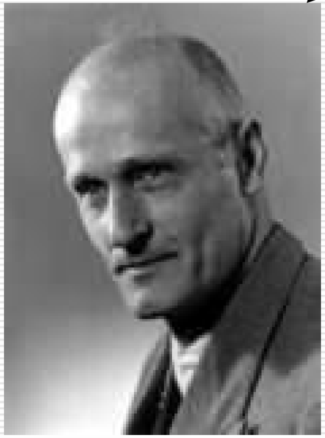
Lecture 3 Regular Expressions

- Regular Languages
- regular expressions
- their equivalence and
- pumping lemma

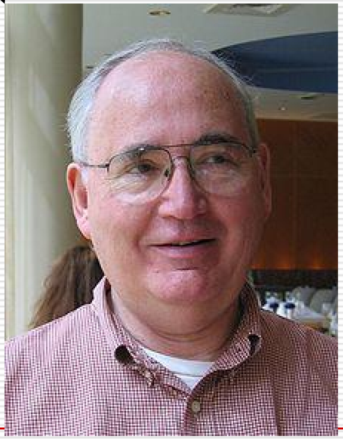
Theory at Princeton: early period

Alonzo Church

Turing



Kleene



Scott

Rabin

Equivalence of NFAs and DFAs

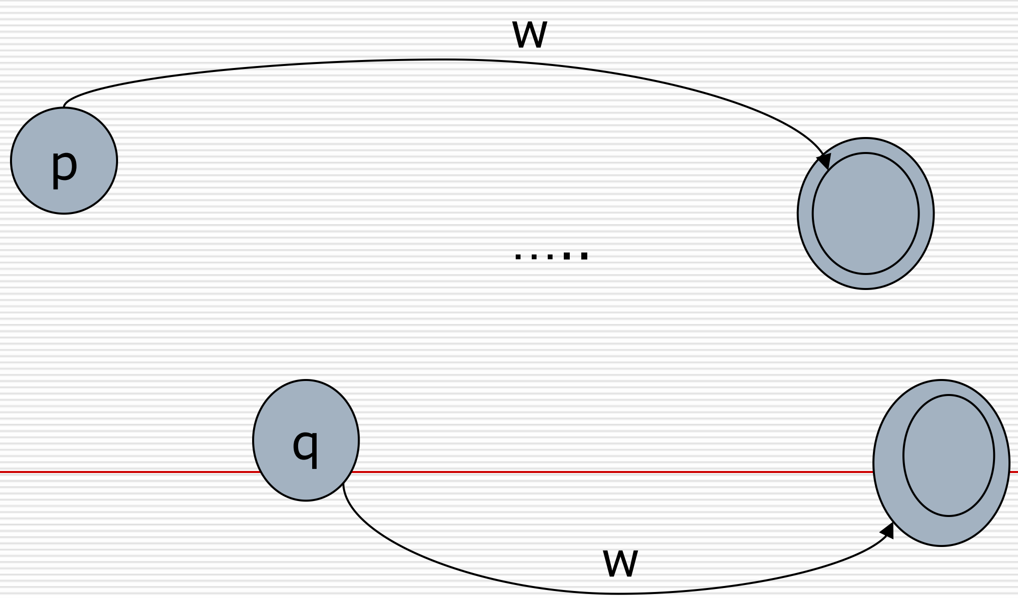
- Every nondeterministic finite automaton has an equivalent deterministic finite automaton.
- Proof: Subset construction by Scott and Rabin(1959).

Summary

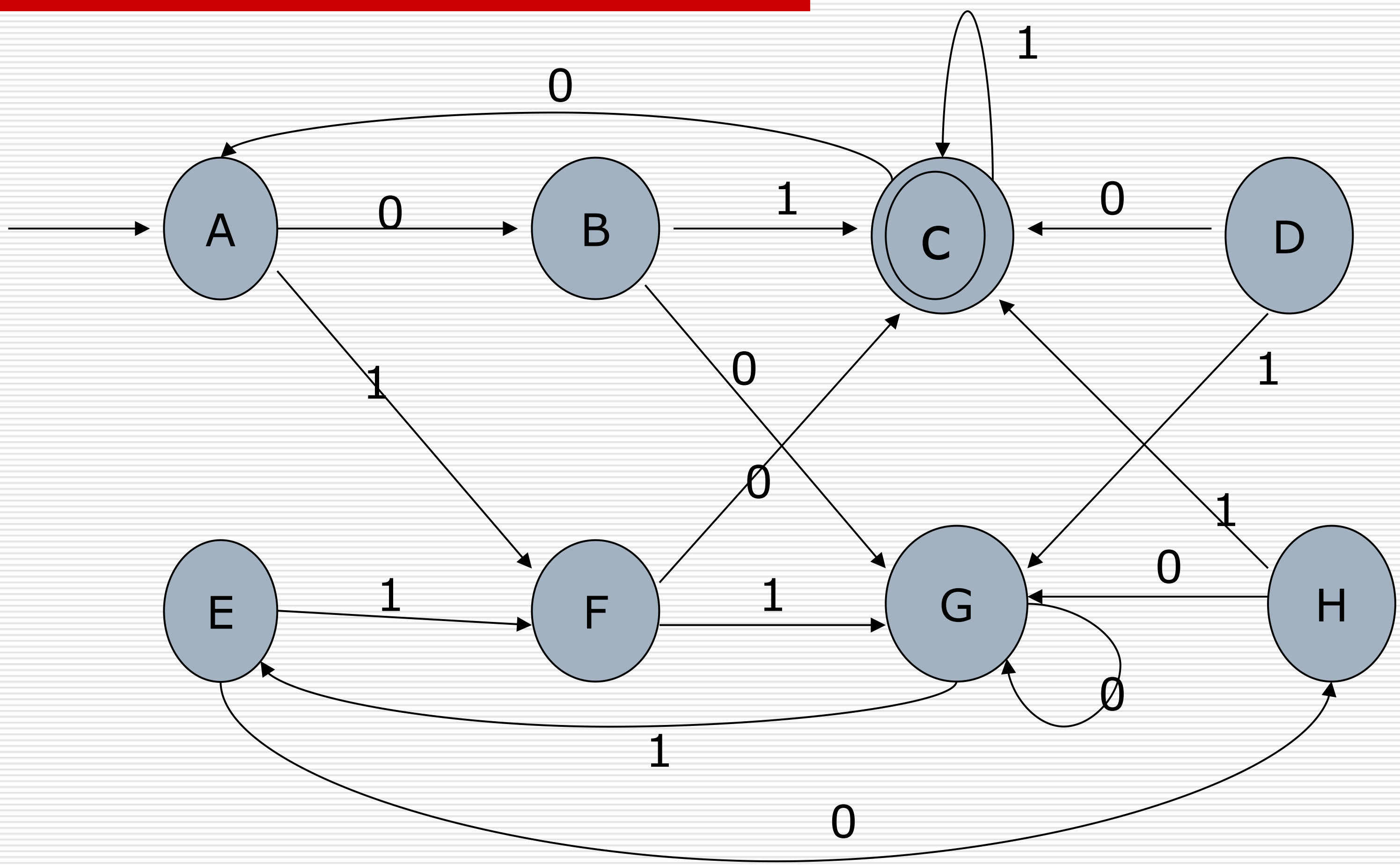
- DFA’s, NFA’s both accept exactly the same set of languages: the regular languages (next time).
- The NFA types are easier to design and may have exponentially fewer states than a DFA. Nondeterminism is a synonyms for guess or search.
- But only a DFA can be implemented!

Equivalence

- Given a DFA, two states p and q are equivalent if
- For all input strings w, $\hat{\delta}(p,w)$ is an accepting state iff $\hat{\delta}(q,w)$ is an accepting state.



Example



Distinguishable states

- Base case: If p is an accepting state and q is a nonaccepting, then the pair {p,q} is distinguishable;
- Induction case: Let p and q be states such that for some input symbol a, $r = \delta(p,a)$ and $s = \delta(q,a)$ are a pair of distinguishable states. Then {p, q} is a pair of distinguishable states.

Are accepting states indistinguishable?

Table-filling algorithm (Where should we start?)

	A	B	C	D	E	F	G	H
A								
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	

Why does this algorithm give the minimal automata?

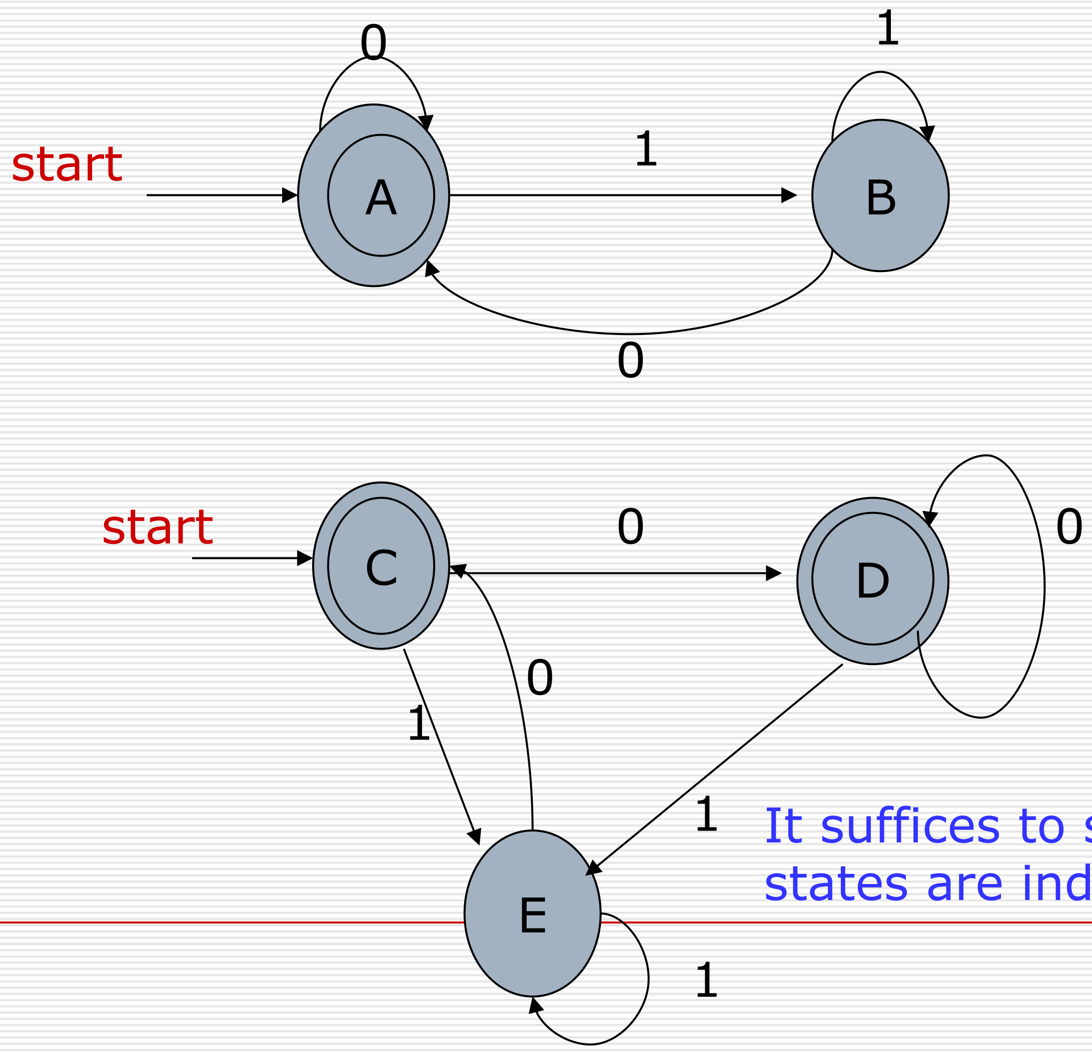
Testing equivalence of DFAs

- Recall that
  - two DFAs are equivalent if they accept the same language.

- Testing equivalence of DFAs can be transformed into testing equivalence of their starting states.

Does this approach work for NFA?

Example: Two equivalent DFAs



It suffices to show that the starting states are indistinguishable.

Design Automata

Suppose $\Sigma = \{0,1\}$ Construct a finite automaton M recognizing the following languages:

- A is the set of all strings with an odd number of 1’s.
- A is the set of all strings 001 as a substring
- A is the set of all strings except 11 and 111
- The set of all strings except the empty string

1. How do you know which one among equivalent automata is the minimal or the optimal?
2. Is there an algorithm for designing automata?

Regular languages and Regular Operations

- A languages is called a regular language if some finite automaton recognizes it.

Let A and B be two languages,

- Union: $A \cup B = \{x \mid (x \in A) \vee (x \in B)\}$
- Concatenation: $A \circ B = \{xy \mid (x \in A) \wedge (y \in B)\}$
- Star:

$$A^* = \{x_1 x_2 \cdots x_k \mid x_i \in A\}(0 \le i \le k)\}$$

The empty string always belongs to A*.

Example

- Let A ={001, 10, 111},B={ε ,001}

Then

$$A \cup B = \{001, 10, 111, \varepsilon\}$$

$$A \circ B = \{001, 001001, 10, 10001, 111, 111001\}$$

$$A^{*} = \{\varepsilon, 001, 10, 111, 001001 10001, 111001, \cdots\}$$

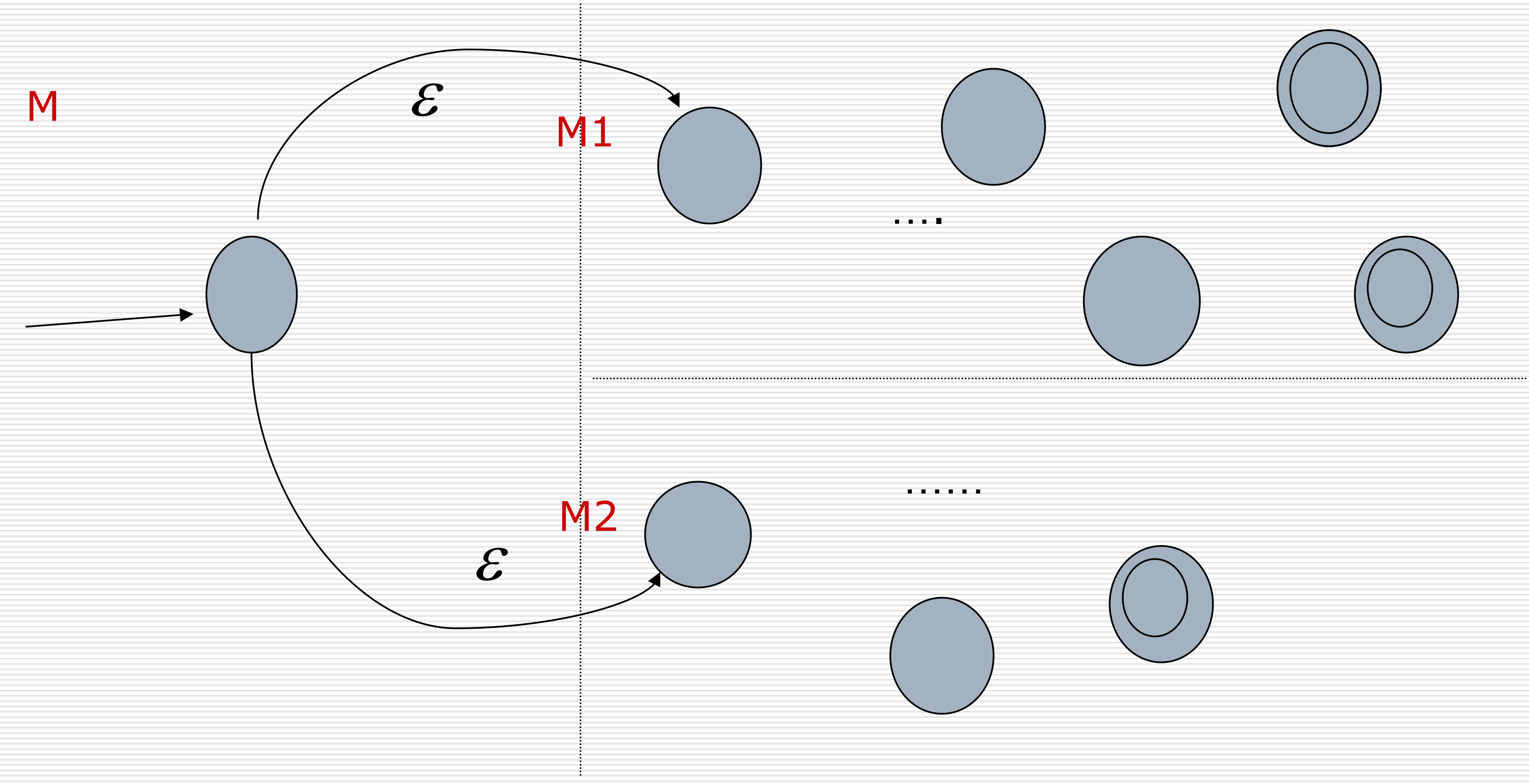
Closure Properties

- The class of regular languages is closed under union operation, and concatenation and star.

That is to say, if both A and B are regular languages, so are $A \cup B, A \circ B, A^*$.

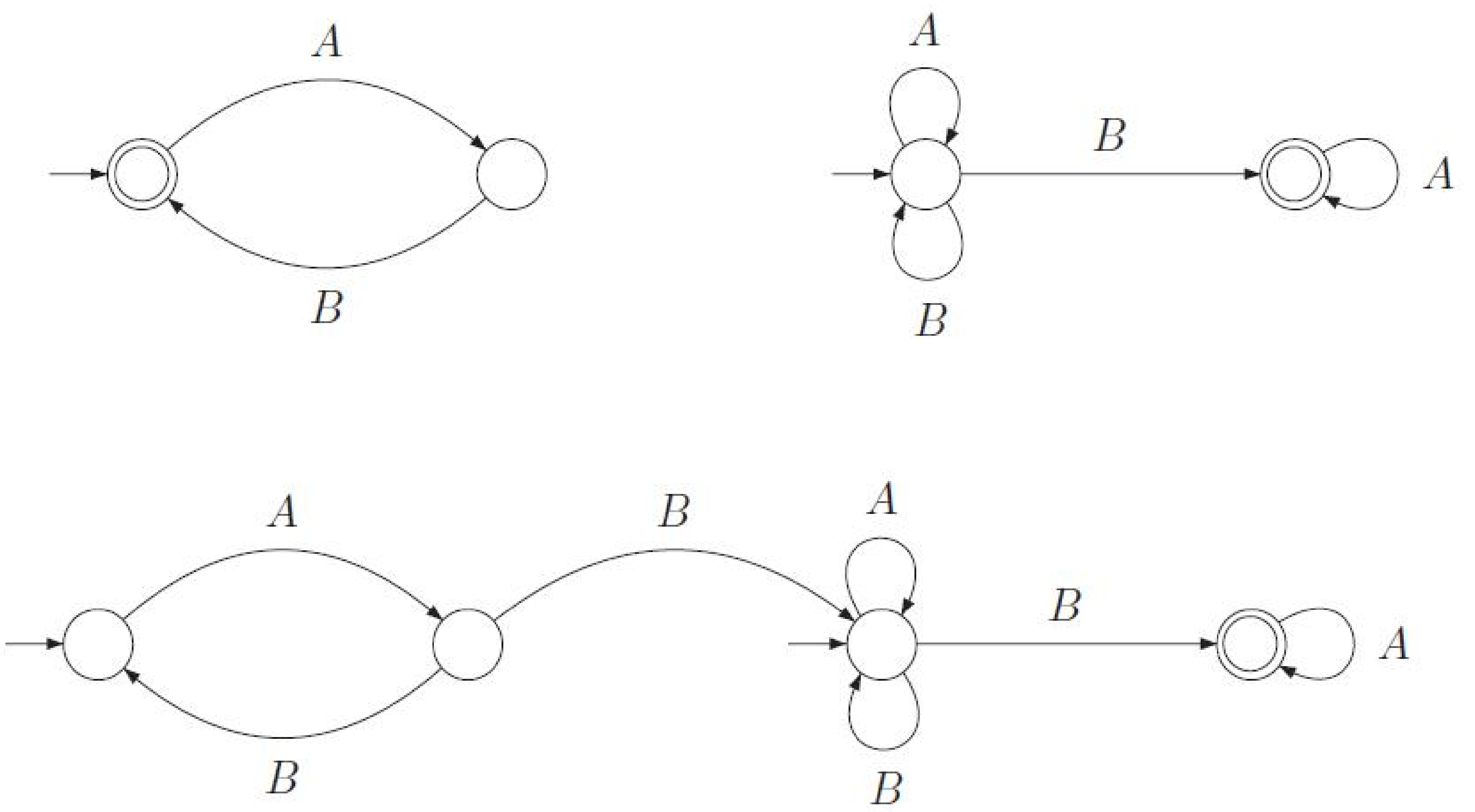
Where did we use Empty string + Non-determinism in the proof?

An illustration: Closed under union

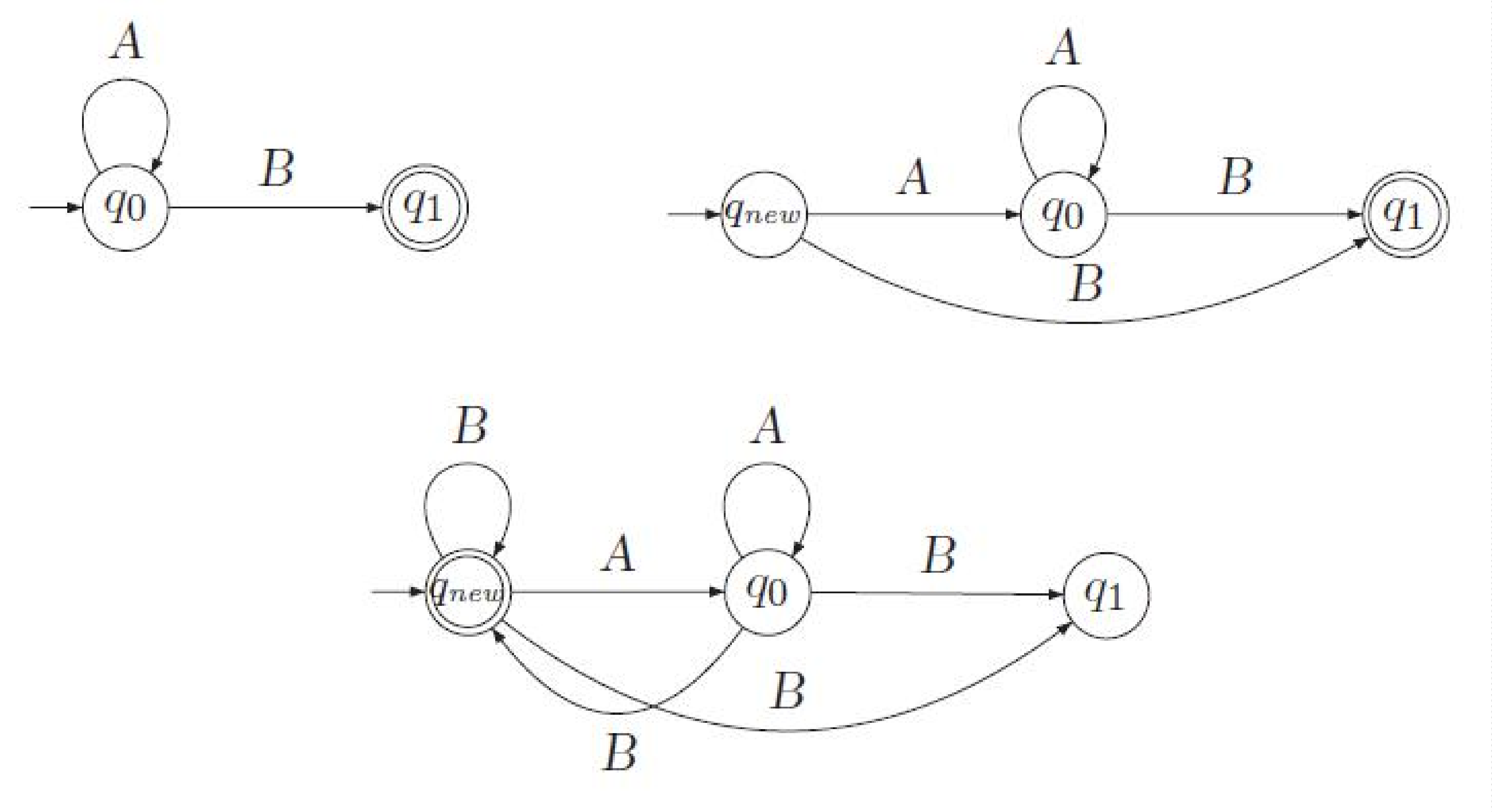


What if we don't use the empty string?

Concatenation: without epsilon



Star-operator: without epsilon



Regular Expressions (Structural Induction)

- R is a regular expression if R is

① a for some a in the alphabet set Σ

② The empty string ε ,

③ Empty set ϕ

Basis

④ $R_1 \cup R_2$, where R_1 and R_2 are regular expressions

⑤ $R_1 \circ R_2$ where R_1 and R_2 are regular expressions

⑥ R_1^* where R_1 is a regular expression

Induction

Let R be a regular expression. L(R) denotes the language represented by R.

What (regular) languages do they represent?

Languages described by regular expressions

- In parallel to the inductive definition of regular expressions, the corresponding languages are

$$L(a) = \{a\}$$

$$L(\varepsilon) = \{\varepsilon\}$$

$$L(\phi) = \phi$$

$$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$$

$$L(R_1 \circ R_2) = L(R_1) \circ L(R_2)$$

$$L(R^*) = (L(R))^*$$

Example: Kleene Star

Let $\Sigma = \{0,1\}$

Then describe the following languages

$$0^*1^*$$

$$\Sigma^* 0 \Sigma^* 1 \Sigma^* 1^*$$

$$(0 \cup 01 \cup 11)\Sigma^*$$

$$\phi^* = \{\varepsilon\}$$

What are the differences between the empty expression ϕ and the empty string ε?

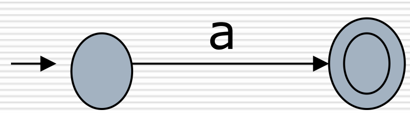
Equivalence

- Main Theorem: A language is regular (i.e.,recognized by a finite automaton) if and only if some regular expression describes it.

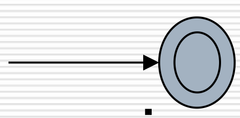
The right-to-left is easy

- According to the closure properties of the class of regular languages, we only need to show the basis cases

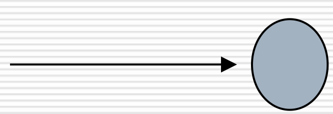
1. R =a some letter in the alphabet set.



2. R=the empty string



3. R=the empty expression



This is a proof
by structural induction!

Example

- Build NFAs from the following regular expressions

$$(ab \cup a)^*$$

$$(a \cup b)^* aba$$

The poof of the other direction is hard

Proof Idea: Given a deterministic FA,

- first we construct an equivalent generalized nondeterministic FA (GNFA)
- next construct an equivalent GNFA by decreasing the number n of states one by one until n=2
- Obtain the regular expression from the 2-state DNFA

Generalized nondeterministic FAs

- A generalized nondeterministic FA is a nondeterministic FA wherein the transition arrows may have any regular expressions as labels.

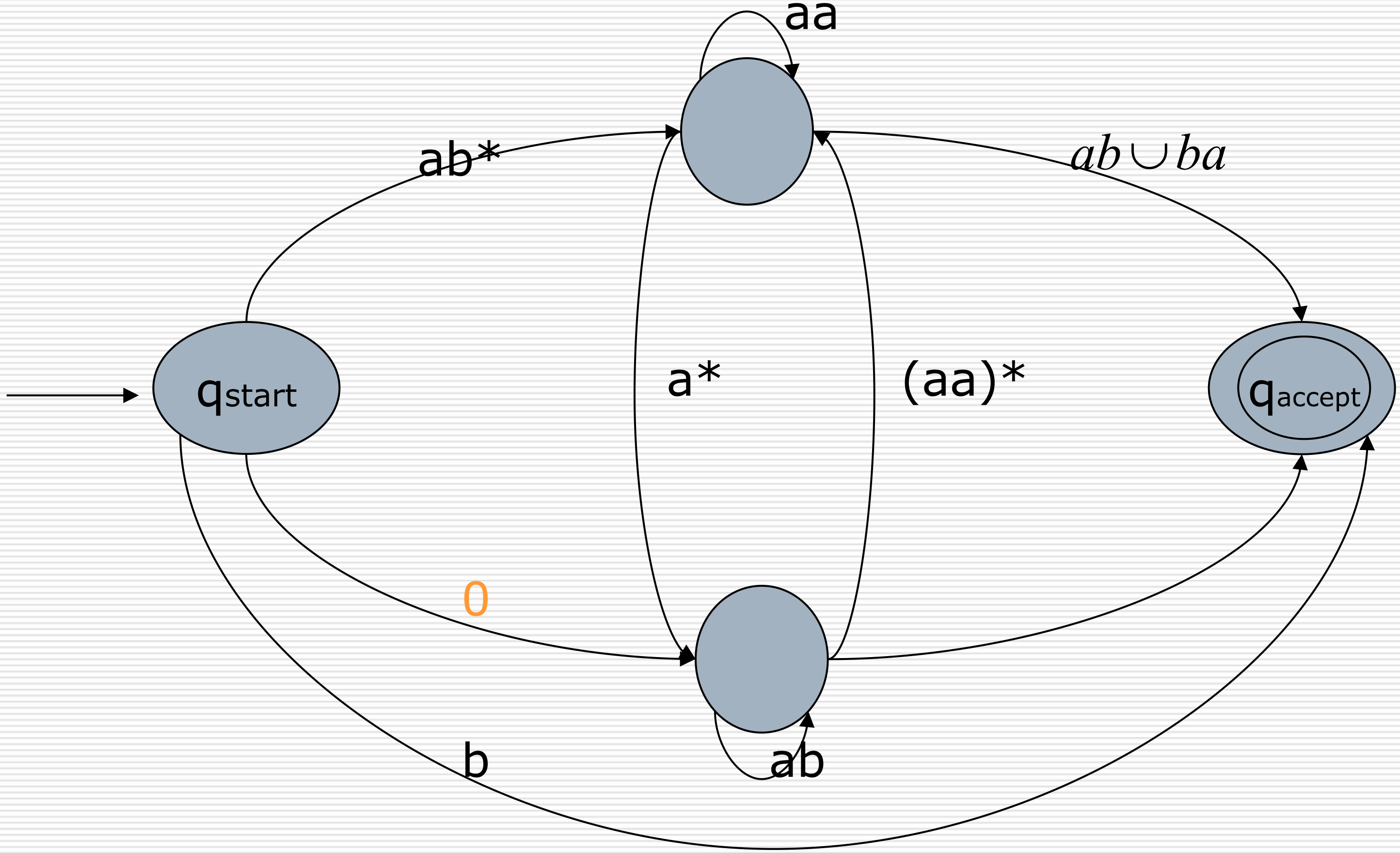
Further we require GNFA satisfy the following three conditions:

1. The start state has transition arrows going to every other state but no arrows coming in from any other state:
2. The dual condition for the accept state, the start state is different from the accept state;
3. Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

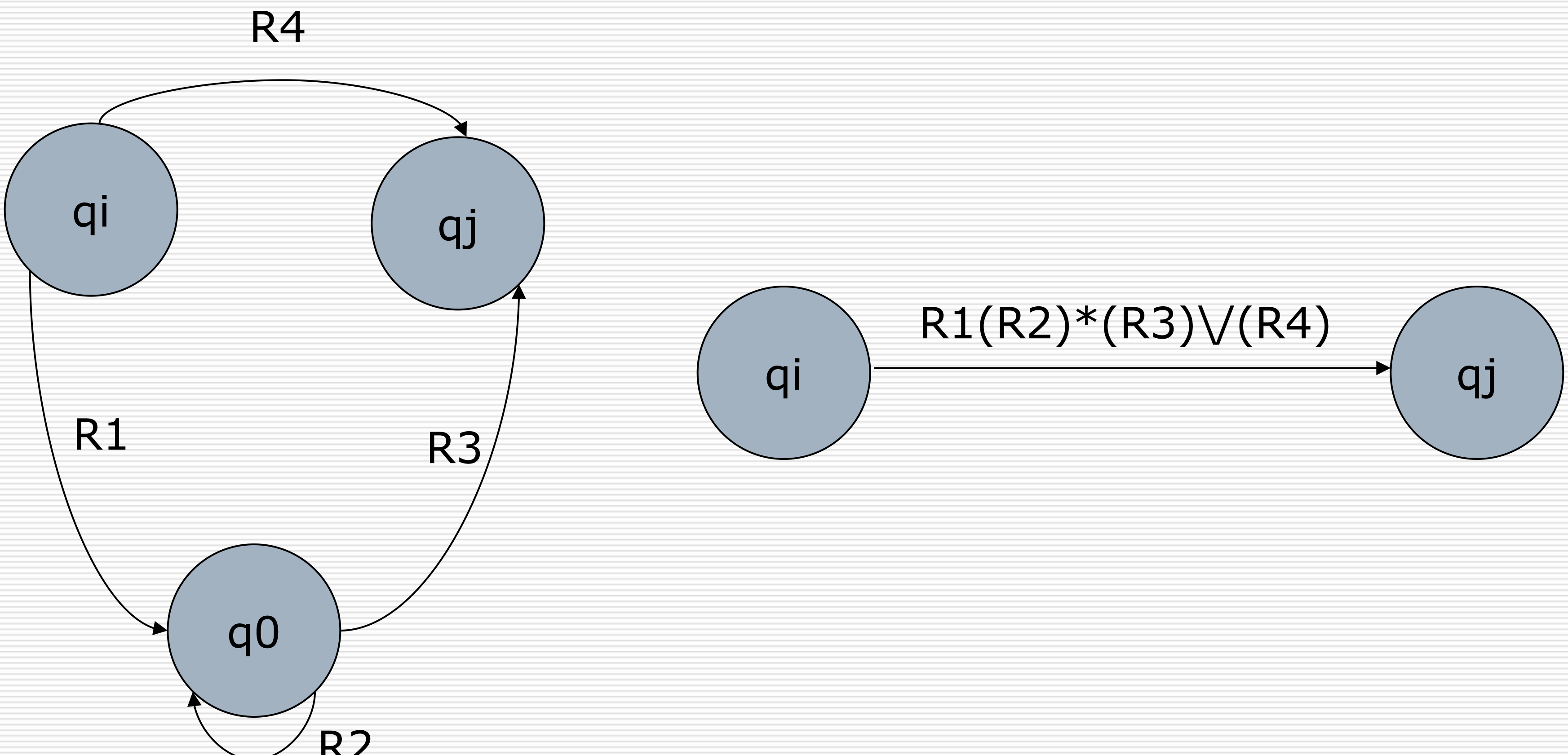
$$\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow \mathrm{R}$$

Are these requirements reasonable?

Example: GNFA



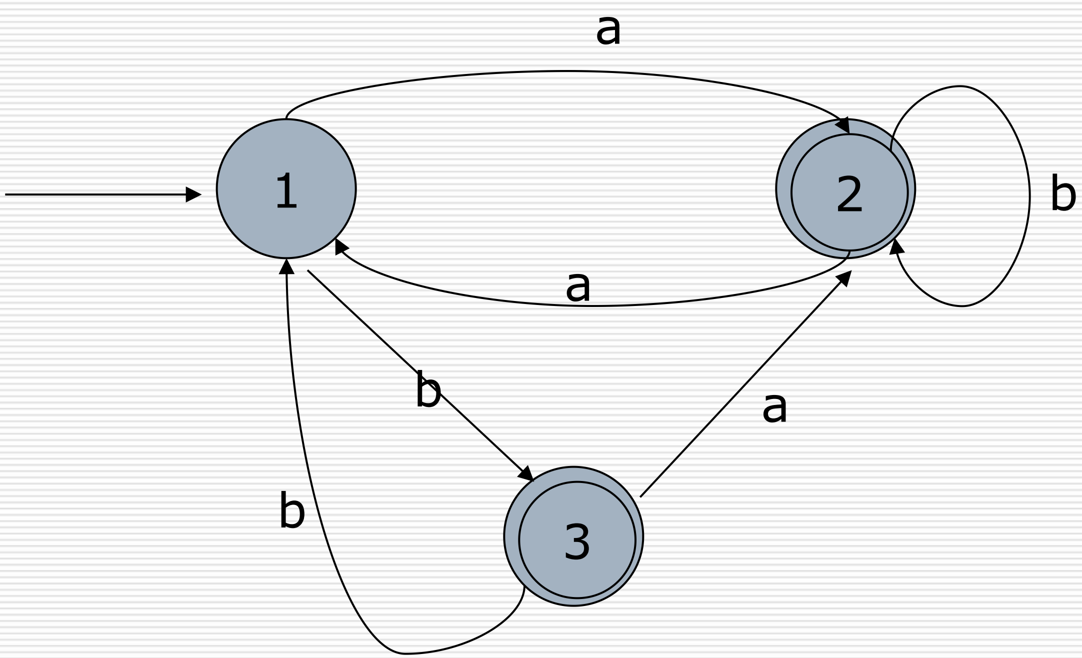
Crucial step



What if qi = qj?

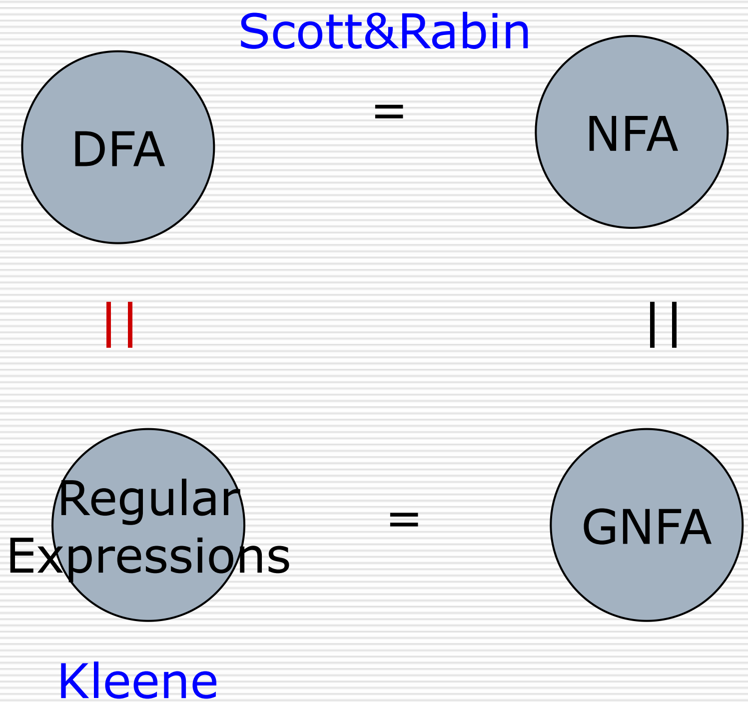
Converting DFAs to Regular expressions

- Example: a 3 state DFA



Automata = Regular expressions

In what sense?



How do we **arithmetize** FAs?

Kleene algebra: the algebra for regular expressions (Dexer Kozen 1994)

$$
\begin{aligned}
a + (b + c) &= (a + b) + c \\
a + b &= b + a \\
a + 0 &= a \\
a + a &= a \\
a(bc) &= (ab)c \\
1a &= a \\
a1 &= a \\
a(b + c) &= ab + ac \\
(a + b)c &= ac + bc
\end{aligned}
$$

$$
\begin{aligned}
0a &= 0 \\
a0 &= 0 \\
1 + aa^* &\leq a^* \\
1 + a^*a &\leq a^*
\end{aligned}
$$

$$
\begin{aligned}
b + ax \leq x &\rightarrow \\
b + xa \leq x &\rightarrow
\end{aligned}
$$

where $\leq$ refers to the natural partial order

$$a \leq b \leftrightarrow a +$$

Exercise: Prove that (a*b*)* = (a+b)*

Negative results delineate the limits of computations!

Pumping Lemma: necessary cond.

- If A is a regular language, then there is a number p (the pumping length) where, if s is any string in A of length at least p, then s may be divided into three pieces, s=xyz, satisfying the following conditions:

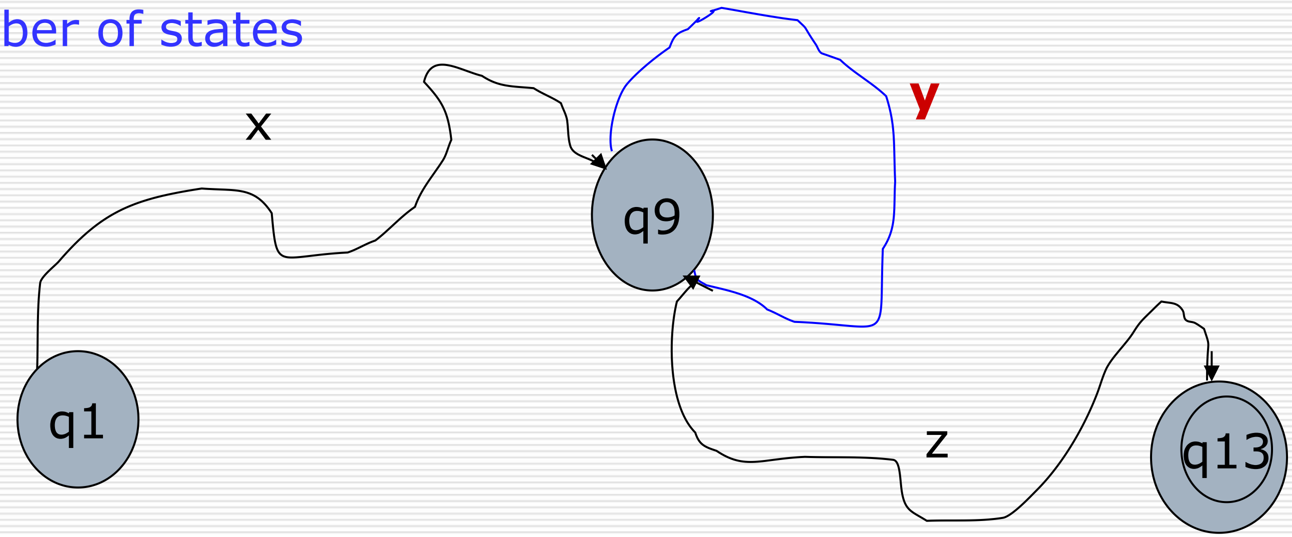
1. For each $i \geq 0, xy^i z \in A$
2. $|y| > 0$, and
3. $|xy| \leq p.$.

Proof Idea

- s=s1 s2 s3 s4 s5 s6….sn string in A

q1 q3 q20 q9 q17 q9 q13 state

P is the number of states



q9 is the **first** state that repeats

Proof steps

- Proof by contradiction: B is nonregular

1. First assume the language B is regular and apply the pumping lemma to obtain the pumping length p
2. Next find a string s in B of length at least p which cannot be pumped.

- Discuss s in several cases

So we arrive at a contradiction. Conclude that B is **NOT** regular

Example

- Show that the language $\{0^n1^n : n \geq 0\}$ is not regular

Proof by contradiction: Suppose that it were regular. Let p be the pumping length given by the Pumping Lemma. Consider the string

$$s = 0^p1^p$$

1. y consists only of 0s;
2. Y consists only of 1s;
3. Y consists of both 0s and 1s.

Discuss in cases

We arrive at a contradiction. So it is not regular.

More examples

- Prove that the following languages are not regular:

1. C={w|w has an equal number of 0s and 1s}.
2. F = $\{ww \mid w \in \{0,1\}^*\}$
3. D = $\{1^{n^2} : n \geq 0\}$
4. E= $\{0^i 1^j : i > j\}$

Exercises: 1.6(a)(b)(c), 1.7(a)(b)(c) 1.16, 1.19, 1.21,1.29(a)(b)