INT201 Decision, Computation and Language

Lecture 5 – Regular Languages (2)

Dr Yushi Li

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

Kleene's Theorem 克莱因定理

设L为一种语言。那么L当且仅当存在描述L的正则表达式时，L是正则的

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

1. • If a language is described by a regular expression, then it is regular.
   如果一种语言是用正则表达式描述的，那么它就是正则语言。
2. • If a language is regular, then it has a regular expression.
   如果一种语言是正则表达式，那么它就有正则表达式。

如果一种语言是用正则表达式描述的，那么它就是正则语言。

① The language described by a regular expression is a regular language

正则表达式R转换为 NFA M

Proof Convert a regular expression R into a NFA M

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

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

2nd case. If $R = \emptyset$, then $L(R) = \emptyset$. The NFA is $M = (\{q\}, \Sigma, \delta, q, \emptyset)$ 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 $L(R) = \{a\}$. The NFA is $M = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$

where:

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

$$\delta(q_1, b) = \emptyset \text{ for all } b \in \Sigma_\epsilon \backslash \{a\}$$ Σ_ϵ 不包含 $\{a\}$

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

The language described by a regular expression is a regular language

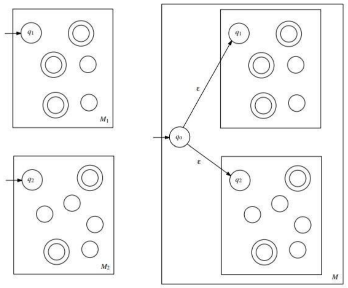
Proof

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

当regular expression为 $R_1 \cup R_2$时，
用以下的图格式转换为 $L(R)$

- $L(R_1)$ has NFA M_1
- $L(R_2)$ has NFA M_2

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



The language described by a regular expression is a regular language

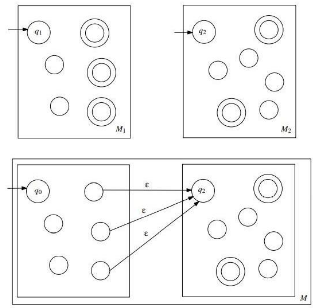
Proof

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

当regular expression中 $R=R_1R_2$时.

- $L(R_1)$ has NFA M_1
- $L(R_2)$ has NFA M_2

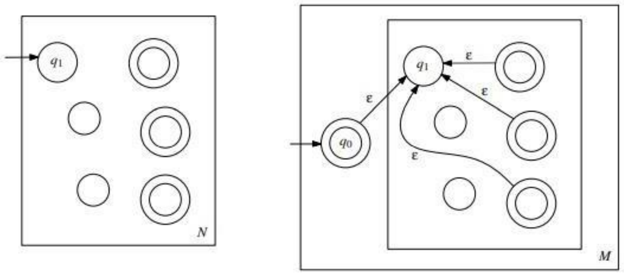
Then $L(R_1) = L(R_1)\ L(R_2)$ has NFA as:



The language described by a regular expression is a regular language

Proof

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

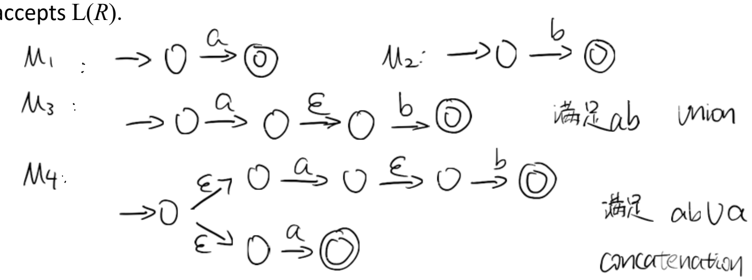


The language described by a regular expression is a regular language

Example

转换为一个成立的NFA

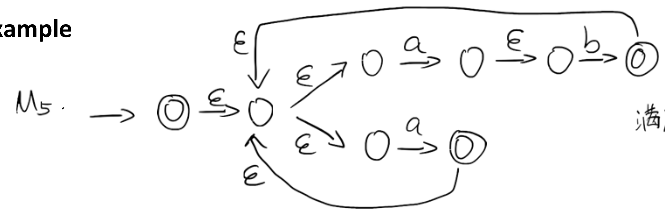
Given a regular expression $R = (ab \cup a)^*$, where the alphabet is $\{a, b\}$. Prove that this regular expression describes a regular language, by constructing a NFA that accepts $L(R)$.



The language described by a regular expression is a regular language

kleene star

Example



满足 $(ab \cup a)^*$

正则语言有正则表达

A regular language has a regular expression

R → RE

Convert DFA into regular expression 将DFA转换回 regular Expression

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

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; 有限的状态集合
- a finite set called the alphabet Σ; 有限的可输入字符集
- a transition function ($\delta : (Q \setminus \{t\}) \times (Q \setminus \{s\}) \rightarrow R$); 转换关系
- a start state ($s \in Q$); 起始态.
- an accept state ($t \in Q$); 接受态.

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

R是所有基于字母集Σ的 regular expression 的集合

A regular language has a regular expression

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

1. Convert DFA $M = (Q, \Sigma, \delta, q, F)$ into equivalent ~~GNFA~~ NFA G:

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

  e.g., "a, b" becomes "$a \cup b$" 把标签转换为 regular expression 形式

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

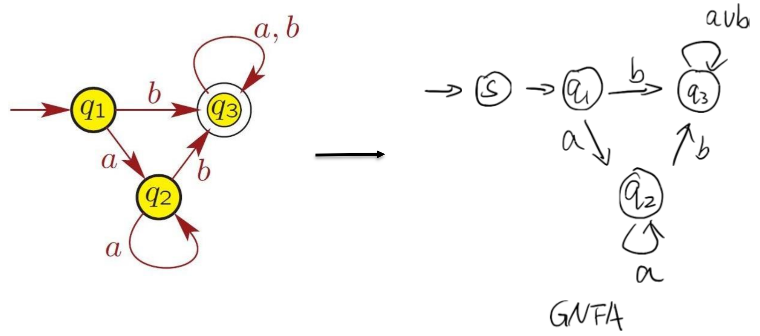
- Need to take into account all possible previous paths.
- Never eliminate new start state s or new accept state t.

Never remove start and accept state

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

remove q_2:

$a \cup b$

$\rightarrow s \xrightarrow{\varepsilon} q_1 \xrightarrow{b \cup aa^*b} q_3 \xrightarrow{\varepsilon} t$

remove q_3

$\rightarrow s \xrightarrow{\varepsilon} q_1 \xrightarrow{(b \cup aa^*b)(a \cup b)} t$

remove q_1, only remain s and t

$\rightarrow s \xrightarrow{(b \cup aa^*b)(a \cup b)^*} t$

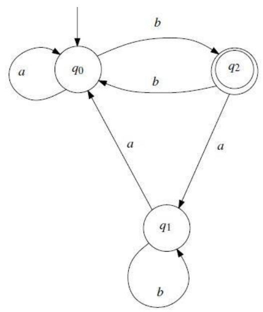
$\Downarrow$

final . $(b \cup aa^*b)(a \cup b)^*$ 2nd step: eliminate states

A regular language has a regular expression

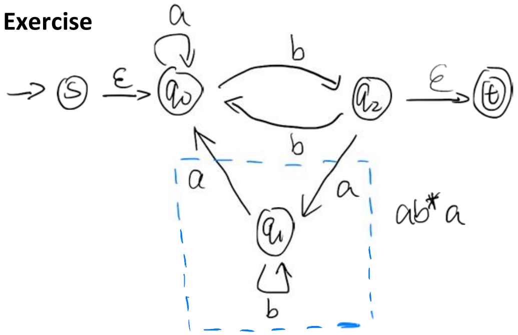
Exercise

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:



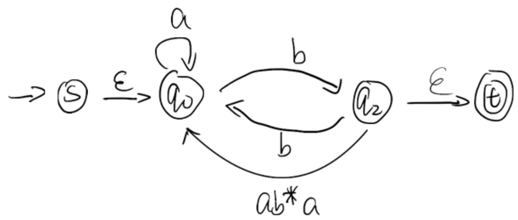
Convert it to a regular expression.

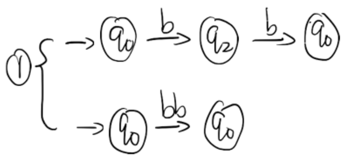
Exercise

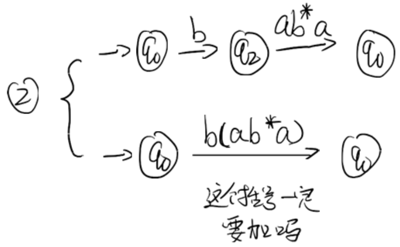


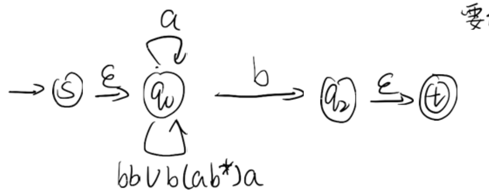
Convert it GNFA.

Exercise









Exercise

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

Pumping Lemma for Regular Languages

包含数学证明.

pumping Lemma 只能验证是不是 non-regular

可用于证明某些语言不规则的工具。该定理说明，所有正则表达式语言都有一个特殊属性。

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.

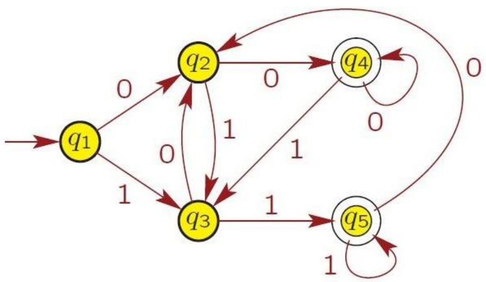
This property states that all strings in the language can be "pumped" if they are at least as long as a certain special value, called the **pumping length**.

- 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.

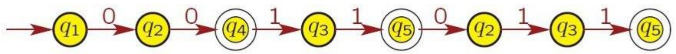


$Q = \{q_1, q_2, q_3, q_4, q_5\}$

Pumping Lemma for Regular Languages

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

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

$$S = \underbrace{0}_{x}\underbrace{0110}_{y}\underbrace{11}_{z}$$

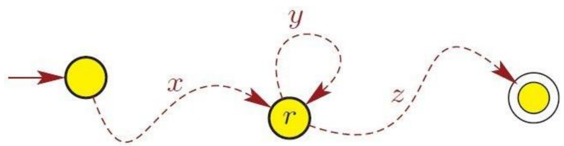
DFA also accepts string

$$xyyz = \underbrace{0}_{x}\ \underbrace{0110}_{y}\ \underbrace{0110}_{y}\ \underbrace{11}_{z}$$

xy^iz , $i \geq 0$ （可以接受多个或1个y部分）

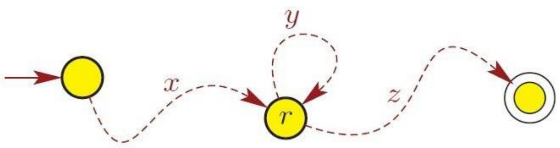
String $xy^iz \in A$ for each $i \geq 0$.

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,

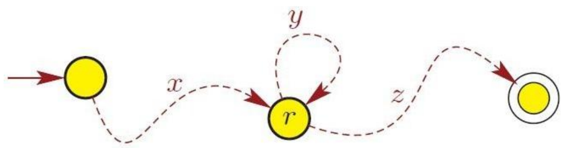
$$xy^iz \in A \text{ for each } i \geq 1.$$

- 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

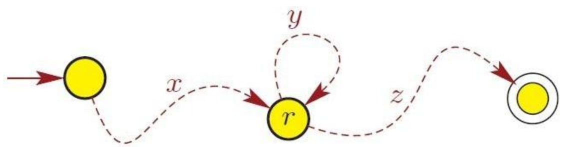


- $|xy| \leq p$, where p is number of states in DFA, because
- ✓ xy are symbols read up to second visit to r.
- ✓ Because r is the first state visited twice, all states visited before second visit to r are unique.
- ✓ So just before visiting r for second time, DFA visited at most p states, which corresponds 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

① Define pumping length — P

② Consider string $s = 0^P1^P \in A$

③ $|s| = 2P \geq P$, pumping lemma will hold

④ So split s into $s = xyz$, satisfying

1. $|y| > 0$
2. $|xy| \leq P$
3. $xy^iz \in A$ for each $i \geq 0$

⑤ Become 1^{st} P symbol of

Example

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

Proof 1. $s = \underbrace{00\cdots0}_{p}\underbrace{11\cdots1}_{p}$ are all 0s

=> x and y consist of only 0s.

=> z will hold all the rest 0s and all 1s.

key: y has some 0s and z contains all 1s => pumping y only changes the number of 0s, but not the numbers of 1s.

6: so we have

$x = 0^j$ for some $j \geq 0$

$y = 0^k$ for some $k \geq 0$

$z = 0^m 1^p$ for some $m \geq 0$

7 $S = xyz$ =>

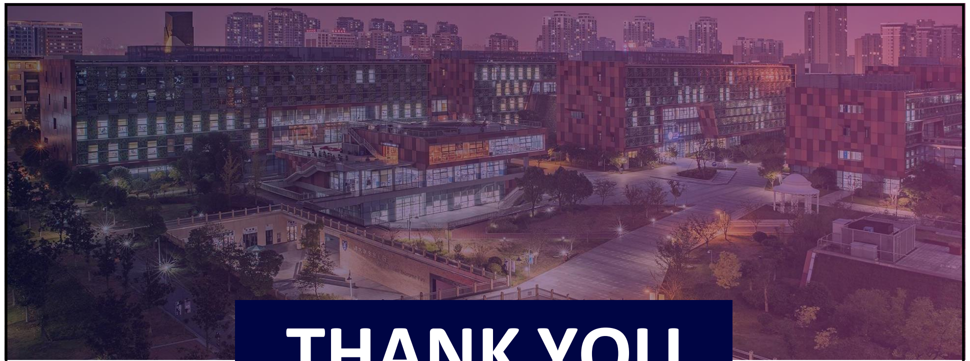
$0^p 1^p = 0^j 0^k 0^m 1^p = 0^{j+k+m} 1^p$ => $j+k+m = p$

8 $|y| > 0$ => $k > 0$

9 假设 $xyyz \in A$ but $xyyz = 0^j 0^k 0^k 0^m 1^p = 0^{j+k+k+m} 1^p = 0^{p+k} 1^p \neq 0^p 1^p$

=> $xyyz \notin A$

=> contradiction => A is non-regular



THANK YOU

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

XJTLU | SCHOOL OF FILM AND TV ARTS