INT201 Decision, Computation and Language

Lecture 2 – Deterministic Finite Automata

Dr Yushi Li

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

Controlling a toll gate

When a car arrives at the toll gate, the gate is closed. The gate opens as soon as the driver has payed 25 cents. We assume that we have only three coin denominations: 5, 10, and 25 cents. Notably, no excess change is returned.

After having arrived at the toll gate, the driver inserts a sequence of coins into the machine. At any moment, the computer has to decide whether or not to open the gate, i.e., whether or not the driver has paid 25 cents (or more).

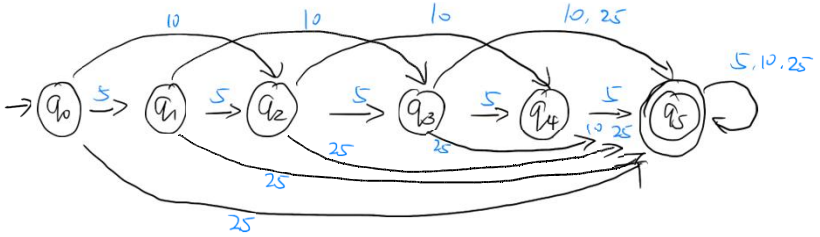
Controlling a toll gate

- The machine is in state q_0, if it has not collected any money yet.
- The machine is in state q_1, if it has collected exactly 5 cents.
- The machine is in state q_2, if it has collected exactly 10 cents.
- The machine is in state q_3, if it has collected exactly 15 cents.
- The machine is in state q_4, if it has collected exactly 20 cents.
- The machine is in state q_5, if it has collected 25 cents or more.

Controlling a toll gate

Question.

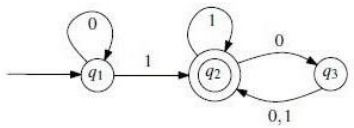
Can we draw a figure to represent the behavior of the computer for all possible sequences of coins?



Deterministic Finite Automata (DFA)

Example

We have a machine state diagram as below:



q_1 is the start state and q_2 is an accept state.

Question.

Can the input string 0101010 be accepted by this machine? No 最终不是位于 q_2

Which kind of strings can be accepted by this machine?

Deterministic Finite Automata (DFA)

Example

满足最终位置位于 q_2 的 string.

(1) Every binary string that ends with 1.

(2) Even number of 0s (zero) following the rightmost one

(3)

Deterministic Finite Automata (DFA)

DFA is a finite-state machine that accepts or rejects a given string of symbols, by running through a state sequence uniquely determined by the string.

It is invented to **recognize a special class of formal language** and have many practical applications:

- Lexical analysis – scans the input program from beginning to end and divides it into tokens like identifiers, constants, and keywords and to remove comments and white space (specify tokens of programming languages).
- Model checking, reasoning about systems with objective of proving they satisfy useful properties.
- Statistical models for analyzing biological and textual sequences.

Deterministic Finite Automata (DFA)

We shall see that DFA can be used to describe

- Any finite set of strings
- Various infinite sets of strings, e.g.
- ✓ strings having exactly 2 occurrences of the letter a
- ✓ strings having more than 6 letters
- ✓ strings in which letter b never comes before letter a

Deterministic Finite Automata (DFA)

DFA cannot be used to describe certain languages such as:

- the set of strings containing more a's than b's
- all words that remain the same if you read them back to front
- well-formed arithmetic expressions, if there is no limit on nesting of parentheses.

Deterministic Finite Automata (DFA)

A DFA is defined as a 5-tuple:

$$M = (Q, \Sigma, \delta, q, F)$$

Where

1. Q is a finite set of **states**,
2. Σ is a finite set of symbols, called the **alphabet** of the automaton,
3. $\delta : Q \times \Sigma \rightarrow Q$ is a function, called the **transition function**,
4. $q \in Q$ is called the **initial state**,
5. $F \subseteq Q$ is a set of **accepting/terminal states**.

F必然位于Q的值

Deterministic Finite Automata (DFA)

State of a machine tells you something about the prefix that has been read so far. If the string is a member of the language of interest, the state reached when the whole string has been scanned will be an accepting state (a member of F).

Transition function δ tells you how state should change when an additional letter is read by the DFA.

Deterministic Finite Automata (DFA)

Example of transition function

Initially the state is i and if the input word is $w = a_1 a_2 \ldots a_n$ then, as each letter is read, the state changes and we get $q_1, q_2 \ldots q_n$ defined by:

$$
\begin{aligned}
q_1 &= \delta(i, a_1) \\
q_2 &= \delta(q_1, a_2) \\
q_3 &= \delta(q_2, a_3) \\
&\vdots \\
q_n &= \delta(q_{n-1}, a_n)
\end{aligned}
$$

Deterministic Finite Automata (DFA)

We can extend the definition of the transition function δ so that it tells us which state we reach after a word (not just a single letter) has been scanned:

In the above notation, extend the map $\delta : Q \times \Sigma \to Q$ to $\delta^* : Q \times \Sigma^* \to Q$ by defining:

$$\delta^*(q, \epsilon) = q \quad \text{for all } q \in Q$$

$$\delta^*(q, wa) = \delta(\delta^*(q, w), a) \quad \text{for all } q \in Q;\ w \in \Sigma^*;\ a \in \Sigma$$

Deterministic Finite Automata (DFA)

Example

$\delta(q_0, a) = q_1$ *and* $\delta(q_1, b) = q_2$

Then

$\delta^*(q_0, ab) = q_2$

Proof Steps:

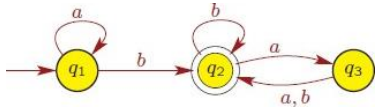
$\delta^*(q_0, ab) = \delta(\delta^*(q_0, a), b)$

$\delta^*(q_0, a) = \delta(q_0, a) = q_1$

$\delta^*(q_0, ab) = \delta(q_1, b) = q_2$

Deterministic Finite Automata (DFA)

A DFA $M = (Q, \Sigma, \delta, q, F)$ is often depicted as a directed graph G_M (called **transition graph**) has exactly $|Q|$ vertices, each one labeled with a different $q_i \in Q$. For each transition function $\delta(q_i, a) = q_j$, the graph has edges (q_i, q_j) labeled (a), (b), and (a, b), . The vertex associated with q_1 is called the initial vertex, while those labeled with q_f are the final vertices.



Deterministic Finite Automata (DFA)

Let $M = (Q, \Sigma, \delta, q, F)$ be a finite automaton and let $w = w_0 w_1 \ldots w_n$ be a string over . Define the sequence $q_0, q_1, \ldots, q_n$ of states, in the following way:

- $q_0 = q$,
- $q_{i+1} = \delta(q_i, w_{i+1})$, for $i = 0, 1, \ldots, n - 1$.

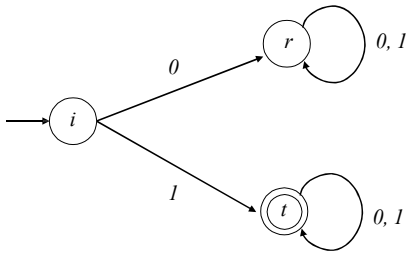
1. If $q_n \in F$, then we say that M accepts w.
2. If $q_n \notin F$, then we say that M rejects w.

F Accepting state

如果根据w中每一个symbol走完整个dfa的关系，并且最终生成了 accept state 则代表 M accept w，否则 M rejects w.

Deterministic Finite Automata (DFA)

Symbolic description of the example DFA



Deterministic Finite Automata (DFA)

Symbolic description of the example DFA

Automaton $M = (Q, \Sigma, \delta, q, F)$

Set of states $Q = \{i, t, r\}$, $\Sigma = \{0, 1\}$, $F = \{t\}$ and the transition function δ is given by

$$\delta(i, 0) = r \qquad \delta(i, 1) = t$$

$$\delta(t, 0) = t \qquad \delta(i, 1) = t$$

$$\delta(r, 0) = r \qquad \delta(r, 1) = r$$

It is simpler to describe a transition function by a table of values. In this example we have:

	0	1
i	*r*	*t*
t	*t*	*t*
r	*r*	*r*

Deterministic Finite Automata (DFA)

Symbolic description of the example DFA

If δ is a partial function (not defined for some state/letter pairs), then the DFA rejects an input if it ever encounters such a pair.

This convention often simplifies the definition of a DFA. In the previous example we could use transition table

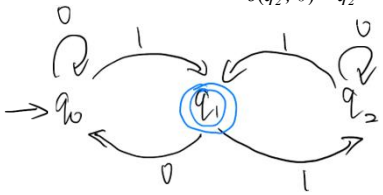
	0	1
i		*t*
t	*t*	*t*

Deterministic Finite Automata (DFA)

Exercise

DFA $M = (\{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_1\})$, where δ is given as:

$$\delta(q_0, 0) = q_0 \qquad \delta(q_0, 1) = q_1$$
$$\delta(q_1, 0) = q_0 \qquad \delta(q_1, 1) = q_2$$
$$\delta(q_2, 0) = q_2 \qquad \delta(q_2, 1) = q_1$$



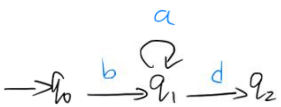
Can the input word 110100 be accepted/recognized by this M ? No

Deterministic Finite Automata (DFA)

Exercise

How about the following infinite language. Can you give a DFA that accepts the words:

bad, baad, baaad, baaaad, ...?



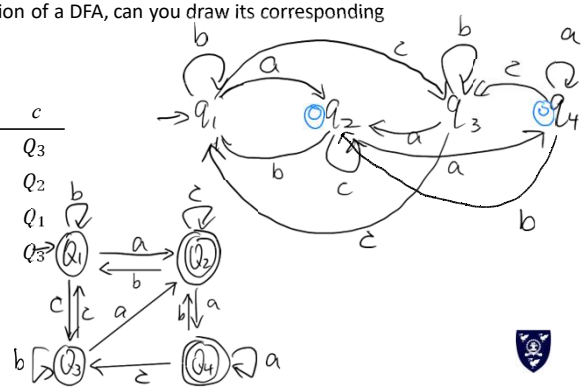
Deterministic Finite Automata (DFA)

Exercise

F is $\{Q_2, Q_4\}$

Given the symbolic description of a DFA, can you draw its corresponding diagram?

δ	a	b	c
Q_1	Q_2	Q_1	Q_3
Q_2	Q_4	Q_1	Q_2
Q_3	Q_2	Q_3	Q_1
Q_4	Q_4	Q_2	Q_3



Deterministic Finite Automata (DFA)

DFA is a finite machine that can accept some specific languages.

Language defined by DFA

Suppose we have a DFA M. A word $w \in \Sigma^*$ is said to be accepted or recognized by M if $\delta^*(q_0, w) \in F$, otherwise it is said to be rejected. The set of all words accepted by M is called the language accepted by M and will be denoted by $L(M)$. Thus

$$\boldsymbol{L}(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$

- Any finite language is accepted by some DFA
- A language $\mathbf{A}$ is called **regular** if there exists a finite automaton M such that $\mathbf{A} = \boldsymbol{L}(M)$

如果想验证A是regular的，设计一个DFA accept A，就能证明A is regular.

任何有限语言都被某些DFA接受

Deterministic Finite Automata (DFA)

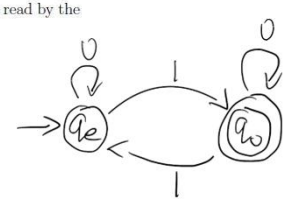
Example

whether the following DFA can accept A

$A = \{w : w \text{ is a binary string containing an odd number of 1s}\}$

- The set of states is $Q = \{q_e, q_o\}$. If the finite automaton is in state q_e, then it has read an even number of 1s; if it is in state q_o, then it has read an odd number of 1s.
- The alphabet is $\Sigma = \{0, 1\}$.
- The start state is q_e, because at the start, the number of 1s read by the automaton is equal to 0, and 0 is even.
- The set F of accept states is $F = \{q_o\}$.
- The transition function δ is given by the following table:

	0	1
q_e	q_e	q_o
q_o	q_o	q_e



Deterministic Finite Automata (DFA)

Exercise

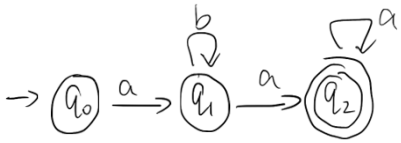
$w = \{a, b, ab, ba\}$

Show that the language $\boldsymbol{L}$ is regular:

$awa = \{aaa, aba, aaba, abaa\}$

$$\boldsymbol{L} = \{awa: w \in \{a,b\}^* \}$$

Can you design a DFA that accepts this language?



Deterministic Finite Automata (DFA)

Regular operations on languages

Let $\mathbf{A}$ and $\mathbf{B}$ be two languages over the same alphabet.

The **union** of $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\mathbf{A} \cup \mathbf{B} = \{w : w \in \mathbf{A} \text{ or } w \in \mathbf{B}\}$$

The concatenation of $\mathbf{A}$ and $\mathbf{B}$ is defined as:

$$\mathbf{AB} = \{ww' : w \in \mathbf{A} \text{ and } w' \in \mathbf{B}\}$$

The star of $\mathbf{A}$ is defined as:

$$\mathbf{A}^* = \{u_1 u_2 \ldots u_k : k \geq 0 \text{ and } u_i \in \mathbf{A} \text{ for all } i = 1, 2, \ldots, k\}$$

Deterministic Finite Automata (DFA)

Example of $\mathbf{A}^*$

Given two languages $\mathbf{A} = \{0, 01\}$ and $\mathbf{B} = \{1, 10\}$. Then

$$\mathbf{A} \cup \mathbf{B} = \{0, 01, 1, 10\}$$

$$\mathbf{AB} = \{01, 010, 011, 0110\}$$

$$\mathbf{A}^* = \{\epsilon, 0, 01, 00, 001, 010, 0101, 000, 0001, 00101, \ldots\}$$

Deterministic Finite Automata (DFA)

Theorem

The set of regular languages is closed under the union operation, i.e., if $\mathbf{A}$ and $\mathbf{B}$ are regular languages over the same alphabet $\mathbf{\Sigma}$, then $\mathbf{A} \cup \mathbf{B}$ is also a regular language.