Finite Automata 02

Thumrongsak Kosiyatrakul
tkosiyat@cs.pitt.edu

Finite Automata

- The computational model called finite automata can be used to simulate a set of **simple** algorithms
  - Check whether a string starts with 010
  - Check whether a string ends with 111
  - Check whether a string contains 0101 as a substring
  - Check whether a string contains substrings 000 and 111 where 000 comes before 111
- It is a powerful tool in compiler
  - Accept or reject your source code based on a programming syntax
  - Example: the `for` statement:
    - starts with `for`
    - followed by (
    - followed by assignment statement(s)
    - followed by ;
    - followed by conditional statement(s)
    - followed by ;
    - followed by assignment statement(s)
    - followed by)

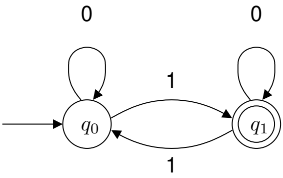
Formal Definition of Computation

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and let $w = w_1 w_2 \ldots w_n$ be a string where each w_i is a member of the alphabet Σ.
  - 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. $\delta(r_i, w_{i+1}) = r_{i+1}$, where $i = 0, \ldots, n-1$
    3. $r_n \in F$
- Think in terms of processing the input string w

$$r_0 \stackrel{w_1}{\rightarrow} r_1 \stackrel{w_2}{\rightarrow} r_2 \stackrel{w_3}{\rightarrow} r_3 \stackrel{w_4}{\rightarrow} r_4 \stackrel{w_5}{\rightarrow} \ldots \stackrel{w_{n-1}}{\rightarrow} r_{n-1} \stackrel{w_n}{\rightarrow} r_n$$

Regular Languages

- A language L over an alphabet Σ is said to be a **regular language** if some finite-state automaton recognizes it.
- Consider the following machine M:



  - What is the language of this machine?
    - $L(M) = \{w \mid w$ contains an odd number of 1s$\}$
  - "The set of all strings consisting of an odd number of 1s" is a **regular language**
  - $L(M)$ is a regular language

Regular Languages

- Why regular language is important in our discussion?
  - **Definition**: A language is regular if some finite-state machines recognize it.
  - If we can prove that a language is **regular**
    - We must be able to construct a finite-state machine to recognize it
    - It maybe hard to build but I know that it exists
  - If we can prove that a language is **not regular**
    - We cannot construct a finite-state machine to recognize it
- This is an example of a limitation of this computational model

Problem and Language

- In theory of computation, a problem is represented as a language
  - A problem of determining whether a string contains 011 as a substring

$$L(M) = \{x \mid x \text{ contains } 011 \text{ as a substring}\}$$

  - We already see a Deterministic Finite Automaton (DFA) M that accepts all strings that contains 011 as a substring and reject those that does not contain 011 as a substring
  - It means this problem is **solvable** by the algorithm captured by the previous DFA
  - In case of algorithm in a form of DFA (not all algorithms)
    - if $L(M)$ is regular, the problem represented by $L(M)$ is solvable
    - if $L(M)$ is not regular, no DFA exists, the problem represented by $L(M)$ is unsolvable

Problems and Languages

- Solvable problems that we see so far
  - The problem of determining whether a string ends with a 1

    $$\{x \mid x \text{ ends with a } 1\}$$

  - The problem of determining whether a string is an empty string or ends in a 0

    $$\{x \mid x \text{ is an empty string or ends in a } 0\}$$

  - The problem of determining whether a string starts and ends with the same symbol

    $$\{x \mid x \text{ starts and ends with the same symbol}\}$$

  - The problem of determining whether a string contains either 11 or 00 as a substring

    $$\{x \mid x \text{ contains either } 11 \text{ or } 00 \text{ as a substring}\}$$

Problems and Languages

- Solvable problems that we see so far (continue)
  - The problem of determining whether a string contains 011 as a substring
$$\{x \mid x \text{ contains } 011 \text{ as a substring}\}$$
  - The problem of determining whether a string ends with 0110
$$\{x \mid x \text{ ends with } 0110\}$$
  - The problem of determining whether a string contains an odd number of 1s
$$\{x \mid x \text{ contains an odd number of } 1\text{s}\}$$
- Each of the above languages is regular since we can construct a DFA that recognizes it.
  - But if a language is very complicate, it will be difficult to construct a DFA that recognizes it
- We need tools to help us to determine whether a language is regular or not

The Regular Operations

- In arithmetic:
  - Objects are numbers
  - Tools are operations for manipulating numbers (e.g., $+$ and $\times$)
    - $1 + 1$ gives you a new number which is 2
- In the theory of computation,
  - Objects are languages (sets of strings)
  - Tools are operations for manipulating languages

Definition 1.23

Let A and B be languages. We define the regular operations 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**: $A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$

Examples (Union)

- Let $\Sigma = \{0, 1\}$
- Consider the following languages A and B
  - $A = \{00,\ 11\}$
  - $B = \{010,\ 101\}$
- The union operations is identical to the set's union operation:

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}$$

- From the above definition:

$$A \cup B = \{00, 11, 010, 101\}$$

Examples (Concatenation)

- Let $\Sigma = \{0, 1\}$
- Consider the following languages A and B
  - $A = \{$00, 11$\}$
  - $B = \{$010, 101$\}$
- The definition of concatenation is defined as

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

- From the above definition:

$$A \circ B = \{00010, 00101, 11010, 11101\}$$

- For simplicity, sometimes we write AB instead of $A \circ B$

Examples (Star)

- Let $\Sigma = \{0, 1\}$
- Consider the following language A
  - $A = \{00,\ 11\}$
- The definition of start is defined as

$$A^* = \{x_1 x_2 \dots x_k \mid k \geq 0 \text{ and } x_i \in A\}$$

  - If $k = 0$, the above definition becomes

$$\{\ \mid 0 \geq 0\} = \{\varepsilon\}$$

  - If $k = 1$, the above definition becomes

$$\{x_1 \mid 1 \geq 0 \text{ and } x_i \in A\} = \{00, 11\}$$

  - If $k = 2$, the above definition becomes

$$\{x_1 x_2 \mid 2 \geq 0 \text{ and } x_i \in A\} = \{0000, 0011, 1100, 1111\}$$

  - If $k = 3$, the above definition becomes

$$\{x_1 x_2 x_3 \mid 3 \geq 0 \text{ and } x_i \in A\} = \{000000, 000011, \dots, 111111\}$$

Examples (Star)

- Let $\Sigma = \{0, 1\}$
- Suppose $A = \{00, 11\}$, what is A^*?

$$A^* = \{\varepsilon, 00, 11, 0000, 0011, 1100, 1111, 000000, \dots\}$$

- Suppose $A = \{011\}$, what is A^*?

$$A^* = \{\varepsilon, 011, 011011, 011011011, 011011011011, \dots\}$$

- Suppose $A = \{0, 1\}$, what is A^*?

$$A^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, \dots\}$$

  This is the set of all strings over $\{0, 1\}$
- Suppose $A = \emptyset$, what is A^*?

$$A^* = \{\varepsilon\}$$

Definition of Closed Under Operations

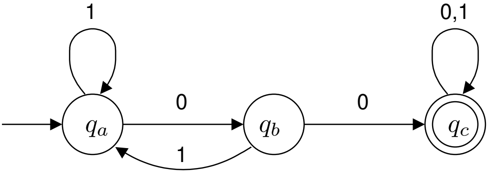
- Let A be a set of objects (a collection of object)
- We say that A is closed under operation $\triangle$ if for any $x \in A$ and $y \in A$, $x \triangle y$ is also in A.
- Example: Let $\mathbb{N}$ be the set of natural number
  - $\mathbb{N}$ is closed under addition
    - For any two natural numbers x and y, $x + y$ is a natural number
  - $\mathbb{N}$ is closed under multiplication
    - For any two natural numbers x and y, $x \times y$ is a natural number
  - $\mathbb{N}$ is **not** closed under subtraction
    - $5 - 7$ is not a natural number

Definition of Closed Under Operations

- Let $\mathbb{L}$ be the set of all **regular languages**
  - This is a set of sets
- Recall that we have three operations, union, concatenation, and star
- Is $\mathbb{L}$ closed under **union** operation?
  - For any regular languages A and B, is $A \cup B$ a regular language?
- Is $\mathbb{L}$ closed under **concatenation** operation?
  - For any regular languages A and B, is $A \circ B$ a regular language?
- Is $\mathbb{L}$ closed under **star** operation?
  - For any regular language A, is A^* a regular language?

$\mathbb{L}$ is regular under union operation

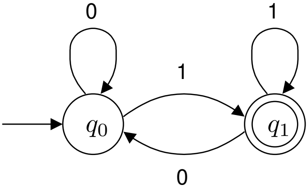
- Let A be a set of strings over $\{0,1\}$ that contain a 00 as a substring
- Is A a regular language?
- Can you construct a DFA that recognizes the language A?
- One of the machine that recognizes A can be as follows:



- Because there exists a DFA that recognizes A, A is a regular language

$\mathbb{L}$ is regular under union operation

- Let B be a set of strings over $\{0, 1\}$ that end with a 1
- Is B a regular language?
- Can you construct a DFA that recognizes the language B?
- One of the machine that recognizes B can be as follows:



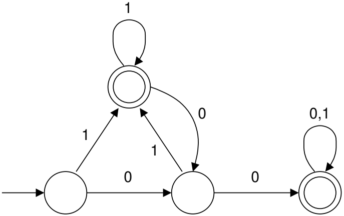
- Because there exists a DFA that recognizes B, B is a regular language

$\mathbb{L}$ is regular under union operation

- We have $A = \{x \mid x \text{ contains } 00 \text{ as a substring}\}$ is regular
- We have $B = \{x \mid x \text{ ends with a } 1\}$ is regular
- How about $A \cup B$?

  $$A \cup B = \{x \mid x \text{ contains } 00 \text{ as a substring or } x \text{ ends with a } 1\}$$

- It is quite straightforward to construct a machine that recognizes $A \cup B$ (try to build one yourself)



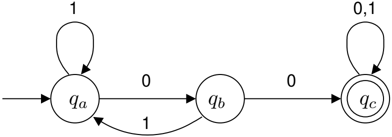
- This does not prove that if A and B are regular, $A \cup B$ is regular
  - This is just one example out of infinite may instances of regular languages

$\mathbb{L}$ is regular under union operation

- We need to show that for any two regular languages A and B, $A \cup B$ is regular
- Given a regular language A over a Σ, what do we know about the language A?
  - There exists a DFA M_A that recognizes A ($L(M_A) = A$)
    - $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$ for some Q_A, δ_A, q_A, and F_A
- Similarly, given a regular language B over a Σ:
  - There exists a DFA M_B that recognizes B ($L(M_B) = B$)
    - $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$ for some Q_B, δ_B, q_B, and F_B
- To show that $A \cup B$ is regular for any regular languages A and B, we need to construct a DFA that recognizes $A \cup B$ from M_A and M_B
  - To understand the process, we are going to work on a specific example

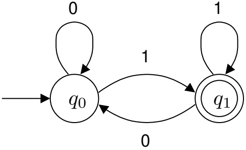
$\mathbb{L}$ is regular under union operation

- Recall the previous two regular languages and its DFAs where $\Sigma = \{0, 1\}$
  - $A = \{x \mid x$ contains 00 as a substring$\}$



$$M_A = (Q_A, \Sigma, \delta_A, q_A, F_A) \text{ and } L(M_A) = A$$

  - $B = \{x \mid x$ ends with a $1\}$



$$M_B = (Q_B, \Sigma, \delta_B, q_B, F_B) \text{ and } L(M_B) = B$$

- Given a string w and these two DFAs, how to check whether w is in $A \cup B$?

$\mathbb{L}$ is regular under union operation

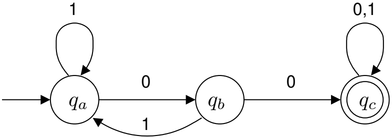
- Recall that $A = L(M_A)$ and $B = L(M_B)$
  - Thus, $A \cup B = L(M_A) \cup L(M_B)$
- $w \in A \cup B$

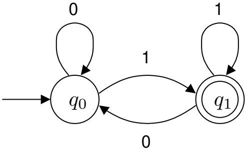
  iff $w \in A$ or $w \in B$
  iff $w \in L(M_A)$ or $w \in L(M_B)$
  iff M_A accepts w or $w \in L(M_B)$
  iff M_A accepts w or M_B accepts w
- In other words,

$$w \in A \cup B \text{ if and only if } M_A \text{ accepts } w \text{ or } M_B \text{ accepts } w$$

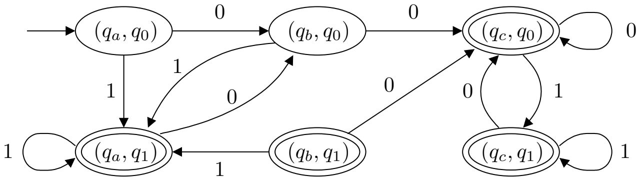
- To check whether $w \in A \cup B$:
  - Run both M_A and M_B on input w
  - If one of them or both accepts w, $w \in A \cup B$
  - If both reject w, $w \notin A \cup B$

- We can run both machines simultaneously





- Let state (p, q) represents the situation where
  - The current state of M_A is p
  - The current state of M_B is q
- With the new notion of states, we have



$\mathbb{L}$ is regular under union operation

- Let M_A recognizes A, where $M_A = (Q_A, \Sigma, \delta_A, q_A, F_A)$
- Let M_B recognizes B, where $M_B = (Q_B, \Sigma, \delta_B, q_B, F_B)$
- Machine $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $A \cup B$ can be constructed as follows:
    1. $Q = \{(r_1, r_2) \mid r_1 \in Q_A$ and $r_2 \in Q_B\}$
    2. For each $(r_1, r_2) \in Q$ and $a \in \Sigma$

       $$\delta((r_1, r_2), a) = (\delta_A(r_1, a), \delta_B(r_2, a))$$

    3. $q_0 = (q_A, q_B)$
    4. $F = \{(r_1, r_2) \mid r_1 \in F_A$ or $r_2 \in F_B\}$
- To recognize $A \cap B$, simply change the set of accept states to

  $$F = \{(r_1, r_2) \mid r_1 \in F_A \text{ and } r_2 \in F_B\}$$

- If A and B are regular languages, $A \cup B$ is regular

Conclusions

- A language is regular if it is recognized by some finite-state machines
  - If you can prove that a language is regular:
    - there exists a finite-state machine that recognizes it
  - If you can prove that a language is **not** regular:
    - there is no finite-state machine that recognizes it
- In formally, we show that if A and B are regular languages, $A \cup B$ is a regular language
- To prove the closure of concatenation and star operators, we need a sightly different computational model