第 3 章　正则语言

(Part 2 of 3)

王　鑫

wangx AT tju.edu.cn

天津大学　智能与计算学部

Outline

Outline

1. Regular Expressions
   - Formal Definition of a Regular Expression
   - Equivalence With Finite Automata
2. Nonregular Languages

Regular Expressions

Arithmetic expressions

- Expressions: $(5+3) \times 4$
- Value: 32

Regular Expressions

Arithmetic expressions

- Expressions: $(5+3)\times 4$
- Value: 32

Regular expressions

- Expressions: $(0\cup 1)0^*$
- Value: a language

Regular Expressions

Arithmetic expressions

- Expressions: $(5 + 3) \times 4$
- Value: 32

Regular expressions

- Expressions: $(0 \cup 1)0^*$
- Value: a language

- 0: $\{0\}$

Regular Expressions

Arithmetic expressions

- Expressions: $(5+3)\times 4$
- Value: 32

Regular expressions

- Expressions: $(0\cup 1)0^*$
- Value: a language

- 0: $\{0\}$
- 1: $\{1\}$

Regular Expressions

Arithmetic expressions

- Expressions: $(5+3)\times 4$
- Value: 32

Regular expressions

- Expressions: $(0\cup 1)0^*$
- Value: a language

- 0: $\{0\}$
- 1: $\{1\}$
- $(0\cup 1)$: $\{0\}\cup\{1\}=\{0,1\}$

Regular Expressions

Arithmetic expressions

- Expressions: $(5+3)\times 4$
- Value: 32

Regular expressions

- Expressions: $(0\cup 1)0^*$
- Value: a language

- 0: $\{0\}$
- 1: $\{1\}$
- $(0\cup 1)$: $\{0\}\cup\{1\}=\{0,1\}$
- 0^*: $\{0\}^*$

Regular Expressions

Arithmetic expressions

- Expressions: $(5 + 3) \times 4$
- Value: 32

Regular expressions

- Expressions: $(0 \cup 1)0^*$
- Value: a language

- 0: $\{0\}$
- 1: $\{1\}$
- $(0 \cup 1)$: $\{0\} \cup \{1\} = \{0, 1\}$
- 0^*: $\{0\}^*$
- $(0 \cup 1)0^*$: $(0 \cup 1) \circ 0^*$

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes the language consisting of all strings over that alphabet

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes the language consisting of all strings over that alphabet
- $\Sigma^* 1$ is the language that

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes the language consisting of all strings over that alphabet
- Σ^*1 is the language that contains all strings that end in a 1

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes the language consisting of all strings over that alphabet
- $\Sigma^* 1$ is the language that contains all strings that end in a 1
- $(0\Sigma^*) \cup (\Sigma^* 1)$ consists of

Regular Expressions: Example

Another example of a regular expression is $(0 \cup 1)^*$

The value of this expression:

the language consisting of all possible strings of 0s and 1s.

- If $\Sigma = \{0, 1\}$, we can write Σ as shorthand for the regular expression $(0 \cup 1)$
- If Σ is any alphabet, the regular expression Σ describes the language consisting of all strings of length 1 over this alphabet
- Σ^* describes the language consisting of all strings over that alphabet
- $\Sigma^* 1$ is the language that contains all strings that end in a 1
- $(0\Sigma^*) \cup (\Sigma^* 1)$ consists of all strings that start with a 0 or end with a 1

Regular Expressions: Precedence

Precedence (优先级)

In regular expressions,

Regular Expressions: Precedence

Precedence (优先级)

In regular expressions,

- the star operation is done first,

Regular Expressions: Precedence

Precedence (优先级)

In regular expressions,

- the star operation is done first,
- followed by concatenation,

Regular Expressions: Precedence

Precedence (优先级)

In regular expressions,

- the star operation is done first,
- followed by concatenation,
- and finally union,

Regular Expressions: Precedence

Precedence (优先级)

In regular expressions,

- the star operation is done first,
- followed by concatenation,
- and finally union,
- unless parentheses change the usual order.

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
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, or

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
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, or
6. (R_1^*), where R_1 is a regular expressions.

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
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, or
6. (R_1^*), where R_1 is a regular expressions.

Don't confuse the regular expressions ε and $\emptyset$

Formal Definition of a Regular Expression

定义 (regular expression 正则表达式)

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
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, or
6. (R_1^*), where R_1 is a regular expressions.

Don't confuse the regular expressions ε and $\emptyset$

inductive definition

Formal Definition of a Regular Expression

For convenience,

- R^+: shorthand for RR^*

Formal Definition of a Regular Expression

For convenience,

- R^+: shorthand for RR^*
- $R^+ \cup \varepsilon = R^*$

Formal Definition of a Regular Expression

For convenience,

- R^+: shorthand for RR^*
- $R^+ \cup \varepsilon = R^*$
- R^k: shorthand for the concatenation of k R's with each other

Formal Definition of a Regular Expression

For convenience,

- R^+: shorthand for RR^*
- $R^+ \cup \varepsilon = R^*$
- R^k: shorthand for the concatenation of k R's with each other

To distinguish between a regular expression R and the language that it describes,

- we write $L(R)$ to be the language of R

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. 0^*10^*

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
3. $\Sigma^*001\Sigma^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring }\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w$ contains a single $1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w$ has at least one $1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as a substring $\}$
4. $1^*(01^+)^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring }\}$
4. $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w$ contains a single $1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w$ has at least one $1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as a substring $\}$
4. $1^*(01^+)^* = \{w \mid$ every 0 in w is followed by at least one $1\}$
5. $(\Sigma\Sigma)^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w$ contains a single $1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w$ has at least one $1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as a substring $\}$
4. $1^*(01^+)^* = \{w \mid$ every 0 in w is followed by at least one $1\}$
5. $(\Sigma\Sigma)^* = \{w \mid w$ is a string of even length $\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w$ contains a single $1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w$ has at least one $1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w$ contains the string 001 as a substring $\}$
4. $1^*(01^+)^* = \{w \mid$ every 0 in w is followed by at least one $1\}$
5. $(\Sigma\Sigma)^* = \{w \mid w$ is a string of even length $\}$
6. $(\Sigma\Sigma\Sigma)^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

1. $0^*10^* = \{w \mid w \text{ contains a single } 1\}$
2. $\Sigma^*1\Sigma^* = \{w \mid w \text{ has at least one } 1\}$
3. $\Sigma^*001\Sigma^* = \{w \mid w \text{ contains the string } 001 \text{ as a substring }\}$
4. $1^*(01^+)^* = \{w \mid \text{every } 0 \text{ in } w \text{ is followed by at least one } 1\}$
5. $(\Sigma\Sigma)^* = \{w \mid w \text{ is a string of even length }\}$
6. $(\Sigma\Sigma\Sigma)^* = \{w \mid \text{the length of } w \text{ is a multiple of } 3\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon)$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
11. $1^*\emptyset$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
11. $1^*\emptyset = \emptyset$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w \text{ starts and ends with the same symbol}\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
11. $1^*\emptyset = \emptyset$
12. $\emptyset^*$

Regular Expression: Examples

例

Assume that the alphabet Σ is $0, 1$

7. $01 \cup 10 = \{01, 10\}$
8. $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 = \{w \mid w$ starts and ends with the same symbol$\}$
9. $(0 \cup \varepsilon)1^* = 01^* \cup 1^*$
10. $(0 \cup \varepsilon)(1 \cup \varepsilon) = \{\varepsilon, 0, 1, 01\}$
11. $1^*\emptyset = \emptyset$
12. $\emptyset^* = \{\varepsilon\}$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset =$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon =$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

Exchanging $\emptyset$ and ε may cause the equalities to fail.

- $R \cup \varepsilon$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

Exchanging $\emptyset$ and ε may cause the equalities to fail.

- $R \cup \varepsilon$ may not equal R

  For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

Exchanging $\emptyset$ and ε may cause the equalities to fail.

- $R \cup \varepsilon$ may not equal R

  For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$
- $R \circ \emptyset$

Regular Expression: Examples

If we let R be any regular expression, we have the following identities.

- $R \cup \emptyset = R$
- $R \circ \varepsilon = R$

Exchanging $\emptyset$ and ε may cause the equalities to fail.

- $R \cup \varepsilon$ may not equal R

  For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \cup \varepsilon) = \{0, \varepsilon\}$
- $R \circ \emptyset$ may not equal R

  For example, if $R = 0$, then $L(R) = \{0\}$ but $L(R \circ \emptyset) = \emptyset$

Regular Expression: Applications

Regular expressions are useful tools in the design of compilers for programming languages.

Elemental objects in a programming language, called ***tokens***, such as the variable names and constants, may be described with regular expressions.

例 (A numerical constant)

$$(+ \cup \text{-} \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

where $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Regular Expression: Applications

Regular expressions are useful tools in the design of compilers for programming languages.

Elemental objects in a programming language, called ***tokens***, such as the variable names and constants, may be described with regular expressions.

例 (A numerical constant)

$$(+ \cup \text{-} \cup \varepsilon)(D^+ \cup D^+.D^* \cup D^*.D^+)$$

where $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Lexical analyzer: the part of a compiler that initially processes the input program

Equivalence With Finite Automata

定理

A language is regular if and only if some regular expression describes it.

Equivalence With Finite Automata

定理

A language is regular if and only if some regular expression describes it.

This theorem has two directions.

State and prove each direction as a separate lemma.

Equivalence With Finite Automata

引理

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

Equivalence With Finite Automata

引理

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

Proof idea:

Equivalence With Finite Automata

引理

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

Proof idea:

- We have a regular expression R describing some language A.

Equivalence With Finite Automata

引理

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

Proof idea:

- We have a regular expression R describing some language A.
- We show how to convert R into an NFA recognizing A.

Equivalence With Finite Automata

引理

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

Proof idea:

- We have a regular expression R describing some language A.
- We show how to convert R into an NFA recognizing A.
- By Corollary 1.40, if an NFA recognizes A then A is regular.

Equivalence With Finite Automata

引理

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

Proof.

Equivalence With Finite Automata

引理

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

Proof.

Let's convert R into an NFA N.

Equivalence With Finite Automata

引理

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

Proof.

Let's convert R into an NFA N.

Consider the 6 cases in the formal definition of regular expressions.

Equivalence With Finite Automata

引理

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

Proof.

Let's convert R into an NFA N.

Consider the 6 cases in the formal definition of regular expressions.

1. $R = a$ for some $a \in \Sigma$.

Equivalence With Finite Automata

引理

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

Proof.

Let's convert R into an NFA N.

Consider the 6 cases in the formal definition of regular expressions.

1. $R = a$ for some $a \in \Sigma$.

   Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.

Equivalence With Finite Automata

引理

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

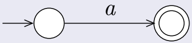
Proof.

Let's convert R into an NFA N.

Consider the 6 cases in the formal definition of regular expressions.

1. $R = a$ for some $a \in \Sigma$.

   Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.



Equivalence With Finite Automata

引理

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

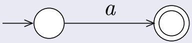
Proof.

Let's convert R into an NFA N.

Consider the 6 cases in the formal definition of regular expressions.

1. $R = a$ for some $a \in \Sigma$.

   Then $L(R) = \{a\}$, and the following NFA recognizes $L(R)$.

   $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$, where $\delta(q_1, a) = \{q_2\}$ and $\delta(r, b) = \emptyset$ for $r \neq q_1$ or $b \neq a$.

Equivalence With Finite Automata

引理

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

Proof.

2. $R = \varepsilon$.

Equivalence With Finite Automata

引理

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

Proof.

2. $R = \varepsilon$.

   Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.

Equivalence With Finite Automata

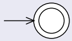
引理

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

Proof.

2. $R = \varepsilon$.

   Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.



Equivalence With Finite Automata

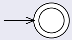
引理

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

Proof.

2. $R = \varepsilon$.

   Then $L(R) = \{\varepsilon\}$, and the following NFA recognizes $L(R)$.

   $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$, where $\delta(r, b) = \emptyset$ for any r and b.

Equivalence With Finite Automata

引理

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

Proof.

3. $R = \emptyset$.

Equivalence With Finite Automata

引理

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

Proof.

3. $R = \emptyset$.

   Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.

Equivalence With Finite Automata

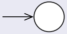
引理

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

Proof.

3. $R = \emptyset$.

   Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.



Equivalence With Finite Automata

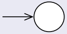
引理

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

Proof.

3. $R = \emptyset$.

   Then $L(R) = \emptyset$, and the following NFA recognizes $L(R)$.

   $N = (\{q\}, \Sigma, \delta, q, \emptyset)$, where $\delta(r, b) = \emptyset$ for any r and b.

Equivalence With Finite Automata

引理

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

Proof.

Equivalence With Finite Automata

引理

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

Proof.

3. $R = R_1 \cup R_2$

Equivalence With Finite Automata

引理

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

Proof.

3. $R = R_1 \cup R_2$
4. $R = R_1 \circ R_2$

Equivalence With Finite Automata

引理

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

Proof.

3. $R = R_1 \cup R_2$
4. $R = R_1 \circ R_2$
5. $R = R_1^*$

Equivalence With Finite Automata

引理

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

Proof.

3. $R = R_1 \cup R_2$
4. $R = R_1 \circ R_2$
5. $R = R_1^*$

For these three cases, we use the constructions given in the proofs that the class of regular languages is closed under the regular operations.

Equivalence With Finite Automata

引理

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

Proof.

3. $R = R_1 \cup R_2$
4. $R = R_1 \circ R_2$
5. $R = R_1^*$

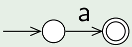
For these three cases, we use the constructions given in the proofs that the class of regular languages is closed under the regular operations.

We construct the NFA for R from the NFAs for R_1 and R_2 (or just R_1 in case 6) and the appropriate closure construction.

Equivalence With Finite Automata

例 (Converting $(ab \cup a)^*$ to an NFA)

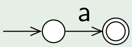
a



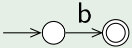
Equivalence With Finite Automata

例 (Converting $(ab \cup a)^*$ to an NFA)

a



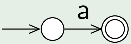
b



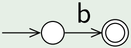
Equivalence With Finite Automata

例 (Converting $(ab \cup a)^*$ to an NFA)

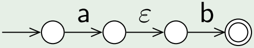
a



b



ab



Equivalence With Finite Automata

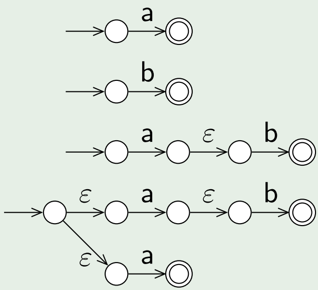
例 (Converting $(ab \cup a)^*$ to an NFA)

a

b

ab

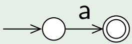
ab∪a



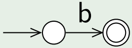
Equivalence With Finite Automata

例 (Converting $(ab \cup a)^*$ to an NFA)

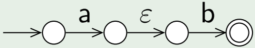
a



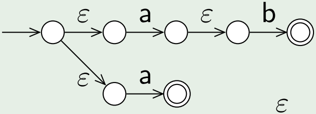
b



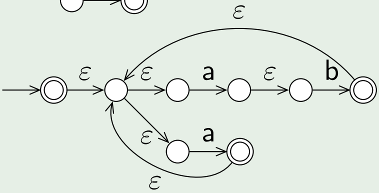
ab



$ab \cup a$



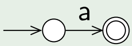
$(ab \cup a)^*$



Equivalence With Finite Automata

例 (Converting $(a \cup b)^* aba$ to an NFA)

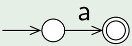
a



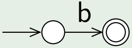
Equivalence With Finite Automata

例 (Converting $(a \cup b)^*aba$ to an NFA)

a



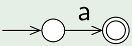
b



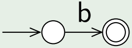
Equivalence With Finite Automata

例 (Converting $(a \cup b)^*aba$ to an NFA)

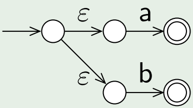
a



b

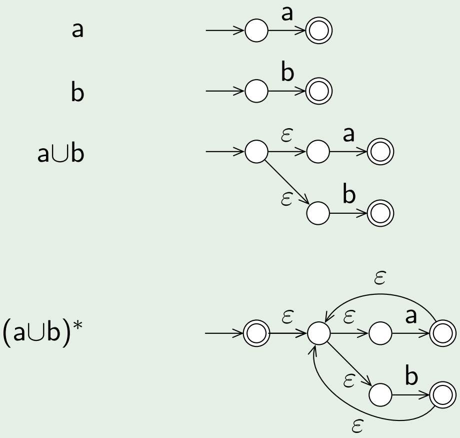


$a \cup b$



Equivalence With Finite Automata

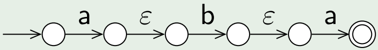
例 (Converting $(a\cup b)^*aba$ to an NFA)



Equivalence With Finite Automata

例 (Converting $(a \cup b)^*aba$ to an NFA)

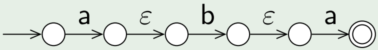
aba



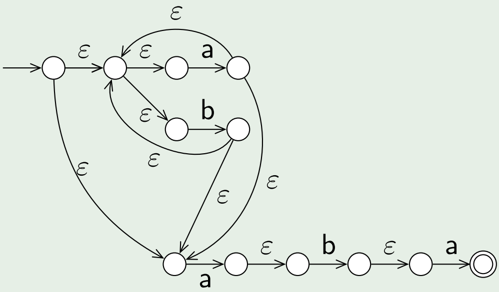
Equivalence With Finite Automata

例 (Converting $(a\cup b)^*aba$ to an NFA)

aba



$(a\cup b)^*aba$



Equivalence With Finite Automata

引理

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

Proof idea

- We need to show that if a language A is regular, a regular expression describes it.
- Because A is regular, it is accepted by a DFA.
- A procedure for converting DFAs into equivalent regular expressions.
  1. How to convert DFAs into GNFAs
  2. GNFAs into regular expressions

Generalized Nondeterministic Finite Automaton

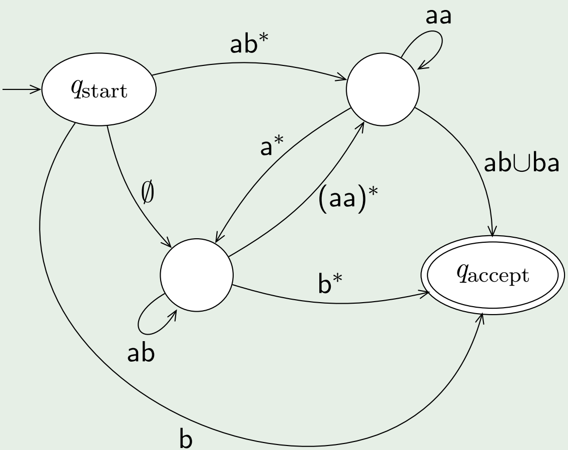
定义 (GNFA)

A ***generalized nondeterministic finite automaton*** (GNFA) (广义非确定型有穷自动机) is a 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \to \mathcal{R}$ is the transition function, where $\mathcal{R}$ is the collection of all regular expressions over the alphabet Σ,
4. q_{start} is the start state, and
5. q_{accept} is the accept state.

Generalized Nondeterministic Finite Automaton

例



Generalized Nondeterministic Finite Automaton

A GNFA is similar to an NFA except for the transition function.

- $\delta : (Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\}) \to \mathcal{R}$
- If $\delta(q_i, q_j) = R$, the arrow from state q_i to state q_j has the regular expression R as its label.
- The domain of δ is $(Q - \{q_{\text{accept}}\}) \times (Q - \{q_{\text{start}}\})$
  - An arrow connects every state to every other state (including itself),
  - except that no arrows are coming from q_{accept} or going to q_{start}.

Generalized Nondeterministic Finite Automaton

A GNFA accepts a string w in Σ^* if $w = w_1 w_2 \cdots w_k$, where $w_i \in \Sigma^*$ and a sequence of states $q_0, q_1, \ldots, q_k$ exists such that

1. $q_0 = q_{\text{start}}$
2. $q_k = q_{\text{accept}}$
3. $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$

Converting a DFA into a GNFA

Converting a DFA into a GNFA.

1. Add a new start state with an ε arrow to the old start state
2. Add a new accept state with ε arrows from the old accept states
3. If any arrows have multiple labels (or if there are multiple arrows going between the same two states in the same direction), we replace each with a single arrow whose label is the **union** of the previous labels
4. Add arrows labeled $\emptyset$ between states that had no arrows

Converting a GNFA into a Regular Expression

Converting a GNFA into a regular expression.

Say that the GNFA has k states. Because a GNFA must have a start and an accept state and they must be different from each other, we know that $k \geq 2$

1. If $k \geq 2$, we construct an equivalent GNFA with $k-1$ states. This step can be repeated on the new GNFA until it is reduced to two states.
2. If $k = 2$, the GNFA has a single arrow that goes from the start state to the accept state. The label of this arrow is the equivalent regular expression.

Converting a GNFA into a Regular Expression

Constructing an equivalent GNFA with one fewer state when $k > 2$

1. Selecting a state, ripping it out of the machine,
2. Repairing the remainder so that the same language is still recognized.

Converting a GNFA into a Regular Expression

Constructing an equivalent GNFA with one fewer state when $k > 2$

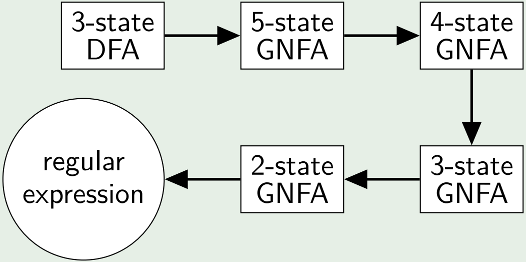
1. Selecting a state, ripping it out of the machine,
2. Repairing the remainder so that the same language is still recognized.

- Any state will do, provided that it is not the start or accept state.
- Let's call the removed state q_{rip}

Converting a GNFA into a Regular Expression

Converting a DFA with 3 states to an equivalent regular expression:

例 (Stages)

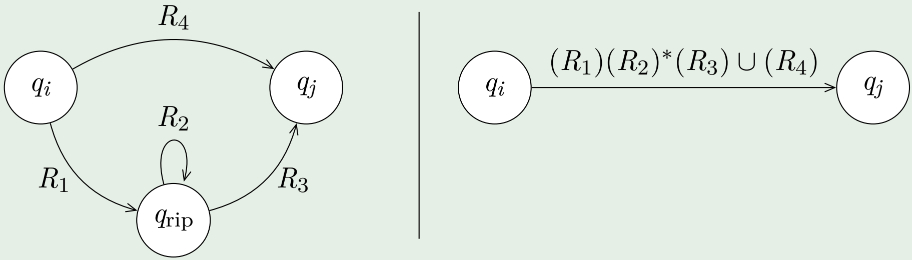


Converting a GNFA into a Regular Expression

Constructing an equivalent GNFA with one fewer state when $k > 2$

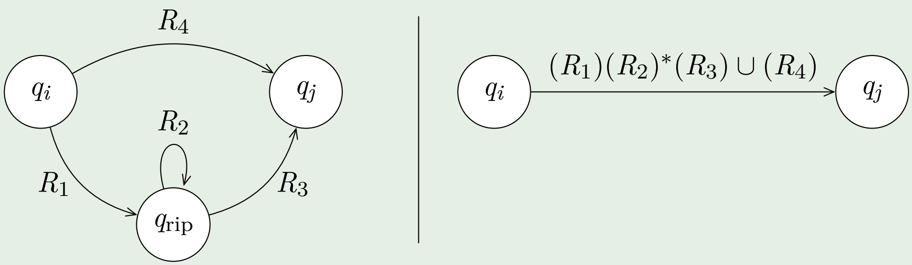
- After removing q_{rip}, the new label from q_i to q_j is a regular expression that describes all strings that would take the machine from q_i to q_j either directly or via q_{rip}

例



Constructing an equivalent GNFA with one fewer state

例



- We make this change for each arrow going from any state q_i to any state q_j, including the case where $q_i = q_j$.
- The new machine recognizes the original language.

Equivalence With Finite Automata

引理

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

Proof.

- We need to show that if a language A is regular, a regular expression describes it.
- Let M be a DFA such that $L(M) = A$
- Convert M to a GNFA G
- The procedure $CONVERT(G)$,
  - takes a GNFA and returns an equivalent regular expression

Equivalence With Finite Automata

$CONVERT(G)$

1. Let k be the number of states of G.
2. If $k = 2$, then G must consist of q_{start}, q_{accept}, and a single arrow connecting them and labeled with a regular expression R. Return R.
3. If $k > 2$, select any state $q_{\text{rip}} \in Q$ different from q_{start} and q_{accept} and let G' be the GNFA $(Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}})$, where
    - $Q' = Q - \{q_{\text{rip}}\}$
    - For any $q_i \in Q' - \{q_{\text{accept}}\}$ and $q_j \in Q' - \{q_{\text{start}}\}$, let $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$ for $R_1 = \delta(q_i, q_{\text{rip}})$, $R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $R_3 = \delta(q_{\text{rip}}, q_j)$, and $R_4 = \delta(q_i, q_j)$
4. Compute $CONVERT(G')$ and return this value.

Equivalence With Finite Automata

Claim

For any GNFA G, $CONVERT(G)$ is equivalent to G.

Proof.

We prove this claim by induction on k, the number of states of the GNFA.

Basis: Prove the claim true for $k = 2$ states.

- If G has only two states, it can have only a single arrow, which goes from q_{start} to q_{accept}.
- The regular expression label on this arrow describes all the strings that allow G to get to the accept state.
- Hence this expression is equivalent to G.

Equivalence With Finite Automata

Proof.

Induction step: Assume that the claim is true for $k-1$ states and use this assumption to prove that the claim is true for k states.

We show that G and G' recognize the same language.

- Suppose that G accepts an input w.
- G enters a sequence of states: $q_{\text{start}}, q_1, q_2, q_3, \dots, q_{\text{accept}}$
- If none of them is the removed state q_{rip}, clearly G' also accepts w.
    - The reason is that each of the new regular expressions labeling the arrows of G' contains the old regular expression as part of a union.

Equivalence With Finite Automata

Proof.

- If q_{rip} does appear,
  - removing each run of consecutive q_{rip} states forms an accepting computation for G'.
  - The states q_i and q_j bracketing a run have a new regular expression on the arrow between them that describes all strings taking q_i to q_j via q_{rip} on G.
- So G' accepts w.

Equivalence With Finite Automata

Proof.

- Conversely, suppose that G' accepts an input w.
- As each arrow between any two states q_i and q_j in G' describes the collection of strings taking q_i to q_j in G, either directly or via q_{rip},
- G must also accept w.

Thus G and G' are equivalent.

Equivalence With Finite Automata

Proof.

The induction hypothesis states that

- when the algorithm calls itself recursively on input G', the result is a regular expression that is equivalent to G'
  because G' has $k-1$ states.

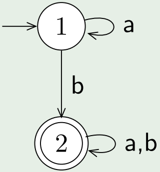
Hence this regular expression also is equivalent to G

For any GNFA G, $CONVERT(G)$ is equivalent to G.

This concludes the proof of the Claim, Lemma, and Theorem.

Converting a DFA into a regular expression

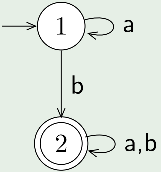
例 (Converting a 2-state DFA into an equivalent regular expression)



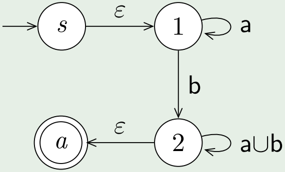
(a)

Converting a DFA into a regular expression

例 (Converting a 2-state DFA into an equivalent regular expression)



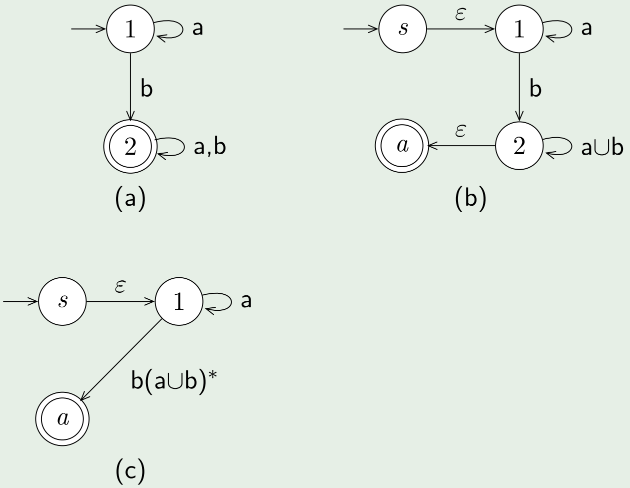
(a)



(b)

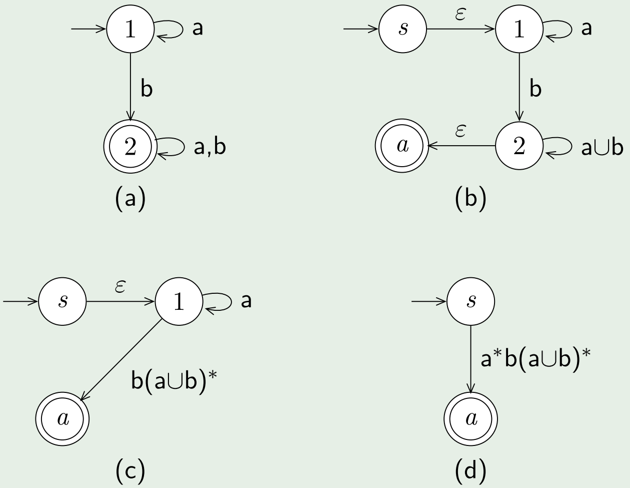
Converting a DFA into a regular expression

例 (Converting a 2-state DFA into an equivalent regular expression)



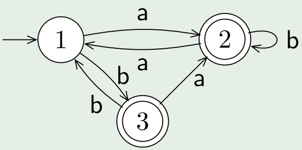
Converting a DFA into a regular expression

例 (Converting a 2-state DFA into an equivalent regular expression)



Converting a DFA into a regular expression

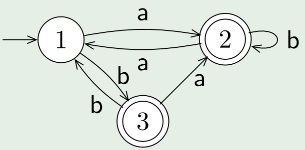
例 (Converting a 3-state DFA to an equivalent regular expression)



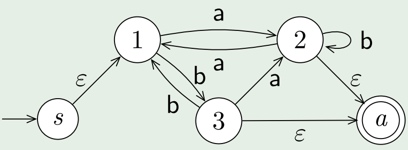
(a)

Converting a DFA into a regular expression

例 (Converting a 3-state DFA to an equivalent regular expression)



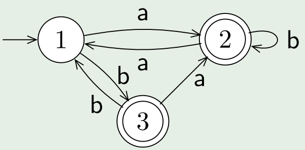
(a)



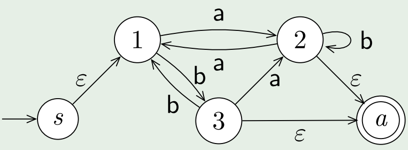
(b)

Converting a DFA into a regular expression

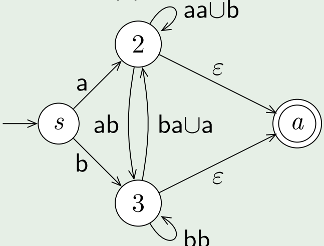
例 (Converting a 3-state DFA to an equivalent regular expression)



(a)



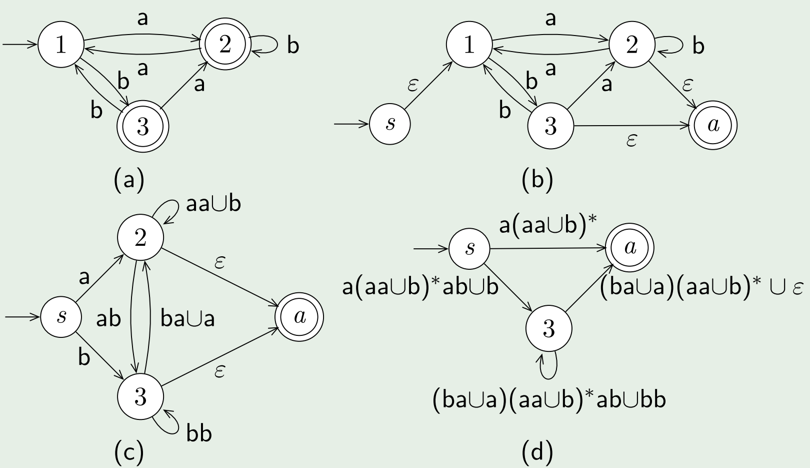
(b)



(c)

Converting a DFA into a regular expression

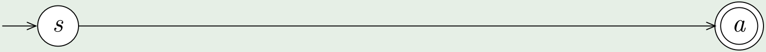
例 (Converting a 3-state DFA to an equivalent regular expression)



Converting a DFA into a regular expression

例

$$(a(aa\cup b)^*ab\cup b)((ba\cup a)(aa\cup b)^*ab\cup bb)^*((ba\cup a)(aa\cup b)^* \cup \varepsilon)\cup a(aa\cup b)^*$$



(d)

Outline

Nonregular Languages

To understand the power of finite automata,
you must also understand their limitations.

In this section, we show

- how to prove that certain languages cannot be recognized by any finite automaton

Nonregular Languages

The language $B = \{0^n 1^n \mid n \geq 0\}$

- The machine seems to need to remember how many 0s have been seen so far as it reads the input.
- Because the number of 0s isn't limited, the machine will have to keep track of an unlimited number of possibilities.
- But it cannot do so with any finite number of states.

Nonregular Languages

Consider two languages over the alphabet $\Sigma = \{0, 1\}$

- $C = \{w \mid w$ has an equal number of 0s and 1s$\}$
- $D = \{w \mid w$ has an equal number of occurrences of 01 and 10 as substrings$\}$

As expected, C is not regular.

Nonregular Languages

Consider two languages over the alphabet $\Sigma = \{0, 1\}$

- $C = \{w \mid w$ has an equal number of 0s and 1s$\}$
- $D = \{w \mid w$ has an equal number of occurrences of 01 and 10 as substrings$\}$

As expected, C is not regular.

But surprisingly D is regular!

Nonregular Languages

Consider two languages over the alphabet $\Sigma = \{0, 1\}$

- $C = \{w \mid w \text{ has an equal number of } 0\text{s and } 1\text{s}\}$
- $D = \{w \mid w \text{ has an equal number of occurrences of } 01 \text{ and } 10 \text{ as substrings}\}$

As expected, C is not regular.

But surprisingly D is regular!

Which is why we need mathematical proofs for certainty.

We show how to prove that certain languages are not regular.

The Pumping Lemma for Regular Languages

The ***pumping lemma*** (正则语言的泵引理)

The Pumping Lemma for Regular Languages

The ***pumping lemma*** (正则语言的泵引理)

- This theorem states that all regular languages have a special property.

The Pumping Lemma for Regular Languages

The ***pumping lemma*** **(正则语言的泵引理)**

- This theorem states that all regular languages have a special property.
- If we can show that a language does not have this property, we are guaranteed that it is not regular.

The Pumping Lemma for Regular Languages

The ***pumping lemma*** (正则语言的泵引理)

- This theorem states that all regular languages have a special property.
- If we can show that a language does not have this property, we are guaranteed that it is not regular.
- The 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.

The Pumping Lemma for Regular Languages

The ***pumping lemma*** (正则语言的泵引理)

- This theorem states that all regular languages have a special property.
- If we can show that a language does not have this property, we are guaranteed that it is not regular.
- The 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.
- That means each such string contains a section that can be repeated any number of times with the resulting string remaining in the language.

The Pumping Lemma for Regular Languages

定理 (Pumping lemma 泵引理)

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:

The Pumping Lemma for Regular Languages

定理 (Pumping lemma 泵引理)

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 $i \geq 0$, $xy^iz \in A$,

The Pumping Lemma for Regular Languages

定理 (Pumping lemma 泵引理)

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* $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, *and*

The Pumping Lemma for Regular Languages

定理 (Pumping lemma 泵引理)

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 $i \geq 0$, $xy^iz \in A$,*
2. *$|y| > 0$, and*
3. *$|xy| \leq p$.*

The Pumping Lemma for Regular Languages

Proof idea

The Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.

The Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.

- We assign the pumping length p to be the number of states of M.

The Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.

- We assign the pumping length p to be the number of states of M.
- We show that any string s in A of length at least p may be broken into the three pieces xyz, satisfying our three conditions.

The Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.

- We assign the pumping length p to be the number of states of M.
- We show that any string s in A of length at least p may be broken into the three pieces xyz, satisfying our three conditions.
- What if no strings in A are of length at least p?

The Pumping Lemma for Regular Languages

Proof idea

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.

- We assign the pumping length p to be the number of states of M.
- We show that any string s in A of length at least p may be broken into the three pieces xyz, satisfying our three conditions.
- What if no strings in A are of length at least p?
- Then our task is even easier because the theorem becomes vacuously true.

The Pumping Lemma for Regular Languages

Proof idea

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

- the sequence of states that M goes through when computing with input s has length $n+1$.

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

- the sequence of states that M goes through when computing with input s has length $n+1$.
- Because n is at least p, we know that $n+1$ is greater than p, the number of states of M.

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

- the sequence of states that M goes through when computing with input s has length $n+1$.
- Because n is at least p, we know that $n+1$ is greater than p, the number of states of M.
- Therefore, the sequence must contain a repeated state.

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

- the sequence of states that M goes through when computing with input s has length $n+1$.
- Because n is at least p, we know that $n+1$ is greater than p, the number of states of M.
- Therefore, the sequence must contain a repeated state.
- This result is an example of the pigeonhole principle.

The Pumping Lemma for Regular Languages

Proof idea

If we let n be the length of s,

- the sequence of states that M goes through when computing with input s has length $n+1$.
- Because n is at least p, we know that $n+1$ is greater than p, the number of states of M.
- Therefore, the sequence must contain a repeated state.
- This result is an example of the pigeonhole principle.

The Pumping Lemma for Regular Languages

Proof idea

The string s and the sequence of states that M goes through when processing s. State q_9 is the one that repeats.

$$\begin{array}{cccccccccccccccccc} s = & \uparrow & a_1 & \uparrow & a_2 & \uparrow & a_3 & \uparrow & a_4 & \uparrow & a_5 & \uparrow & a_6 & \uparrow & \cdots & \uparrow & a_n & \uparrow \\ & q_1 & & q_3 & & q_{20} & & {\color{red}q_9} & & q_{17} & & {\color{red}q_9} & & q_6 & & q_{35} & & q_{13} \end{array}$$

The Pumping Lemma for Regular Languages

Proof idea

The Pumping Lemma for Regular Languages

Proof idea

We now divide s into the three pieces x, y, and z.

The Pumping Lemma for Regular Languages

Proof idea

We now divide s into the three pieces x, y, and z.

- Piece x is the part of s appearing before q_9,

The Pumping Lemma for Regular Languages

Proof idea

We now divide s into the three pieces x, y, and z.

- Piece x is the part of s appearing before q_9,
- piece y is the part between the two appearances of q_9,

The Pumping Lemma for Regular Languages

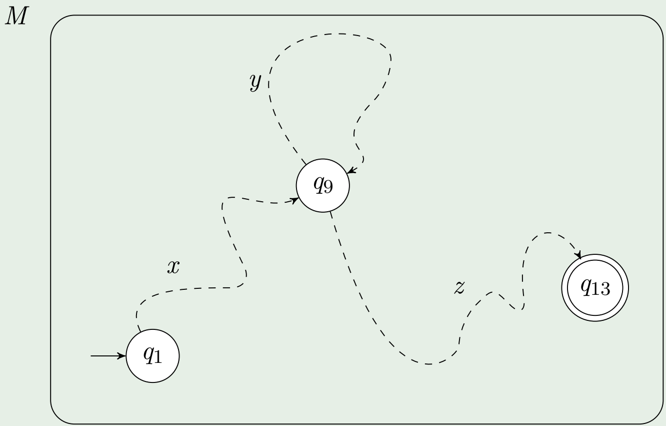
Proof idea

We now divide s into the three pieces x, y, and z.

- Piece x is the part of s appearing before q_9,
- piece y is the part between the two appearances of q_9,
- and piece z is the remaining part of s, coming after the second occurrence of q_9.

例 (Showing how the strings x, y, and z affect M)

So x takes M from the state q_1 to q_9, y takes M from q_9 back to q_9, and z takes M from q_9 to the accept state q_{13}.



The Pumping Lemma for Regular Languages

Proof.

The Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.

The Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.

- Let $s = a_1 a_2 \cdots a_n$ be a string in A of length n, where $n \geq p$.

The Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.

- Let $s = a_1 a_2 \cdots a_n$ be a string in A of length n, where $n \geq p$.
- Let $r_1, \cdots, r_{n+1}$ be the sequence of states that M enters while processing s, so $r_{i+1} = \delta(r_i, a_i)$ for $1 \leq i \leq n$.

The Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.

- Let $s = a_1 a_2 \cdots a_n$ be a string in A of length n, where $n \geq p$.
- Let $r_1, \cdots, r_{n+1}$ be the sequence of states that M enters while processing s, so $r_{i+1} = \delta(r_i, a_i)$ for $1 \leq i \leq n$.
- This sequence has length $n + 1$, which is at least $p + 1$.

The Pumping Lemma for Regular Languages

Proof.

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.

- Let $s = a_1 a_2 \cdots a_n$ be a string in A of length n, where $n \geq p$.
- Let $r_1, \cdots, r_{n+1}$ be the sequence of states that M enters while processing s, so $r_{i+1} = \delta(r_i, a_i)$ for $1 \leq i \leq n$.
- This sequence has length $n + 1$, which is at least $p + 1$.
- Among the first $p + 1$ elements in the sequence, two must be the same state, by the pigeonhole principle

The Pumping Lemma for Regular Languages

Proof.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.
- Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.
- Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$.
- Now let $x = a_1 \cdots a_{j-1}$, $y = a_j \cdots a_{l-1}$, and $z = a_l \cdots a_n$.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.
- Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$.
- Now let $x = a_1 \cdots a_{j-1}$, $y = a_j \cdots a_{l-1}$, and $z = a_l \cdots a_n$.
- As x takes M from r_1 to r_j, y takes M from r_j to r_j , and z takes M from r_j to r_{n+1}, which is an accept state, M must accept xy^iz for $i \geq 0$.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.
- Because r_l occurs among the first $p + 1$ places in a sequence starting at r_1, we have $l \leq p + 1$.
- Now let $x = a_1 \cdots a_{j-1}$, $y = a_j \cdots a_{l-1}$, and $z = a_l \cdots a_n$.
- As x takes M from r_1 to r_j, y takes M from r_j to r_j , and z takes M from r_j to r_{n+1}, which is an accept state, M must accept $xy^i z$ for $i \geq 0$.
- We know that $j \neq l$, so $|y| > 0$; and $l \leq p + 1$, so $|xy| \leq p$.

The Pumping Lemma for Regular Languages

Proof.

- We call the first of these r_j and the second r_l.
- Because r_l occurs among the first $p+1$ places in a sequence starting at r_1, we have $l \leq p+1$.
- Now let $x = a_1 \cdots a_{j-1}$, $y = a_j \cdots a_{l-1}$, and $z = a_l \cdots a_n$.
- As x takes M from r_1 to r_j, y takes M from r_j to r_j , and z takes M from r_j to r_{n+1}, which is an accept state, M must accept xy^iz for $i \geq 0$.
- We know that $j \neq l$, so $|y| > 0$; and $l \leq p+1$, so $|xy| \leq p$.
- Thus we have satisfied all conditions of the pumping lemma.

□

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

- first assume that B is regular in order to obtain a contradiction.

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

- first assume that B is regular in order to obtain a contradiction.
- Then use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in B can be pumped.

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

- first assume that B is regular in order to obtain a contradiction.
- Then use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in B can be pumped.
- Next, find a string s in B that has length p or greater but that cannot be pumped.

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

- first assume that B is regular in order to obtain a contradiction.
- Then use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in B can be pumped.
- Next, find a string s in B that has length p or greater but that cannot be pumped.
- Finally, demonstrate that s cannot be pumped by considering all ways of dividing s into x, y, and z and, for each such division, finding a value i where $xy^iz \notin B$.

The Pumping Lemma for Regular Languages

To use the pumping lemma to prove that a language B is not regular,

- first assume that B is regular in order to obtain a contradiction.
- Then use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in B can be pumped.
- Next, find a string s in B that has length p or greater but that cannot be pumped.
- Finally, demonstrate that s cannot be pumped by considering all ways of dividing s into x, y, and z and, for each such division, finding a value i where $xy^iz \notin B$.

The existence of s contradicts the pumping lemma if B were regular.

Hence B cannot be regular.

The Pumping Lemma for Regular Languages

例

Let B be the language $\{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to prove that B is not regular.

The Pumping Lemma for Regular Languages

例

Let B be the language $\{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to prove that B is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that B is regular.

The Pumping Lemma for Regular Languages

例

Let B be the language $\{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to prove that B is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that B is regular.
- Let p be the pumping length given by the pumping lemma.

The Pumping Lemma for Regular Languages

例

Let B be the language $\{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to prove that B is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that B is regular.
- Let p be the pumping length given by the pumping lemma.
- Choose s to be the string 0^p1^p.

The Pumping Lemma for Regular Languages

例

Let B be the language $\{0^n1^n \mid n \geq 0\}$. Use the pumping lemma to prove that B is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that B is regular.
- Let p be the pumping length given by the pumping lemma.
- Choose s to be the string 0^p1^p.
- Because s is a member of B and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in B.

We consider three cases to show that this result is impossible.

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.

We consider three cases to show that this result is impossible.

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
2. The string y consists only of 1s. This case also gives a contradiction.

We consider three cases to show that this result is impossible.

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
2. The string y consists only of 1s. This case also gives a contradiction.
3. The string y consists of both 0s and 1s. In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence it is not a member of B, which is a contradiction.

We consider three cases to show that this result is impossible.

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
2. The string y consists only of 1s. This case also gives a contradiction.
3. The string y consists of both 0s and 1s. In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence it is not a member of B, which is a contradiction.

Thus a contradiction is unavoidable if we make the assumption that B is regular, so B is not regular.

We consider three cases to show that this result is impossible.

1. The string y consists only of 0s. In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma. This case is a contradiction.
2. The string y consists only of 1s. This case also gives a contradiction.
3. The string y consists of both 0s and 1s. In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s. Hence it is not a member of B, which is a contradiction.

Thus a contradiction is unavoidable if we make the assumption that B is regular, so B is not regular.

Note that we can simplify this argument by applying condition 3 of the pumping lemma to eliminate cases 2 and 3.

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that C is regular.

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that C is regular.
- Let p be the pumping length given by the pumping lemma.

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that C is regular.
- Let p be the pumping length given by the pumping lemma.
- Choose s to be the string 0^p1^p.

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that C is regular.
- Let p be the pumping length given by the pumping lemma.
- Choose s to be the string 0^p1^p.
- With s being a member of C and having length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in C.

We would like to show that this outcome is impossible.

But wait, it is possible!

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

- If we let x and z be the empty string and y be the string 0^p1^p,

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

- If we let x and z be the empty string and y be the string $0^p 1^p$,
- then xy^iz always has an equal number of 0s and 1s and hence is in C.

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

- If we let x and z be the empty string and y be the string 0^p1^p,
- then xy^iz always has an equal number of 0s and 1s and hence is in C.

So it seems that s can be pumped.

Here condition 3 in the pumping lemma is useful.

- It stipulates that when pumping s, it must be divided so that $|xy| \leq p$.

例

Let $C = \{w \mid w \text{ has an equal number of } 0\text{s and } 1\text{s}\}$. Use the pumping lemma to prove that C is not regular.

- If we let x and z be the empty string and y be the string 0^p1^p,
- then xy^iz always has an equal number of 0s and 1s and hence is in C.

So it seems that s can be pumped.

Here condition 3 in the pumping lemma is useful.

- It stipulates that when pumping s, it must be divided so that $|xy| \leq p$.
- If $|xy| \leq p$, then y must consist only of 0s, so $xyyz \notin C$.

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

- If we let x and z be the empty string and y be the string 0^p1^p,
- then xy^iz always has an equal number of 0s and 1s and hence is in C.

So it seems that s can be pumped.

Here condition 3 in the pumping lemma is useful.

- It stipulates that when pumping s, it must be divided so that $|xy| \leq p$.
- If $|xy| \leq p$, then y must consist only of 0s, so $xyyz \notin C$.
- s cannot be pumped. That gives us the desired contradiction.

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

Need more care.

- If we had chosen $s = (01)^p$ instead, we would have run into trouble because we need a string that cannot be pumped and that string can be pumped, even taking condition 3 into account.

例

Let $C = \{w \mid w$ has an equal number of 0s and 1s$\}$. Use the pumping lemma to prove that C is not regular.

Need more care.

- If we had chosen $s = (01)^p$ instead, we would have run into trouble because we need a string that cannot be pumped and that string can be pumped, even taking condition 3 into account.
- Can you see how to pump it?

例

Let $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$. Use the pumping lemma to prove that C is not regular.

Need more care.

- If we had chosen $s = (01)^p$ instead, we would have run into trouble because we need a string that cannot be pumped and that string can be pumped, even taking condition 3 into account.
- Can you see how to pump it?
  - One way to do so sets $x = \varepsilon$, $y = 01$, and $z = (01)^{p-1}$.
  - Then $xy^iz \in C$ for every value of i.

If you fail on your first attempt to find a string that cannot be pumped, don't despair. Try another one!

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that F is regular.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that F is regular.
- Let p be the pumping length given by the pumping lemma.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that F is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s to be the string $0^p 1 0^p 1$.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that F is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s to be the string 0^p10^p1.
- Because s is a member of F and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, satisfying the three conditions of the lemma.

We show that this outcome is impossible.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof.

- Condition 3 is once again crucial because without it we could pump s if we let x and z be the empty string.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof.

- Condition 3 is once again crucial because without it we could pump s if we let x and z be the empty string.
- With condition 3 the proof follows because y must consist only of 0s, so $xyyz \notin F$

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof.

- Condition 3 is once again crucial because without it we could pump s if we let x and z be the empty string.
- With condition 3 the proof follows because y must consist only of 0s, so $xyyz \notin F$

Observe that we chose $s = 0^p10^p1$ to be a string that exhibits the "essence" of the nonregularity of F, as opposed to, say, the string 0^p0^p.

例

Let $F = \{ww \mid w \in \{0,1\}^*\}$. Use the pumping lemma to prove that F is not regular.

Proof.

- Condition 3 is once again crucial because without it we could pump s if we let x and z be the empty string.
- With condition 3 the proof follows because y must consist only of 0s, so $xyyz \notin F$

Observe that we chose $s = 0^p10^p1$ to be a string that exhibits the "essence" of the nonregularity of F, as opposed to, say, the string 0^p0^p. Even though 0^p0^p is a member of F, it fails to demonstrate a contradiction because it can be pumped.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that D is regular.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that D is regular.
- Let p be the pumping length given by the pumping lemma.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that D is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s to be the string 1^{p^2}.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that D is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s to be the string 1^{p^2}.
- Because s is a member of D and s has length at least p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, where for any $i \geq 0$ the string xy^iz is in D.

We show that this outcome is impossible.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.
- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.
- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.
- But $p^2 + p < p^2 + 2p + 1 = (p + 1)^2$.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.
- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.
- But $p^2 + p < p^2 + 2p + 1 = (p+1)^2$.
- Condition 2 implies that $|y| > 0$ and so $|xy^2z| > p^2$.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.
- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.
- But $p^2 + p < p^2 + 2p + 1 = (p+1)^2$.
- Condition 2 implies that $|y| > 0$ and so $|xy^2z| > p^2$.
- Therefore, $p^2 < |xy^2z| < (p+1)^2$. Hence this length cannot be a perfect square itself.

例

Let $D = \{1^{n^2} \mid n \geq 0\}$. Use the pumping lemma to prove that D is not regular.

Now consider the two strings xyz and xy^2z.

- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.
- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.
- But $p^2 + p < p^2 + 2p + 1 = (p+1)^2$.
- Condition 2 implies that $|y| > 0$ and so $|xy^2z| > p^2$.
- Therefore, $p^2 < |xy^2z| < (p+1)^2$. Hence this length cannot be a perfect square itself.
- So we arrive at the contradiction $xy^2z \notin D$ and conclude that D is not regular.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that E is regular.

例

Let $E = \{0^i1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that E is regular.
- Let p be the pumping length given by the pumping lemma.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that E is regular.
- Let p be the pumping length given by the pumping lemma.
- Let $s = 0^{p+1}1^p$.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

Proof. (The proof is by contradiction.)

- Assume to the contrary that E is regular.
- Let p be the pumping length given by the pumping lemma.
- Let $s = 0^{p+1}1^p$.
- Because s is a member of E and s has length at least p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, satisfying the conditions of the pumping lemma.

We show that this outcome is impossible.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

- By condition 3, y consists only of 0s.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

- By condition 3, y consists only of 0s.
- Let's examine the string $xyyz$ to see whether it can be in E.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

- By condition 3, y consists only of 0s.
- Let's examine the string $xyyz$ to see whether it can be in E.
- Adding an extra copy of y increases the number of 0s.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

- By condition 3, y consists only of 0s.
- Let's examine the string $xyyz$ to see whether it can be in E.
- Adding an extra copy of y increases the number of 0s.
- Increasing the number of 0s will still give a string in E.

No contradiction occurs. We need to try something else.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

The pumping lemma states that $xy^i z \in E$ even when $i = 0$,

- so let's consider the string $xy^0 z = xz$.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

The pumping lemma states that $xy^iz \in E$ even when $i = 0$,

- so let's consider the string $xy^0z = xz$.
- Because $|y| > 0$ and s has just one more 0 than 1,

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

The pumping lemma states that $xy^iz \in E$ even when $i = 0$,

- so let's consider the string $xy^0z = xz$.
- Because $|y| > 0$ and s has just one more 0 than 1,
- xz cannot have more 0s than 1s.

例

Let $E = \{0^i 1^j \mid i > j\}$. Use the pumping lemma to prove that D is not regular.

The pumping lemma states that $xy^iz \in E$ even when $i = 0$,

- so let's consider the string $xy^0z = xz$.
- Because $|y| > 0$ and s has just one more 0 than 1,
- xz cannot have more 0s than 1s.
- So it cannot be a member of E. Thus we obtain a contradiction.

Conclusion

Conclusion

- Regular Expressions
  - Formal Definitions
  - Equivalence With Finite Automata
    - From REs to NFAs
    - From DFAs to REs

Conclusion

- Regular Expressions
  - Formal Definitions
  - Equivalence With Finite Automata
    - From REs to NFAs
    - From DFAs to REs
- Nonregular Languages
  - The Pumping Lemma