a DFA by the method of subset construction
  3. Complement the states of the DFA
  4. Convert the complemented DFA to an RE, e.g., by the state elimination method

Boolean Operations (cont.)

- Theorem: if L and M are regular languages, L ∩ M is a regular language as well
  - Consider the two FAs in parallel, with the same input and consider the pair of states of the FAs intersection; accept only if both accept.



  - Product of the DFAs and then select as final states the ones consisting of only final states

Reverse

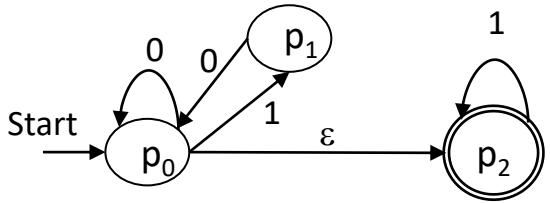
- Consider $w=a_1a_2\ldots a_n$. The reverse of w, w^R, is $a_n\ldots a_2a_1$
- The reverse of a language L^R is the language that consists of the reverse of the string of L
- Theorem: if L is a regular language, then L^R is a regular language as well
  - If L is the language of an FA A, L(A)
    1. Revert all the edges of the transition diagram of A
    2. Mark the initial state of A as the unique accept state
    3. Include a new initial state p_0 with transitions ε to all the accept states of A
  - The result is an FA that represents A reverted and that accepts w^R iff A accepts w

Proof (Reverse)

- Assume L defined by the E RE. Lets show that there exist other RE E^R such that $L(E^R) = (L(E))^R$
- Proof by structural induction in E
  - Basis: if E is ε, $\varnothing$ or $a \in \Sigma$, E^R is the same of E.
  - Induction: three cases
    - $E = E_1 + E_2$; $E^R = E_1^R + E_2^R$; reverse of the union is the union of the reverses
    - $E = E_1 E_2$; $E^R = E_2^R E_1^R$; reverse of the concatenation is the concatenation of the reverses by the inverse order
      - Example: $w = w_1 w_2 = 1101\ 0010$; $w^R = 0100\ 1011 = w_2^R\ w_1^R$
    - $E = E_1^*$; $E^R = (E_1^R)^*$; each w in L is $w = w_1 w_2 \ldots w_n$, with w_i in L, but $w^R = w_n^R w_{n-1}^R \ldots w_1^R$. Each w_i^R is in $L(E^R)$ then $w \in L((E^R)^*)$

Example

- Consider the language of the strings that after the first pair of adjacent 1s does not include 0s
  - Obtain an RE defining the language
    - (0+10)*1*
  - Show an FA for the same language.

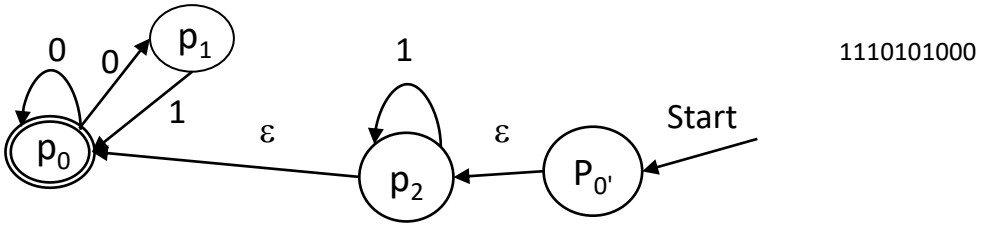


0001010111

Example (cont.)

- Repeat the steps for the reverse language

$$w^R = ((0+10)^*1^*)^R = (1^*)^R((0+10)^*)^R = (1^R)^*((0+10)^R)^* = 1^*(0^R+(10)^R)^* = 1^*(0+01)^*$$



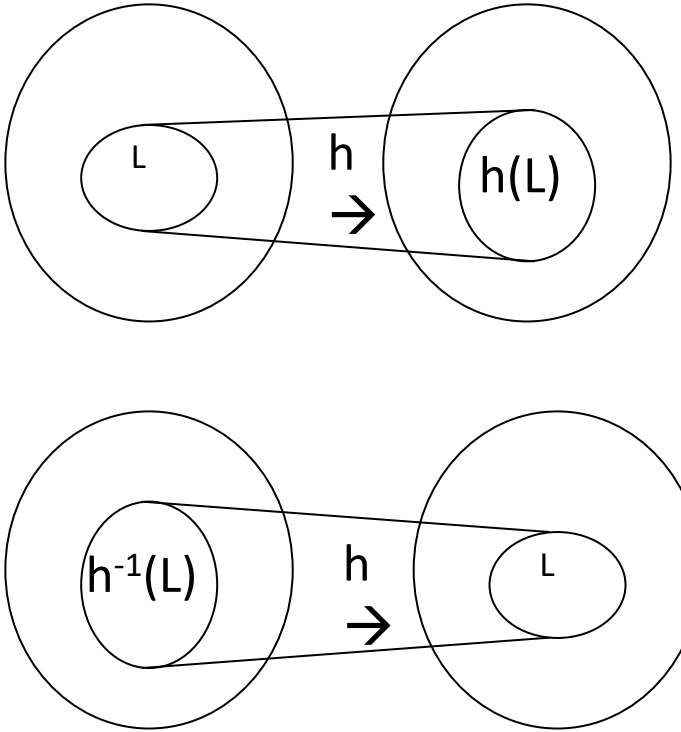
- i.e., may start with a sequence of 1s and, after the first 0, there are no pairs of adjacent 1s.

Homomorphism

- Function of symbols in strings
  - Example: h(0)=ab and h(1)=ε applied to 0011 results in abab
- Chain: if w= $a_1a_2…a_n$ then h(w)=h(a_1)h(a_2)…h(a_n)
- Language: h(L) = {h(w) | w ∈ L}
- RE: L(h(R)) = h(L(R))
  - Example: L = L(10*1)
  - h(L(10*1)) = L(h(10*1)) = L((ab)*)
- Theorem: if L is a regular language over Σ and h a homomorphism in Σ, then h(L) is a regular language as well

Homomorphism Inverse

- The idea is that the homomorphism can be applied in the inverse order
- The regular property of the language is maintained
- Σ={a,b} T={0,1}
- L1 = L((00+1)*)
- h: h(a) = 01
  h(b) = 10
- $h^{-1}(L1)$= L((ba)*)



Decision Properties for RLs

- How to study a language?
  - Infinite → unfeasible to perform exhaustive analysis
  - Finite representation: DFA, NFA, ε-NFA, RE → RL
- To answer to a question about the language
  - Find an algorithm which answers yes or no
  - In many cases there are algorithms for RLs, but not for non-regular languages, even if there are finite representations for some of them
- Three examples of questions:
  - Is the language empty?
  - Does string w belong to the language?
  - Two language descriptions represent the same language?

Is the Language Empty?

- Answer: L=∅ is empty;
- Let's assume the language is represented by an FA or by an RE
- FA
  - Question can resume to the accessibility in the respective graph: if none final state is accessible from the initial state, the answer is true
  - Algorithm with temporal complexity proportional to the number of edges, $O(n^2)$, being *n* the number of nodes
- RE (length *n*)
  - Convert to ε-NFA, results in O(n) states, $O(n^2)$ algorithm, or
  - Inspecting the RE (in the case it contains ∅ and…)

Does w belong to the Language?

- L represented by an FA
  - DFA
    - Simulate the processing of the chain and answer yes if it terminates in a final/accept state
    - O(|w|)
  - NFA, ε-NFA
    - Convert to DFA and apply the previous method
    - Algorithm can have exponential temporal complexity in the size of the representation
    - Sometimes it is simpler and more efficient to simulate the NFA or the ε-NFA directly, maintaining the set of the s states in which the FA can be for each transition
    - $O(|w|s^2)$, being s the number of states
- L represented by an RE with length s
  - Convert to ε-NFA with at most 2s states in time O(s) and apply the previous method

L_1 and L_2 are Equivalent?

- Two RL descriptions are equivalent if they define the same language
  - We can find a minimal and unique representation, possibly with the exception of the name of the sates
- We can use the cartesian product of DFAs and:
  - a) verify $L_1 - L_2 = \varnothing$ and $L_2 - L_1 = \varnothing$, or:
  - b) verify $L_1 \cap \overline{L_2} = \varnothing$ and $\overline{L_1} \cap L_2 = \varnothing$

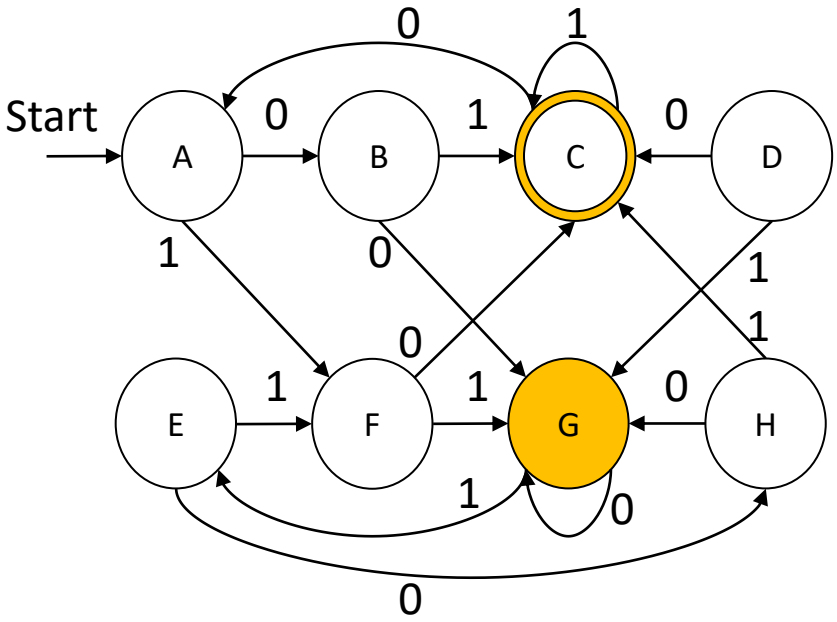
Equivalence of States in DFAs and DFA Minimization

L_1 and L_2 are Equivalent?

- Equivalence of the sates of a DFA
  - Two states p and q are equivalents if for all the strings w, δ^(p,w) is a final state iff δ^(q,w) is a final state as well
  - We cannot distinguish p and q only from the accept or non-accept result of any string
    - δ^(p,w) and δ^(q,w) may not be the same state but only that they be both accept states or both non-accept states
  - If two states are not equivalent, then they are distinguishable states
    - There is at least one string w in which one of δ^(p,w) and δ^(q,w) is a final state and the other not

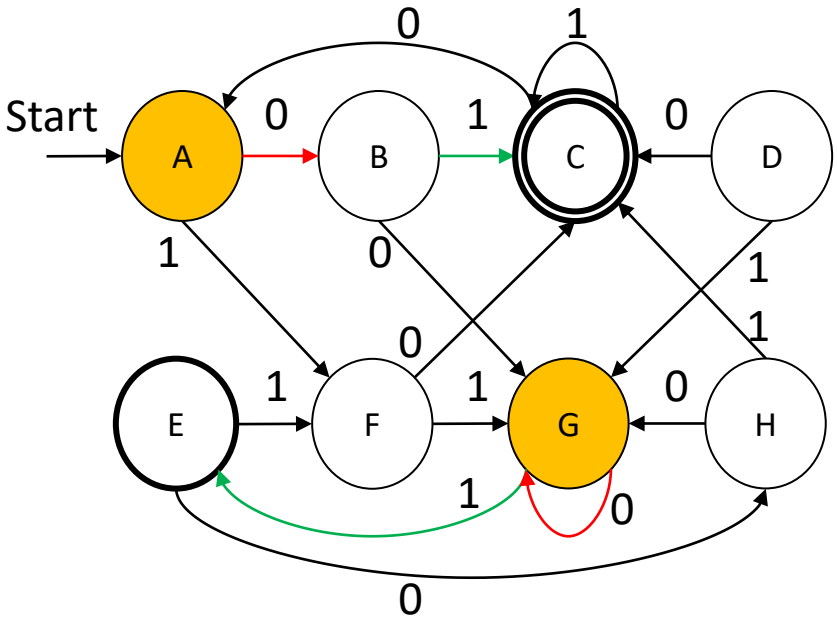
State Equivalence

- δ^(C,ε) = C is final state, and δ^(G,ε) not → C and G are not equivalent



State Equivalence

- δ^(C,ε) = C is final state, and δ^(G,ε) not → C and G are not equivalent
- A and G:
  - ε, 0, 1 do not allow to distinguish, but δ^(A,01)=C and δ^(G,01)=E allow



State Equivalence

- δ^(C,ε) = C is final state and δ^(G,ε) not → C and G are not equivalent
- A and G:
  - ε, 0, 1 do not allow to distinguish, but δ^(A,01)=C and δ^(G,01)=E allow
- A and E:
  - None of them is a final state
  - With 1 both transit to F, thus w=1x does not allow to distinguish; 0 as well;
  - δ^(A,00) = G = δ^(E,00); δ^(A,01) = C = δ^(E,01); thus A and E are equivalent

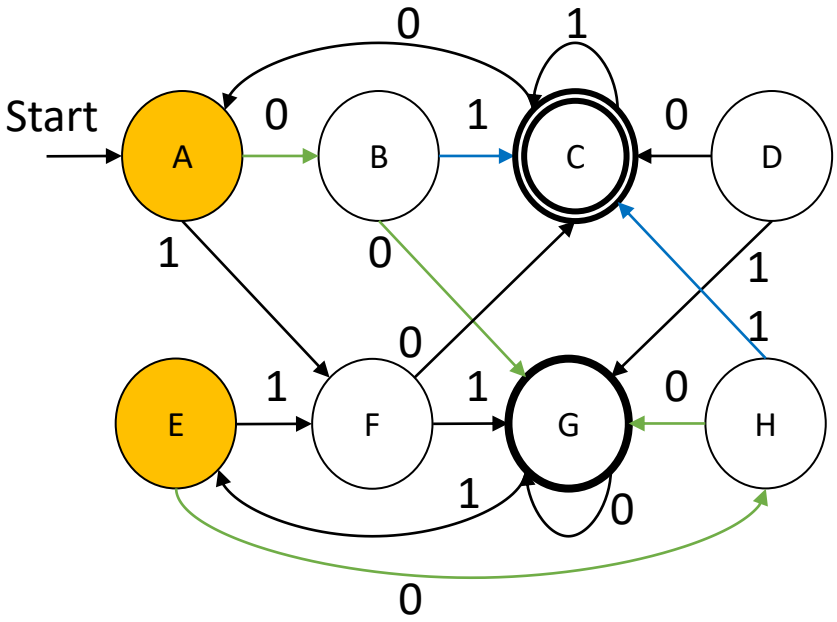
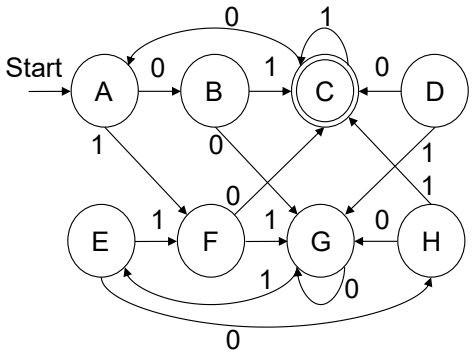


Table of Distinguishable States

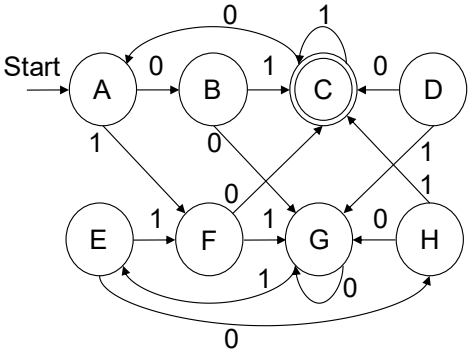
- A table indicating if each pair of states is distinguishable or not



	A	B	C	D	E	F	G
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

Algorithm to Fill the Table

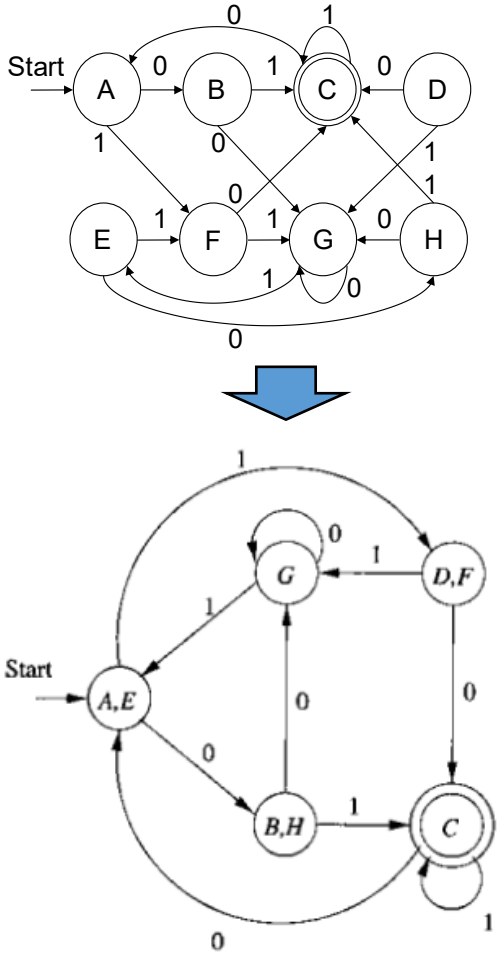
- Given a DFA A=(Q,Σ,δ,q_0,F) find the equivalent states
  - Basis: if p is a final state and q not, the pair {p,q} is distinguishable
  - Induction: be p and q states such that, for each symbol a, r=δ(p,a) and s=δ(q,a) are distinguishable; then {p,q} are distinguishable



	A	B	C	D	E	F	G
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

DFA Minimization

- It is possible to find a minimum representation in terms of the number of states for a given language
  - Eliminate states not reached from the initial state
  - Coalesce the remaining states in groups of equivalent states (the state equivalence is transitive)
  - Consider each group a state and modify δ accordingly
- Example from the table of distinguishable states (previous slide)
  - Groups: ({A,E}, {B,H}, {C}, {D,F}, {G})



Equivalence of Regular Languages (RLs)

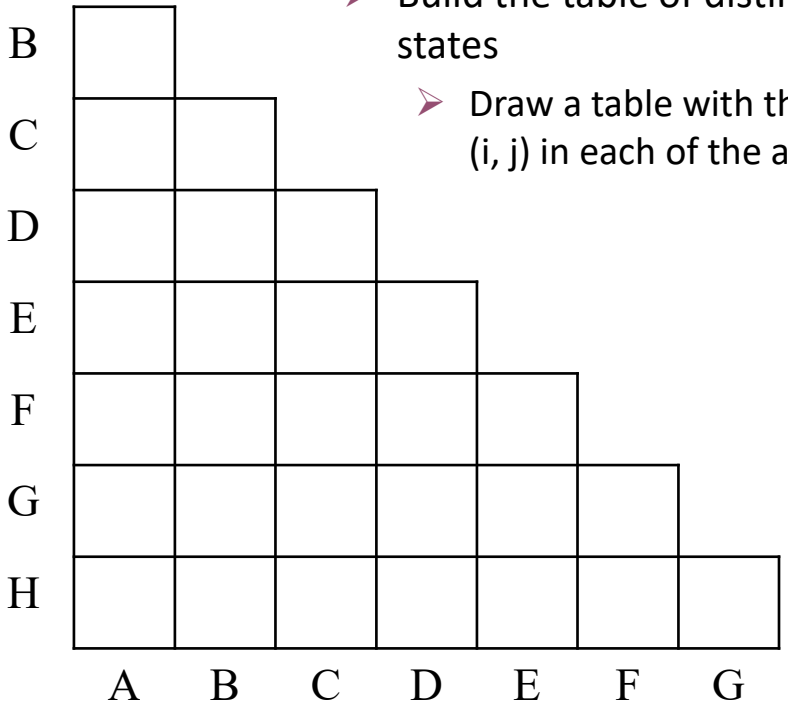
- Given two representations for the same languages L and M
  - Convert each one to a DFA
  - Imagine a DFA with the states formed by the union of the states of the two DFAs and use the algorithm to fill the table of distinguishable states to verify if the two initial states are equivalent, if they are then L=M
- Another possibility:
  - Convert each one to a DFA
  - Minimize both DFAs
  - If they are equal (apart the names of the sates), then L=M
- One additional possibility: the use of the product of the two DFAs and…

Exercise 2

- Given the DFA in the right
  - Build the table of distinguishable states
  - Obtain the DFA equivalent with the minimum number of states

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

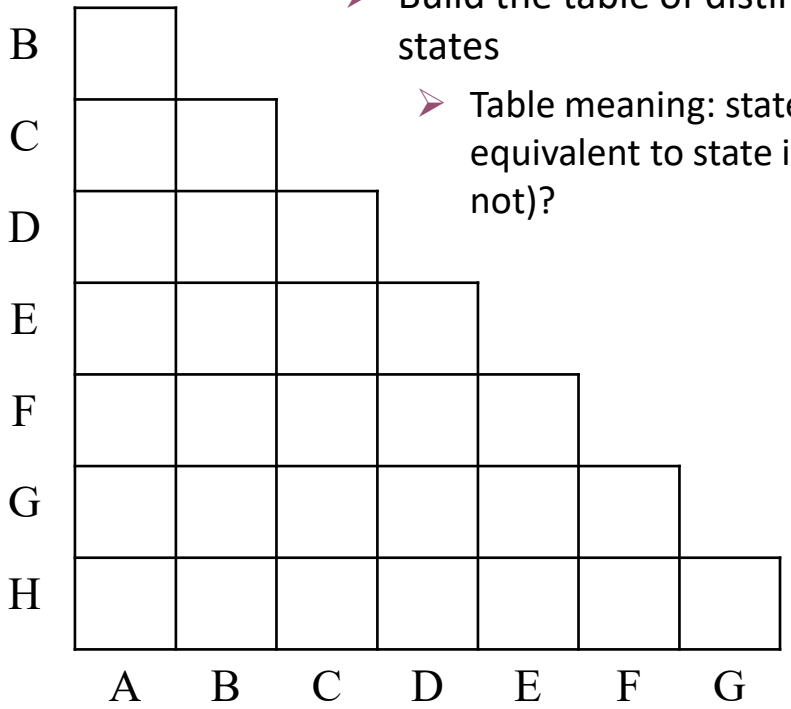
Exercise 2: solution



- Build the table of distinguishable states
  - Draw a table with the DFA states (i, j) in each of the axis

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

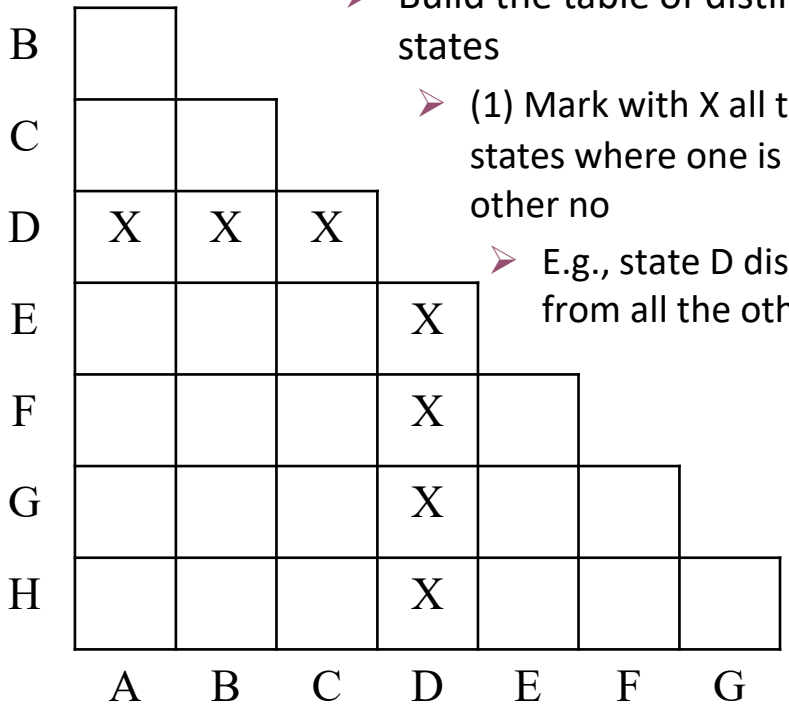
Exercise 2: solution



- Build the table of distinguishable states
  - Table meaning: state j is equivalent to state i (X indicates not)?

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Exercise 2: solution



- Build the table of distinguishable states
  - (1) Mark with X all the pairs of states where one is final and the other no
    - E.g., state D distinguishes from all the other states

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Exercise 2: solution

- Build the table of distinguishable states
  - (2) Include in the cells the equality that must exist in order two states (j, i) are equivalent
    - E.g., between B and A: A=C

B	A=C						
C	B=D A=B	A=D C=B					
D	X	X	X				
E	B=D A=F	A=D C=F	B=F	X			
F	B=G A=E	A=G C=E	D=G B=E	X	D=G		
G	B=F	A=F C=G	D=F B=G	X	D=F F=G	E=G	
H	B=G A=D	A=G C=D	D=G B=D	X	D=G F=D	E=D	F=G G=D
	A	B	C	D	E	F	G

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

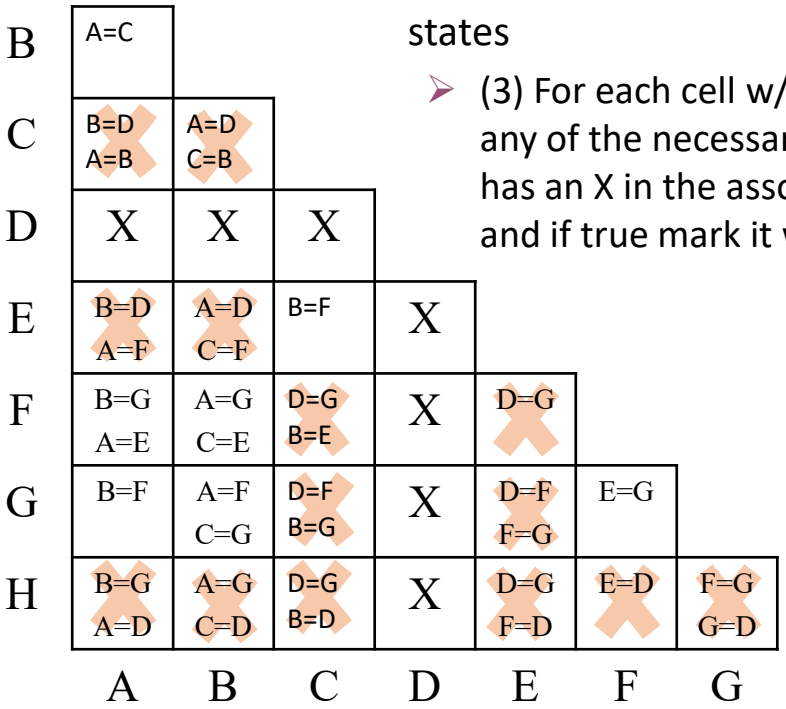
Exercise 2: solution

- Build the table of distinguishable states
  - (3) For each cell w/o X verify if any of the necessary equalities has an X in the associated cell and if true mark it with X

	A	B	C	D	E	F	G
B	A=C						
C	B=D A=B	A=D C=B					
D	X	X	X				
E	B=D A=F	A=D C=F	B=F	X			
F	B=G A=E	A=G C=E	D=G B=E	X	D=G		
G	B=F	A=F C=G	D=F B=G	X	D=F F=G	E=G	
H	B=G A=D	A=G C=D	D=G B=D	X	D=G F=D	E=D	F=G G=D

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

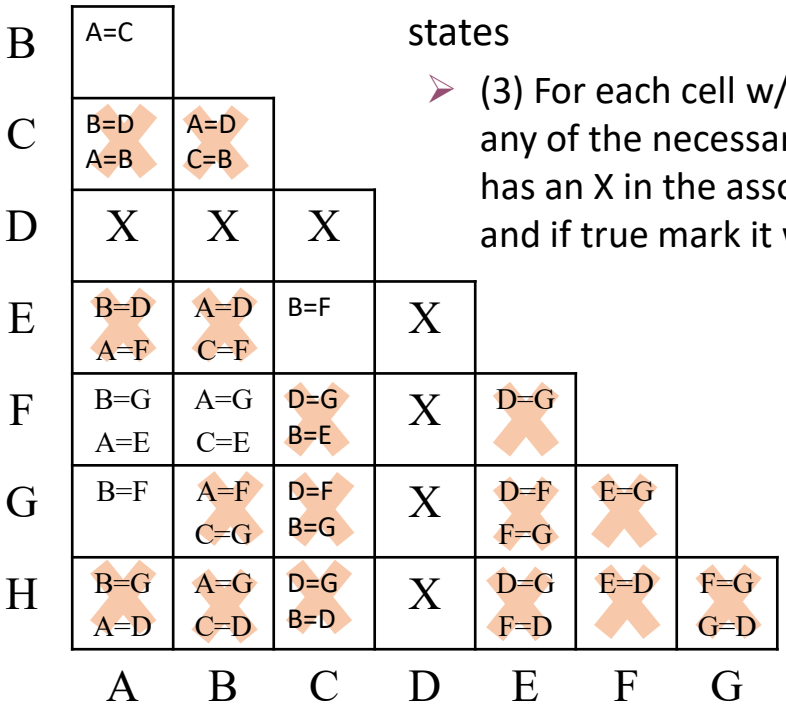
Exercise 13: solution



- Build the table of distinguishable states
  - (3) For each cell w/o X verify if any of the necessary equalities has an X in the associated cell and if true mark it with X

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

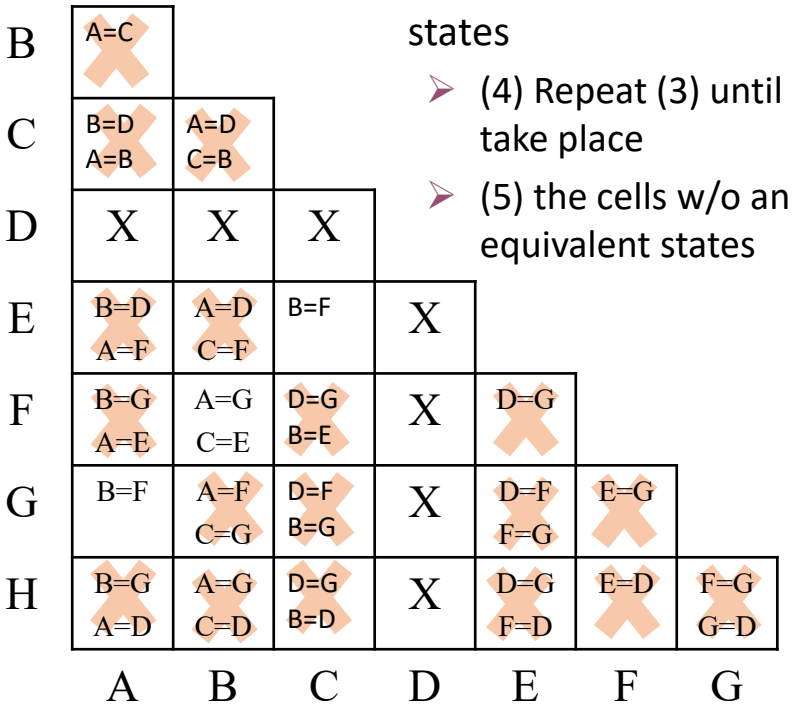
Exercise 2: solution



- Build the table of distinguishable states
  - (3) For each cell w/o X verify if any of the necessary equalities has an X in the associated cell and if true mark it with X

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Exercise 2: solution

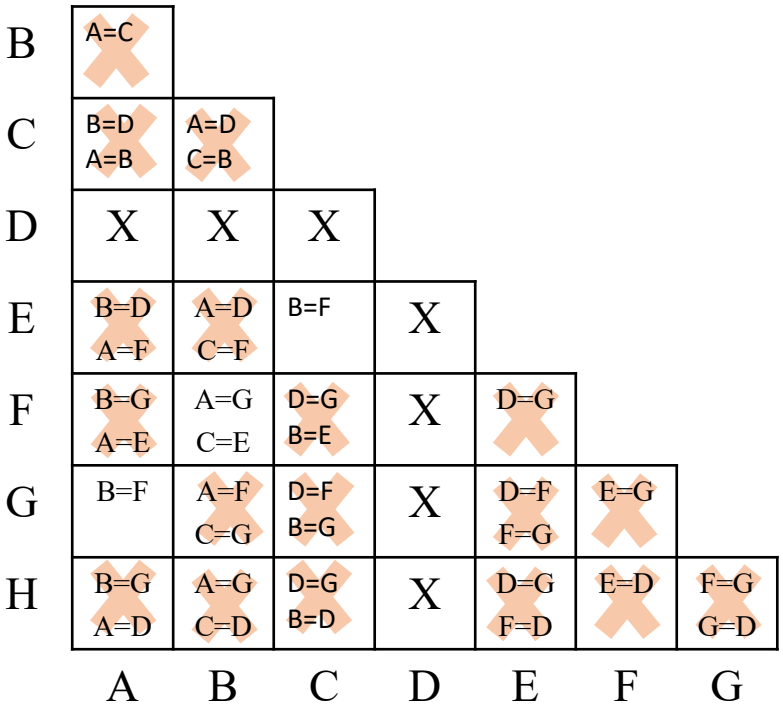


- Build the table of distinguishable states
  - (4) Repeat (3) until no change take place
  - (5) the cells w/o an X identify equivalent states

	0	1
→A	B	A
B	A	C
C	D	B
*D	D	A
E	D	F
F	G	E
G	F	G
H	G	D

Exercise 2: solution

- Coalesce the equivalent states in groups and draw the DFA with the minimum number of states



	0	1
→A,G	B,F	A,G
B,F	A,G	C,E
C,E	D	B,F
*D	D	A,G
H	A,G	D

Additional Questions About Regular Languages

Some Questions

- Given an FA $M = (Q, \sum, \delta, q0, F)$:
- **1. i**s $L(M)$ nonempty?
  - Simulate the FA with the possible strings w, being $w \in \sum^*$ *and* $0 \leq |w| \leq n-1$, *where n is the number of states of M*
  - *Other possibility would be the one based on the graph and previously presented to determine if a regular language is empty*
- **2.** is $L(M)$ infinite?
  - Simulate the FA with the possible strings w, being $w \in \sum^*$ *and* $n \leq |w| < 2n$, *where n is the number of states of M*
  - *Other possibility would be to identify at least one loop in a path from the start state to an accept state in the graph*
- **2.** is $L(M)=\sum^*$?