Applications

- Finite state machines are a useful model for many important kinds of hardware and software. e.g.
  - Software for designing digital circuits
  - Lexical analyzer of a compiler
  - Searching for keywords in a file or on the Web
  - Software for verifying finite state systems, such as communication protocols
  - Operating System (UNIX grep)
  - Text Editors
  - Markup Languages (HTML, XML)
  - Natural Language Processing

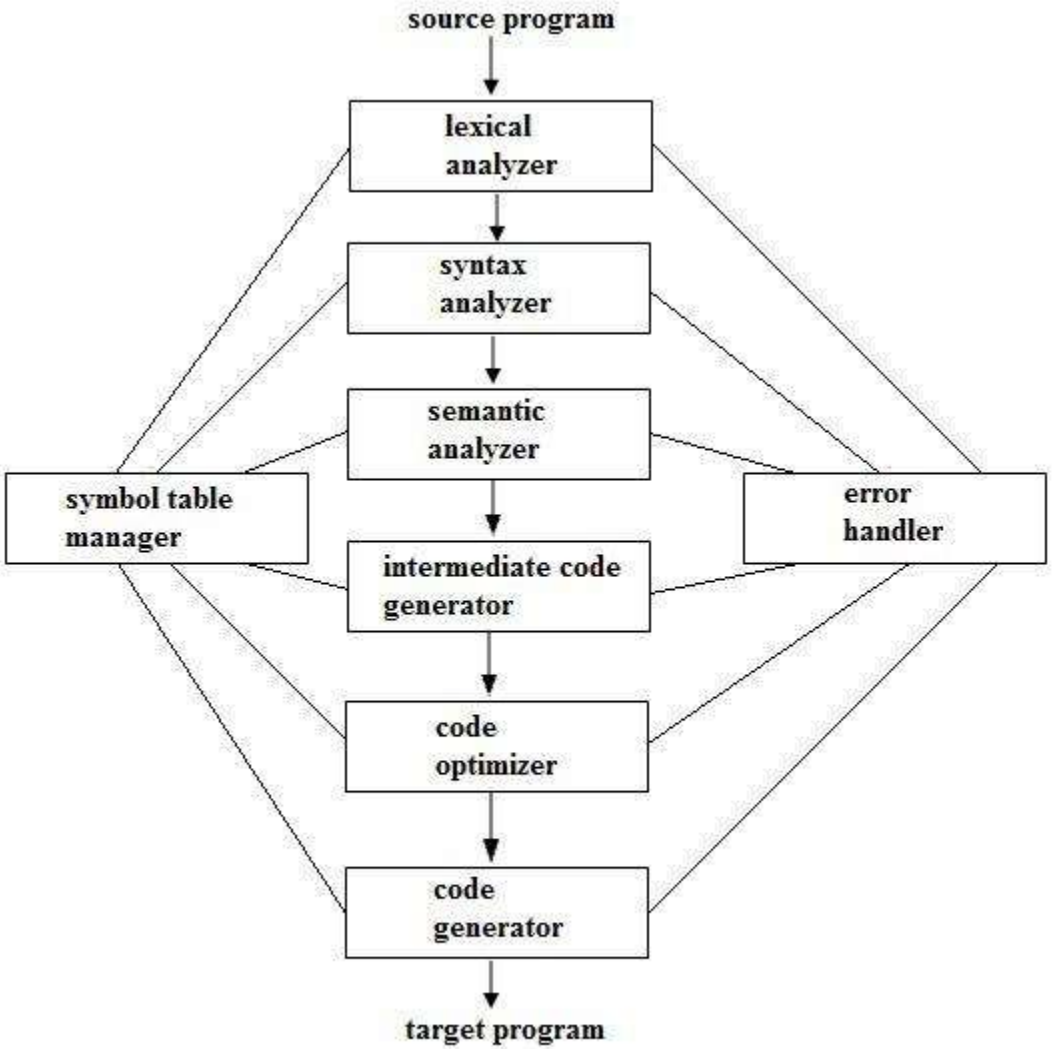
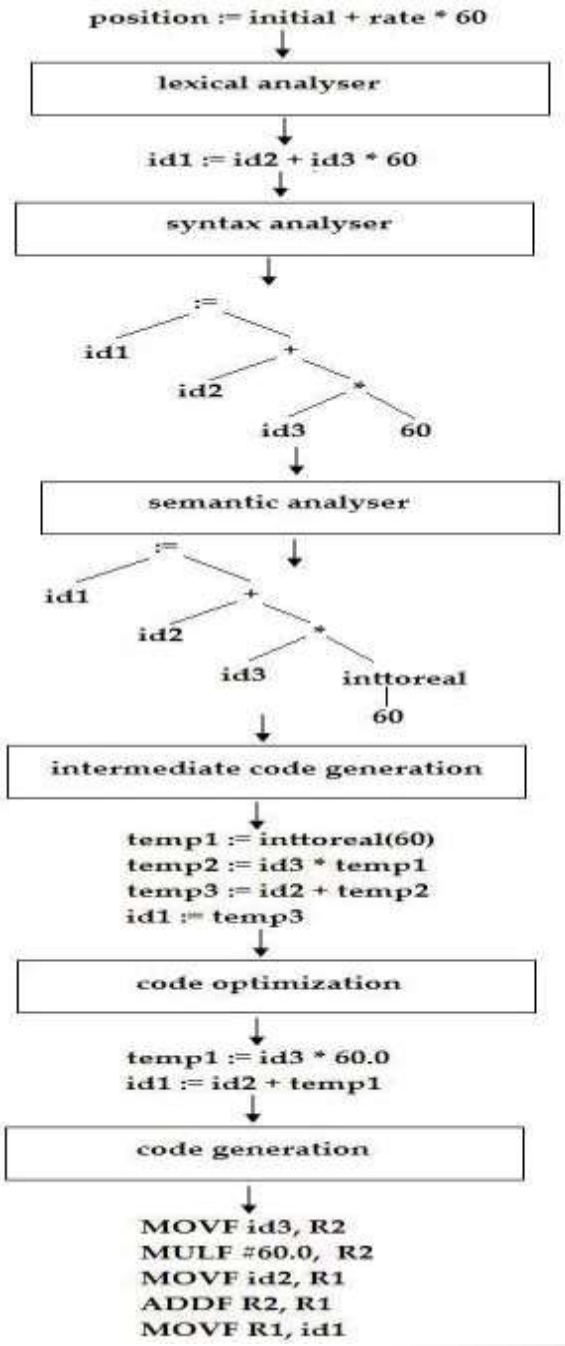


Fig 1.5 Phases of a compiler



Lexical Analyzer

- **Lexical Analyzer** reads the source program character by character and returns the *tokens* of the source program.
- A *token* describes a pattern of characters having same meaning in the source program. (such as identifiers, operators, keywords, numbers, delimeters and so on)

Ex: newval := oldval + 12 => tokens:

newval	identifier
:=	assignment operator
oldval	identifier
+	add operator
12	a number

- Puts information about identifiers into the symbol table.
- Regular expressions are used to describe tokens (lexical constructs).
- A (Deterministic) Finite State Automaton can be used in the implementation of a lexical analyzer.

Properties of Regular Languages

- *Pumping Lemma*. Every regular language satisfies the pumping lemma. If somebody presents you with fake regular language, use a pumping lemma to show a contradiction.
- *Closure properties*. Building automata from components through operations, e.g. given L and M we can build an automaton for $L \cap M$.
- *Decision properties*. Computational analysis of a automata, e.g. are two automata equivalent.
- *Minimization techniques*. We can save money since we can build smaller machines.

Non-regular languages

(Pumping Lemma)

Non-regular languages

$$\{a^n b^n : \ n \geq 0\}$$

$$\{vv^R : \ v \in \{a,b\}^*\}$$

Regular languages

a^*b

$b^*c + a$

$b + c(a+b)^*$

$etc...$

How can we prove that a language L is not regular?

Prove that there is no DFA or NFA or RE that accepts L

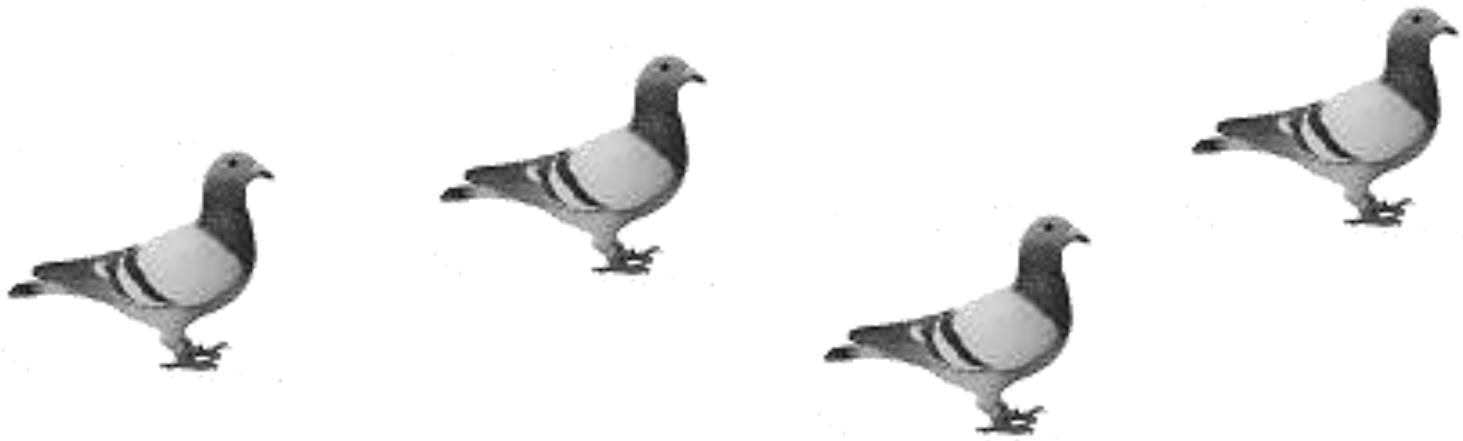
Difficulty: this is not easy to prove

(since there is an infinite number of them)

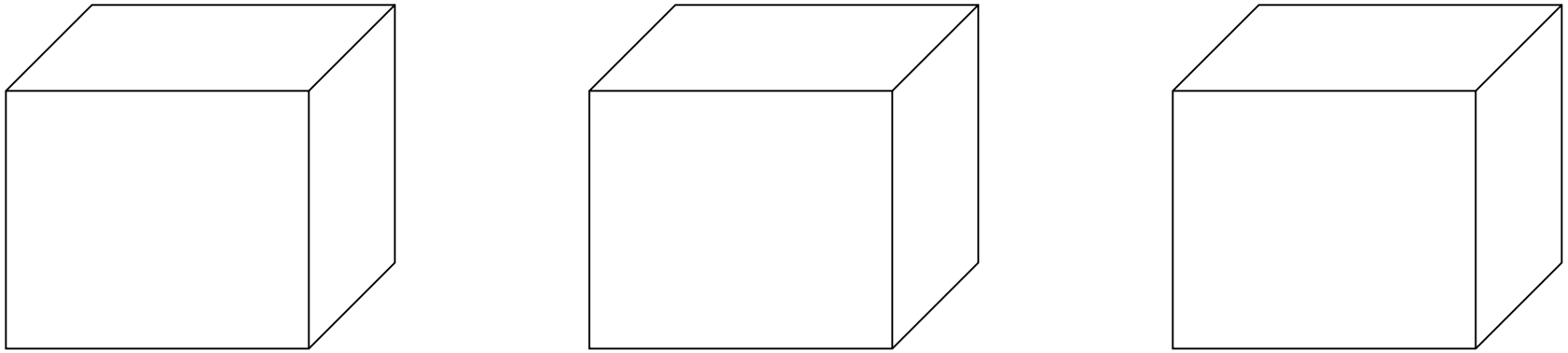
Solution: use the Pumping Lemma !!!

The Pigeonhole Principle

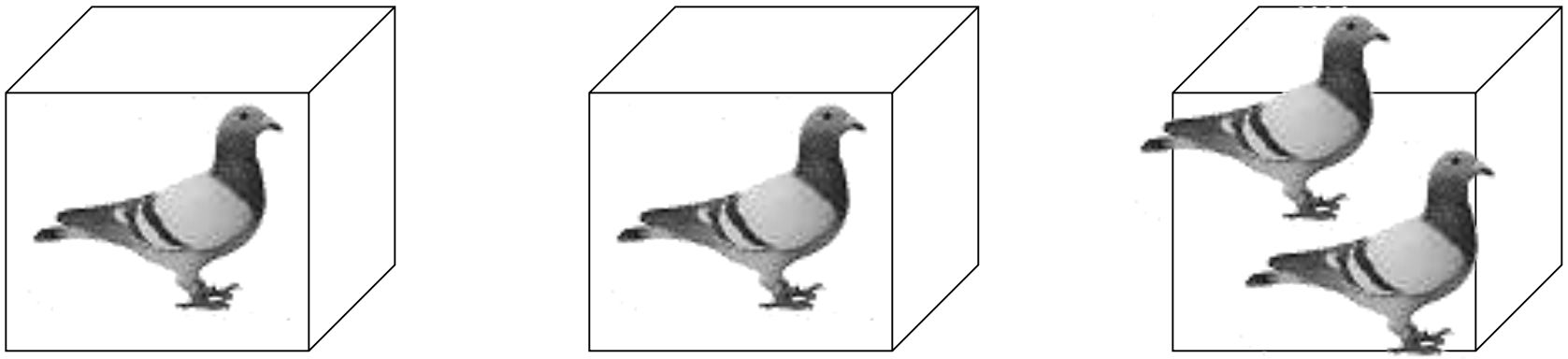
4 pigeons



3 pigeonholes

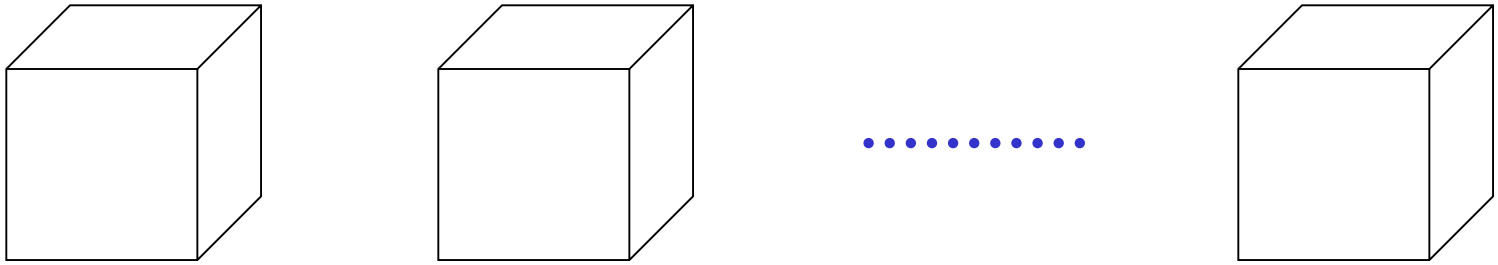


A pigeonhole must
contain at least two pigeons



n pigeons

m pigeonholes $n > m$



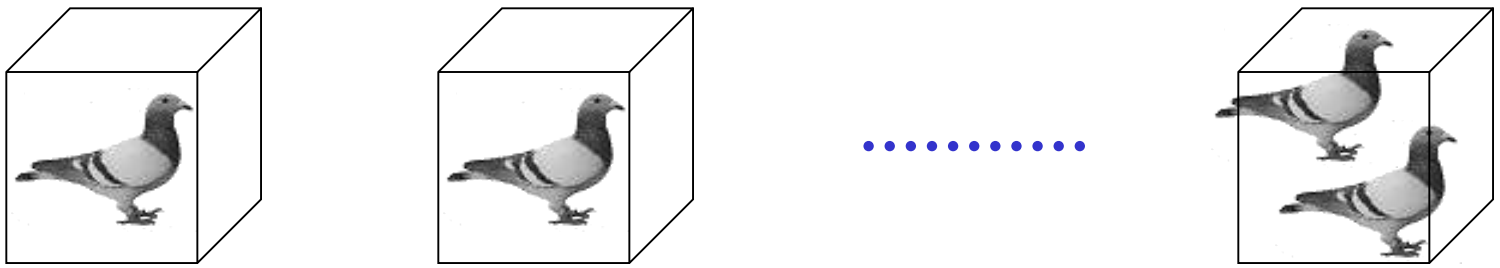
The Pigeonhole Principle

n pigeons

m pigeonholes

$n > m$

There is a pigeonhole with at least 2 pigeons

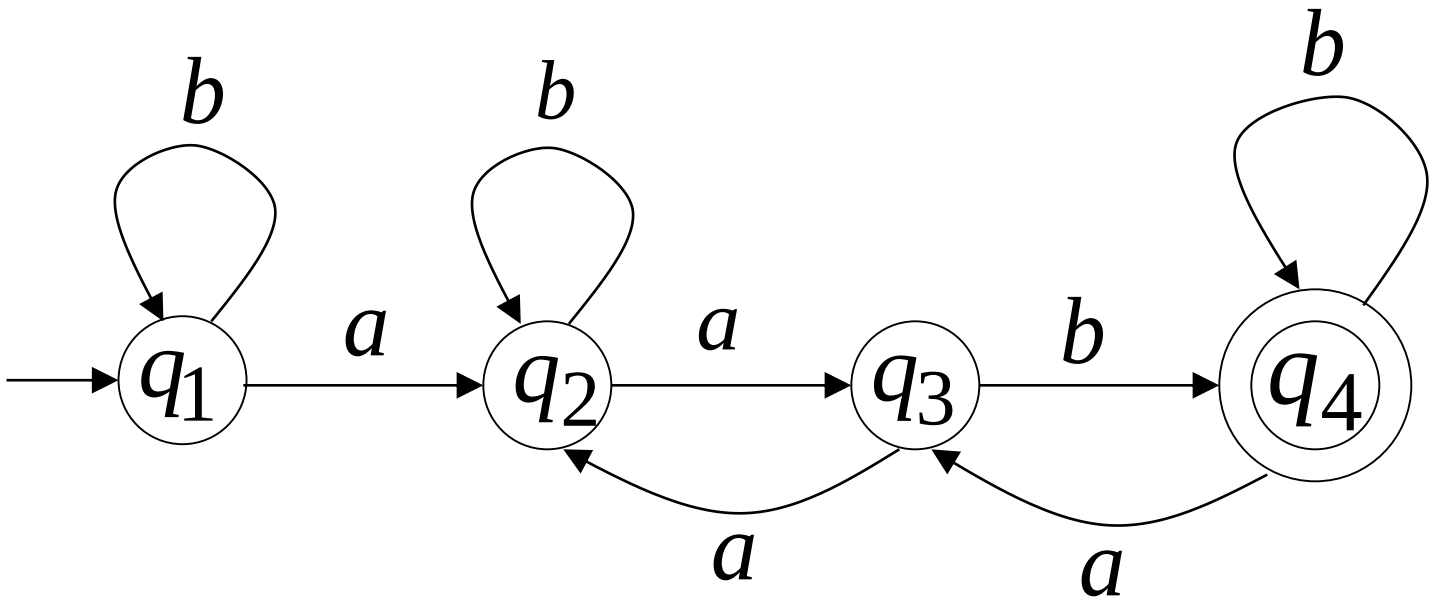


The Pigeonhole Principle

and

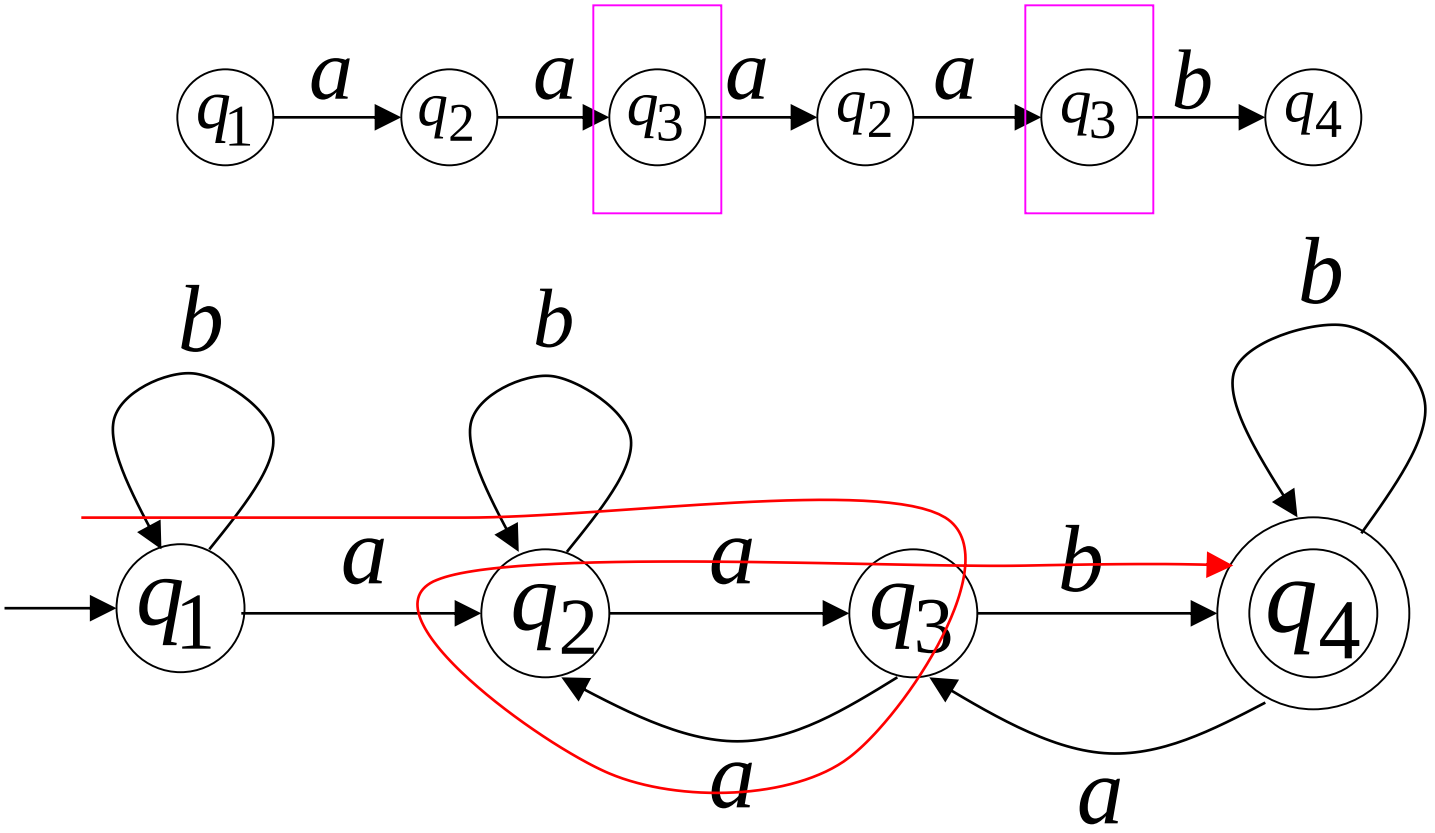
DFAs

Consider a DFA with 4 states

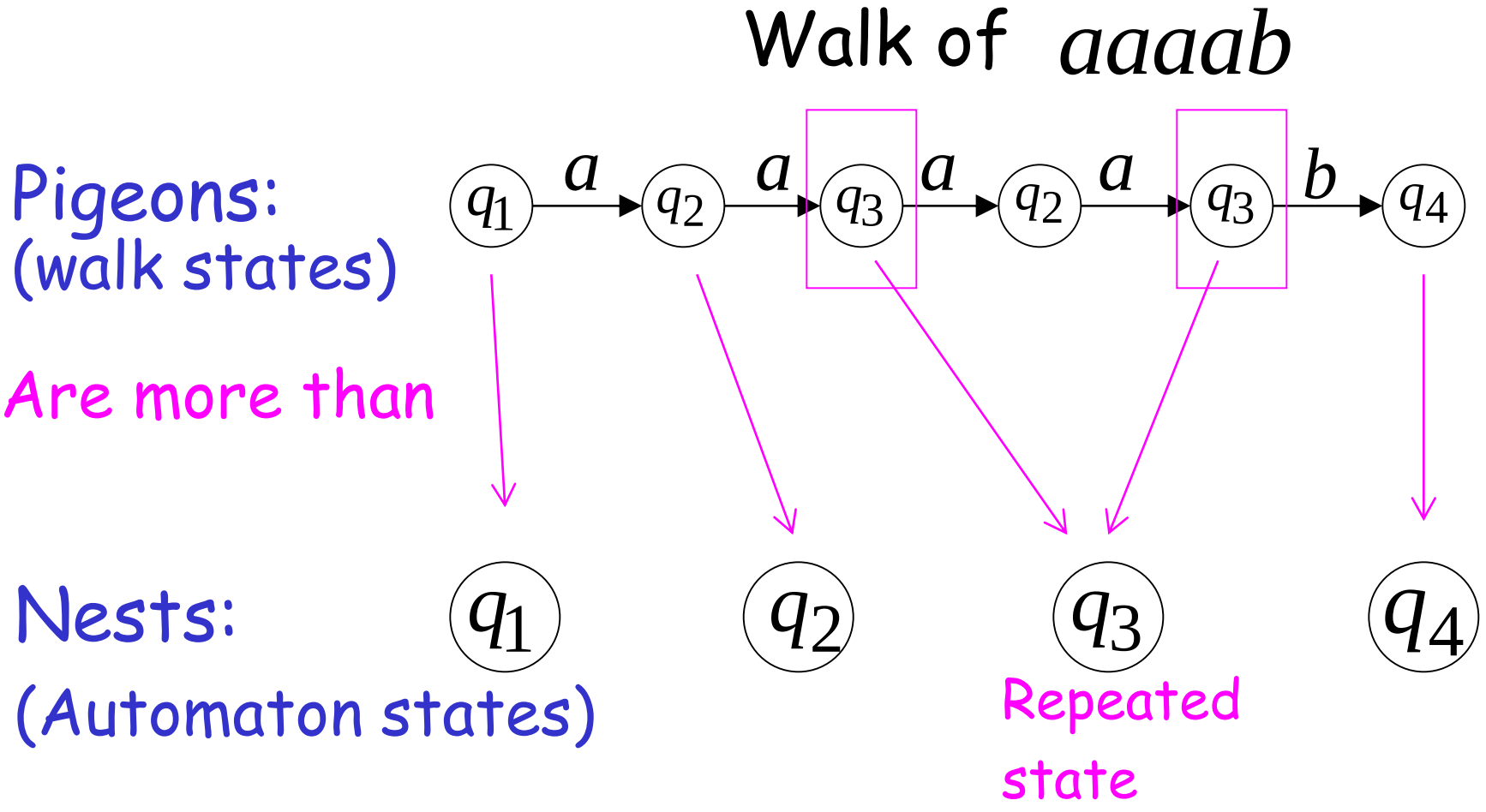


Consider the walk of a "long" string: $aaaab$ (length at least 4)

A state is repeated in the walk of $aaaab$



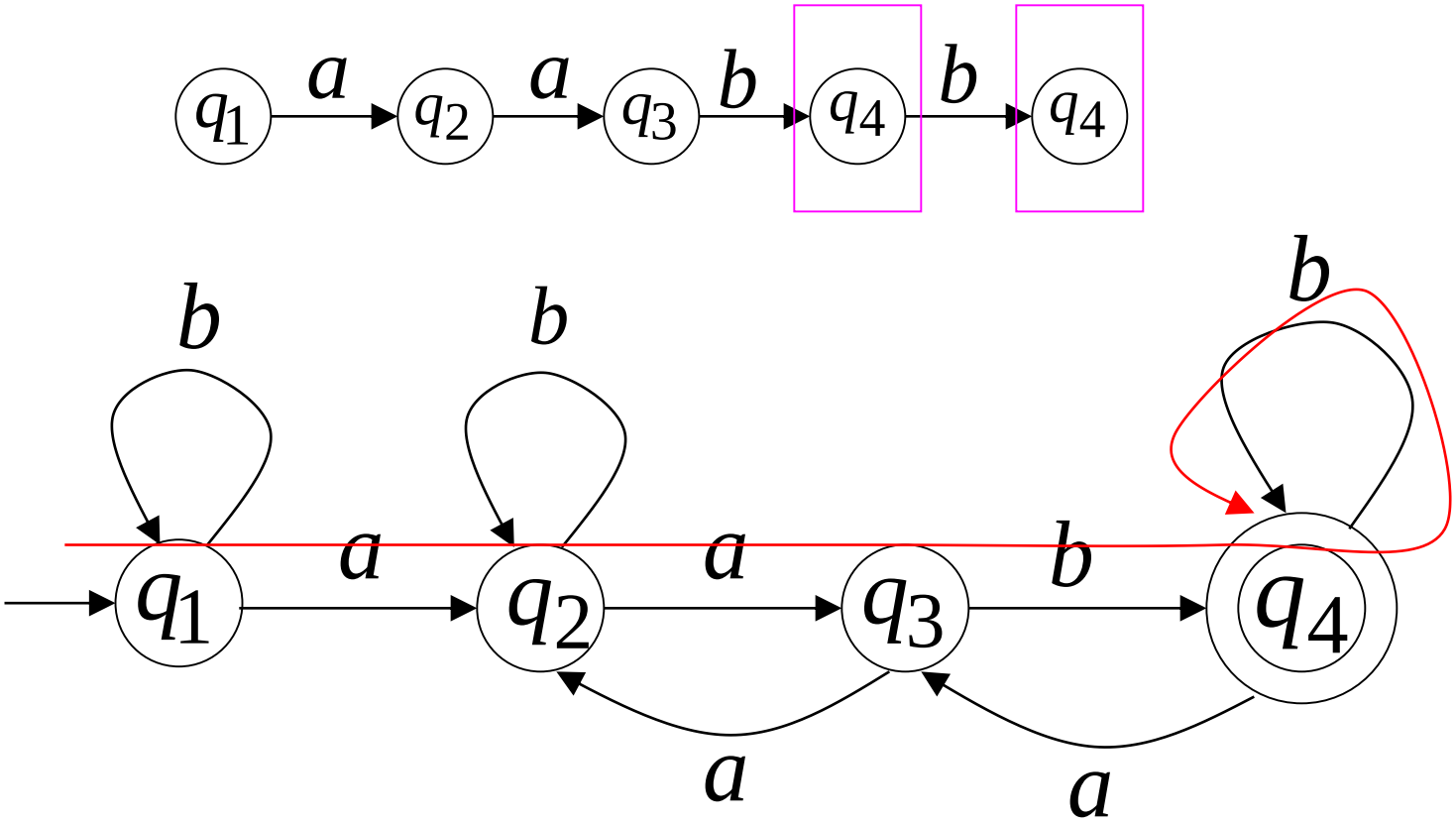
The state is repeated as a result of the pigeonhole principle



Consider the walk of a "long" string: $aabb$
(length at least 4)

Due to the pigeonhole principle:

A state is repeated in the walk of $aabb$

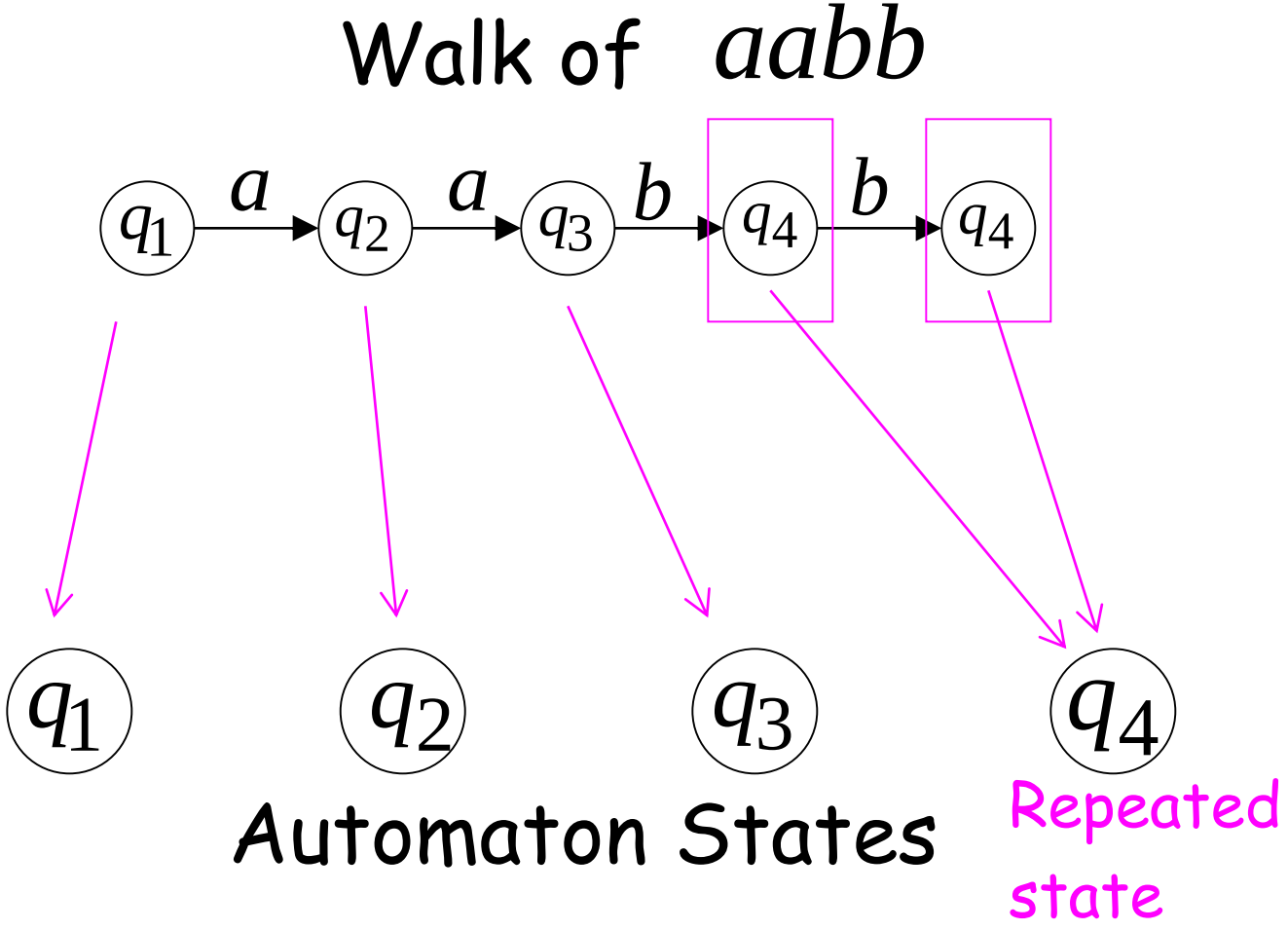


The state is repeated as a result of the pigeonhole principle

Pigeons:
(walk states)

Are more than

Nests:
(Automaton states)

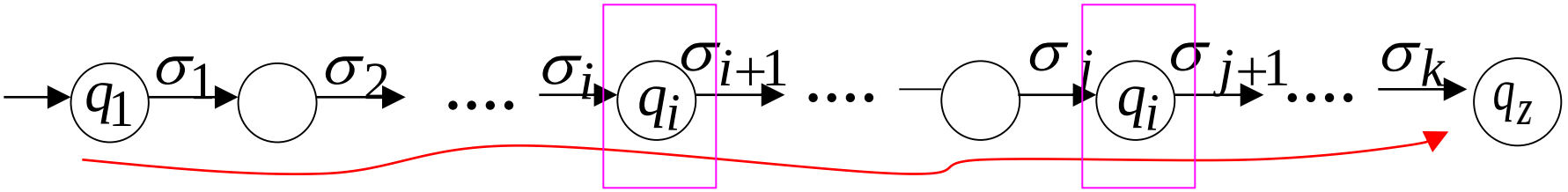


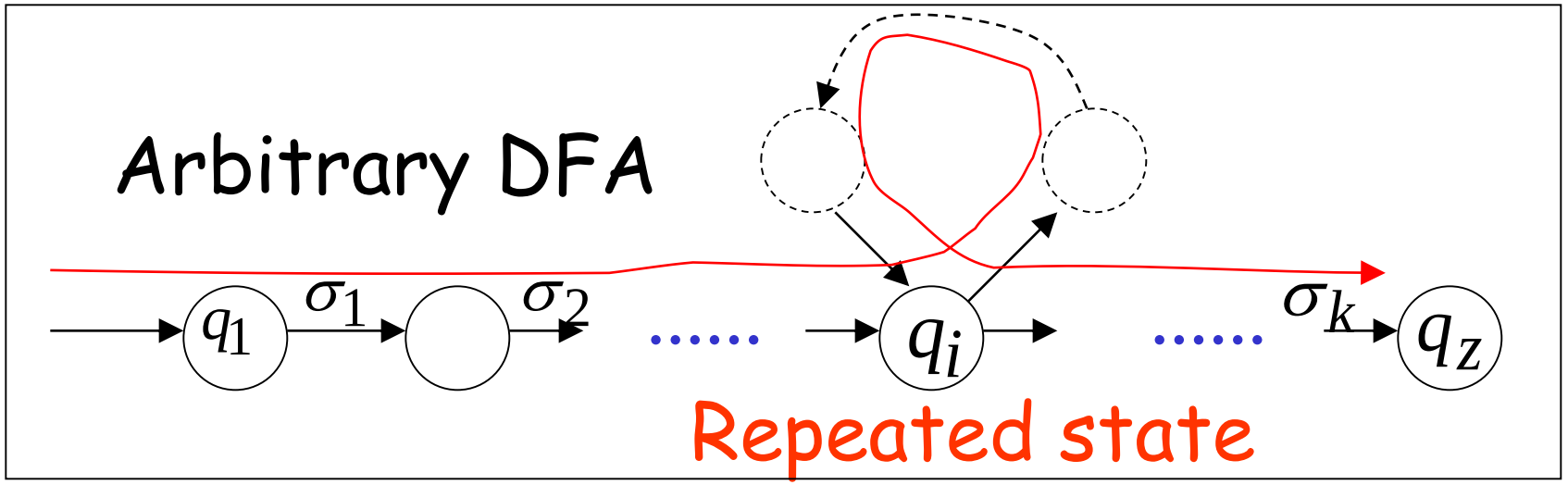
In General: If $|w| \geq$ #states of DFA,

by the pigeonhole principle,

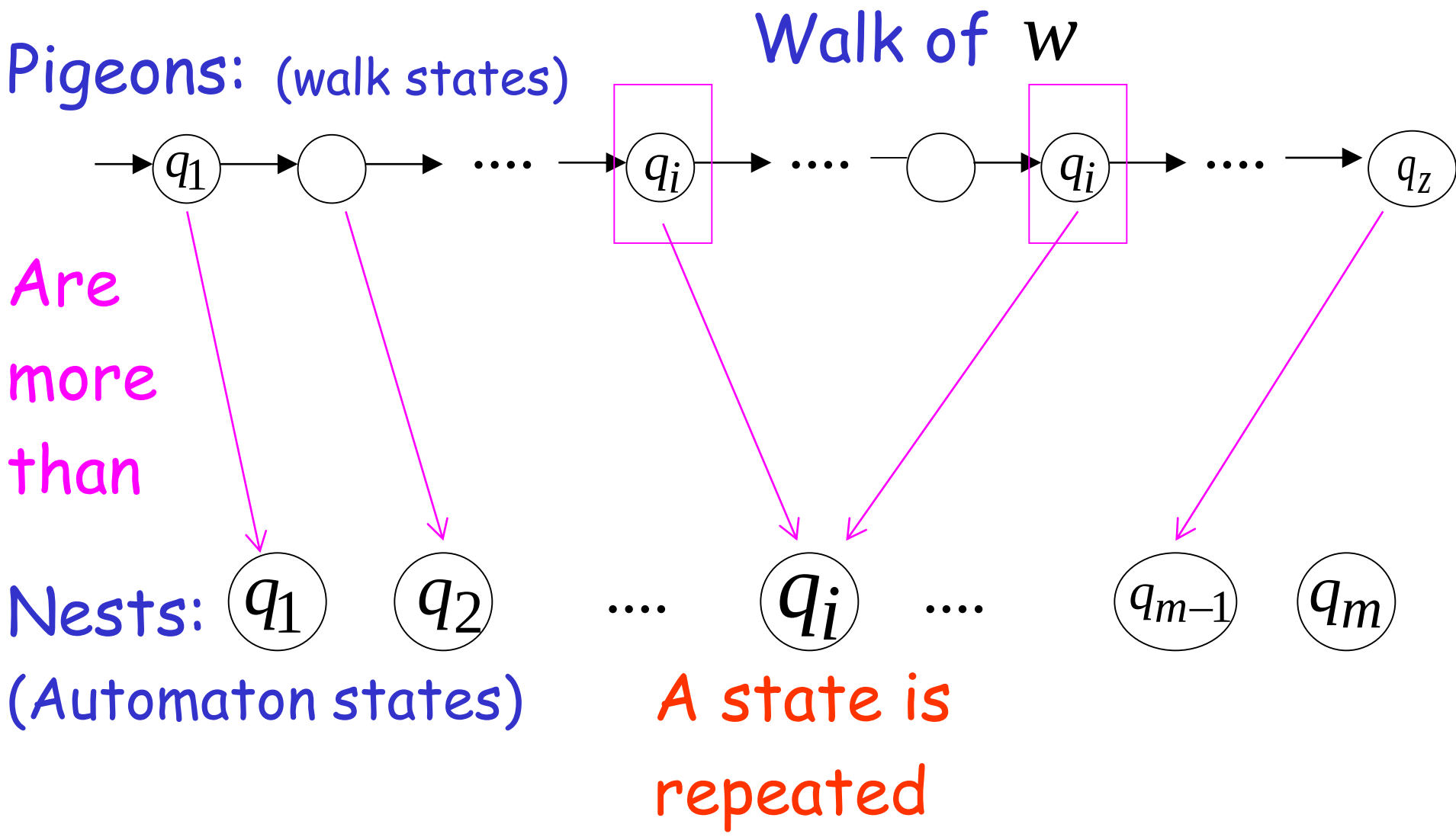
a state is repeated in the walk w

Walk of $w = \sigma_1 \sigma_2 \cdots \sigma_k$





$$|w| \geq \#\text{states of DFA} = m$$

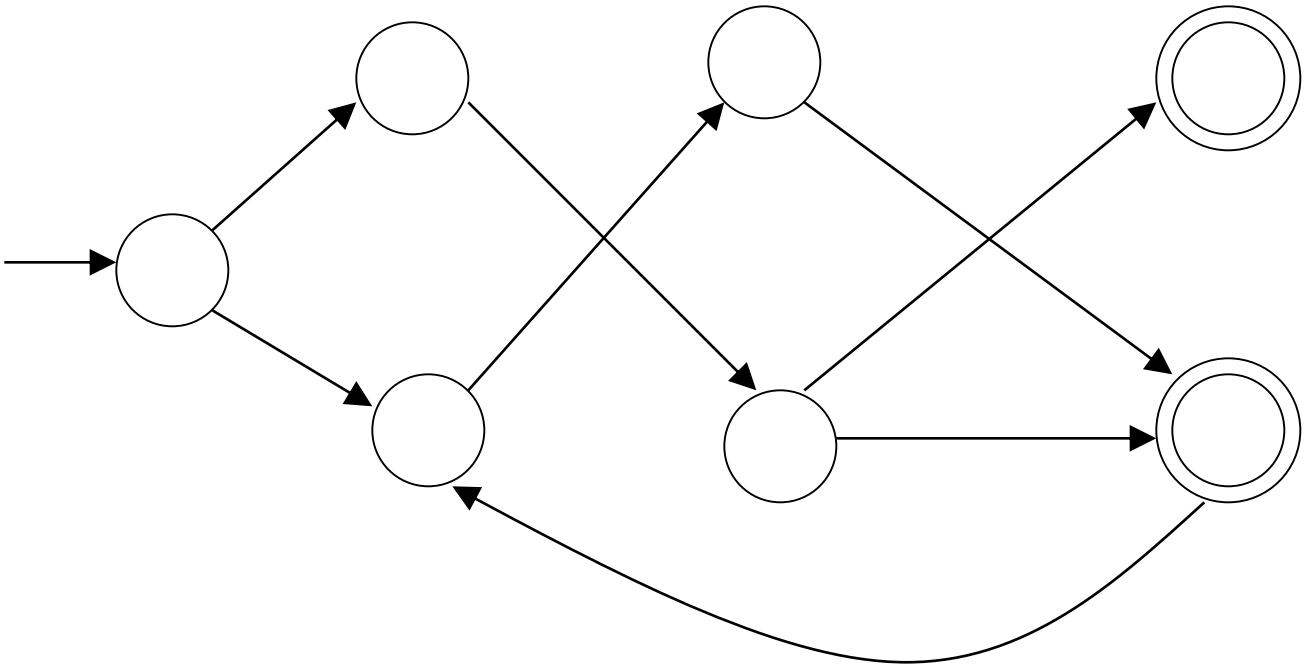


The Pumping Lemma

Take an infinite regular language L

(contains an infinite number of strings)

There exists a DFA that accepts L

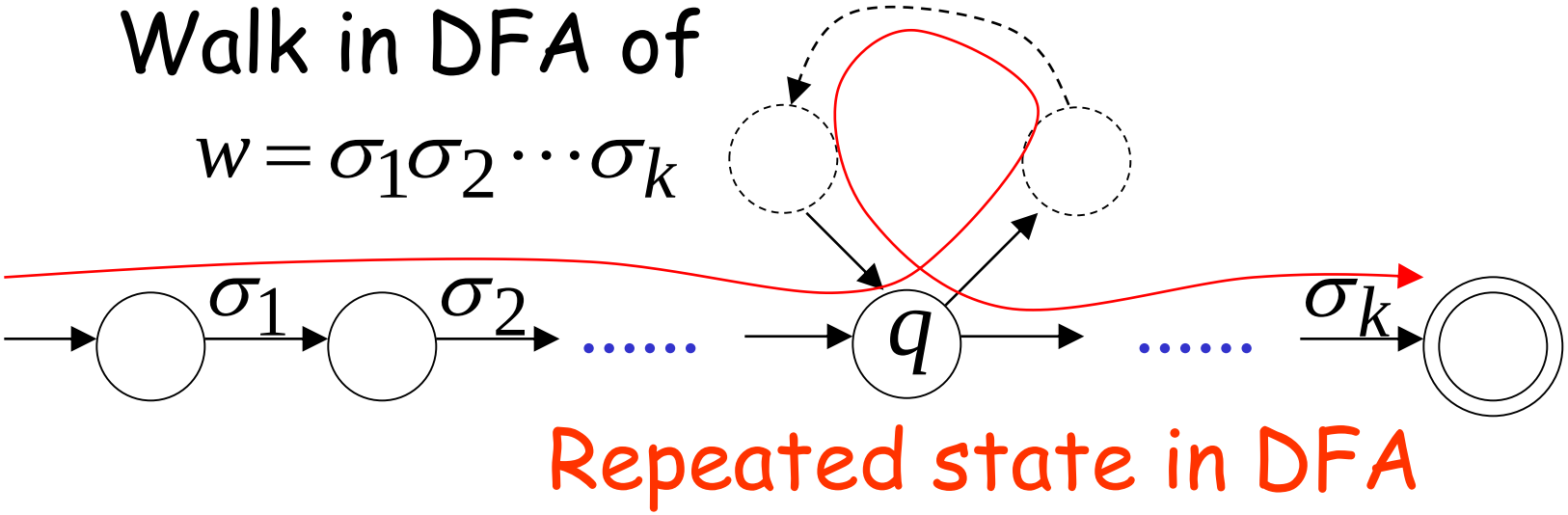


m states

Take string $w \in L$ with $|w| \geq m$

(number of states of DFA)

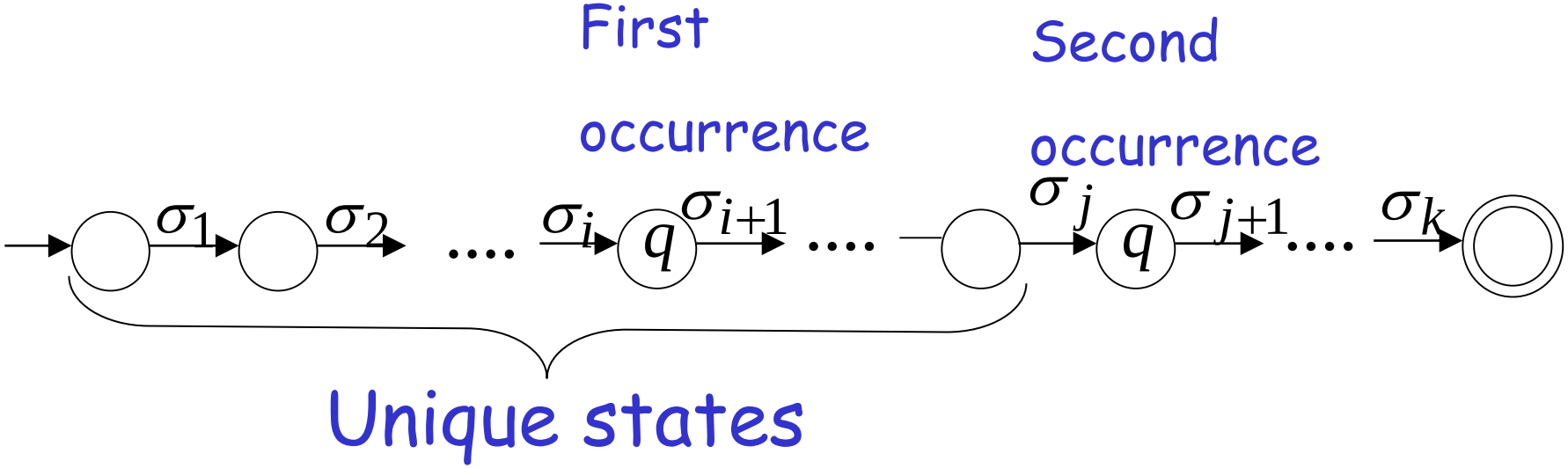
then, at least one state is repeated in the walk of w



There could be many states repeated

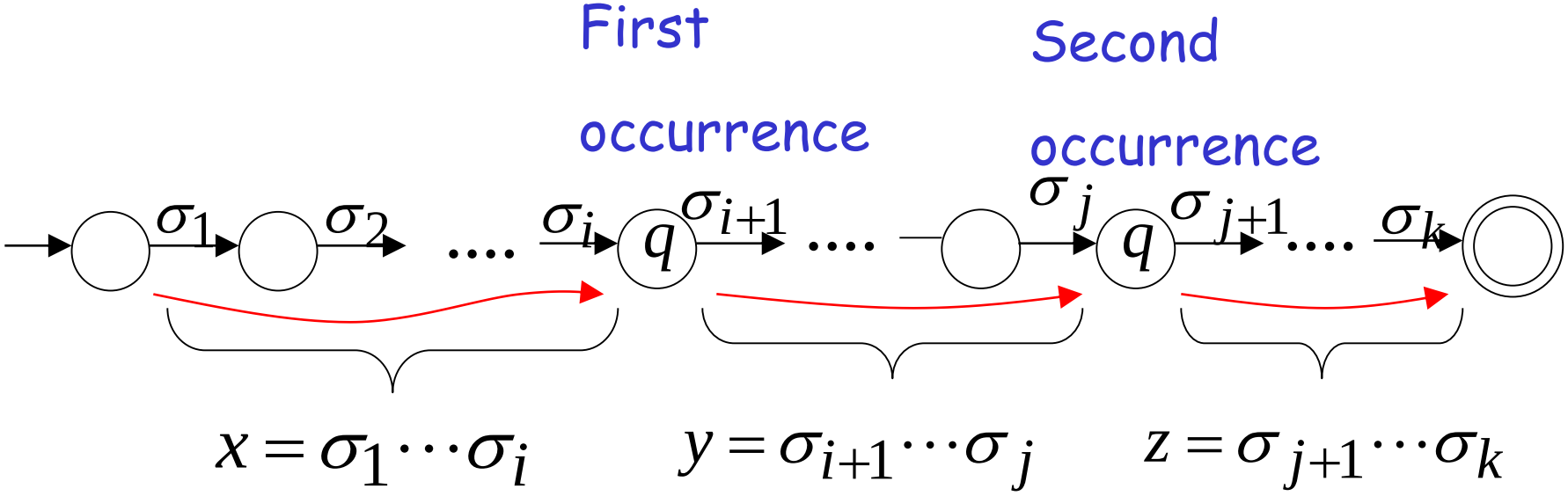
Take q to be the first state repeated

One dimensional projection of walk w:



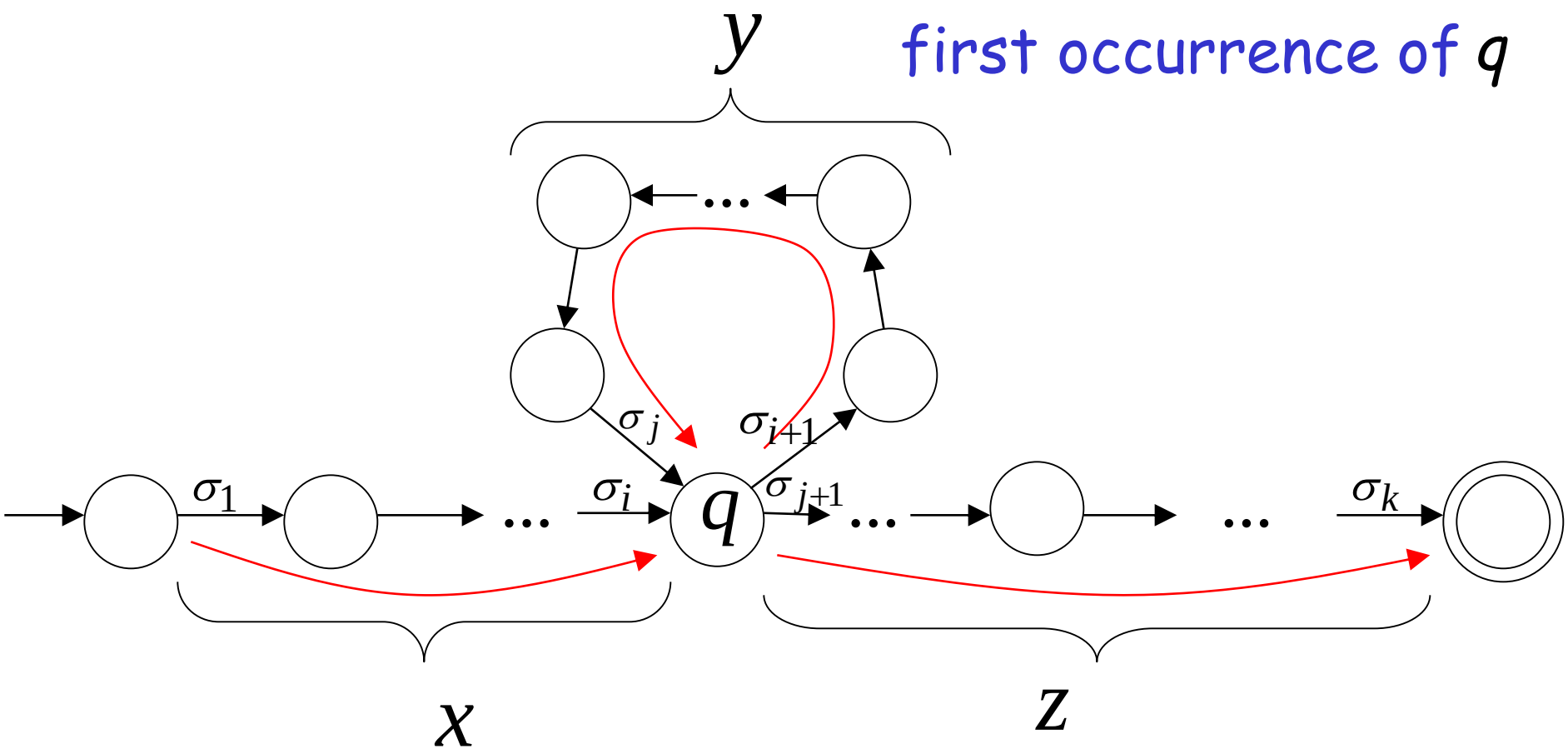
We can write $w = xyz$

One dimensional projection of walk w:

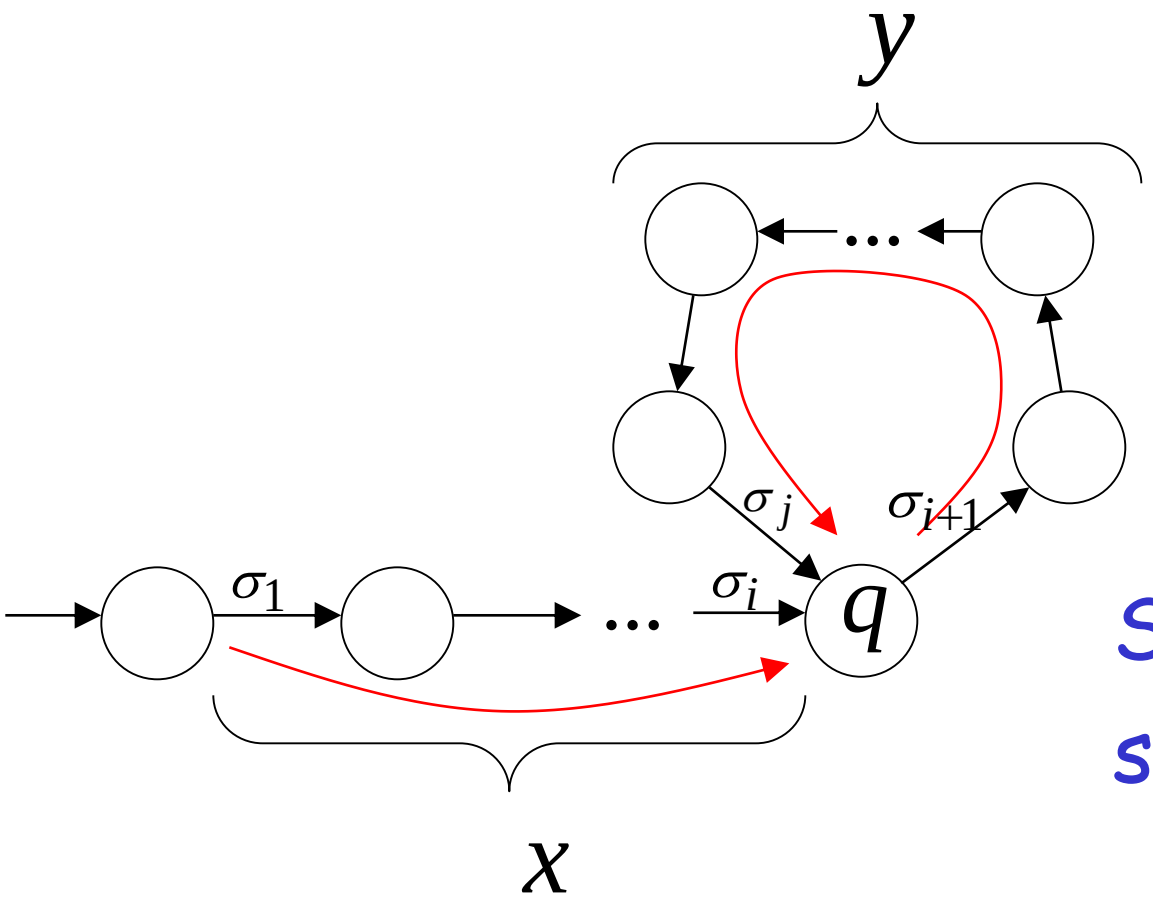


In DFA: $w = x\ y\ z$

contains only
first occurrence of q



Observation: length $|xy| \leq m$ number of states of DFA



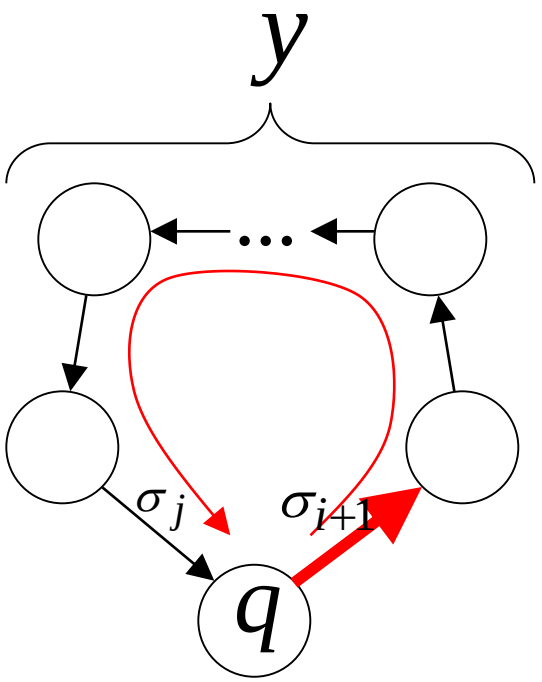
Unique States

Since, in xy no state is repeated

(except q)

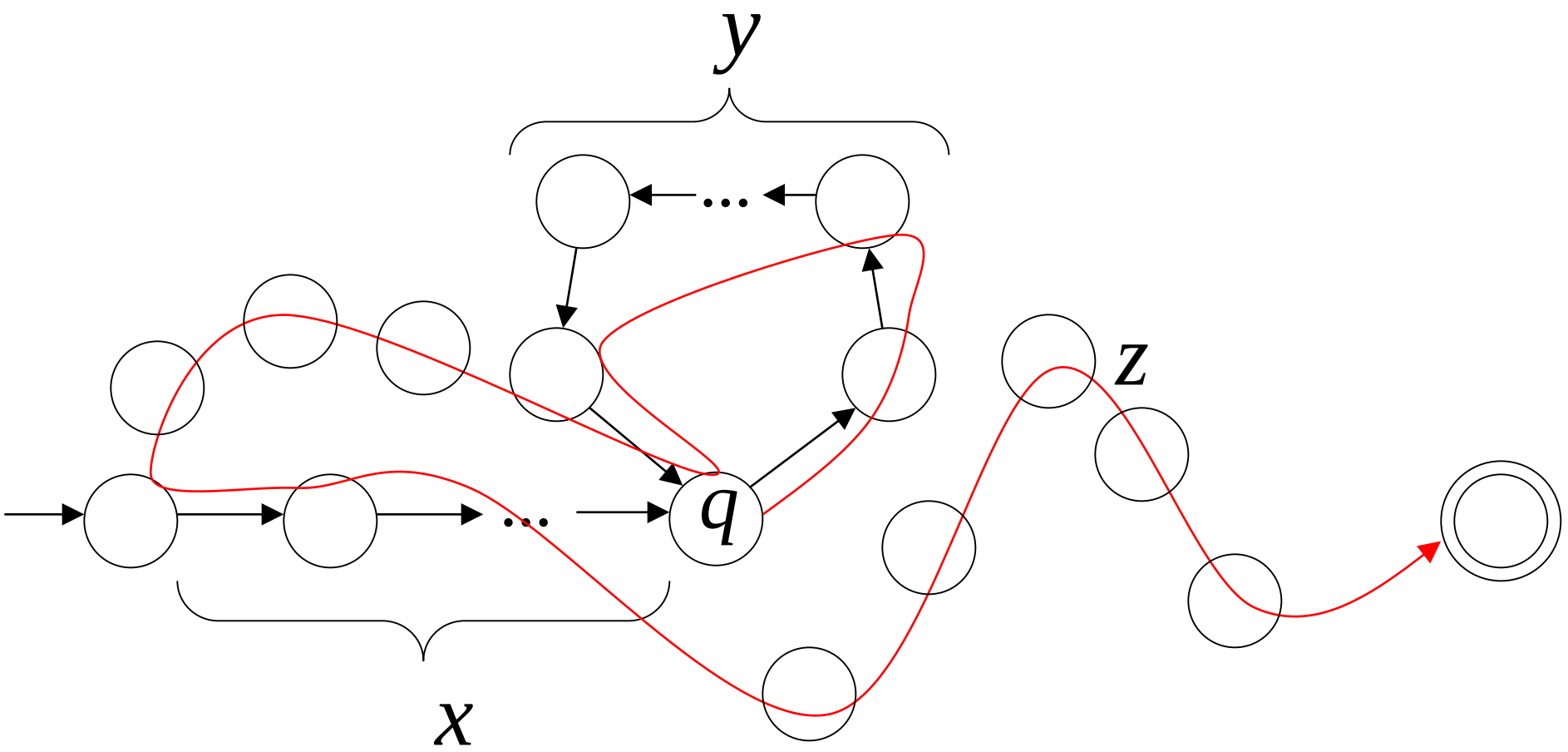
Observation: length $|y| \geq 1$

Since there is at least one transition in loop



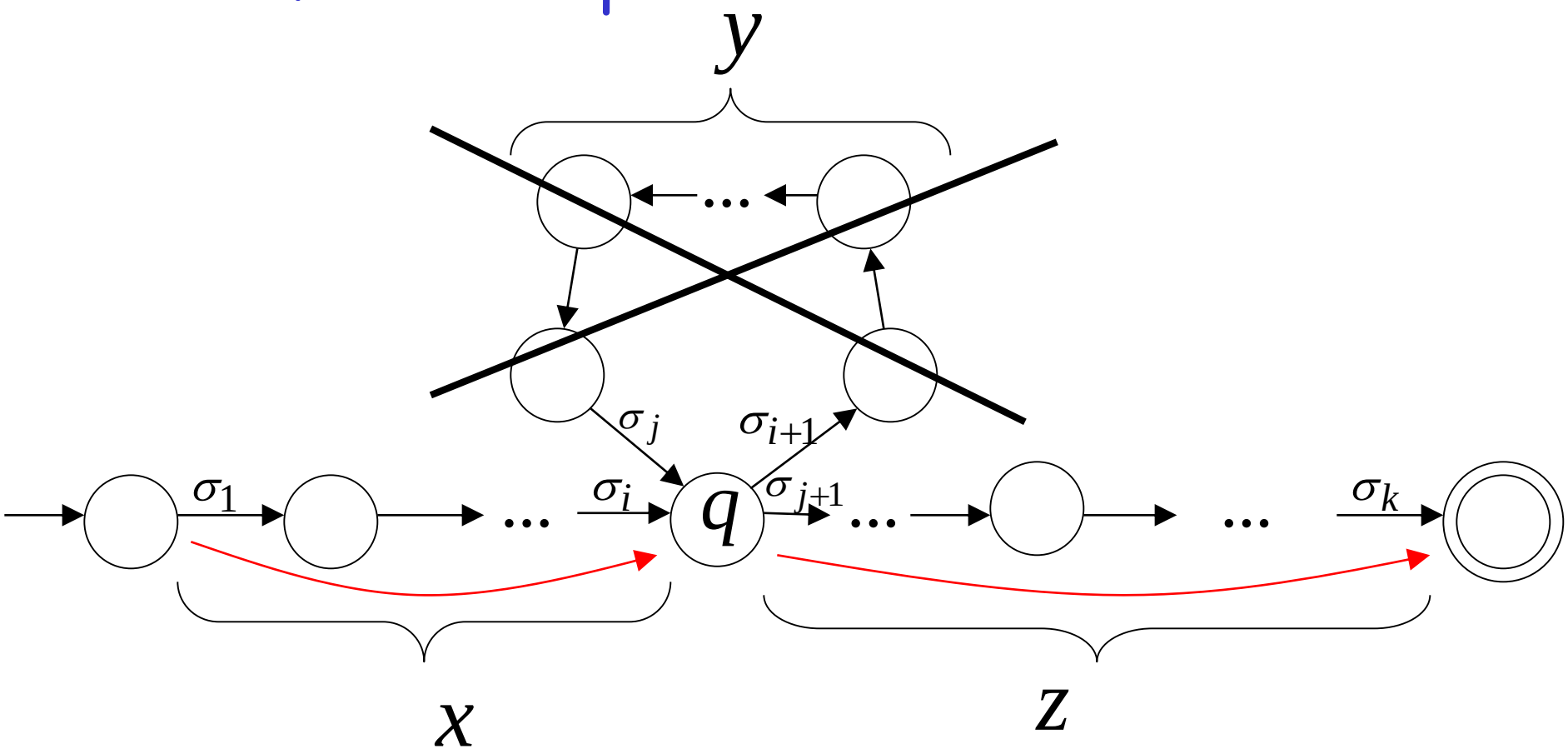
We do not care about the form of string z

z may actually overlap with the paths of x and y



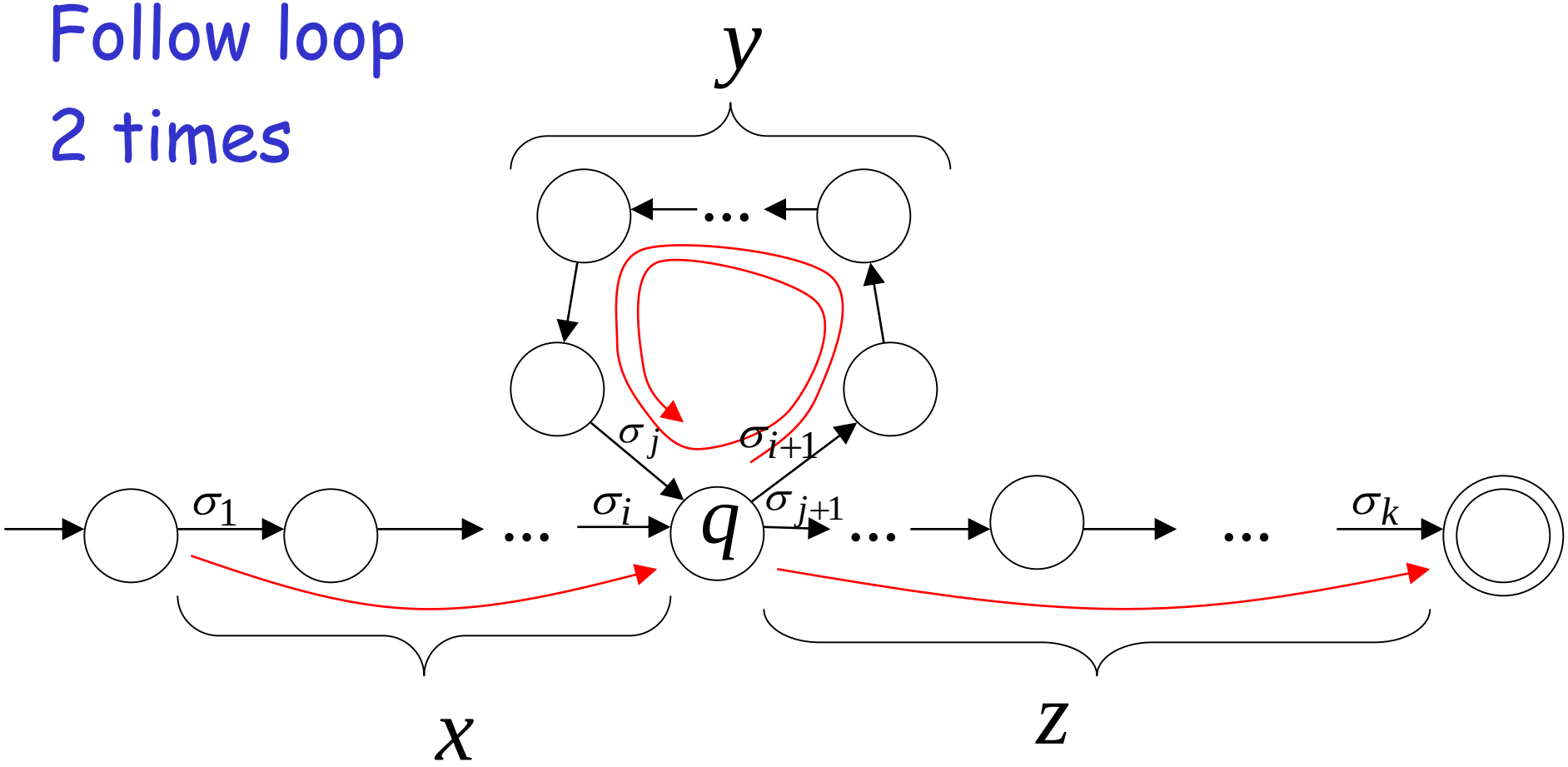
Additional string: The string $x\ z$ is accepted

Do not follow loop



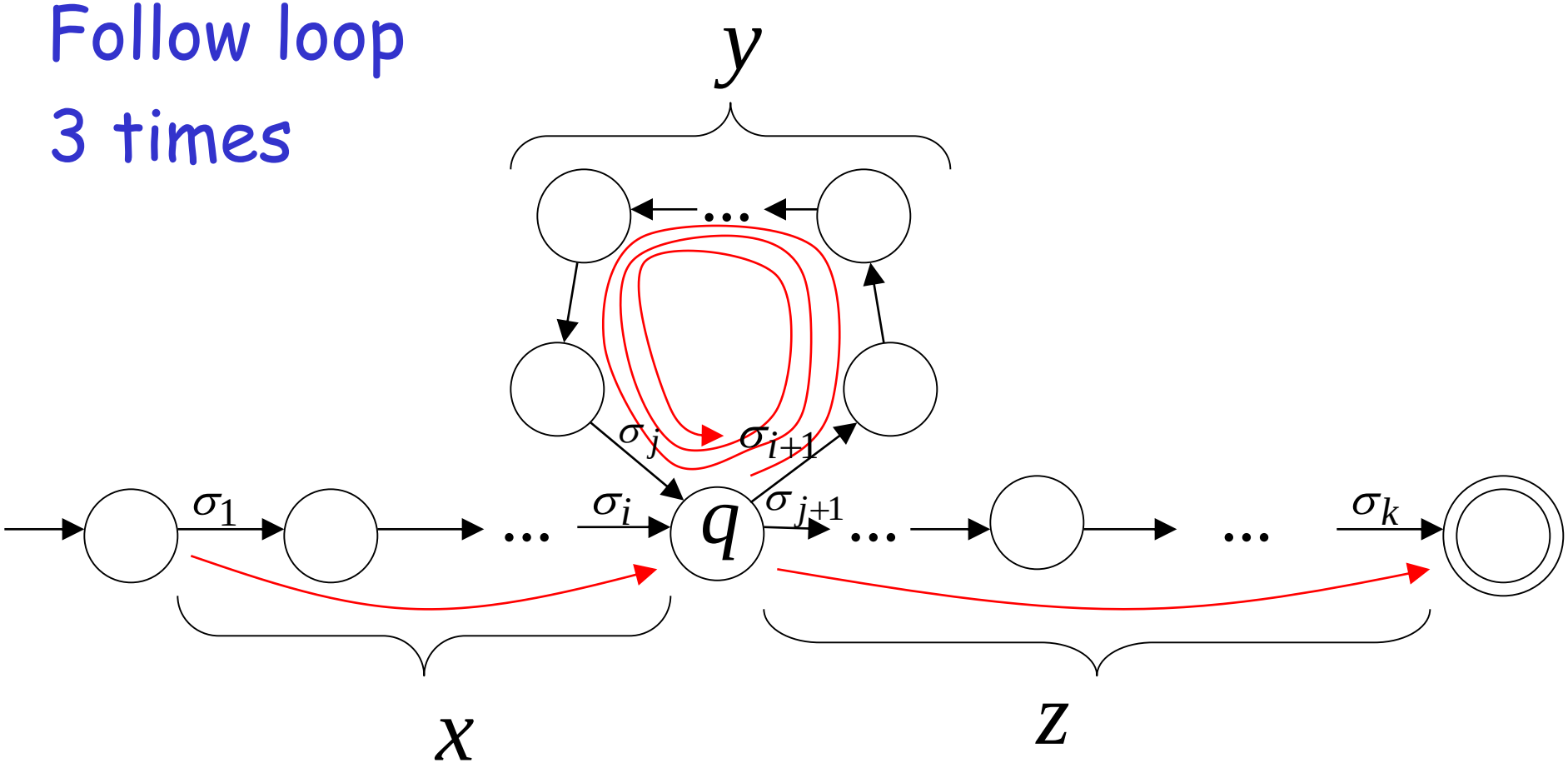
Additional string: The string $x\ y\ y\ z$ is accepted

Follow loop 2 times



Additional string: The string $x\ y\ y\ y\ z$ is accepted

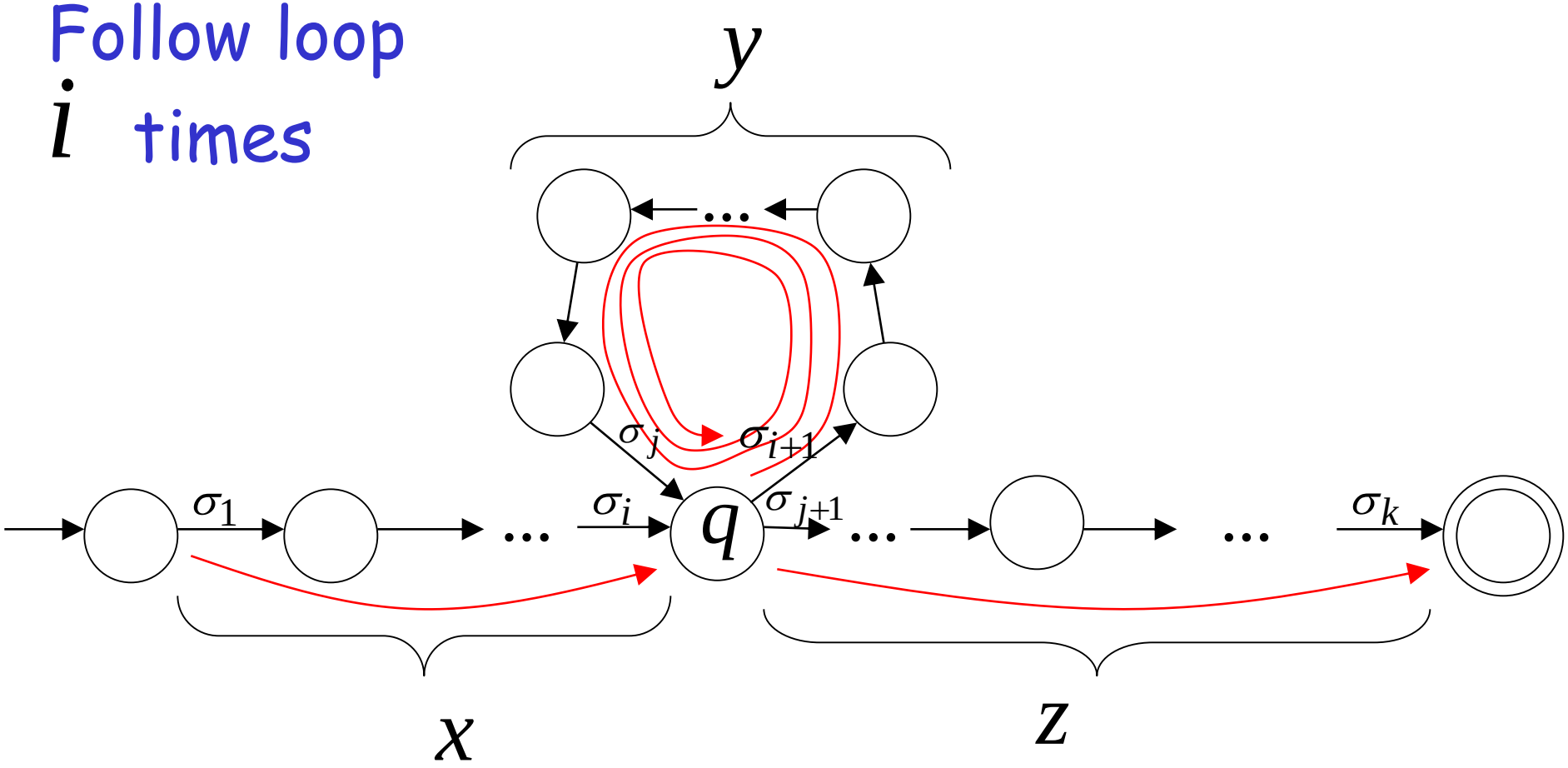
Follow loop 3 times



In General: The string $x\,y^i\,z$

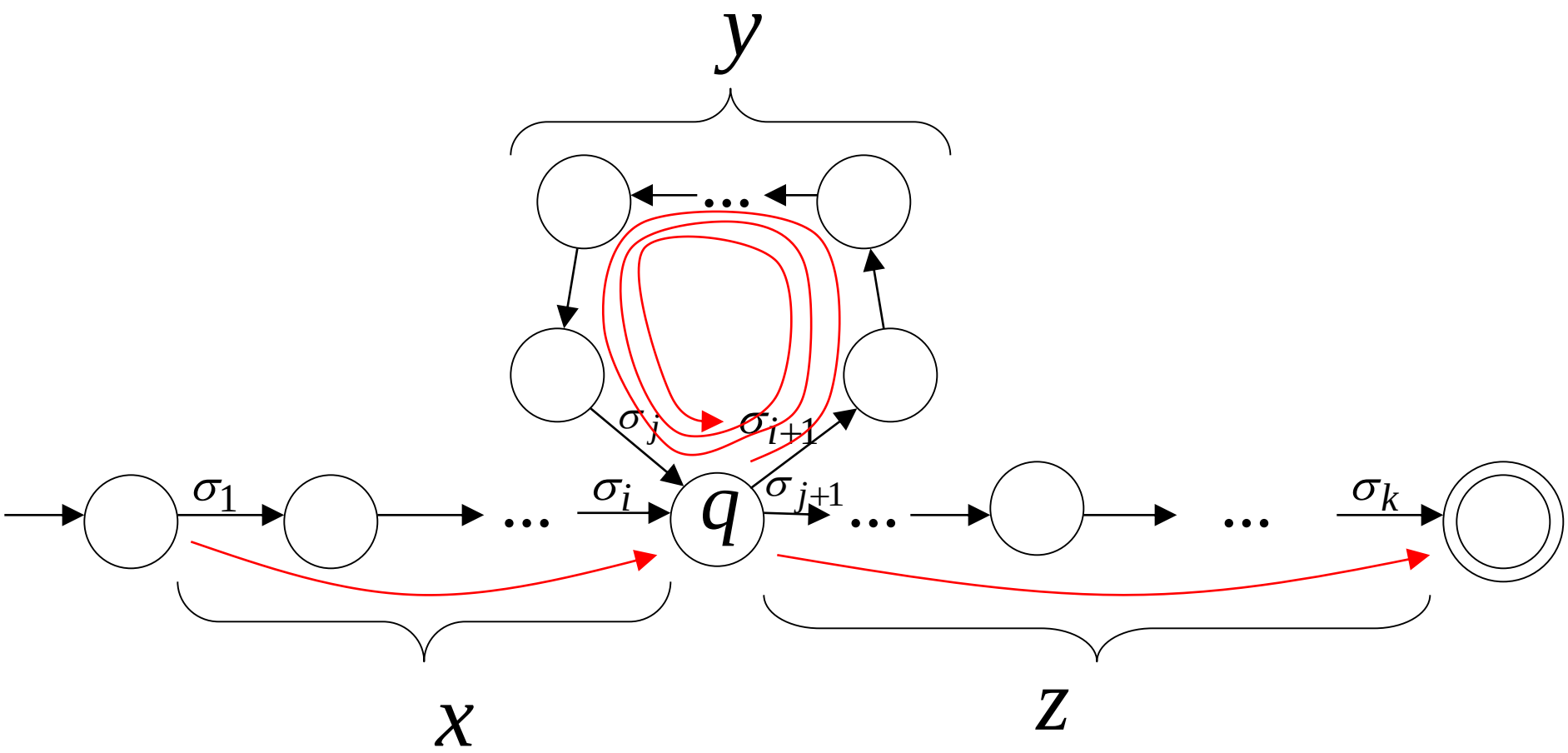
is accepted $i = 0, 1, 2, \ldots$

Follow loop i times

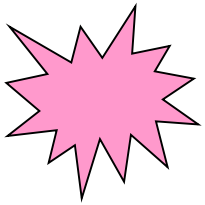


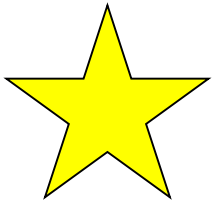
Therefore: $x\, y^i\, z \in L \qquad i = 0, 1, 2, \ldots$

Language accepted by the DFA

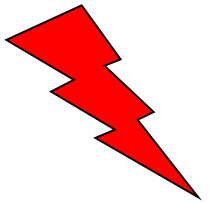


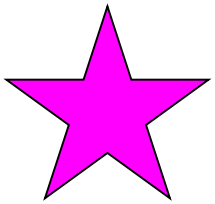
In other words, we described:





The Pumping Lemma !!!





The Pumping Lemma:

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

In the book:

Critical length m = Pumping length p

Applications

of

the Pumping Lemma

Observation:

Every language of finite size has to be regular

(we can easily construct an NFA
that accepts every string in the language)

Therefore, every non-regular language
has to be of infinite size

(contains an infinite number of strings)

Suppose you want to prove that
An infinite language L is not regular

1. Assume the opposite: L is regular

2. The pumping lemma should hold for L

3. Use the pumping lemma to obtain a contradiction

4. Therefore, L is not regular

Explanation of Step 3: How to get a contradiction

1. Let m be the critical length for L

2. Choose a particular string $w \in L$ which satisfies the length condition $|w| \geq m$

3. Write $w = xyz$

4. Show that $w' = xy^i z \notin L$ for some $i \neq 1$

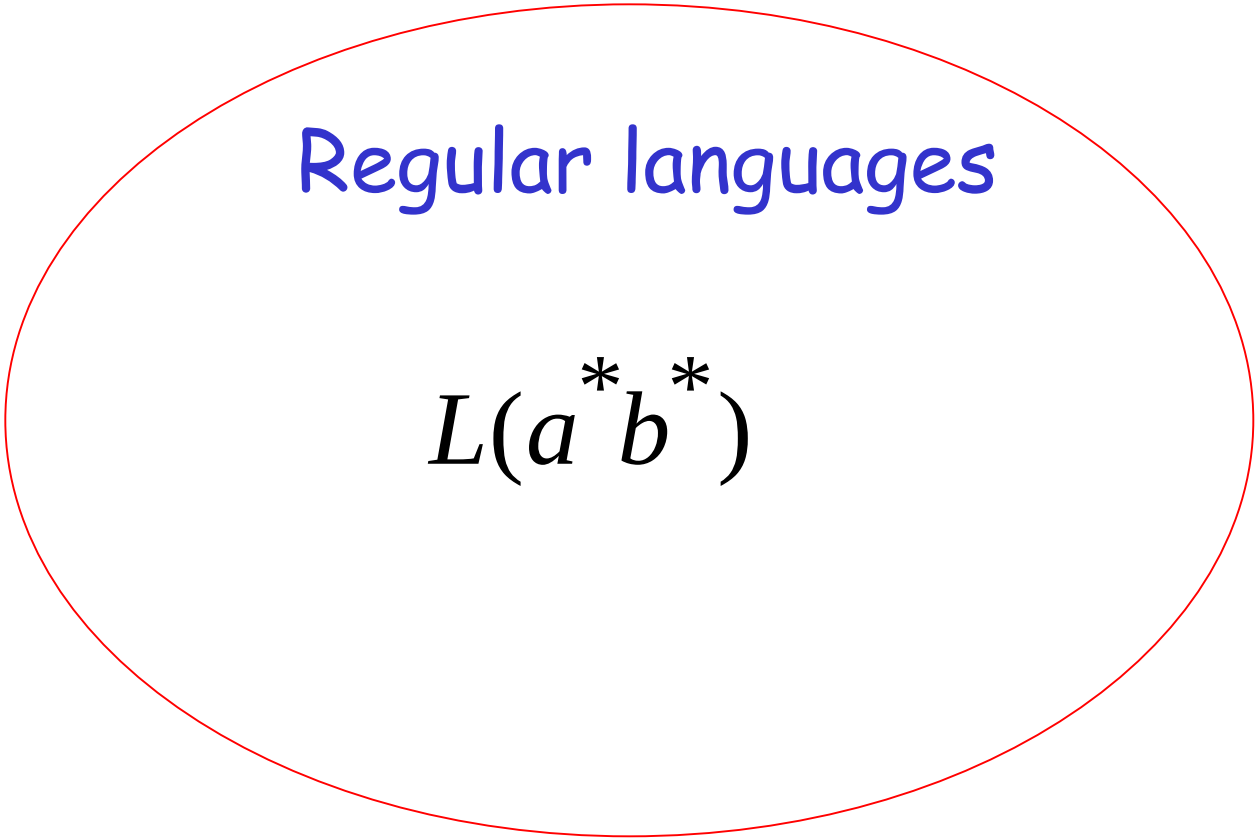
5. This gives a contradiction, since from pumping lemma $w' = xy^i z \in L$

Example of Pumping Lemma application

Theorem: The language $L = \{a^n b^n : n \geq 0\}$ is not regular

Proof: Use the Pumping Lemma

Non-regular language $\{a^n b^n : \ n \geq 0\}$



Example of Pumping Lemma application

Theorem: The language $L = \{a^n b^n : n \geq 0\}$ is not regular

Proof: Use the Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

Assume for contradiction
that L is a regular language

Since L is infinite
we can apply the Pumping Lemma

$$L = \{a^n b^n : n \geq 0\}$$

Let m be the critical length for L

Pick a string w such that: $w \in L$

and length $|w| \geq m$

We pick $w = a^m b^m$

From the Pumping Lemma:

we can write $w = a^m b^m = x\ y\ z$

with lengths $|x\ y| \leq m, \quad |y| \geq 1$

$$w = xyz = a^m b^m = \overbrace{a\ldots a}^{m}\overbrace{a\ldots a}^{m}\overbrace{a\ldots a}^{m}\overbrace{b\ldots b}^{m}$$

$$x \quad y \quad z$$

Thus: $y = a^k, \quad 1 \leq k \leq m$

$$x\,y\,z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^i\,z \in L$

$$i = 0, 1, 2, \ldots$$

Thus: $x\,y^2\,z \in L$

$$x\,y\,z = a^m b^m \qquad y = a^k, \quad 1 \le k \le m$$

From the Pumping Lemma: $x\,y^2\,z \in L$

$$xy^2z = \overbrace{a\ldots a}^{x}\overbrace{a\ldots a}^{y}\overbrace{a\ldots a}^{y}\overbrace{a\ldots ab\ldots b}^{z} \in L$$

$$\overbrace{\qquad\qquad\qquad\qquad}^{m+k} \qquad \overbrace{\quad}^{m}$$

Thus: $a^{m+k}b^m \in L$

$$a^{m+k}b^m \in L \qquad k \geq 1$$

BUT: $L = \{a^n b^n : n \geq 0\}$

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

CONTRADICTION!!!

Therefore: Our assumption that L is a regular language is not true

Conclusion: L is not a regular language

END OF PROOF

Non-regular language $\{a^n b^n : \ n \geq 0\}$