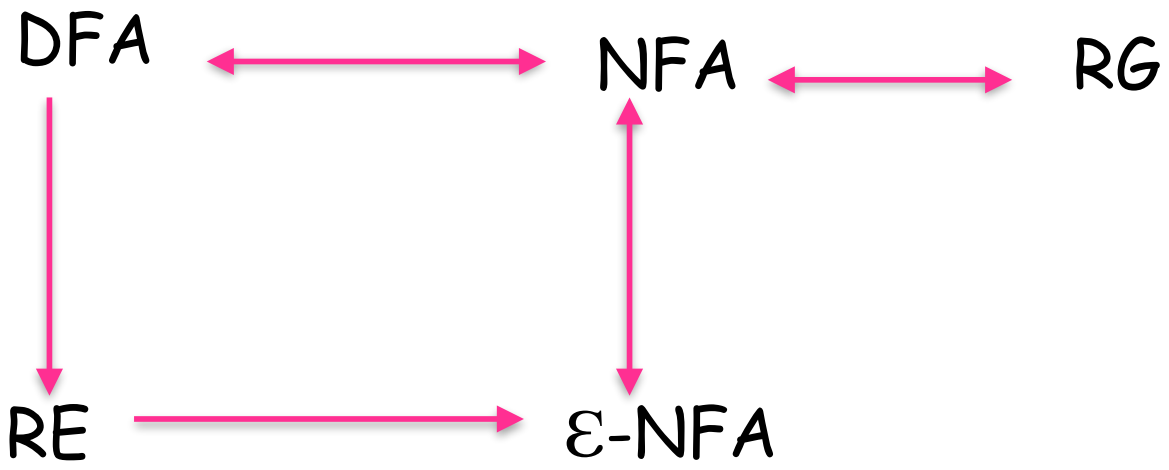
Roadmap



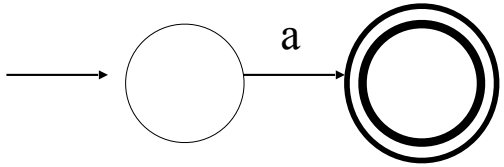
Converting a RE to an Automata

- We can convert a RE to an ε-NFA
  - Inductive construction
  - Start with a simple basis, use that to build more complex parts of the NFA

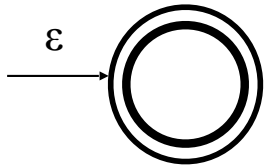
RE to ε-NFA

- Basis:

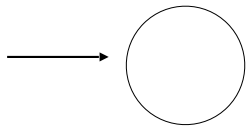
R=a



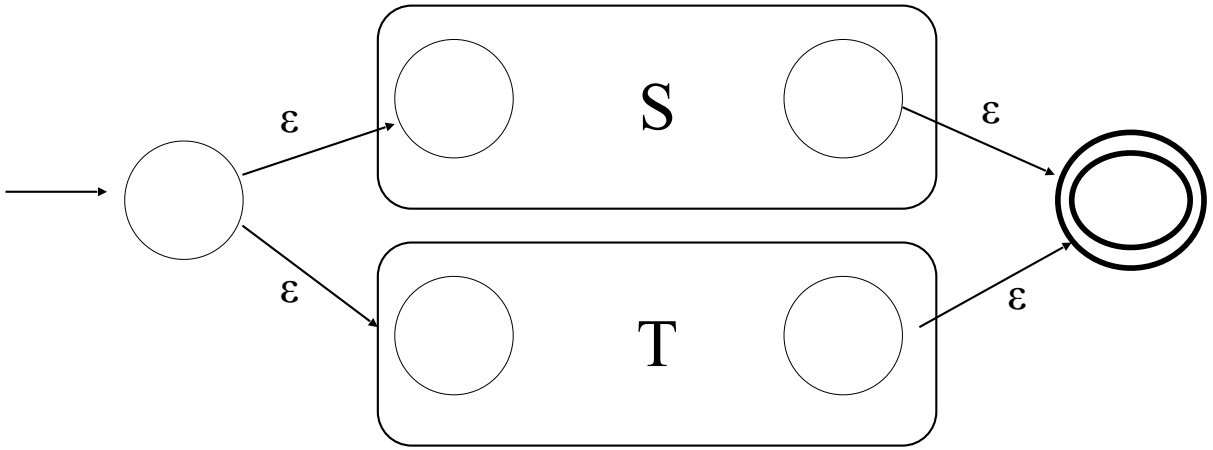
R=ε



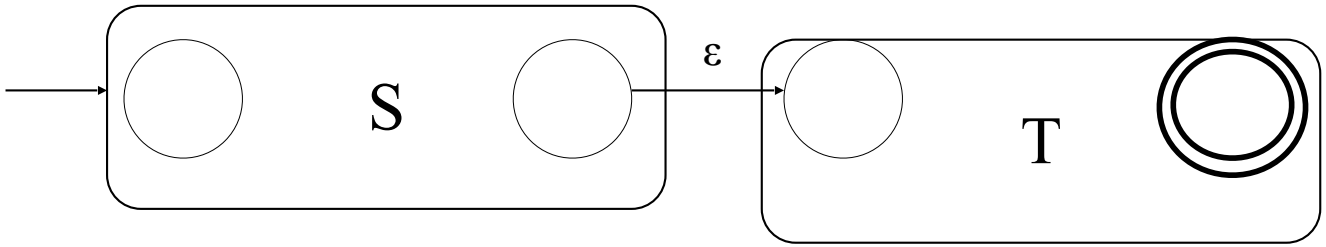
R=Ø



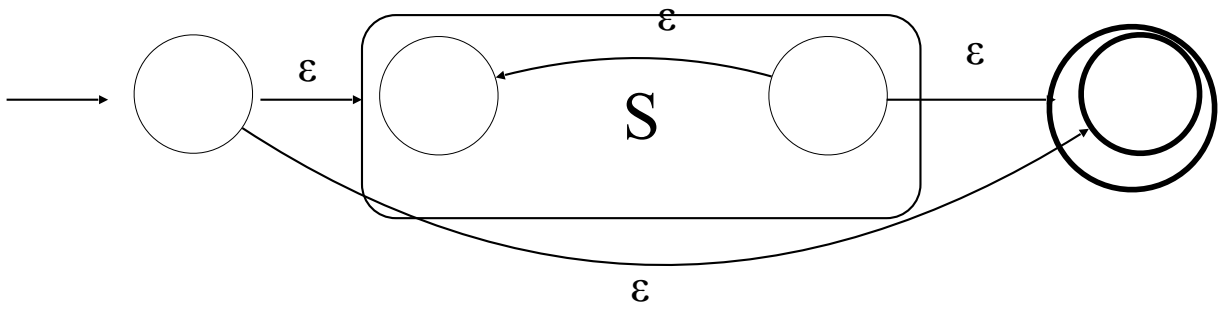
R=S+T



R=ST

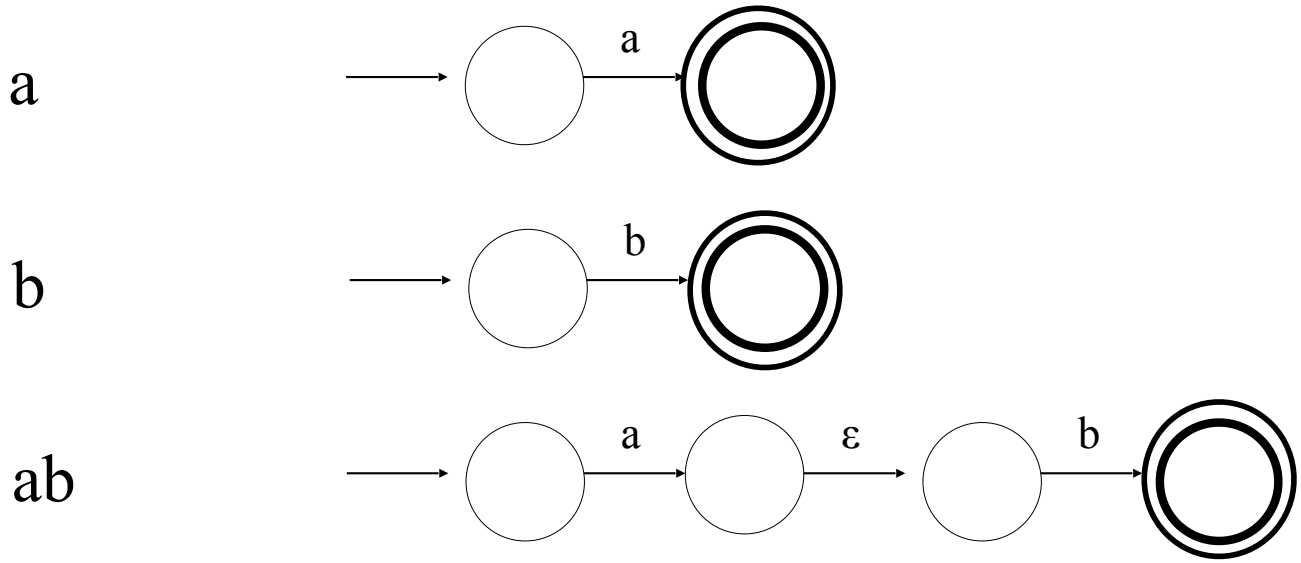


R=S*



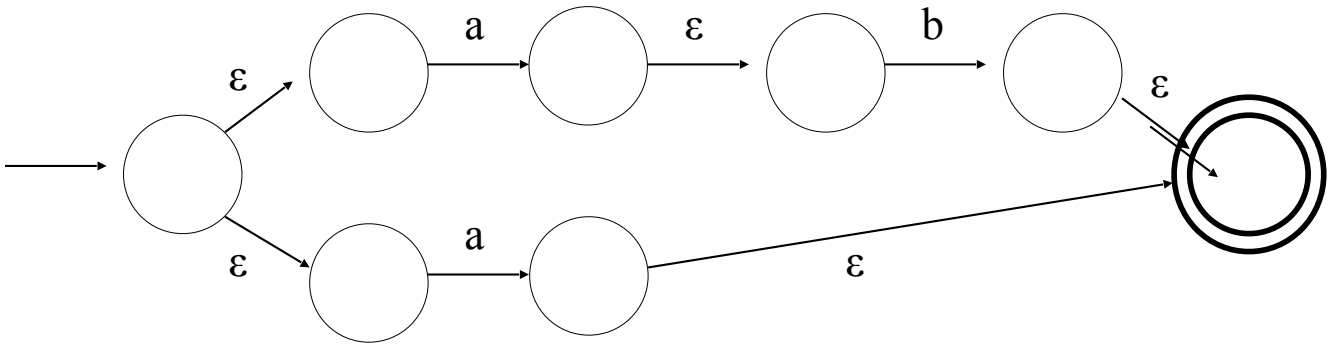
RE to ε-NFA Example

- Convert R= (ab+a)* to an NFA
  - We proceed in stages, starting from simple elements and working our way up

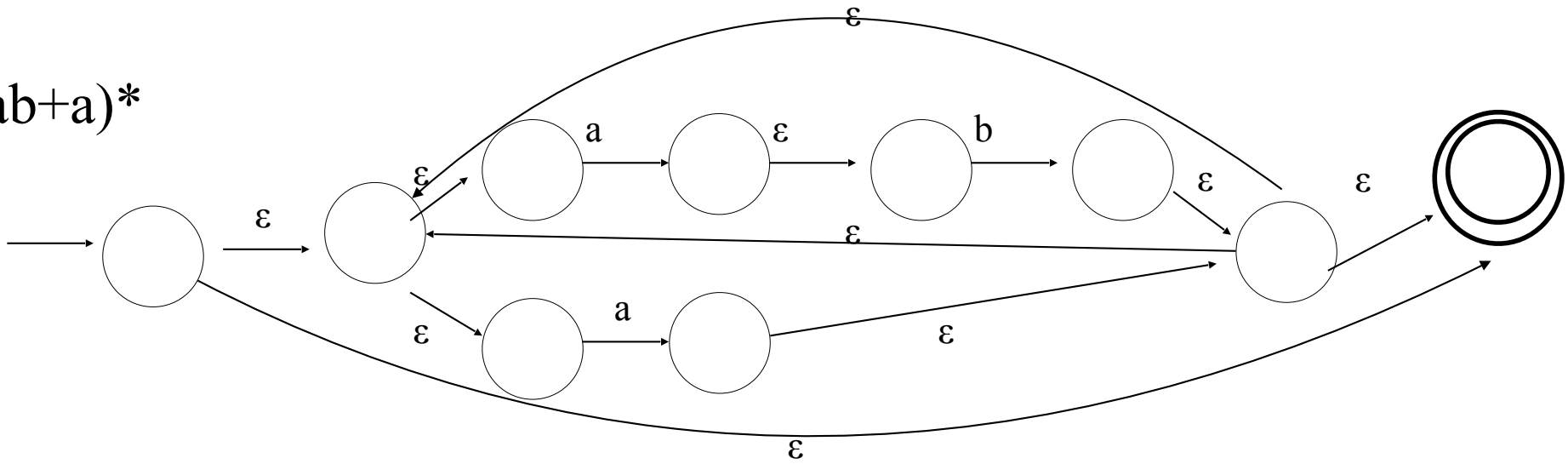


RE to ε-NFA Example (2)

ab+a



(ab+a)*

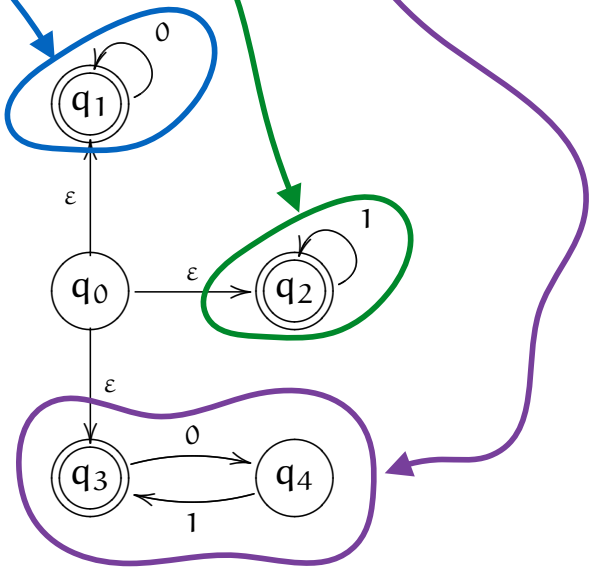


Esempio: from RE to ε-*NFA*

$$L = \{\, x \mid \exists n \in \mathbb{N}.\ x = 0^n \ \vee\ x = 1^n \ \vee\ x = (01)^n \,\}$$

$(0^* + 1^* + (01)^*)$.

ε-*NFA*



What have we shown?

- Regular expressions, finite state automata and regular grammars are really different ways of expressing the same thing.
- In some cases you may find it easier to start with one and move to the other
  - E.g., the language of an even number of one's is typically easier to design as a NFA or DFA and then convert it to a RE

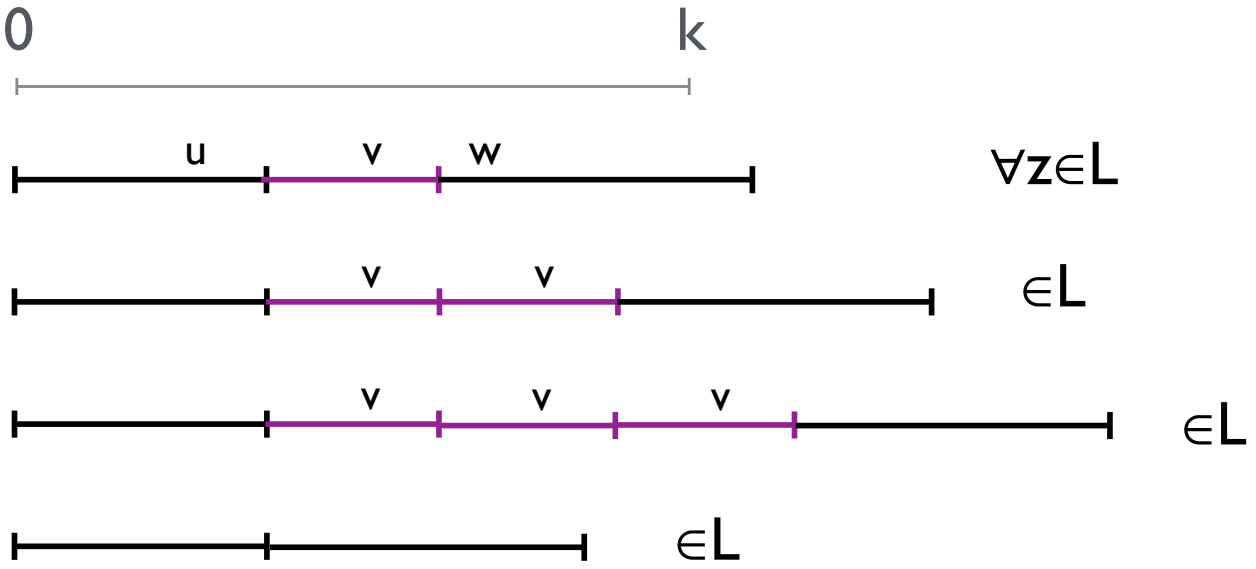
Not all languages are regular!

- $L=\{ a^n b^n \mid n \in Nat \}$

Pumping Lemma

Given L an infinite regular language then there exists an integer k such that for any string $z \in L . |z| \geq k$ it is possible to split z into 3 substrings

$$z = uvw \text{ with } |uv| \leq k, |v| > 0 \text{ such that } \forall i \in \mathbf{N}, uv^i w \in L$$



Negating the PL

The PL gives a necessary condition, that can be used to prove that a language is not regular!

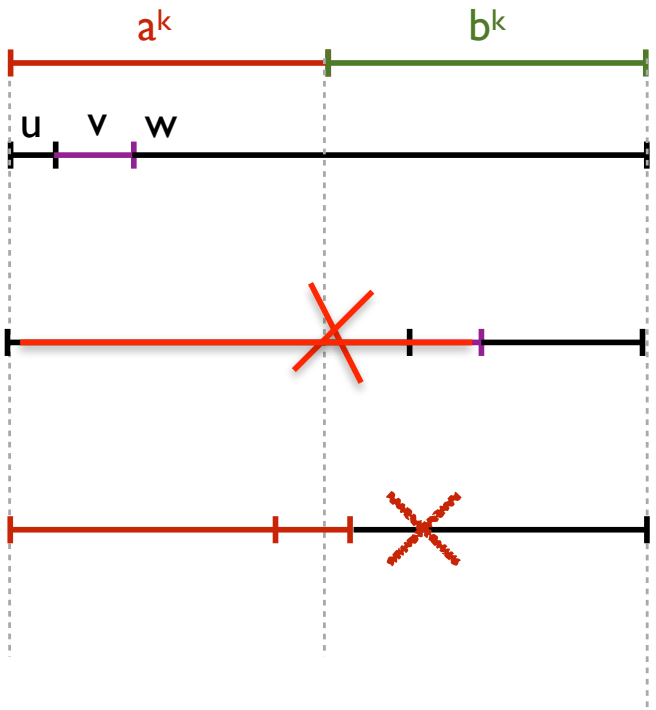
If $\forall k \in \mathbf{N} \;\; \exists z \in L . |z| \geq k$ for all possible splitting

$z = uvw$ with $|uv| \leq k, |v| > 0 \;\; \exists i \in \mathbf{N}$ such that $uv^i w \notin L$

then L is not a regular language!

Esempio

- $L=\{ a^n b^n \mid n \in Nat \}$, consider $k \in N$
- Let $z = a^k b^k$

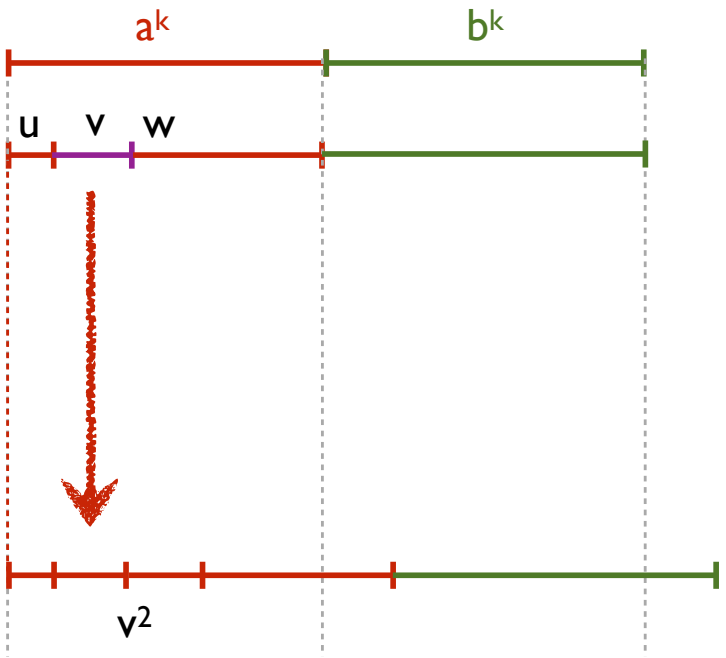


$z \in L$, $|v|=i$,

$|uv| > k$

Esempio

- $L=\{ a^n b^n \mid n \in Nat \}$, consider $k \in N$
- Let $z = a^k b^k$



$z \in L$, $|v|=i$,

$a^{k+i} b^k \notin L$

$i \neq 0$

Exercises

Prove that the following are not regular languages

$$L_{pal} = \{w \in \{0,1\}^* \mid w = w^R\} = \{\epsilon, 0, 1, 00, 11, 00100, 01110, \ldots\}$$

$$L_0 = \{0^n 1 0^n | n \geq 1\}$$

$$L_1 = \{0^n 1^m | n \leq m\}$$

Property of Regular languages

The regular languages are closed with respect to the union, concatenation and Kleene closure.

The complement of a regular language is always regular.

- The regular language are closed under intersection

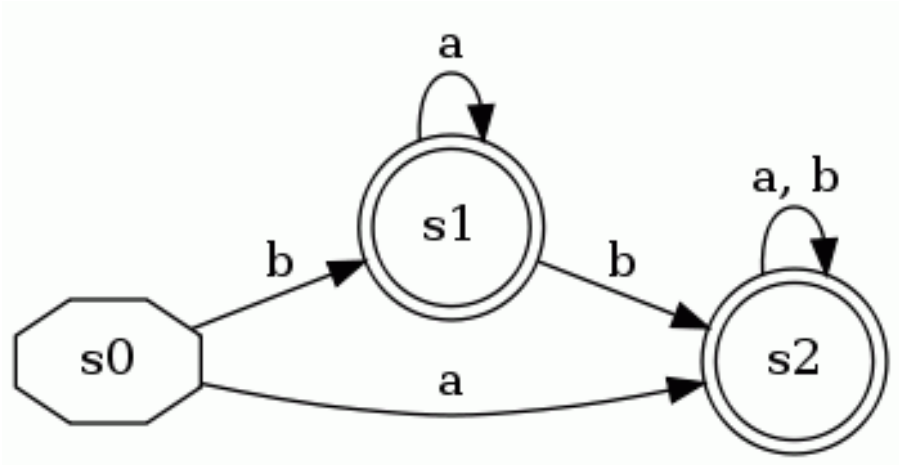
Decision Properties:

Approximately all the properties are decidable in case of finite automaton.

(i) Emptiness
(ii) Non-emptiness
(iii) Finiteness
(iv) Infiniteness
(v) Membership

DFA Minimization

- Some states can be redundant:
  - The following DFA accepts (a|b)+
  - State s1 is not necessary



DFA Minimization

- The task of *DFA minimization*, then, is to automatically transform a given DFA into a state-minimized DFA
  - Several algorithms and variants are known

DFA Minimization Algorithm

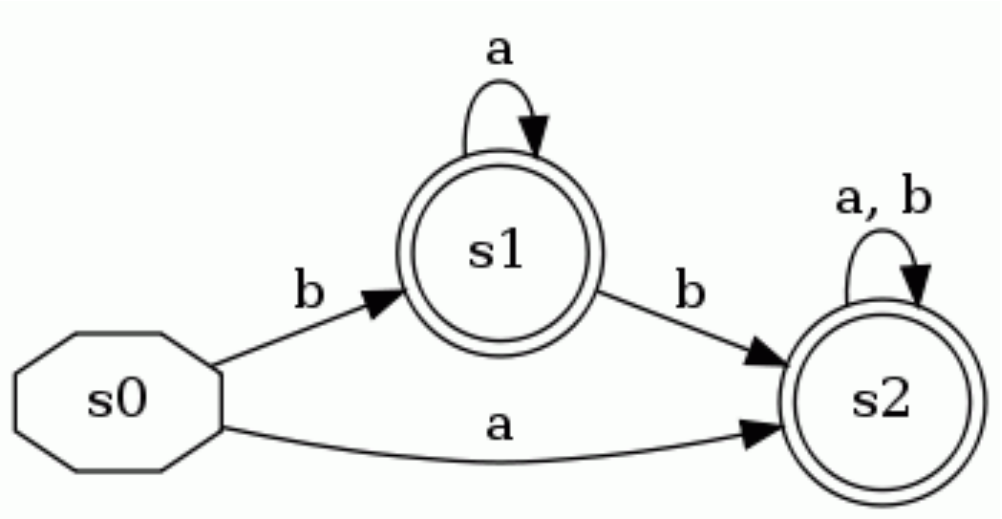
- Recall that a DFA $M=(Q, \Sigma, \delta, q_0, F)$
- Two states p and q are distinct if
  - p in F and q not in F or vice versa, or
  - for some a in Σ, $\delta(p, a)$ and $\delta(q, a)$ are distinct
- Using this inductive definition, we can calculate which states are distinct

DFA Minimization Algorithm

- Create lower-triangular table DISTINCT, initially blank
- For every pair of states (p,q):
  - If p is final and q is not, or vice versa
    - DISTINCT(p,q) = ε
- Loop until no change for an iteration:
  - For every pair of states (p,q) and each symbol a
    - If DISTINCT(p,q) is blank and DISTINCT($\delta(p,a)$, $\delta(q,a)$) is not blank
      - DISTINCT(p,q) = a
- Combine all states that are not distinct

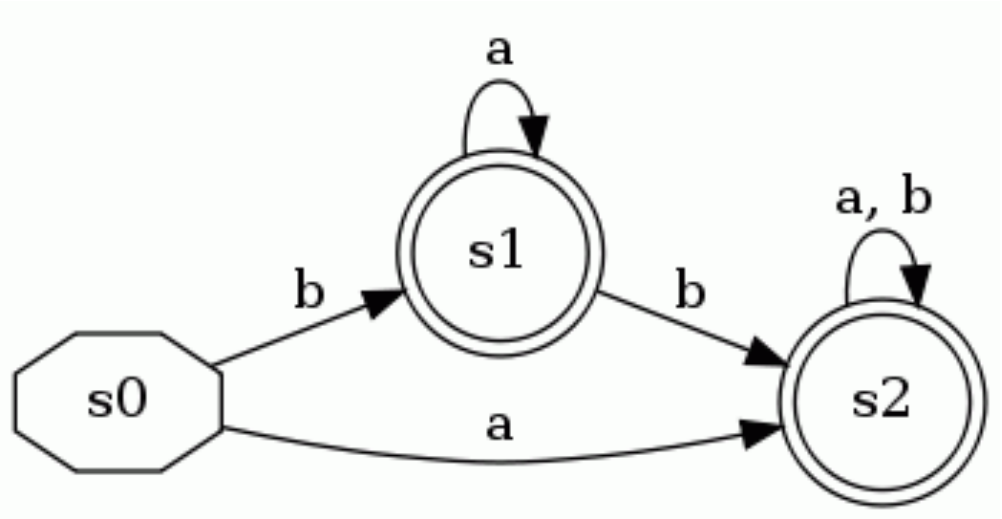
Very Simple Example

s0			
s1			
s2			
	s0	s1	s2



Very Simple Example

s0			
s1	ε		
s2	ε		
	s0	s1	s2

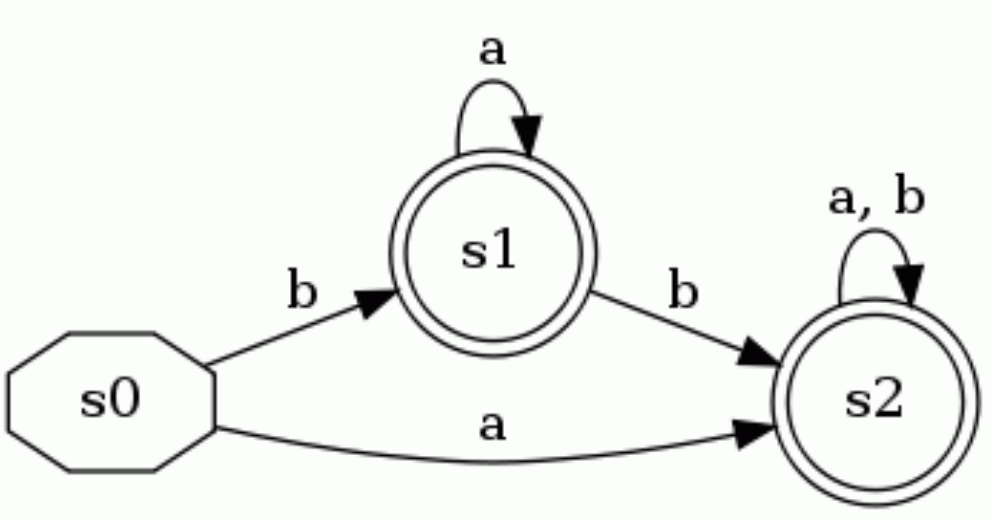


Label pairs with ε where one is a final state and the other is not

Very Simple Example

- DISTINCT(p,q) is blank and DISTINCT($\delta(p,\alpha)$, $\delta(q,\alpha)$) is not blank
  - DISTINCT(p,q) = α

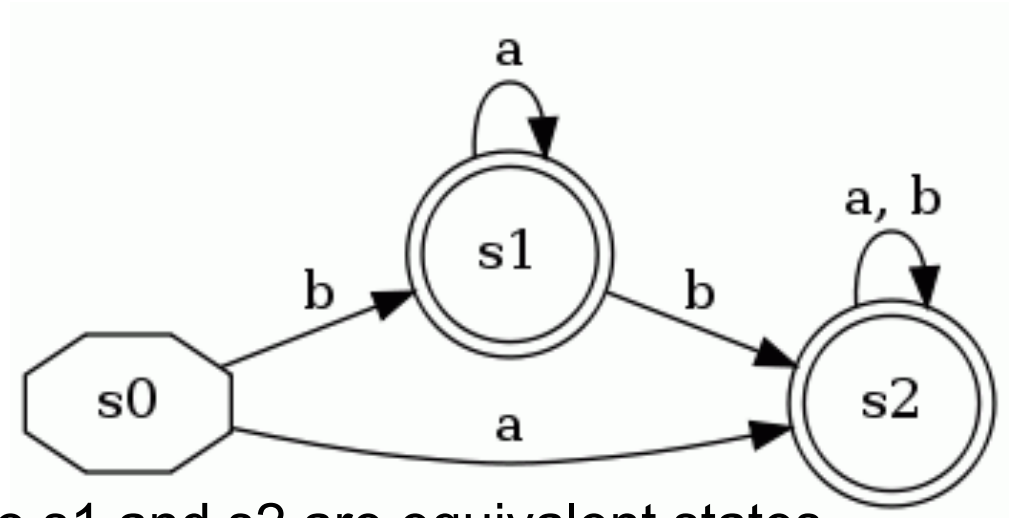
s0			
s1	ε		
s2	ε		
	s0	s1	s2



Main loop (no changes occur)

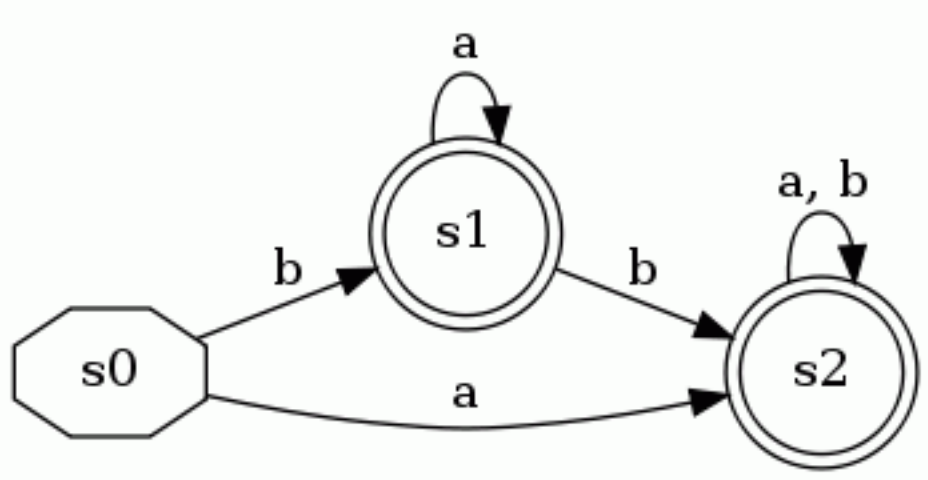
Very Simple Example

s0			
s1	ε		
s2	ε		
	s0	s1	s2

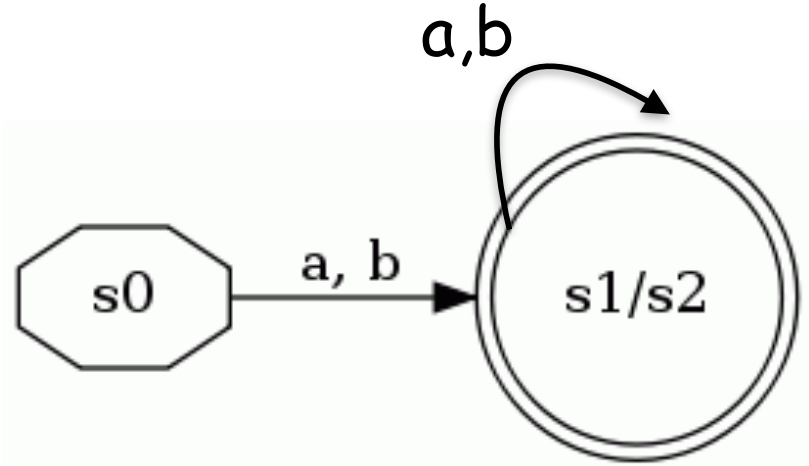


DISTINCT(s1, s2) is empty, so s1 and s2 are equivalent states

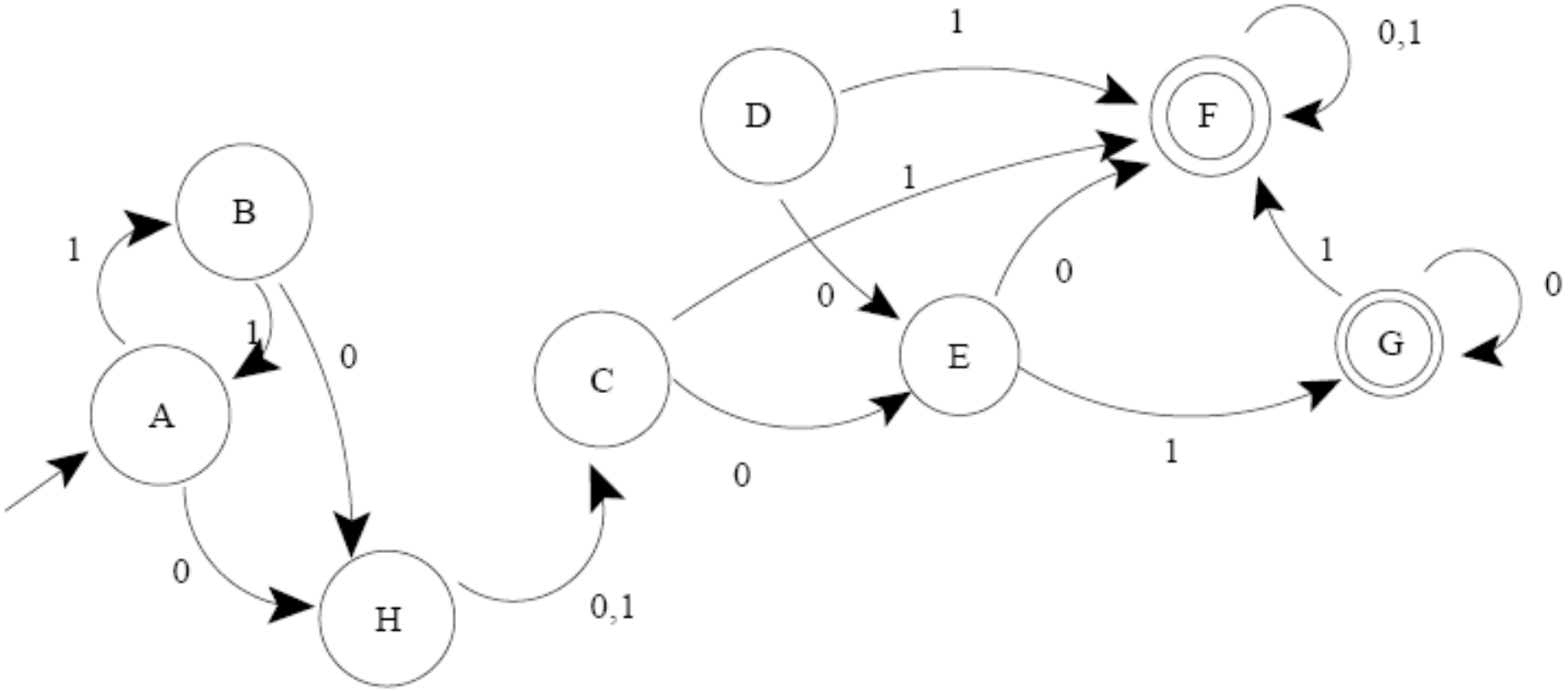
Very Simple Example



Merge s1 and s2

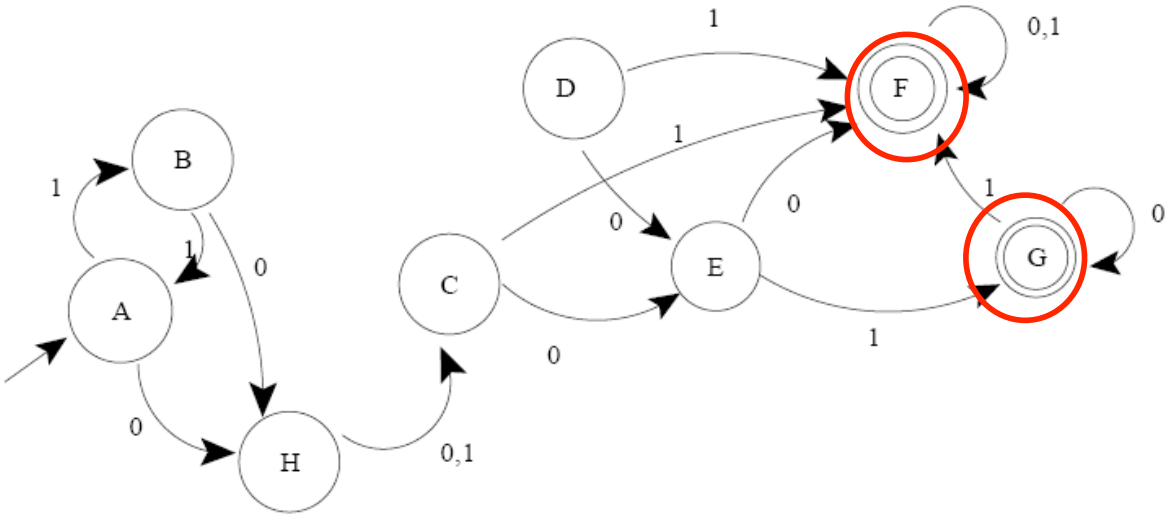


More Complex Example



More Complex Example

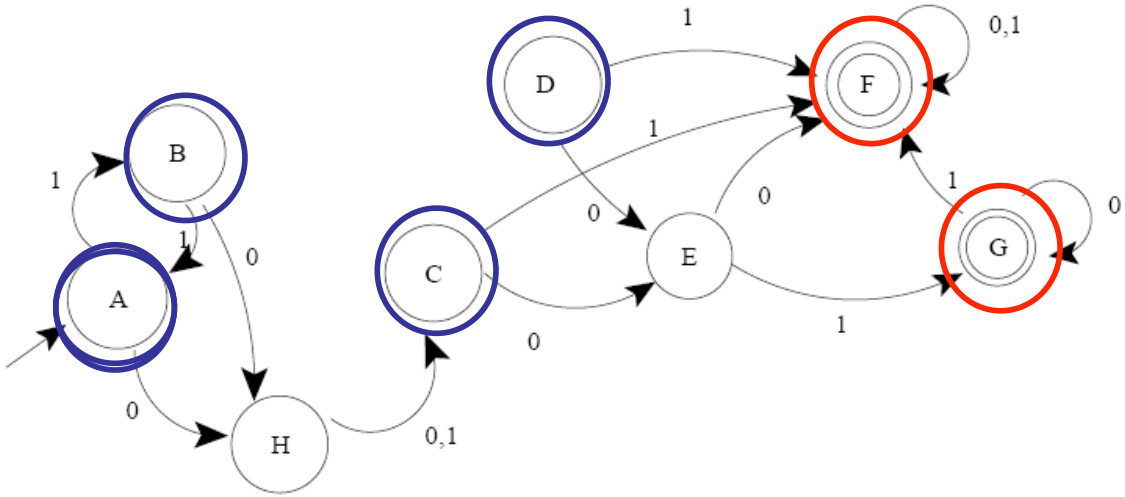
- Check for pairs with one state final and one not:



b							
c							
d							
e							
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h						ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

- First iteration of main '

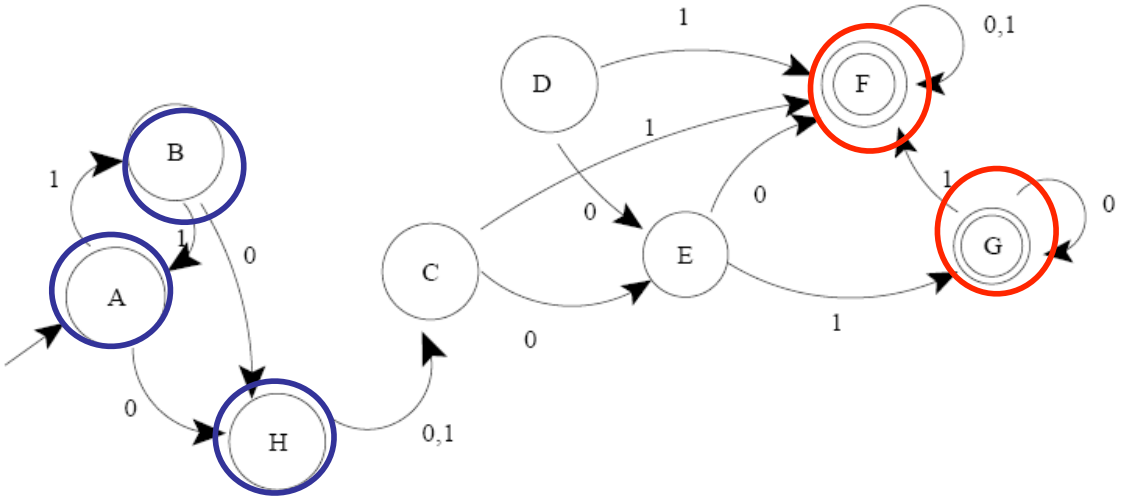


b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h			1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

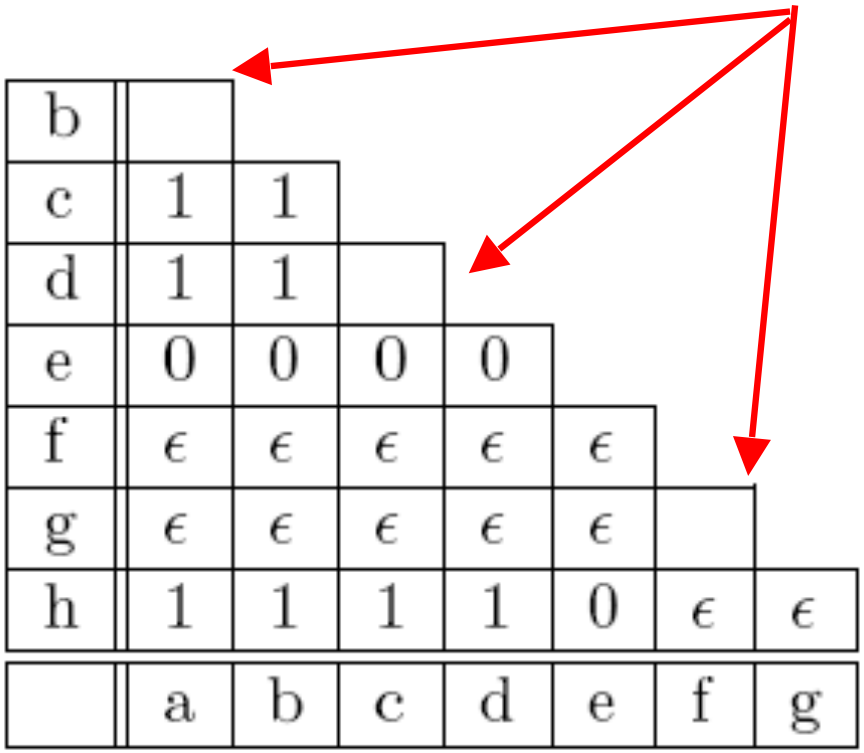
- Second iteration of main loop:

b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h	1	1	1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g



More Complex Example

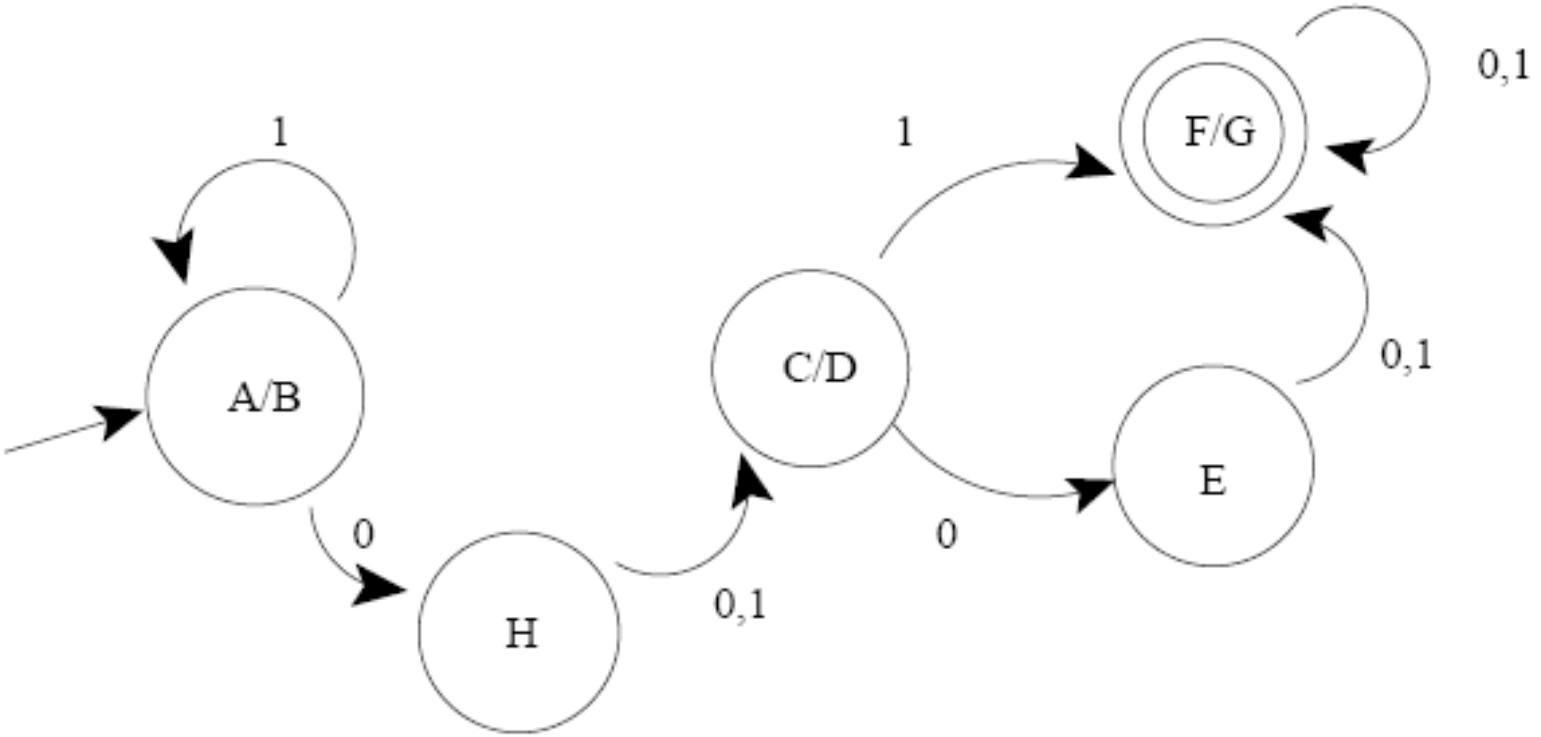
- Third iteration makes no changes
  - Blank cells are equivalent pairs of states



b							
c	1	1					
d	1	1					
e	0	0	0	0			
f	ϵ	ϵ	ϵ	ϵ	ϵ		
g	ϵ	ϵ	ϵ	ϵ	ϵ		
h	1	1	1	1	0	ϵ	ϵ
	a	b	c	d	e	f	g

More Complex Example

- Combine equivalent states for minimized DFA:



Conclusion

- DFA Minimization is a fairly understandable process, and is useful in several areas
  - Regular expression matching implementation
  - Very similar algorithm is used for compiler optimization to eliminate duplicate computations
- The algorithm described is $O(kn^2)$
  - John Hopcraft describes another more complex algorithm that is $O(k\ (n \log n)\ )$

Exercises

Minimize the following automata

		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	H

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