Decision Properties of Regular Languages

General Discussion of "Properties"

The Pumping Lemma

Membership, Emptiness, Etc.

Properties of Language Classes

Char → String → Language → Lang Class

- A *language class* is a set of languages.
  - We have one example: the regular languages.
  - We'll see many more in this class.
- Language classes have two important kinds of properties:
  1. Decision properties. Yes/No
  2. Closure properties.

Representation of Languages

- Representations can be formal or informal.

ทางการ

- Example (formal): represent a language by a RE or DFA defining it.

ไม่เป็นทางการ

- Example: (informal): a logical or prose statement about its strings:
  - $\{0^n1^n \mid n \text{ is a nonnegative integer}\}$
  - "The set of strings consisting of some number of 0's followed by the same number of 1's."

Y N

Decision Properties

- A *decision property* for a class of languages is an algorithm that takes a formal description of a language (e.g., a DFA) and tells whether or not some property holds.
- Example: Is language L empty?

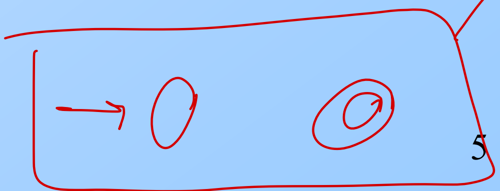
Subtle Point: Representation Matters

- You might imagine that the language is described informally, so if my description is "the empty language" then yes, otherwise no.

$\emptyset \neq \varepsilon$

- But the representation is a DFA (or a RE that you will convert to a DFA).
- Can you tell if $L(A) = \emptyset$ for DFA A?

DFA



No connect

$\emptyset$

Yes, No Answer DFA.

Why Decision Properties?

- When we talked about protocols represented as DFA's, we noted that important properties of a good protocol were related to the language of the DFA.
- Example: "Does the protocol terminate?" = "Is the language finite?"
- Example: "Can the protocol fail?" = "Is the language nonempty?"

DFA

Why Decision Properties – (2)

- We might want a "smallest" representation for a language, e.g., a minimum-state DFA or a shortest RE.
- If you can't decide "Are these two languages the same?"
  - I.e., do two DFA's define the same language?

You can't find a "smallest."

Closure Properties

- A *closure property* of a language class says that given languages in the class, an operator (e.g., union) produces another language in the same class.
- Example: the regular languages are obviously closed under union, concatenation, and (Kleene) closure.
  - Use the RE representation of languages.

+ $*$

Why Closure Properties?

1. Helps construct representations.
2. Helps show (informally described) languages not to be in the class.

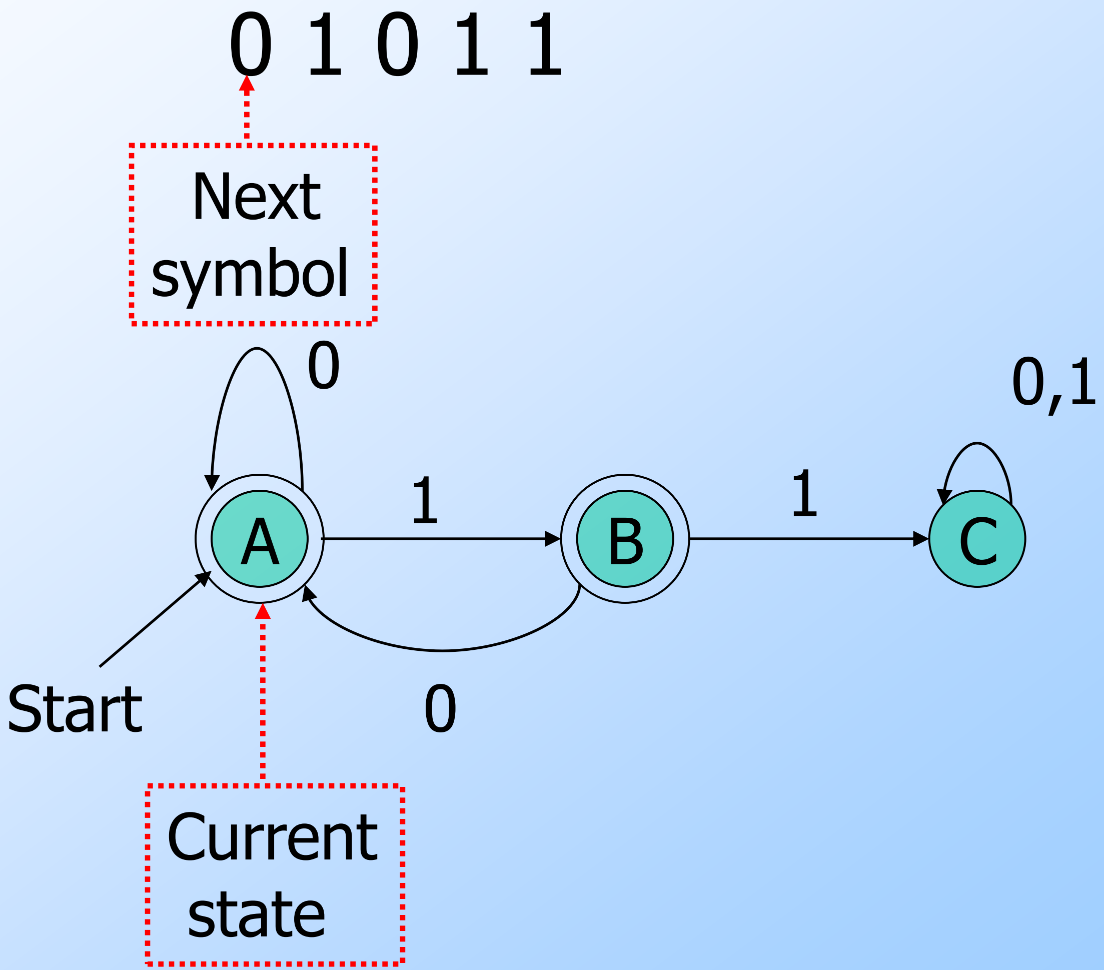
Example: Use of Closure Property

- We can easily prove $L_1 = \{0^n 1^n \mid n \geq 0\}$ is not a regular language.
- L_2 = the set of strings with an = number of 0's and 1's isn't either, but that fact is trickier to prove.
- Regular languages are closed under $\cap$.
- If L_2 were regular, then $L_2 \cap L(\mathbf{0}^*\mathbf{1}^*) = L_1$ would be, but it isn't.

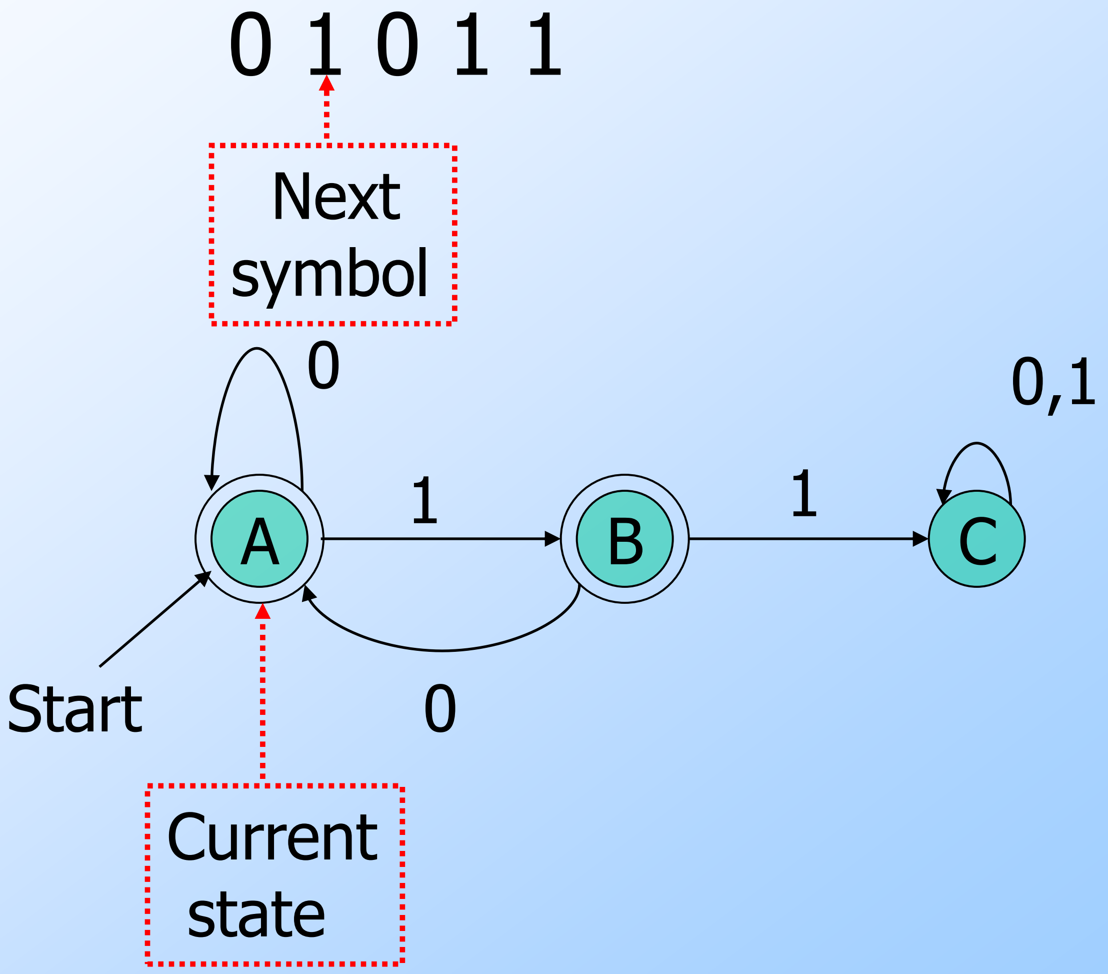
The Membership Question

- Our first decision property is the question: “is string w in regular language L?”
- Assume L is represented by a DFA A.
- Simulate the action of A on the sequence of input symbols forming w.

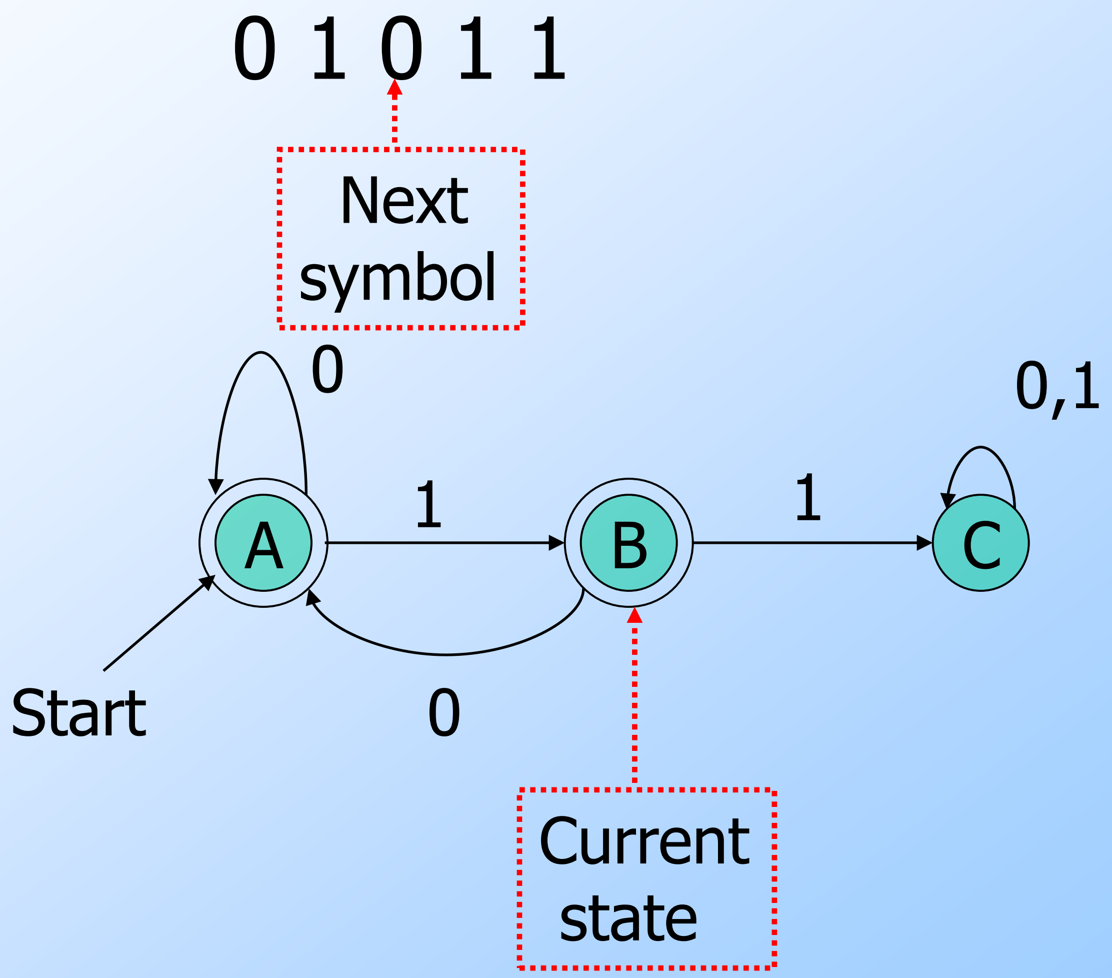
Example: Testing Membership



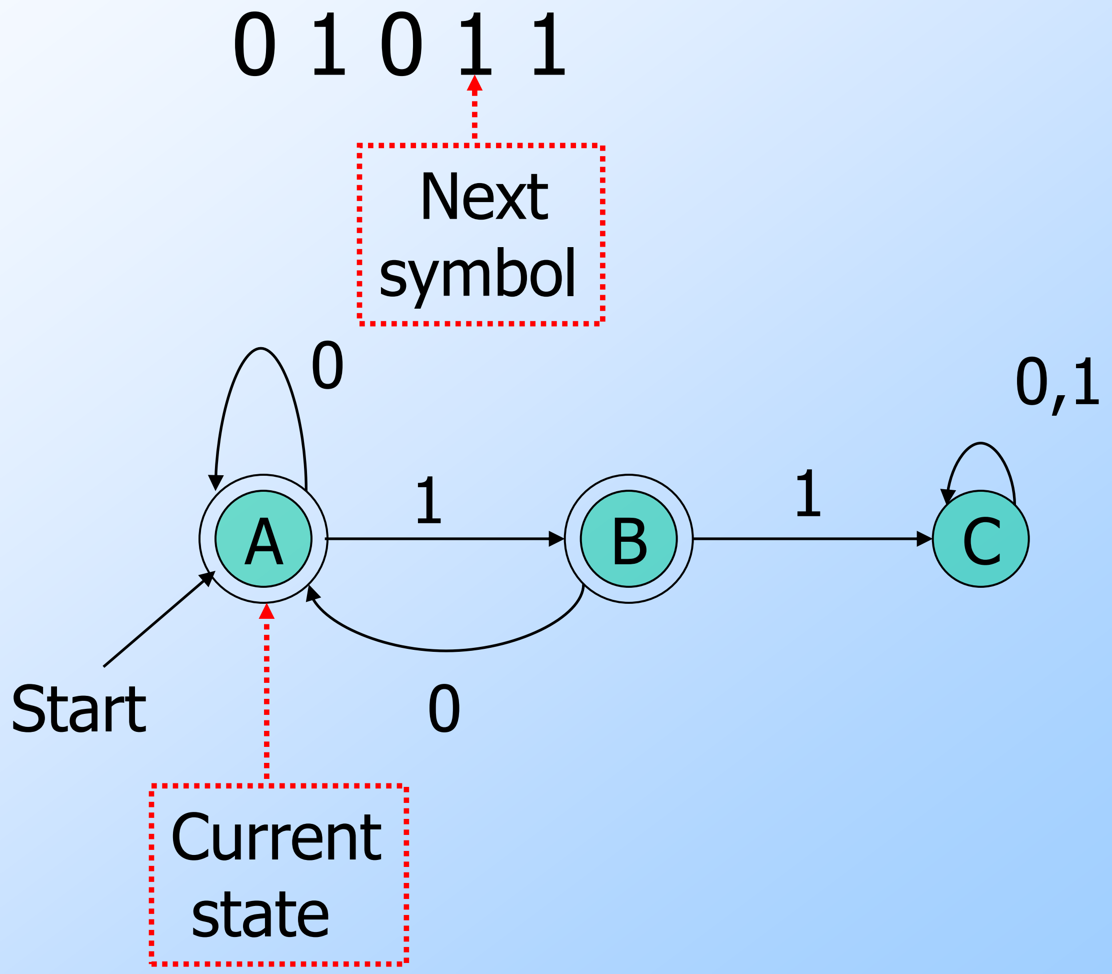
Example: Testing Membership



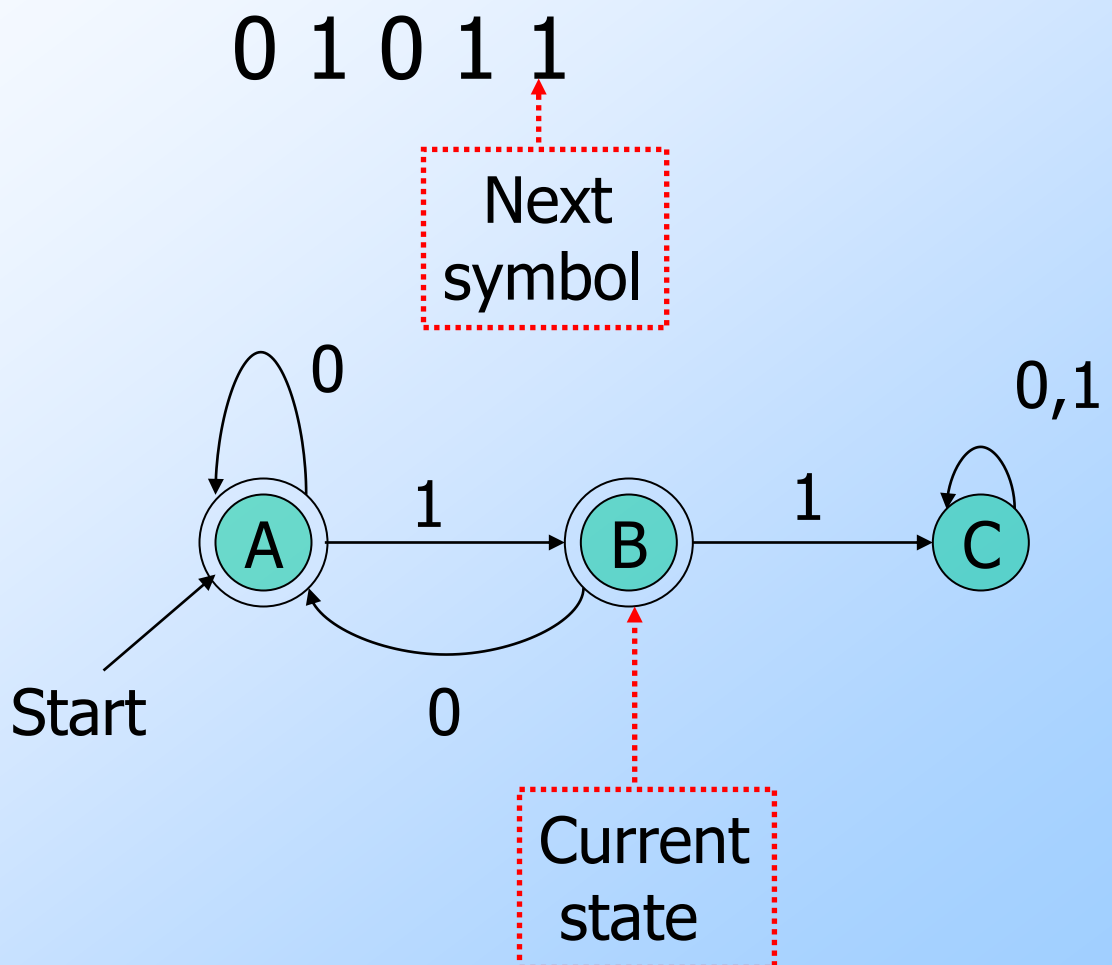
Example: Testing Membership



Example: Testing Membership

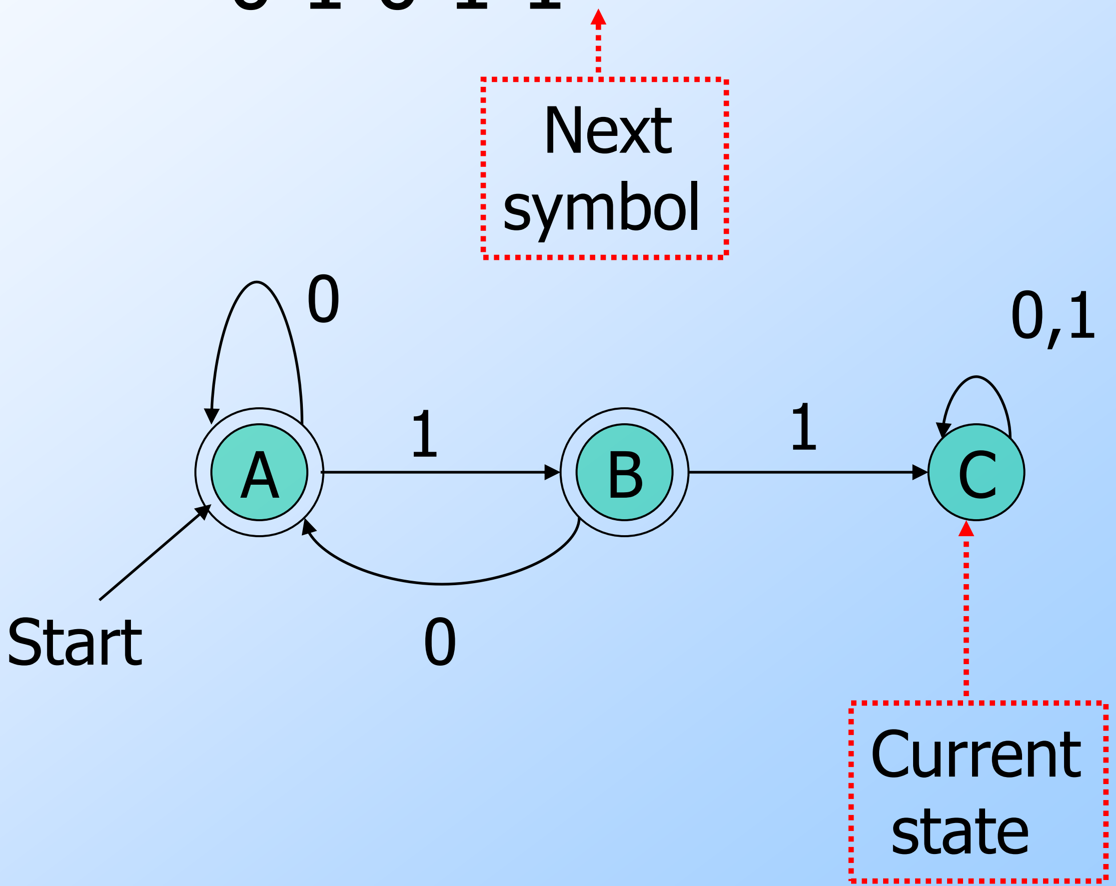


Example: Testing Membership



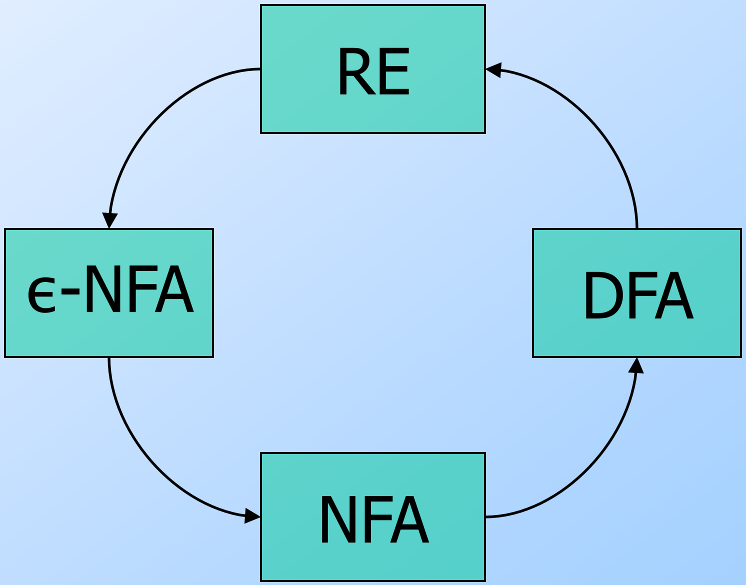
Example: Testing Membership

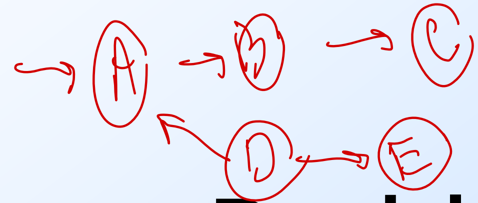
0 1 0 1 1



What if the Regular Language Is not Represented by a DFA?

- There is a circle of conversions from one form to another:





D, E

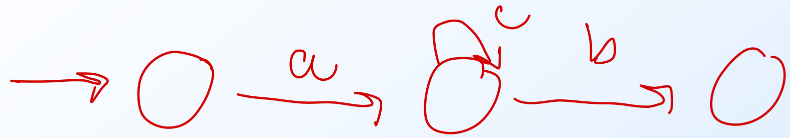
Emptiness Problem

The Emptiness Problem

- Given a regular language, does the language contain any string at all.
- Assume representation is DFA.
- Construct the transition graph.
- Compute the set of states reachable from the start state.

เข้าถึง State.

- If any final state is reachable, then yes, else no.



Example

The Infiniteness Problem

- Is a given regular language infinite?
- Start with a DFA for the language.
- Key idea: if the DFA has *n* states, and the language contains any string of length *n* or more, then the language is infinite.
- Otherwise, the language is surely finite.
  - Limited to strings of length *n* or less.

Infiniteness – Continued

- We do not yet have an algorithm.
- There are an infinite number of strings of length > n, and we can't test them all.
- Second key idea: if there is a string of length $\geq$ n (= number of states) in L, then there is a string of length between n and 2n-1.

Finding Cycles

1. Eliminate states not reachable from the start state.
2. Eliminate states that do not reach a final state.
3. Test if the remaining transition graph has any cycles.

L → ~~RL~~

The Pumping Lemma

- We have, almost accidentally, proved a statement that is quite useful for showing certain languages are not regular.
- Called the *pumping lemma for regular languages*.

ถ้า L สามารถพิสูจน์ได้คือ RL.

L = RL? → DFA
NFA, ε-NFA
RE

แต่ถ้าอยากบอกว่า ถ้าไม่ใช่ RL
ต้องใช้ Pumping Lemma

Statement of the Pumping Lemma

For every regular language L

There is an integer n, such that

For every string w in L of length $\geq$ n

We can write w = xyz such that:

1. $|xy| \leq n$.
2. $|y| > 0$.
3. For all $i \geq 0$, xy^iz is in L.

Number of states of DFA for L

Labels along first cycle on path labeled w

Example: Use of Pumping Lemma

- We have claimed $\{0^k1^k \mid k \geq 1\}$ is not a regular language. 00000 11111
- Suppose it were. Then there would be an associated n for the pumping lemma.
- Let $w = 0^n1^n$. We can write $w = xyz$, where x and y consist of 0's, and $y \neq \epsilon$.
- But then xyyz would be in L, and this string has more 0's than 1's.

Decision Property: Equivalence

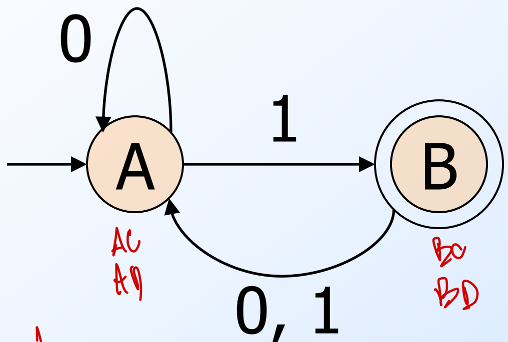
- Given regular languages L and M, is L = M?
- Algorithm involves constructing the *product DFA* from DFA's for L and M.
- Let these DFA's have sets of states Q and R, respectively.
- Product DFA has set of states $Q \times R$.
  - I.e., pairs [q, r] with q in Q, r in R.

L = M ?

Example: Product DFA

L

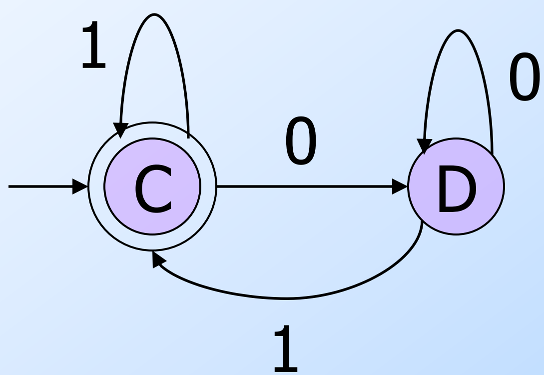
2state

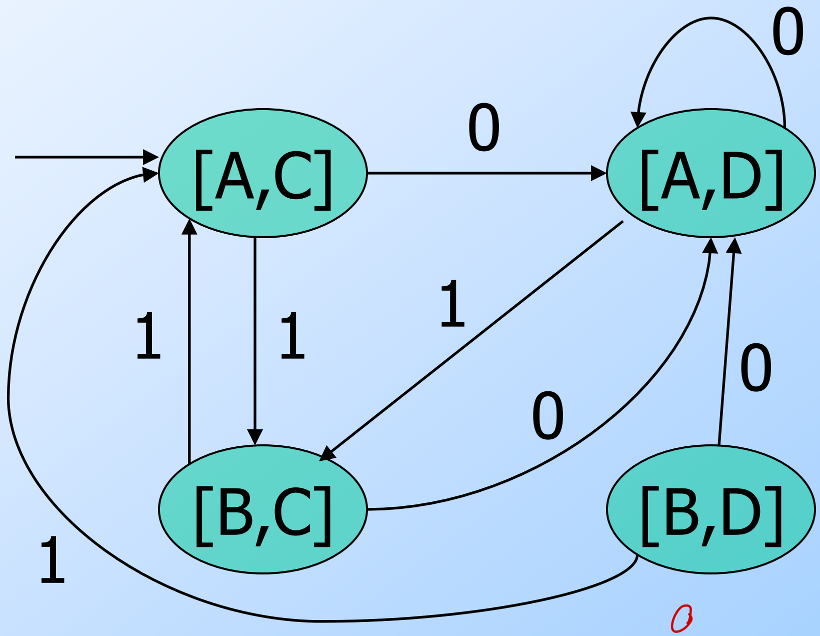


2x2 = 4 state.

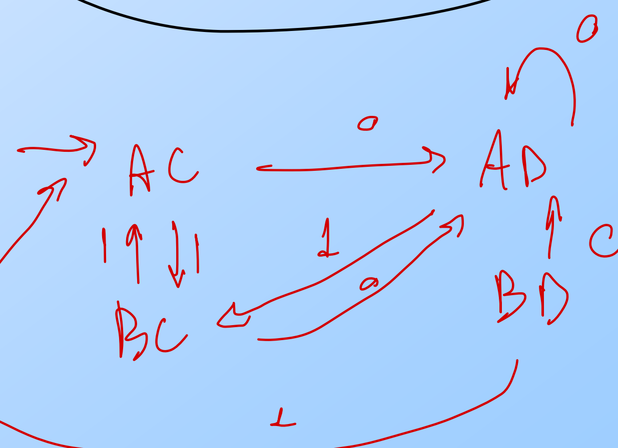
M

2state





ดูไปทีละคู่ AC, AD, BC, BD



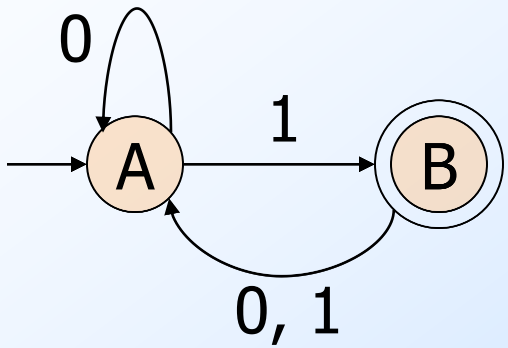
Equivalence Algorithm

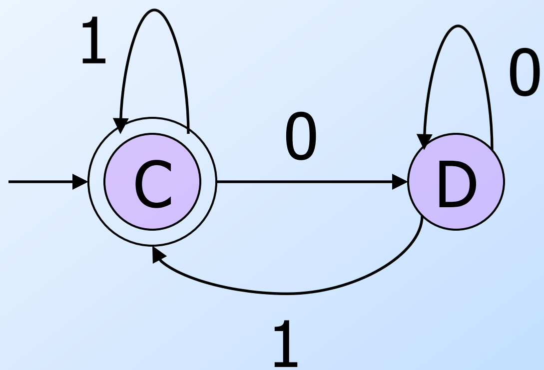
- Make the final states of the product DFA be those states [q, r] such that exactly one of q and r is a final state of its own DFA.
- Thus, the product accepts w iff w is in exactly one of L and M.

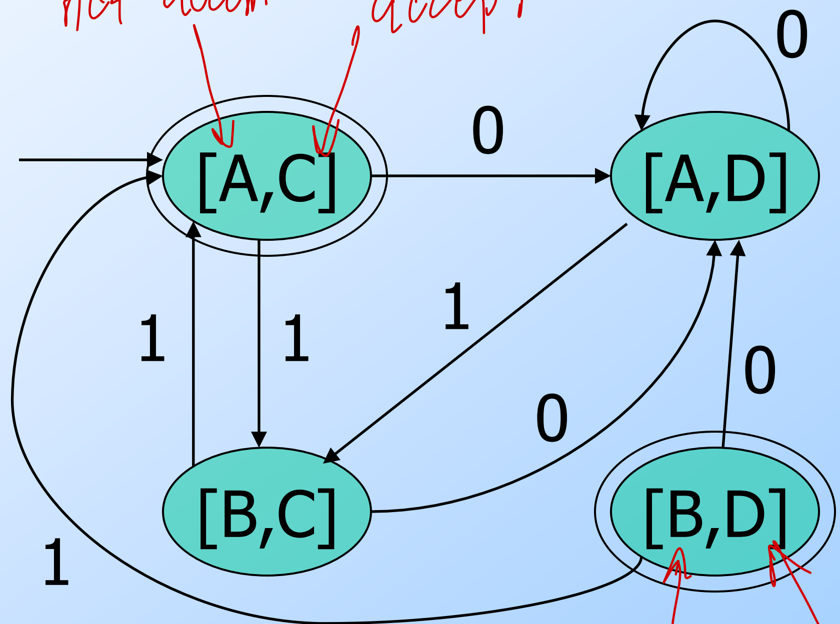
$L = M$?

ถ้า accept $L \neq M$

Example: Equivalence







ดูกัน

Equivalence Algorithm – (2)

ไม่สามารถเข้าถึง final

- The product DFA's language is empty iff $L = M$.
- But we already have an algorithm to test whether the language of a DFA is empty.

Decision Property: Containment

- Given regular languages L and M, is $L \subseteq M$?
- Algorithm also uses the product automaton.
- How do you define the final states [q, r] of the product so its language is empty if $L \subseteq M$?

Answer: q is final; r is not.

$L \subseteq M$? $M \subseteq L$?

ต้องหาว่าถึง final state

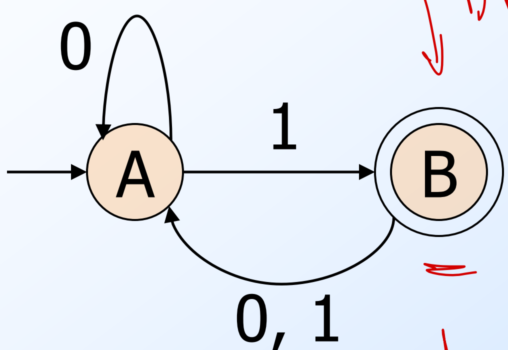
BD AC

Example: Containment

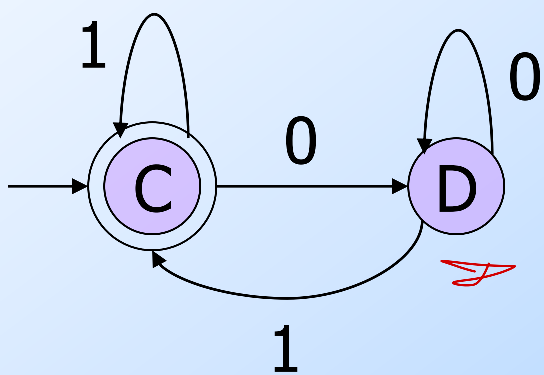
ต้องไม่ accept

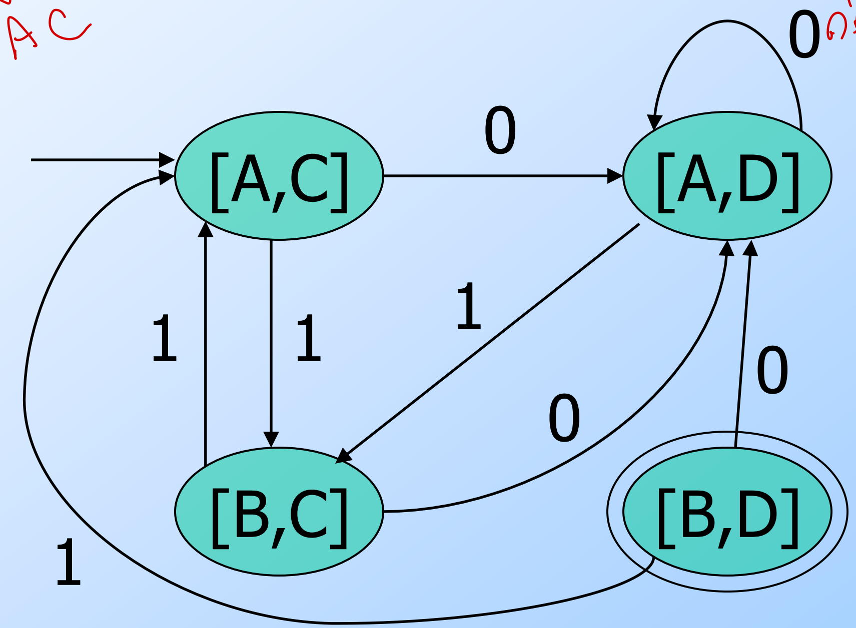
ถึงคือ contain

L



M





Note: the only final state is unreachable, so containment holds.

The Minimum-State DFA for a Regular Language

- In principle, since we can test for equivalence of DFA's we can, given a DFA *A* find the DFA with the fewest states accepting L(A).
- Test all smaller DFA's for equivalence with *A*.
- But that's a terrible algorithm.

Efficient State Minimization

- Construct a table with all pairs of states.

ทขก แยกความต่าง ต่าง.

- If you find a string that *distinguishes* two states (takes exactly one to an accepting state), mark that pair.
- Algorithm is a recursion on the length of the shortest distinguishing string.

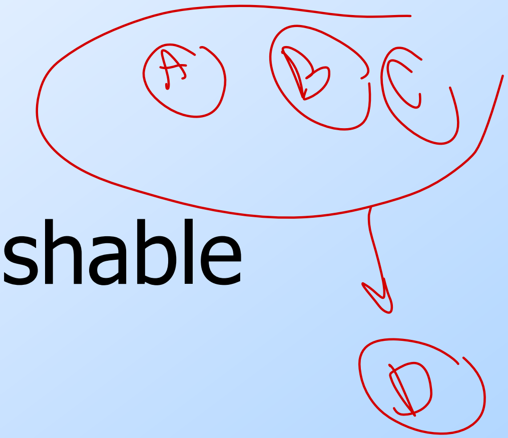
State Minimization – (2)

- Basis: Mark a pair if exactly one is a final state.
- Induction: mark [q, r] if there is some input symbol *a* such that [δ(q,a), δ(r,a)] is marked.
- After no more marks are possible, the unmarked pairs are equivalent and can be merged into one state.

Transitivity of "Indistinguishable"

- If state p is indistinguishable from q, and q is indistinguishable from r, then p is indistinguishable from r.
- Proof: The outcome (accept or don't) of p and q on input w is the same, and the outcome of q and r on w is the same, then likewise the outcome of p and r.

Constructing the Minimum-State DFA



- Suppose $q_1,\ldots,q_k$ are indistinguishable states.
- Replace them by one state q.
- Then $\delta(q_1, a),\ldots, \delta(q_k, a)$ are all indistinguishable states.
  - Key point: otherwise, we should have marked at least one more pair.
- Let $\delta(q, a)$ = the representative state for that group.

Example: State Minimization

DFA

	r	b
→ {1}	{2,4}	{5}
{2,4}	{2,4,6,8}	{1,3,5,7}
{5}	{2,4,6,8}	{1,3,7,9}
{2,4,6,8}	{2,4,6,8}	{1,3,5,7,9}
{1,3,5,7}	{2,4,6,8}	{1,3,5,7,9}
* {1,3,7,9}	{2,4,6,8}	{5}
* {1,3,5,7,9}	{2,4,6,8}	{1,3,5,7,9}

	r	b
→ A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
* F	D	C
* G	D	G

Here it is with more convenient state names

Remember this DFA? It was constructed for the chessboard NFA by the subset construction.

Example – Continued

		r	b
→	A	B	C
	B	D	E
	C	D	F
	D	D	G
	E	D	G
*	F	D	C
*	G	D	G

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

Start with marks for the pairs with one of the final states F or G.

Example – Continued

		r	b
→	A	B	C
	B	D	E
	C	D	F
	D	D	G
	E	D	G
*	F	D	C
*	G	D	G

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

Input r gives no help, because the pair [B, D] is not marked.

ข้ามตัวที่ทำการ BD ไม่มีอะไร เลยไม่ต้อง Mark ส่วน DD มัน ก็สร้างมาจากตัวมันเอง

Example – Continued

อันที่ตกทก.

ดูตัวนี้

	r	b
A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
* F	D	C
* G	D	G

mark

ดีกว่า

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

But input b distinguishes {A,B,F} from {C,D,E,G}. For example, [A, C] gets marked because [C, F] is marked.

Example – Continued

		r	b
→	A	B	C
	B	D	E
	C	D	F
	D	D	G
	E	D	G
*	F	D	C
*	G	D	G

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

[C, D] and [C, E] are marked because of transitions on b to marked pair [F, G].

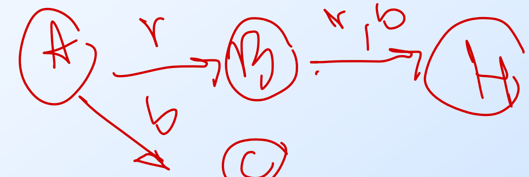
Example – Continued

	r	b
→ A	B	C
B	D	E
C	D	F
D	D	G
E	D	G
* F	D	C
* G	D	G

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

[A, B] is marked because of transitions on r to marked pair [B, D].

[D, E] can never be marked, because on both inputs they go to the same state.



Example – Concluded

4 state

	r	b
→ A	B	C
B	~~D~~	~~E~~
C	~~D~~	F
H ~~D~~	~~D~~	G
~~E~~	~~D~~	G
* F	~~D~~	C
* G	~~D~~	G

1 state.

	r	b
→ A	B	C
B	H	H
C	H	F
H	H	G
* F	H	C
* G	H	G

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

Replace D and E by H.
Result is the minimum-state DFA.

Eliminating Unreachable States

- Unfortunately, combining indistinguishable states could leave us with unreachable states in the "minimum-state" DFA.
- Thus, before or after, remove states that are not reachable from the start state.