INT201 Decision, Computation and Language

Lecture 5 – Regular Languages (2)

Dr Yushi Li

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Kleene's Theorem

Let L be a language. Then L is **regular** if and only if there exists a regular expression that describes L.

- If a language is described by a regular expression, then it is regular.

- If a language is regular, then it has a regular expression.

The language described by a regular expression is a regular language

Proof Convert a regular expression R into a NFA M

1st case. If $R = \epsilon$, then $\mathrm{L}(R) = \{\epsilon\}$. The NFA is $\mathrm{M} = (\{q\}, \Sigma, \delta, q, \{q\})$ where:

$$\delta(q, a) = \emptyset \text{ for all } a \in \Sigma_\epsilon$$

2nd case. If $R = \emptyset$, then $\mathrm{L}(R) = \emptyset$. The NFA is $\mathrm{M} = (\{q\}, \Sigma, \delta, q, \emptyset)$ where:

$$\delta(q, a) = \emptyset \text{ for all } a \in \Sigma_\epsilon$$

The language described by a regular expression is a regular language

Proof

3rd case. If $R = a$ for $a \in \Sigma$, then $\mathrm{L}(R) = \{a\}$. The NFA is $\mathrm{M} = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$

where:

$$\delta(q_1, a) = \{q_2\}$$

$$\delta(q_1, b) = \emptyset \text{ for all } b \in \Sigma_\epsilon \setminus \{a\}$$

$$\delta(q_2, b) = \emptyset \text{ for all } b \in \Sigma_\epsilon$$

The language described by a regular expression is a regular language

Proof

4th case (union). If $R = (R_1 \cup R_2)$ and

- $\mathrm{L}(R_1)$ has NFA M_1
- $\mathrm{L}(R_2)$ has NFA M_2

Then $\mathrm{L}(R_1)$ = $\mathrm{L}(R_1) \cup \mathrm{L}(R_2)$ has NFA as:

The language described by a regular expression is a regular language

Proof

5th case (concatenation). If $R = R_1\ R_2$ and

- $\mathrm{L}(R_1)$ has NFA M_1
- $\mathrm{L}(R_2)$ has NFA M_2

Then $\mathrm{L}(R_1)$ = $\mathrm{L}(R_1)\ \mathrm{L}(R_2)$ has NFA as:

The language described by a regular expression is a regular language

Proof

6th case (Kleene star). If $R = (R_1)^*$ and $\mathrm{L}(R_1)$ has NFA N, then $\mathrm{L}(R) = (\ \mathrm{L}(R_1)\ )^*$ has NFA M as:

The language described by a regular expression is a regular language

Example

Given a regular expression $R = (ab \cup a)^*$, where the alphabet is $\{a, b\}$. Prove that this regular expression describes a regular language, by constructing a NFA that accepts $\mathrm{L}(R)$.

The language described by a regular expression is a regular language

Example

A regular language has a regular expression

Convert DFA into regular expression

Every DFA M can be converted to a regular expression that describes the language $\mathrm{L(M)}$.

Generalized NFA (GNFA)

A GNFA can be defined as a 5-tuple, $(Q, \Sigma, \delta, \{s\}, \{t\})$, consisting of

- a finite set of states Q;
- a finite set called the alphabet Σ;
- a transition function $(\delta : (Q \setminus \{t\}) \times (Q \setminus \{s\}) \rightarrow R)$;
- a start state $(\mathrm{s} \in Q)$;
- an accept state $(\mathrm{t} \in Q)$;

where R is the collection of all regular expressions over the alphabet Σ.

A regular language has a regular expression

Iterative procedure for converting a DFA $\mathbf{M} = (Q, \Sigma, \delta, q, F)$ into a regular expression:

1. Convert DFA $\mathrm{M} = (Q, \Sigma, \delta, q, F)$ into equivalent GFNA G:

- Introduce new start state s
- Introduce new start state t
- Change edge labels into regular expressions

  e.g., "a, b" becomes "$a \cup b$"

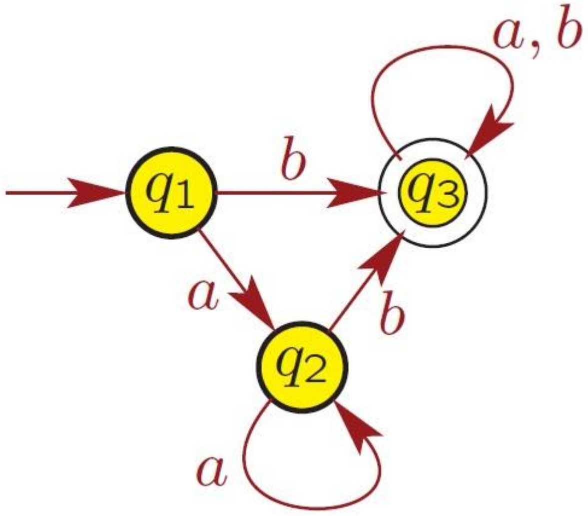
2. Iteratively eliminate a state from GNFA G until only 2 states remaining: start and accept.

- Need to take into account all possible previous paths.
- Never eliminate new start state s or new accept state t.

A regular language has a regular expression

Example

Convert the given DFA into regular expression



1st step: DFA -> GNFA

A regular language has a regular expression

Example

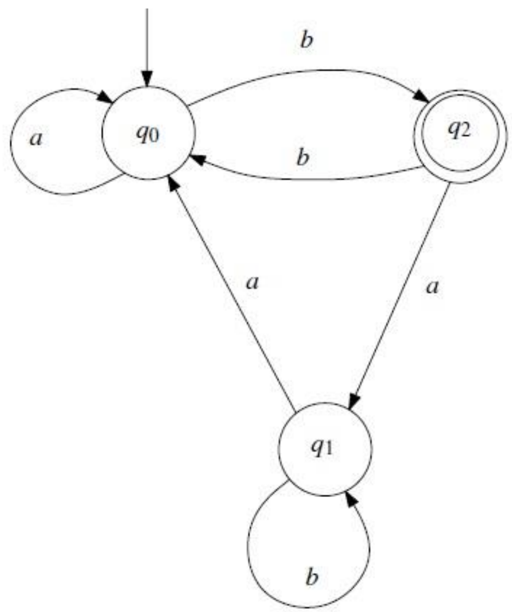
Convert the given DFA into regular expression

2nd step: eliminate states

A regular language has a regular expression

Exercise

M = (Q, Σ, δ, q_0, F), where $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $F = \{q_2\}$, and δ is given as:



Convert it to a regular expression.

Exercise

Convert it GNFA.

Exercise

Exercise

Pumping Lemma for Regular Languages

A tool that can be used to prove that certain languages are not regular. This theorem states that all regular languages have a special property.

This property states that all strings in the language can be "pumped" if they are at least as long as a certain special value, called the **pumping length**.

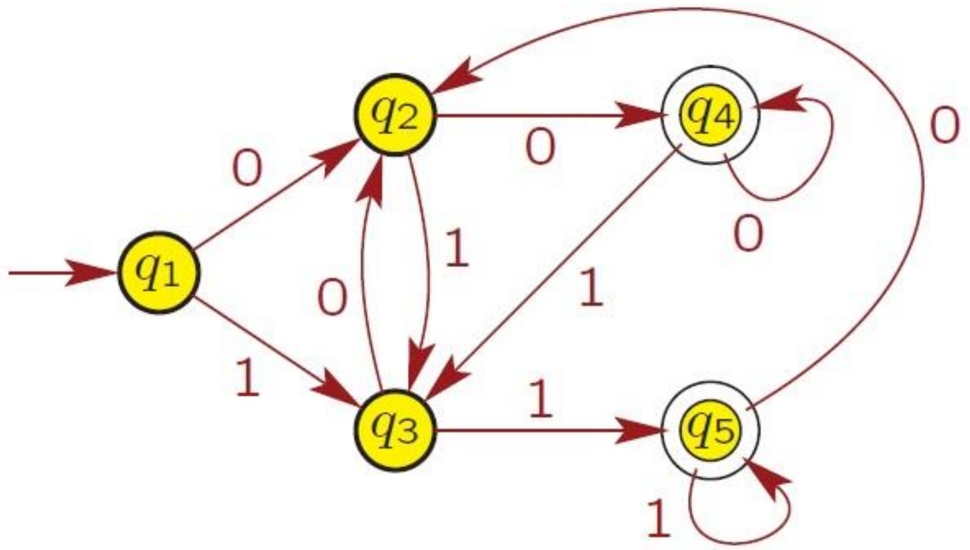
- If a language L is regular, it always satisfies pumping lemma. If there exists at least one string made from pumping which is not in L, then L is surely not regular.

- The opposite may not be true. If pumping lemma holds, it does not mean that the language is regular.

Pumping Lemma for Regular Languages

Example

DFA with $\Sigma = \{0, 1\}$ for language A.

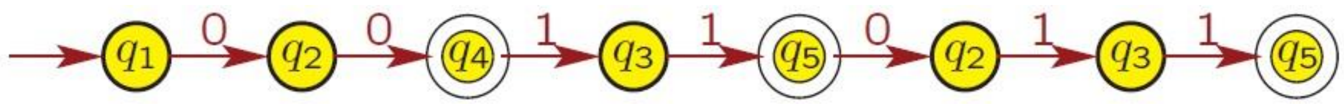


$Q = \{q_1 , q_2 , q_3 , q_4 , q_5\}$

Pumping Lemma for Regular Languages

For any string s with $|\, s \,| \geq 5$, guaranteed to visit some state twice by the **pigeonhole principle**.

String $s = 0011011$ is accepted by DFA, i.e., $s \in A$



q_2 is first state visited twice.

Using q_2, divide string s into 3 parts x, y, z such that $s = xyz$.

- $x = 0$, the symbols read until first visit to q_2.
- $y = 0110$, the symbols read from first to second visit to q_2.
- $z = 11$, the symbols read after second visit to q_2.

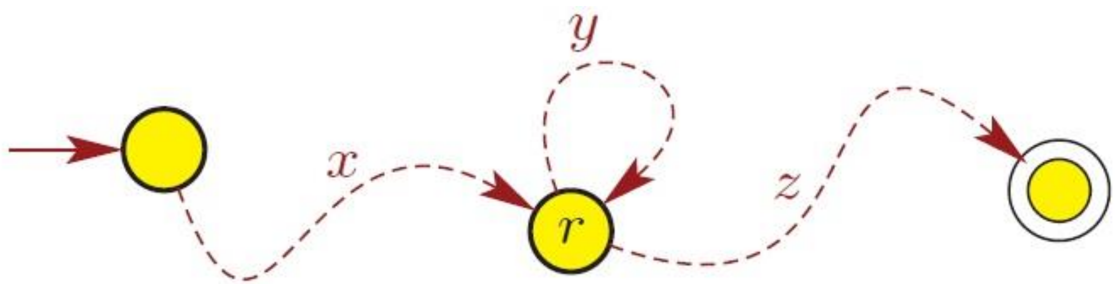
Pumping Lemma for Regular Languages

DFA accepts string

DFA also accepts string

String $xy^iz \in \mathrm{A}$ for each $i \geq 0$.

Pumping Lemma for Regular Languages



- More generally, consider

✓ language A with DFA M having p states (where p is number of states in DFA).

✓ string $s \in A$ with $|s| \geq p$.

- When processing s on M, guaranteed to visit some state twice.

- Let r be first state visited twice.

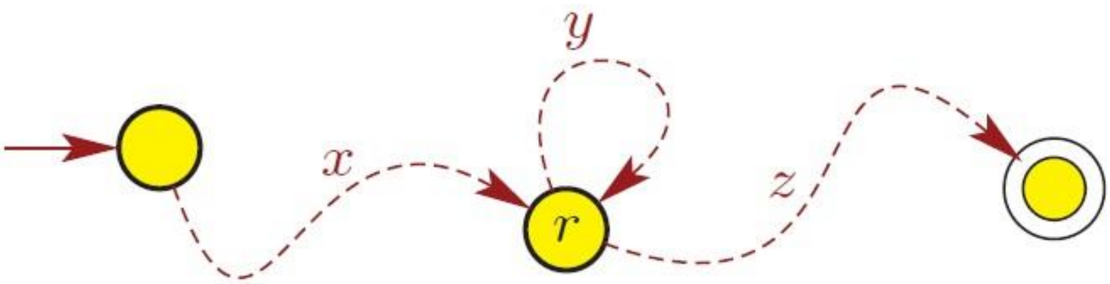
- Using state r, can divide s as $s = xyz$.

✓ x are symbols read until first visit to r.

✓ y are symbols read from first to second visit to r.

✓ z are symbols read from second visit to r to end of s.

Pumping Lemma for Regular Languages



- Because y corresponds to starting in r and returning to r,

$$xy^iz \in A \text{ for each } i \geq 1.$$

- Also, note $xy^0z = xz \in A$, so

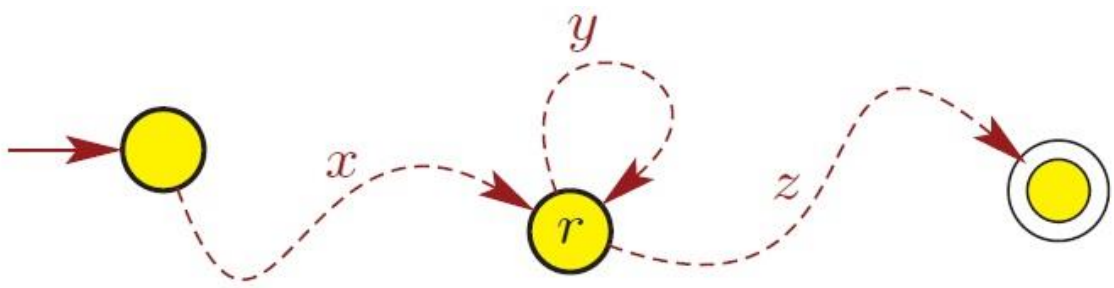
$$xy^iz \in A \text{ for each } i \geq 0.$$

- $|y| > 0$ because

✓ y corresponds to starting in r and coming back;

✓ this consumes at least one symbol (because DFA), so y can't be empty

Pumping Lemma for Regular Languages



- $|xy| \le p$, where p is number of states in DFA, because

✓ xy are symbols read up to second visit to r.

✓ Because r is the first state visited twice, all states visited before second visit to r are unique.

✓ So just before visiting r for second time, DFA visited at most p states, which corresponds to reading at most $p - 1$ symbols.

✓ The second visit to r, which is after reading 1 more symbol,corresponds to reading at most p symbols.

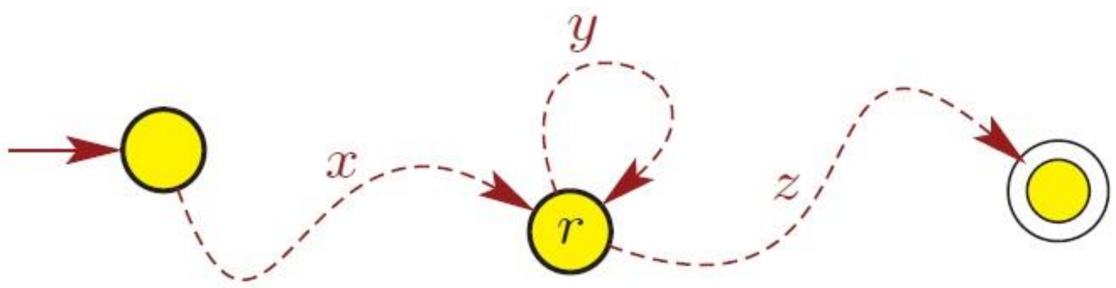
Pumping Lemma for Regular Languages

Let A be a regular language. Then there exists an integer $p \geq 1$, called the pumping length, such that the following holds: Every string s in A, with $|s| \geq p$, can be written as $s = xyz$, such that

1. $y \neq \epsilon$ (i.e., $|y| \geq 1$),

2. $|xy| \leq p$, and

3. for all $i \geq 0$, $xy^iz \in A$.



Example

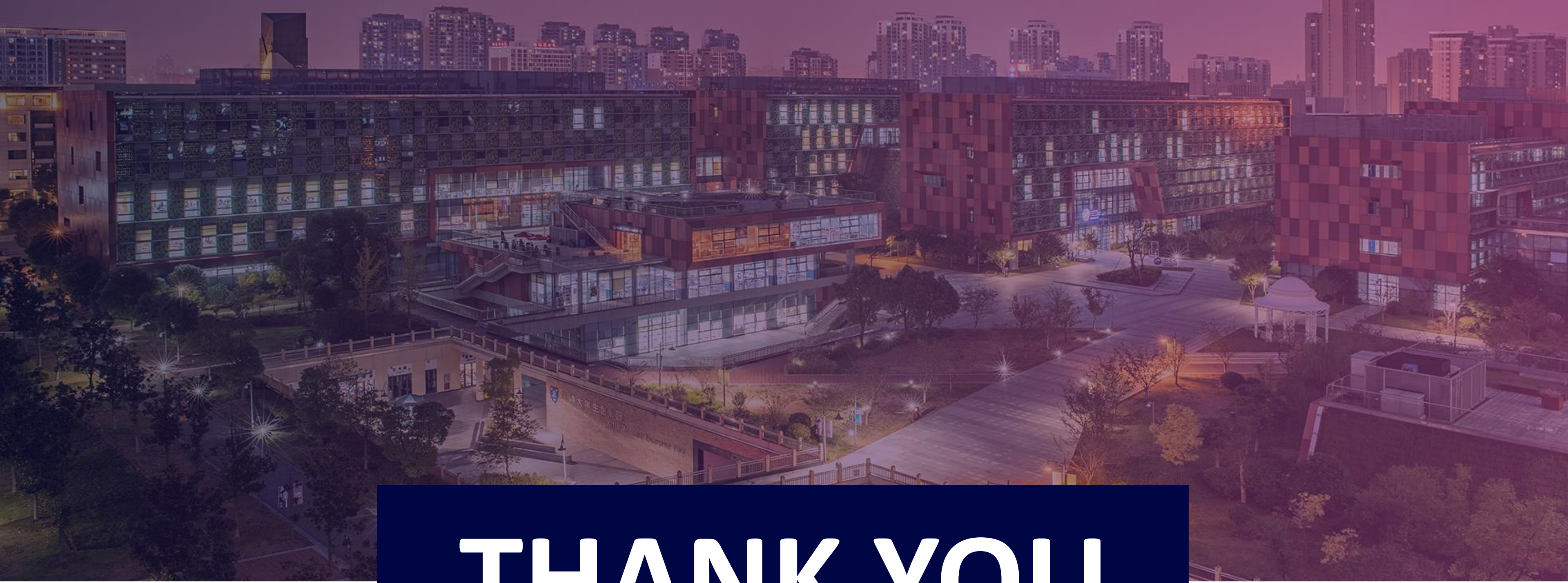
Language $A = \{ 0^n1^n \mid n \geq 0 \}$ is Nonregular

Proof

Example

Language $A = \{ 0^n 1^n \mid n \geq 0 \}$ is Nonregular

Proof



THANK YOU

Xi'an Jiaotong-Liverpool University
西交利物浦大学

XJTLU | SCHOOL OF FILM AND TV ARTS