第 3 章　正则语言

(Part 1 of 3)

王　鑫

wangx AT tju.edu.cn

天津大学　智能与计算学部

Outline

A Computational Model

The theory of computation begins with a question:

What is a computer?

A Computational Model

The theory of computation begins with a question:

What is a computer?

- ***Computational model***: an idealized computer.

A Computational Model

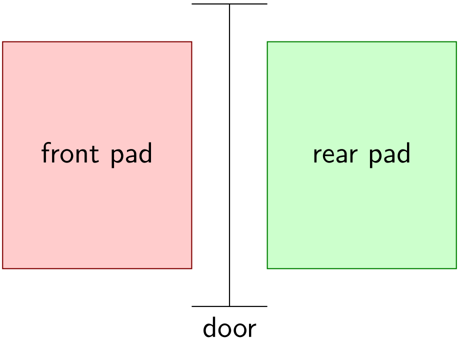
The theory of computation begins with a question:

What is a computer?

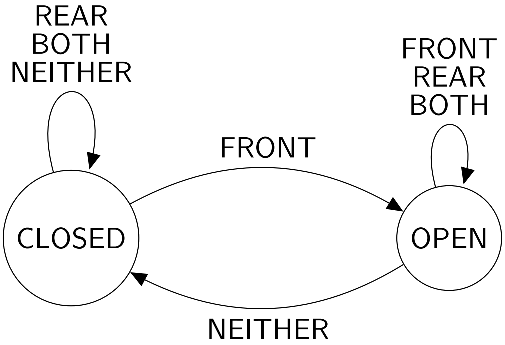
- ***Computational model***: an idealized computer.
- Several different computational models
  - ***Finite automata*** or ***finite state machine*** 有穷自动机
  - ***Pushdown automata*** 下推自动机
  - ***Linear-bounded automata*** 线性有界自动机
  - ***Turing machine*** 图灵机

Outline

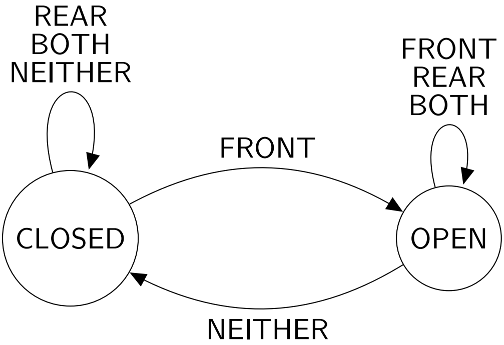
Example: An Automatic Door



Example: An Automatic Door



Example: An Automatic Door

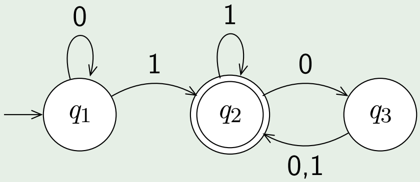


inpute signal

		NEITHER	FRONT	REAR	BOTH
state	CLOSED	CLOSED	OPEN	CLOSED	CLOSED
	OPEN	CLOSED	OPEN	OPEN	OPEN

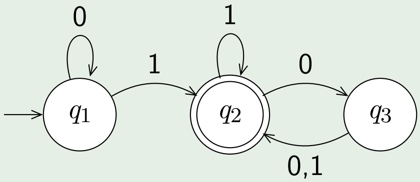
Example: A Finite Automaton

例 (A finite automaton M_1)



Example: A Finite Automaton

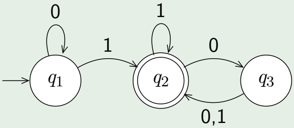
例 (A finite automaton M_1)



- the ***state diagram*** of M_1
- three ***states***: q_1, q_2, and q_3
- the ***start state***: q_1
- the ***accept state***: q_2
- ***transitions***: the arrows going from one state to another

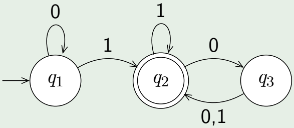
Example: A Finite Automaton

例 (A finite automaton M_1)



Example: A Finite Automaton

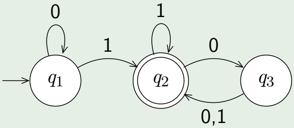
例 (A finite automaton M_1)



Feed the input string 1101 to the machine M_1

Example: A Finite Automaton

例 (A finite automaton M_1)

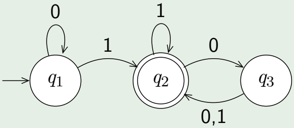


Feed the input string 1101 to the machine M_1

1. Start in state q_1

Example: A Finite Automaton

例 (A finite automaton M_1)

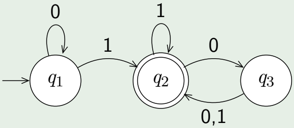


Feed the input string 1101 to the machine M_1

1. Start in state q_1
2. Read 1, follow transition from q_1 to q_2

Example: A Finite Automaton

例 (A finite automaton M_1)

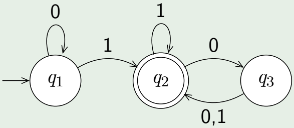


Feed the input string 1101 to the machine M_1

1. Start in state q_1
2. Read 1, follow transition from q_1 to q_2
3. Read 1, follow transition from q_2 to q_2

Example: A Finite Automaton

例 (A finite automaton M_1)

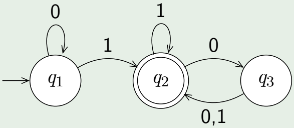


Feed the input string 1101 to the machine M_1

1. Start in state q_1
2. Read 1, follow transition from q_1 to q_2
3. Read 1, follow transition from q_2 to q_2
4. Read 0, follow transition from q_2 to q_3

Example: A Finite Automaton

例 (A finite automaton M_1)

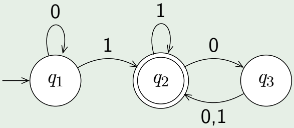


Feed the input string 1101 to the machine M_1

1. Start in state q_1
2. Read 1, follow transition from q_1 to q_2
3. Read 1, follow transition from q_2 to q_2
4. Read 0, follow transition from q_2 to q_3
5. Read 1, follow transition from q_3 to q_2

Example: A Finite Automaton

例 (A finite automaton M_1)



Feed the input string 1101 to the machine M_1

1. Start in state q_1
2. Read 1, follow transition from q_1 to q_2
3. Read 1, follow transition from q_2 to q_2
4. Read 0, follow transition from q_2 to q_3
5. Read 1, follow transition from q_3 to q_2
6. ***Accept*** because M_1 is in an accept state q_2 at the end of the input

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \to Q$ is the ***transition function***,

Formal Definition of a Finite Automaton

定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \to Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and

Formal Definition of a Finite Automaton

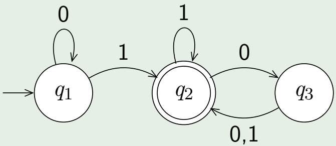
定义 (DFA (确定型有穷自动机))

A ***deterministic finite automaton*** (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \to Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

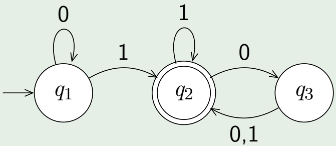
Using the Definition of DFA

例 (A finite automaton M_1)



Using the Definition of DFA

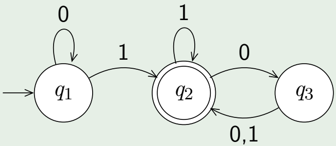
例 (A finite automaton M_1)



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

Using the Definition of DFA

例 (A finite automaton M_1)

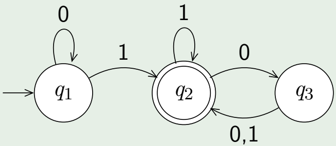


$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$

Using the Definition of DFA

例 (A finite automaton M_1)

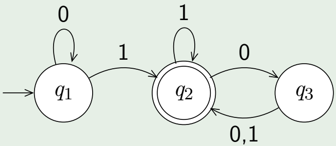


$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$

Using the Definition of DFA

例 (A finite automaton M_1)

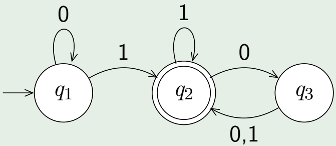


$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ is described as: $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$, $\delta(q_2, 0) = q_3$, $\delta(q_2, 1) = q_2$, $\delta(q_3, 0) = q_2$, $\delta(q_3, 1) = q_2$

Using the Definition of DFA

例 (A finite automaton M_1)

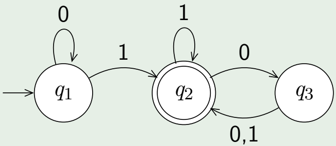


$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ is described as: $\delta(q_1, 0) = q_1, \quad \delta(q_1, 1) = q_2,$
   $\delta(q_2, 0) = q_3, \quad \delta(q_2, 1) = q_2, \quad \delta(q_3, 0) = q_2, \quad \delta(q_3, 1) = q_2$
4. q_1 is the start state, and

Using the Definition of DFA

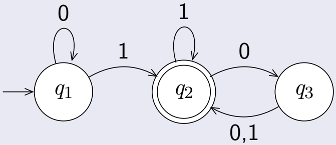
例 (A finite automaton M_1)



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

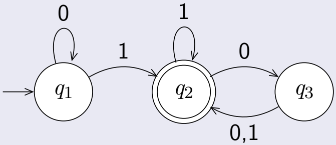
1. $Q = \{q_1, q_2, q_3\}$
2. $\Sigma = \{0, 1\}$
3. δ is described as: $\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_2$, $\delta(q_2, 0) = q_3$, $\delta(q_2, 1) = q_2$, $\delta(q_3, 0) = q_2$, $\delta(q_3, 1) = q_2$
4. q_1 is the start state, and
5. $F = \{q_2\}$

Language of DFA



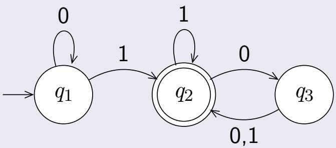
- If A is the set of all strings that machine M accepts, we say that A is the ***language of machine*** M and write $L(M) = A$.

Language of DFA



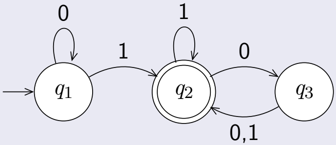
- If A is the set of all strings that machine M accepts, we say that A is the ***language of machine*** M and write $L(M) = A$.
- We say that M ***recognizes*** A.

Language of DFA



- If A is the set of all strings that machine M accepts, we say that A is the ***language of machine*** M and write $L(M) = A$.
- We say that M ***recognizes*** A.
- A machine may accept several strings, but it always recognizes only one language.

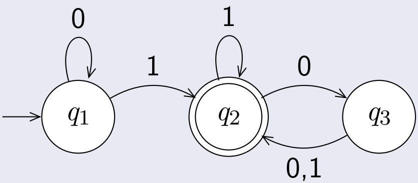
Language of DFA



- If A is the set of all strings that machine M accepts, we say that A is the ***language of machine*** M and write $L(M) = A$.
- We say that M ***recognizes*** A.
- A machine may accept several strings, but it always recognizes only one language.
- What about the machine accepts no strings?

Language of DFA M_1

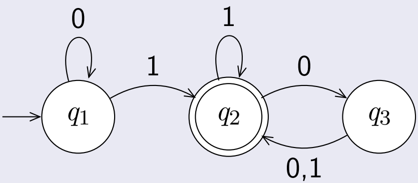
DFA M_1



$L(M_1) = ?$

Language of DFA M_1

DFA M_1



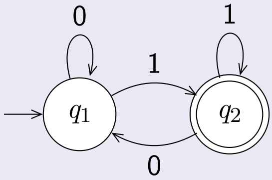
$L(M_1) = ?$

$L(M_1) =$

$A = \{w \mid w$ contains at least one 1 and an even number of 0s follow the last 1 $\}$

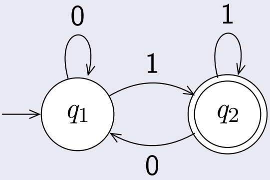
Example: DFA M_2

DFA M_2



Example: DFA M_2

DFA M_2



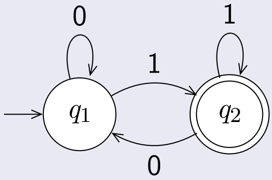
$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$$

δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

Example: DFA M_2

DFA M_2



$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$$

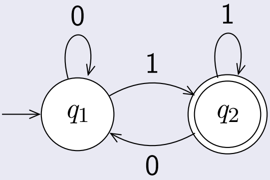
δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2) = ?$

Example: DFA M_2

DFA M_2



$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$$

δ:

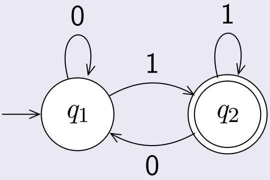
	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2) = ?$

try 1101,

Example: DFA M_2

DFA M_2



$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$$

δ:

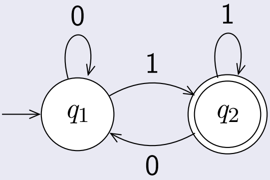
	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_2) = ?$

try 1101, try 110

Example: DFA M_2

DFA M_2



$$M_2 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_2\})$$

δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

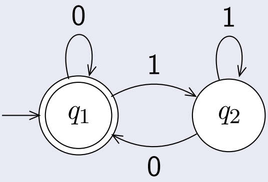
$L(M_2) = ?$

try 1101, try 110

$$L(M_2) = \{w \mid w \text{ ends in a } 1\}$$

Example: DFA M_3

DFA M_3



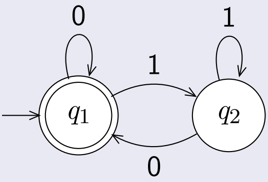
$$M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$$

δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

Example: DFA M_3

DFA M_3



$$M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$$

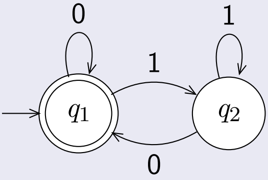
δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_3) = ?$

Example: DFA M_3

DFA M_3



$$M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$$

δ:

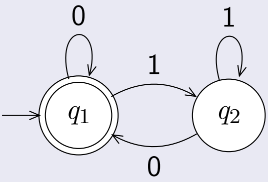
	0	1
q_1	q_1	q_2
q_2	q_1	q_2

$L(M_3) = ?$

$L(M_3) = \{w \mid w$ is the empty string ε or ends in a 0$\}$

Example: DFA M_3

DFA M_3



$$M_3 = (\{q_1, q_2\}, \{0, 1\}, \delta, q_1, \{q_1\})$$

δ:

	0	1
q_1	q_1	q_2
q_2	q_1	q_2

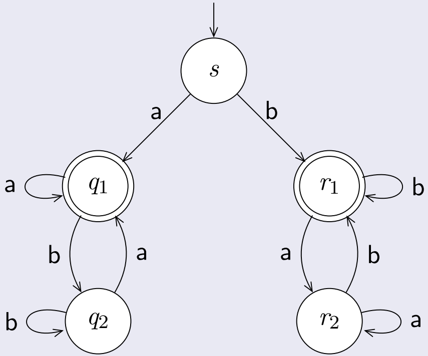
$L(M_3) = ?$

$$L(M_3) = \{w \mid w \text{ is the empty string } \varepsilon \text{ or ends in a } 0\}$$

What is the relationship between $L(M_2)$ and $L(M_3)$?

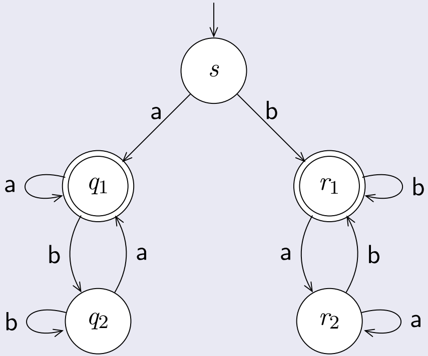
Example: DFA M_4

DFA M_4



Example: DFA M_4

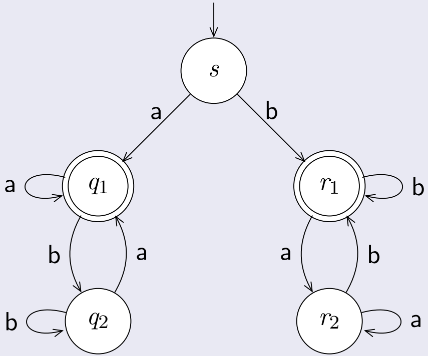
DFA M_4



$L(M_4) =$

Example: DFA M_4

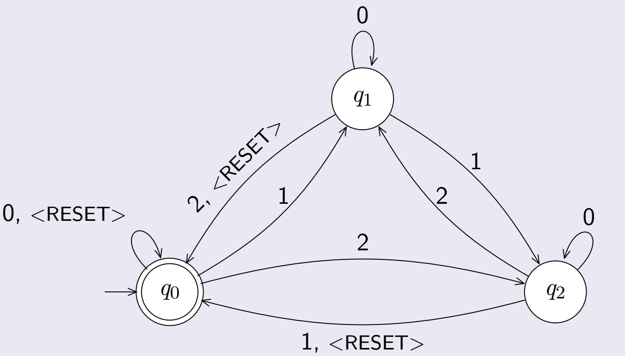
DFA M_4



$L(M_4) = \{w \mid w$ starts and ends with the same symbol $\}$

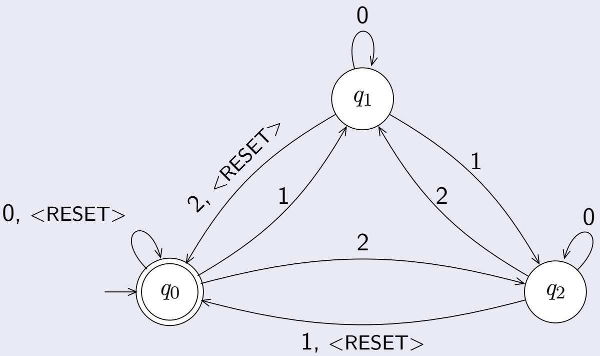
Example: DFA M_5

DFA M_5



Example: DFA M_5

DFA M_5



$L(M_5) =$

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears.

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears.

- For each A_i we give a DFA B_i, recognizing A_i.

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol $<$RESET$>$ appears.

- For each A_i we give a DFA B_i, recognizing A_i.
- $B_i = \{Q_i, \Sigma, \delta_i, q_0, \{q_0\}\}$

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears.

- For each A_i we give a DFA B_i, recognizing A_i.
- $B_i = \{Q_i, \Sigma, \delta_i, q_0, \{q_0\}\}$
  - $Q_i = \{q_0, q_1, q_2, \ldots, q_{i-1}\}$

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears.

- For each A_i we give a DFA B_i, recognizing A_i.
- $B_i = \{Q_i, \Sigma, \delta_i, q_0, \{q_0\}\}$
  - $Q_i = \{q_0, q_1, q_2, \ldots, q_{i-1}\}$
  - We design the transition function δ_i so that for each j, if B_i is in q_j, the running sum is j, modulo i.

Example: Generalization of M_5

- $\Sigma = \{$<RESET>, 0, 1, 2$\}$
- For each $i \geq 1$ let A_i be the language of all strings where the sum of the numbers is a multiple of i, except that the sum is reset to 0 whenever the symbol <RESET> appears.

- For each A_i we give a DFA B_i, recognizing A_i.
- $B_i = \{Q_i, \Sigma, \delta_i, q_0, \{q_0\}\}$
  - $Q_i = \{q_0, q_1, q_2, \ldots, q_{i-1}\}$
  - We design the transition function δ_i so that for each j, if B_i is in q_j, the running sum is j, modulo i.
  - $\delta_i(q_j, 0) = q_j$

    $\delta_i(q_j, 1) = q_k$, where $k = j + 1$ modulo i

    $\delta_i(q_j, 2) = q_k$, where $k = j + 2$ modulo i

    $\delta_i(q_j, \text{<RESET>}) = q_0$

Formal Definition of Computation for a DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- Let $w = a_1 a_2 \ldots a_n$ be a string where $a_i \in \Sigma$.
- Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

Formal Definition of Computation for a DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- Let $w = a_1 a_2 \dots a_n$ be a string where $a_i \in \Sigma$.
- Then M accepts w if a sequence of states $r_0, r_1, \dots, r_n$ in Q exists with three conditions:
  1. $r_0 = q_0$

Formal Definition of Computation for a DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- Let $w = a_1 a_2 \ldots a_n$ be a string where $a_i \in \Sigma$.
- Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:
  1. $r_0 = q_0$
  2. $\delta(r_i, a_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$

Formal Definition of Computation for a DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- Let $w = a_1 a_2 \ldots a_n$ be a string where $a_i \in \Sigma$.
- Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:
  1. $r_0 = q_0$
  2. $\delta(r_i, a_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$
  3. $r_n \in F$

Formal Definition of Computation for a DFA

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA.
- Let $w = a_1 a_2 \ldots a_n$ be a string where $a_i \in \Sigma$.
- Then M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:
  1. $r_0 = q_0$
  2. $\delta(r_i, a_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$
  3. $r_n \in F$

We say that M recognizes language A if $A = \{w \mid M \text{ accepts } w\}$

Regular Language

定义 (regular language)

A language is called a regular language if some DFA recognizes it.

Regular Language

定义 (regular language)

A language is called a regular language if some DFA recognizes it.

例

- Take DFA M_5
- $w =$10<RESET>22<RESET>012

Regular Language

定义 (regular language)

A language is called a regular language if some DFA recognizes it.

例

- Take DFA M_5
- $w =$10<RESET>22<RESET>012
- The sequence of states M_5 enters when computing on w is

  $q_0, q_1, q_1, q_0, q_2, q_1, q_0, q_0, q_1, q_0$

  which satisfies the three conditions.

Regular Language

定义 (regular language)

A language is called a regular language if some DFA recognizes it.

例

- Take DFA M_5
- w =10<RESET>22<RESET>012
- The sequence of states M_5 enters when computing on w is

  $q_0, q_1, q_1, q_0, q_2, q_1, q_0, q_0, q_1, q_0$

  which satisfies the three conditions.

$L(M_5) = \{w \mid$ the sum of the symbols in w is 0 modulo 3, except that <RESET> resets the count to 0 $\}$

Designing Finite Automata

An approach helpful: “reader as automaton"

- put ***yourself*** in the place of the machine you are trying to design
- and then see how you would go about performing the machine’s task

Designing Finite Automata

An approach helpful: “reader as automaton"

- put ***yourself*** in the place of the machine you are trying to design
- and then see how you would go about performing the machine’s task

例

- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.

Designing Finite Automata

例

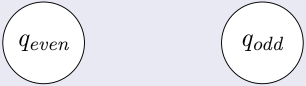
- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.
  1. q_{even}: even so far
  2. q_{odd}: odd so far

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.
  1. q_{even}: even so far
  2. q_{odd}: odd so far

DFA E_1

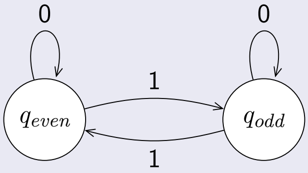


Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.
  1. q_{even}: even so far
  2. q_{odd}: odd so far

DFA E_1

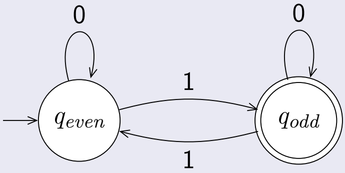


Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language consists of all strings with an odd number of 1s.
- Construct a finite automaton E_1 to recognize this language.
  1. q_{even}: even so far
  2. q_{odd}: odd so far

DFA E_1



Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language of all strings that contain the string 001 as a substring.
- To design a finite automaton E_2 to recognize this language.

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language of all strings that contain the string 001 as a substring.
- To design a finite automaton E_2 to recognize this language.
  1. q: haven't just seen any symbols of the pattern

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language of all strings that contain the string 001 as a substring.
- To design a finite automaton E_2 to recognize this language.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language of all strings that contain the string 001 as a substring.
- To design a finite automaton E_2 to recognize this language.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00

Designing Finite Automata

例

- $\Sigma = \{0, 1\}$
- The language of all strings that contain the string 001 as a substring.
- To design a finite automaton E_2 to recognize this language.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

Designing Finite Automata

例

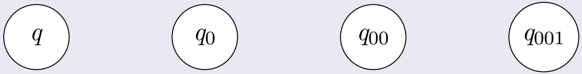
- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

DFA E_2



Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

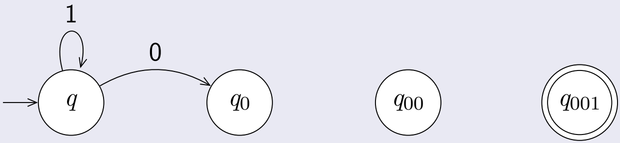
DFA E_2

Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

DFA E_2

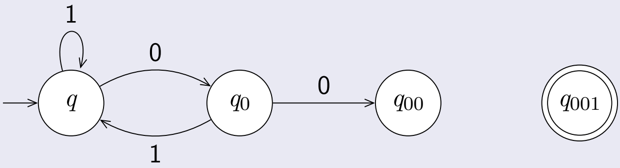


Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

DFA E_2

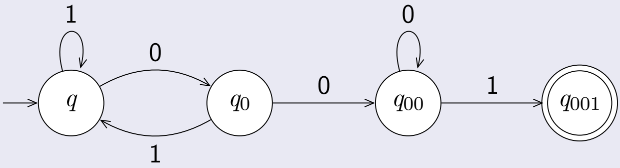


Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

DFA E_2

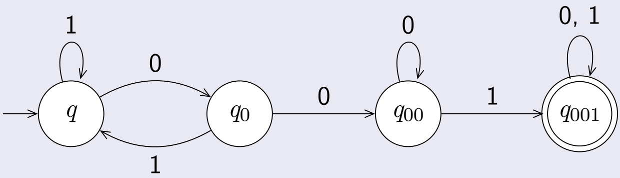


Designing Finite Automata

例

- The language of all strings that contain the string 001 as a substring.
  1. q: haven't just seen any symbols of the pattern
  2. q_0: have just seen a 0
  3. q_{00} : have just seen 00
  4. q_{001}: have seen the entire pattern 001

DFA E_2



The Regular Operations

定义 (regular operations)

Let A and B be language. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

The Regular Operations

定义 (regular operations)

Let A and B be language. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$

The Regular Operations

定义 (regular operations)

Let A and B be language. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$

The Regular Operations

定义 (regular operations)

Let A and B be language. We define the regular operations ***union***, ***concatenation***, and ***star*** as follows:

- **Union**: $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$
- **Concatenation**: $A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$
- **Star**: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

The Regular Operations

例

Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

If $A = \{$ good, bad$\}$ and $B = \{$boy, girl$\}$, then

The Regular Operations

例

Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

If $A = \{$ good, bad$\}$ and $B = \{$boy, girl$\}$, then

- $A \cup B = \{$good, bad, boy, girl$\}$

The Regular Operations

例

Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

If $A = \{$ good, bad$\}$ and $B = \{$boy, girl$\}$, then

- $A \cup B = \{$good, bad, boy, girl$\}$
- $A \circ B = \{$goodboy, goodgirl, badboy, badgirl$\}$

The Regular Operations

例

Let the alphabet Σ be the standard 26 letters {a, b, . . . , z}.

If $A = \{$ good, bad$\}$ and $B = \{$boy, girl$\}$, then

- $A \cup B = \{$good, bad, boy, girl$\}$
- $A \circ B = \{$goodboy, goodgirl, badboy, badgirl$\}$
- $A^* = \{\varepsilon$, good, bad, goodgood, goodbad, badgood, badbad, goodgoodgood, goodgoodbad, goodbadgood, goodbadbad,. . . $\}$

The Regular Operations

Closed

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and

M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and

M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and

M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$

The Regular Operations

Closed

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

Let M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, and

M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cup A_2$, where $M = (Q, \Sigma, \delta, q_0, F)$.

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
2. Σ is the same as in M_1 and M_2

□

The Regular Operations

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

3. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let

   $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

The Regular Operations

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

3. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let
   $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$
4. q_0 is the pair (q_1, q_2)

The Regular Operations

定理

The class of regular languages is closed under the union operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.

Proof.

3. For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let
$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$
4. q_0 is the pair (q_1, q_2)
5. $F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ or } r_2 \in F_2\}$

□

The Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.

The Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.

Problem: M doesn't know where to break the input string?

Outline

1 Finite Automata

2 Nondeterminism

- Formal Definition of a Nondeterministic Finite Automaton
- Equivalence of NFAs and DFAs
- Closure Under the Regular Operations

Nondeterminism 非确定性

- ***Determinism***: When the machine is in a given state and reads the next input symbol, we know what the next state will be it is determined. We call this ***deterministic*** computation.
- ***Nondeterminism***: In a ***nondeterministic*** machine, several choices may exist for the next state at any point.

Nondeterminism 非确定性

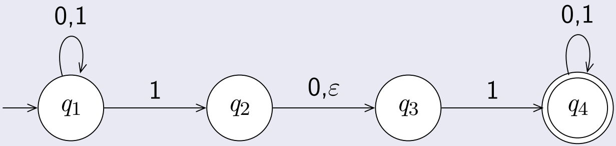
- ***Determinism***: When the machine is in a given state and reads the next input symbol, we know what the next state will be it is determined. We call this ***deterministic*** computation.
- ***Nondeterminism***: In a ***nondeterministic*** machine, several choices may exist for the next state at any point.

- Nondeterminism is a generalization of determinism,
- so every deterministic finite automaton is automatically a nondeterministic finite automaton.

Nondeterministic Finite Automata

NFA N_1

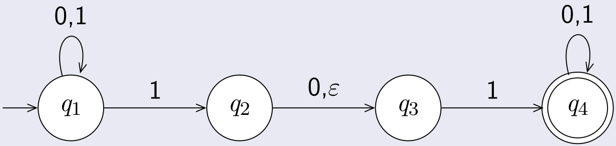
- Nondeterministic finite automata may have additional features.



Nondeterministic Finite Automata

NFA N_1

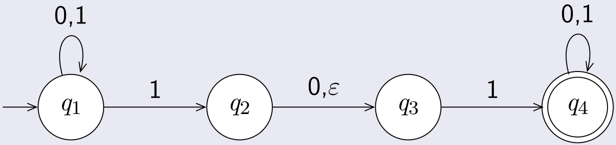
- Nondeterministic finite automata may have additional features.



- DFA: deterministic finite automaton 确定型有穷自动机
- NFA: nondeterministic finite automaton 非确定型有穷自动机

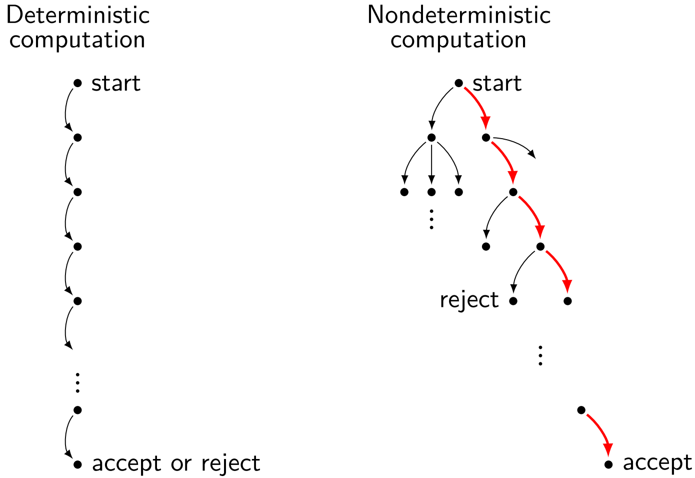
NFAs

NFA N_1

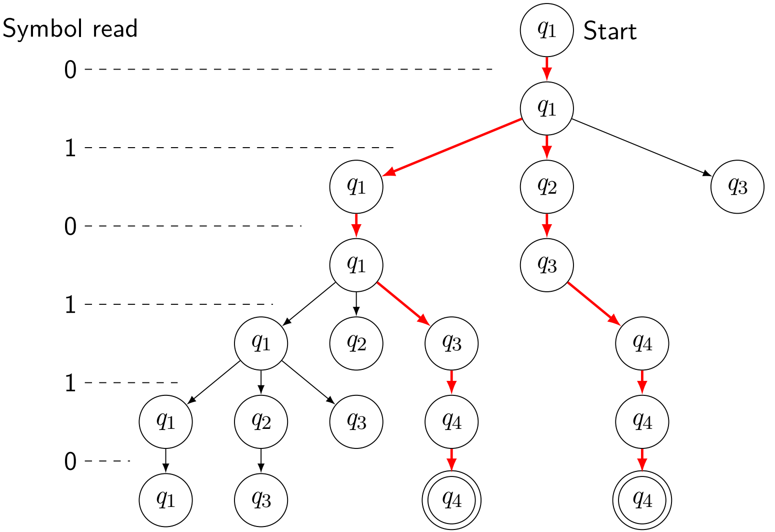


- DFA:
  1. every state of a DFA always has exactly one exiting transition arrow for each symbol in the alphabet.
- NFA:
  1. a state may have zero, one, or many exiting arrows for each alphabet symbol.
  2. an NFA may have arrows labeled with members of the alphabet or ε.

Deterministic and Nondeterministic Computations

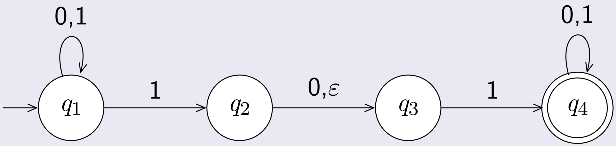


How Does an NFA Compute?



NFAs

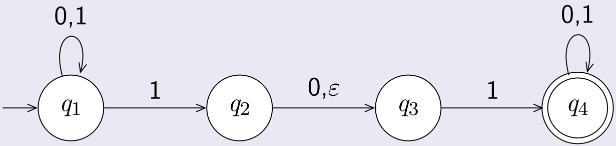
NFA N_1



- $L(N_1) = ?$

NFAs

NFA N_1



- $L(N_1) = ?$ $\{w \mid w$ contain either 101 or 11 as a substring$\}$

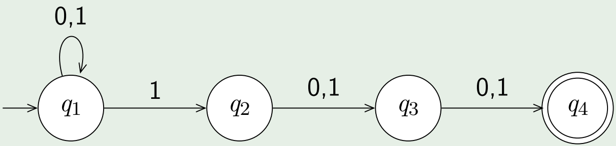
Example: NFA N_2

- The language A:
    - $\{$the language consisting of all strings over $\{0, 1\}$ containing a 1 in the third position from the end$\}$
    - e.g., 000100 $\in A$, 0011 $\notin A$

Example: NFA N_2

- The language A:
    - {the language consisting of all strings over $\{0, 1\}$ containing a 1 in the third position from the end}
    - e.g., 000100 $\in A$, 0011 $\notin A$

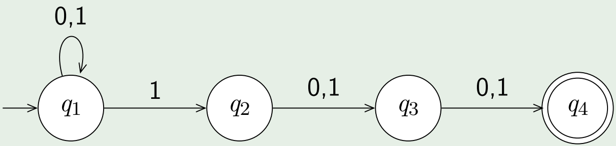
例 (NFA N_2)



Example: NFA N_2

- The language A:
  - {the language consisting of all strings over $\{0, 1\}$ containing a 1 in the third position from the end}
  - e.g., 000100 $\in A$, 0011 $\notin A$

例 (NFA N_2)

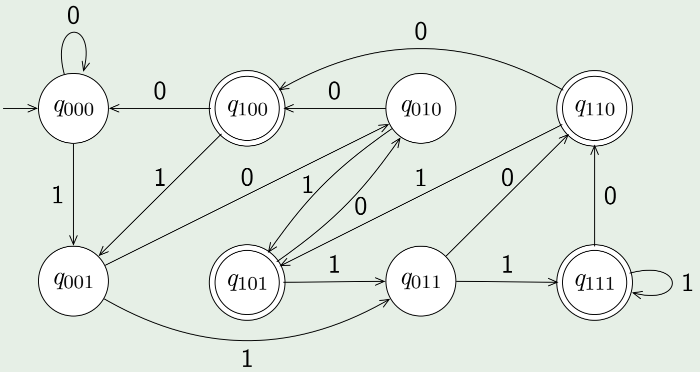


$L(N_2) = A$

Example: NFA N_2

Every NFA can be converted into an equivalent DFA.

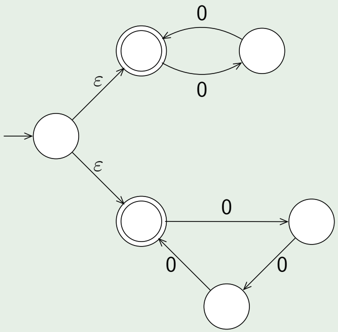
例 (The equivalent DFA of NFA N_2)



Example: NFA N_3

The convenience of having ε arrows

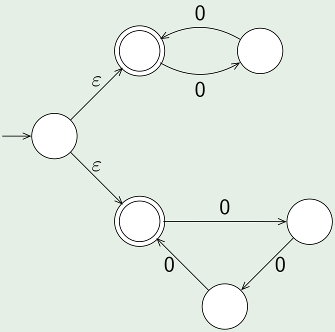
例 (NFA N_3)



Example: NFA N_3

The convenience of having ε arrows

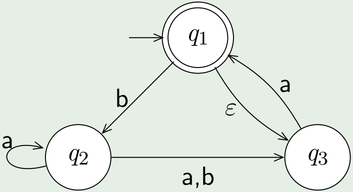
例 (NFA N_3)



$L(N_3) = \{$all strings of the form 0^k where k is a multiple of 2 or 3. $\}$

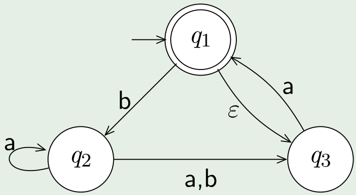
Example: NFA N_4

例 (NFA N_4)



Example: NFA N_4

例 (NFA N_4)



- it accepts the strings ε, a, baba, baa
- it accepts it doesn't accept the strings b, bb, babba

Formal Definition of a Nondeterministic Finite Automaton

定义 (NFA)

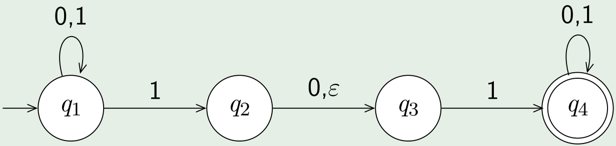
A ***nondeterministic finite automaton*** (NFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set of states,
2. Σ is a finite alphabet,
3. $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$ is the transition function,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

- $\mathcal{P}(Q)$ is the power set of Q
- $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

Example: The Formal Definition of NFA N_1

例 (Recall the NFA N_1)



$N_1 = (Q, \Sigma, \delta, q_1, F)$, where

- $Q = \{q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0, 1\}$
- δ is given as
- q_1 is the start state
- $F = \{q_4\}$

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	$\emptyset$
q_2	$\{q_3\}$	$\emptyset$	$\{q_3\}$
q_3	$\emptyset$	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$	$\emptyset$

Formal Definition of Computation for an NFA

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be an NFA.
- Let w be a string over Σ.
- Then N accepts w if we can write w as $w = a_1 a_2 \cdots a_n$, where $a_i \in \Sigma_\varepsilon$ and a sequence of states $r_0, r_1, \ldots, r_n$ exists in Q with three conditions:
  1. $r_0 = q_0$
  2. $r_{i+1} \in \delta(r_i, a_{i+1})$, for $i = 0, \ldots, n-1$
  3. $r_n \in F$

Equivalence of NFAs and DFAs

DFA and NFA recognize the same class of languages.

Equivalence of NFAs and DFAs

DFA and NFA recognize the same class of languages.

- Surprising: NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages

Equivalence of NFAs and DFAs

DFA and NFA recognize the same class of languages.

- Surprising: NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages
- Useful: describing an NFA for a given language sometimes is much easier than describing a DFA for that language

Equivalence of NFAs and DFAs

DFA and NFA recognize the same class of languages.

- Surprising: NFAs appear to have more power than DFAs, so we might expect that NFAs recognize more languages
- Useful: describing an NFA for a given language sometimes is much easier than describing a DFA for that language

Equivalent

Say that two machines are ***equivalent*** if they recognize the same language.

Equivalence of NFAs and DFAs

定理

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Equivalence of NFAs and DFAs

定理

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Proof.

- Let $N = (Q, \Sigma, \delta, q_0, F)$ be the NFA recognizing some language A.
- We construct a DFA $M = (Q', \Sigma, \delta', q'_0, F')$ recognizing A.
- Before doing the full construction, let's first consider the easier case wherein N has no ε arrows. Later we take the ε arrows into account.

□

Equivalence of NFAs and DFAs

定理

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Proof.

1. $Q' = \mathcal{P}(Q)$
2. For $R \in Q'$ and $a \in \Sigma$,
   $\delta'(R, a) = \{q \in Q \mid q \in \delta(r, a)$ for some $r \in R\}$

$$\delta'(R, a) = \bigcup_{r \in R} \delta(r, a)$$

$\square$

Equivalence of NFAs and DFAs

定理

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

Proof.

3. $q'_0 = \{q_0\}$
4. $F' = \{R \in Q' \mid R \text{ contains an accept state of } N\}$

$\square$

Equivalence of NFAs and DFAs

Proof.

Now we need to consider the ε arrows.

Equivalence of NFAs and DFAs

Proof.

Now we need to consider the ε arrows.

- For any state R of M,
  $E(R) = \{q \mid q$ can be reached from R by traveling along 0 or more ε arrows$\}$
  - $E(R)$ is the collection of states that can be reached from members of R by going only along ε arrows, including the members of R themselves.
- $\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a))$ for some $r \in R\}$
- $q_0' = E(\{q_0\})$

Equivalence of NFAs and DFAs

Proof.

Now we need to consider the ε arrows.

- For any state R of M,
  $E(R) = \{q \mid q$ can be reached from R by traveling along 0 or more ε arrows$\}$
    - $E(R)$ is the collection of states that can be reached from members of R by going only along ε arrows, including the members of R themselves.
- $\delta'(R, a) = \{q \in Q \mid q \in E(\delta(r, a))$ for some $r \in R\}$
- $q_0' = E(\{q_0\})$

We have now completed the construction of the DFA M that simulates the NFA N. $\square$

Equivalence of NFAs and DFAs

定理

Every nondeterministic finite automaton has an equivalent deterministic finite automaton.

推论

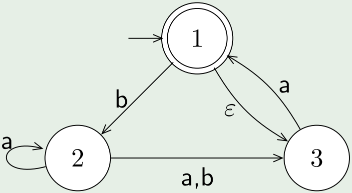
A language is regular if and only if some nondeterministic finite automaton recognizes it.

Equivalence of NFAs and DFAs

例 (NFA N_4)

NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$

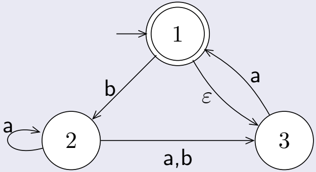
- $Q = \{1, 2, 3\}$
- $\Sigma = \{\text{a}, \text{b}\}$
- δ
- $q_0 = 1$
- $F = \{1\}$



Construct a DFA D that is equivalent to N_4

Equivalence of NFAs and DFAs

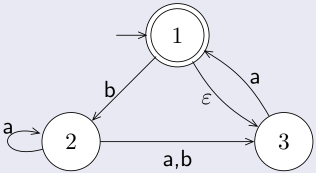
NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$



DFA $D = (Q', \Sigma, \delta', q_0', F')$

Equivalence of NFAs and DFAs

NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$

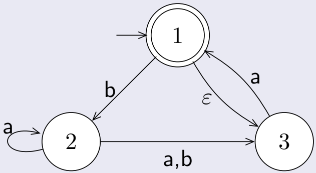


DFA $D = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

Equivalence of NFAs and DFAs

NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$

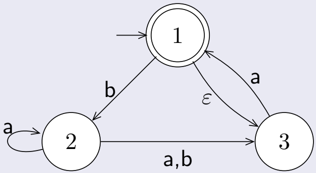


DFA $D = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
- $\Sigma = \{$a, b$\}$

Equivalence of NFAs and DFAs

NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$

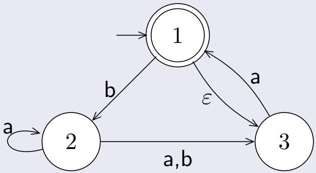


DFA $D = (Q', \Sigma, \delta', q_0', F')$

- $Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
- $\Sigma = \{\mathsf{a}, \mathsf{b}\}$
- $q_0' = E(\{q_0\}) = E(\{1\}) = \{1, 3\}$

Equivalence of NFAs and DFAs

NFA $N_4 = (Q, \Sigma, \delta, q_0, F)$

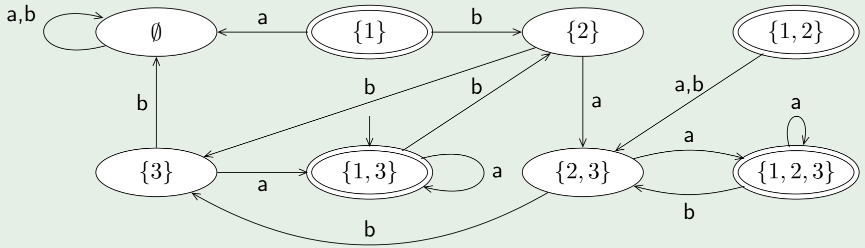


DFA $D = (Q', \Sigma, \delta', q'_0, F')$

- $Q' = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$
- $\Sigma = \{\text{a, b}\}$
- $q'_0 = E(\{q_0\}) = E(\{1\}) = \{1,3\}$
- $F' = \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\}$

Equivalence of NFAs and DFAs

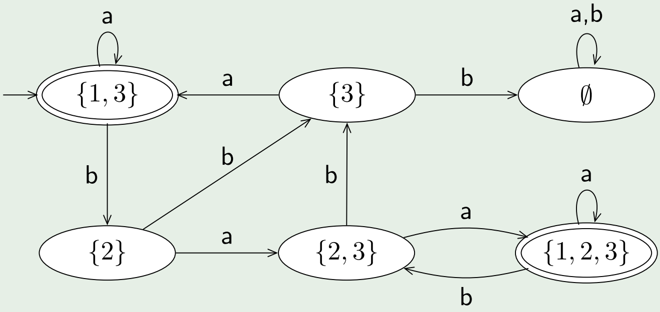
例 (DFA D that is equivalent to the NFA N_4)



Equivalence of NFAs and DFAs

例 (DFA D after removing unnecessary states)

- No arrows point at states $\{1\}$ and $\{1, 2\}$
- They may be removed without affecting the performance of DFA.



Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
2. q_0 is the start state of N

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
2. q_0 is the start state of N
3. $F = F_1 \cup F_2$

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
2. q_0 is the start state of N
3. $F = F_1 \cup F_2$
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the union operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$.

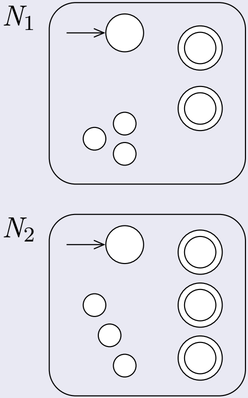
1. $Q = \{q_0\} \cup Q_1 \cup Q_2$
2. q_0 is the start state of N
3. $F = F_1 \cup F_2$
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 \\ \delta_2(q, a) & q \in Q_2 \\ \{q_1, q_2\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

□

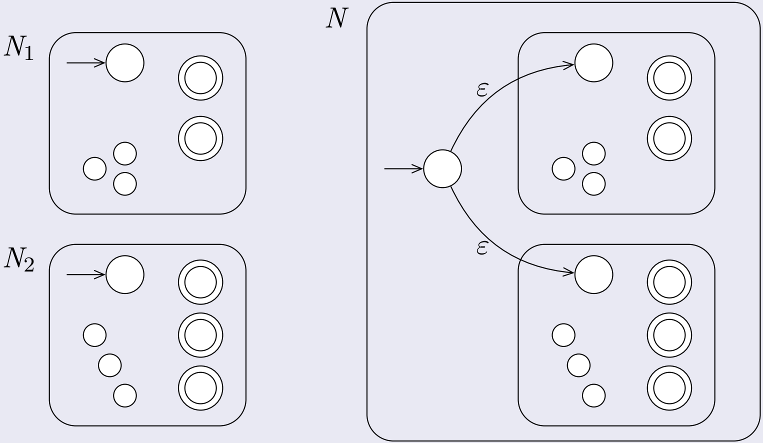
Closure Under the Regular Operations

Construction of an NFA N to recognize $A_1 \cup A_2$



Closure Under the Regular Operations

Construction of an NFA N to recognize $A_1 \cup A_2$



Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and

$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$
2. q_1 is the same as the start state of N_1

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$
2. q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and $N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.

Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

1. $Q = Q_1 \cup Q_2$
2. q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the concatenation operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1, and
$N_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognize A_2.
Construct $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$.

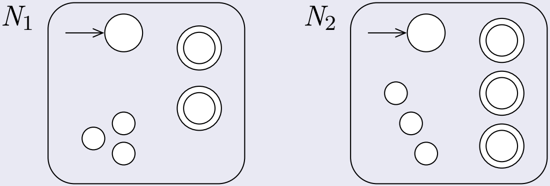
1. $Q = Q_1 \cup Q_2$
2. q_1 is the same as the start state of N_1
3. The accept states F_2 are the same as the accept states of N_2
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_2\} & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q,a) & q \in Q_2 \end{cases}$$

□

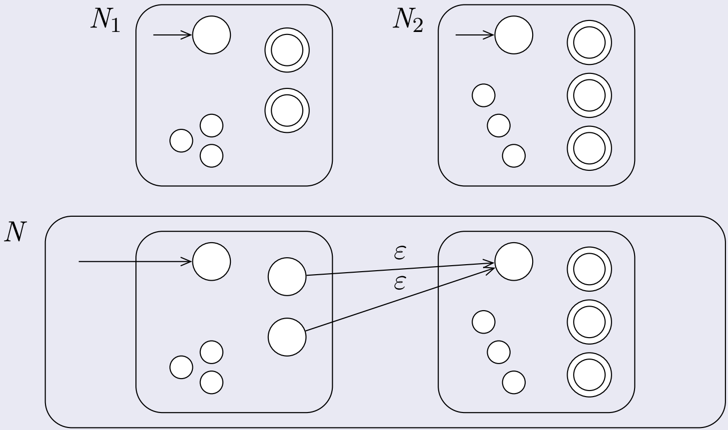
Closure Under the Regular Operations

Construction of N to recognize $A_1 \circ A_2$



Closure Under the Regular Operations

Construction of N to recognize $A_1 \circ A_2$



Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$
2. q_0 is the new start state.

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$
2. q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$

□

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$
2. q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

$\square$

Closure Under the Regular Operations

定理

The class of regular languages is closed under the star operation.

Proof.

Let $N_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A_1.

Construct $N = (Q, \Sigma, \delta, q_0, F)$ to recognize A_1^*.

1. $Q = \{q_0\} \cup Q_1$
2. q_0 is the new start state.
3. $F = \{q_0\} \cup F_1$
4. For any $q \in Q$ and any $a \in \Sigma_\varepsilon$

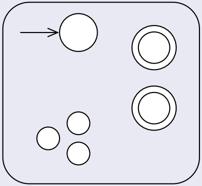
$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q,a) & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\} & q = q_0 \text{ and } a = \varepsilon \\ \emptyset & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

□

Closure Under the Regular Operations

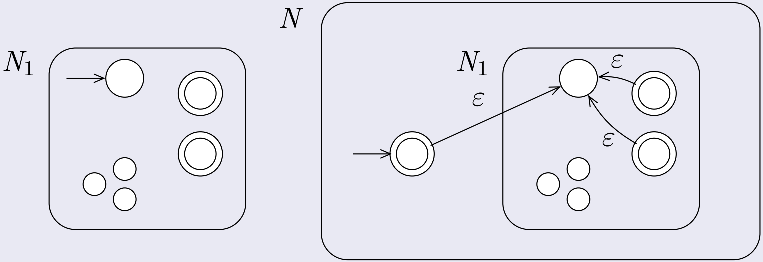
Construction of N to recognize A_1^*

N_1



Closure Under the Regular Operations

Construction of N to recognize A_1^*



Conclusion

Conclusion

- DFA
  - Formal Definitions of a DFA
  - Computation of a DFA
  - From DFAs to languages
  - From languages to DFAs
  - The Regular Operations

Conclusion

- DFA
  - Formal Definitions of a DFA
  - Computation of a DFA
  - From DFAs to languages
  - From languages to DFAs
  - The Regular Operations
- NFA
  - Formal Definitions of an NFA
  - Equivalence of NFAs and DFAs
  - Closure Under the Regular Operations