Finite Automata and Regular Languages

Huan Long

Shanghai Jiao Tong University

Acknowledgements

Part of the slides comes from a similar course given by Prof. Yijia Chen.

`http://basics.sjtu.edu.cn/~chen/`

Textbook

Introduction to the theory of computation
Michael Sipser, MIT
Third edition, 2012

Outline

Finite automata and regular language

Nondeterminism automata

Equivalence of DFA and NFA

Regular expression

Pumping lemma for regular languages

Some decision problems related to FA

Finite Automata

Definition

A deterministic finite automata (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,
5. $F \subseteq Q$ is the set of accept states.

Computation by DFA

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let $w = w_1 w_2 \cdots w_n$ be a string with $w_i \in \Sigma$ for all $i \in [n]$. Then M accepts w if there exists a sequence of states $r_0, r_1, \ldots, r_n$ in Q such that:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for $i = 0, \ldots, n-1$,
3. $r_n \in F$.

For a set A, we say that M recognizes A if

$$A = \{\ell \mid M \text{ accepts } \ell\}$$

Regular languages

Definition
A language is called regular if some finite automata recognizes it.

The regular operators

Definition

Let A and B be languages. We define the following three regular operations:

- Union: $A \cup B = \{x \mid x \in A \ \vee \ x \in B\}$
- Concatenation: $A \circ B = \{xy \mid x \in A \ \wedge \ y \in B\}$
- Kleene star: $A^\star = \{x_1 x_2 \ldots x_k \mid k \geq 0 \ \wedge \ x_i \in A\}$

Nondeterminism

Definition

A nondeterministic finite automata (NFA) 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. $\delta : Q \times \Sigma_\epsilon \to \mathcal{P}(Q)$ is the transition function, where $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$,
4. $q_0 \in Q$ is the start state,
5. $F \subseteq Q$ is the set of accept states.

Computation by NFA

Let $N = (Q, \Sigma, \delta, q_0, F)$ be a NFA and let $w = y_1 y_2 \cdots y_m$ be a string with $y_i \in \Sigma_\epsilon$ for all $i \in [m]$. Then N accepts w if there exists a sequence of states $r_0, r_1, \ldots, r_m$ in Q such that:

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for $i = 0, \ldots, m-1$,
3. $r_m \in F$.

Equivalence of NFAs and DFAs

Theorem

Every NFA has an equivalent DFA, i.e., they recognize the same language.

Proof (1)

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

Main idea: view a NFA as occupying a set of states at any moment.

Step 1: For any state $q \in Q$, compute its *silently reachable class* $E(q)$:

$$
\begin{array}{l}
\textbf{initially set } E(q) = \{q\}; \\
\quad \textbf{repeat} \\
\qquad E'(q) = E(q) \\
\qquad \forall x \in E(q), \textbf{if } \exists y \in \delta(x, \epsilon) \wedge y \notin E(q), E(q) = E(q) \cup \{y\} \\
\quad \textbf{until } E(q) = E'(q) \\
\textbf{return } E(q).
\end{array}
$$

Proof (2)

Step 2: build the equivalent DFA

NFA: $N = (Q, \Sigma, \delta, q_0, F) \Rightarrow$ DFA: $M = (Q', \Sigma, \delta', q_0', F')$

1. $Q' = \mathcal{P}(Q)$;
2. Let $R \in Q'$ and $a \in \Sigma$, define

$$\delta'(R, a) = \bigcup \{E(q) \mid q \in Q \wedge (\exists r \in R)(q \in \delta(r, a))\} ;$$

3. $q_0' = E(q_0)$;
4. $F' = \{R \in Q' \mid R \cap F \neq \emptyset\}$.

Corollary

A language is regular if and only if some NFA recognizes it.

Recall: regular operators

Definition

Let A and B be languages. We define the following three regular operations:

- Union: $A \cup B = \{x \mid x \in A \ \vee \ x \in B\}$
- Concatenation: $A \circ B = \{xy \mid x \in A \ \wedge \ y \in B\}$
- Kleene star: $A^{\star} = \{x_1 x_2 \ldots x_k \mid k \geq 0 \ \wedge \ x_i \in A\}$

Closure under the regular operators

Theorem

The class of regular languages is closed under the $\cup$, $\circ$, * operations.

Proof.

Let $N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ recognize A_1,
$N_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ recognize A_2;

We will build NFAs which recognize $A_1 \cup A_2$, $A_1 \circ A_2$, $A_1^\star$ respectively. $\square$

I. Closure under *union* : $A_1 \cup A_2$ is regular

$N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ recognize A_1,
$N_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ recognize A_2;

Define the NFA as:

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

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

II. Closure under *concatenation* : $A_1 \circ A_2$ is regular

$N_1 = (Q_1, \Sigma_1, \delta_1, q_1, F_1)$ recognize A_1,
$N_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$ recognize A_2;

Define the NFA as:

1. $Q = Q_1 \cup Q_2$;
2. the start state is q_1;
3. the set of accept states is F_2;
4. For any $q \in Q$ and any $a \in \Sigma_\epsilon$

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & q \in Q_1 - F_1 \\ \delta_1(q, a) & q \in F_1 \wedge a \neq \epsilon \\ \delta_1(q, a) \cup \{q_2\} & q \in F_1 \wedge a = \epsilon \\ \delta_2(q, a) & q \in Q_2 \end{cases}$$

III. Closure under *Kleene star* : $A_1^\star$ is regular

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

Define the NFA as:

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

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & q \in Q_1 - F_1 \\ \delta_1(q,a) & q \in F_1 \wedge a \neq \epsilon \\ \delta_1(q,a) \cup \{q_1\} & q \in F_1 \wedge a = \epsilon \\ \{q_1\} & q = q_0 \wedge a = \epsilon \\ \emptyset & q = q_0 \wedge a \neq \epsilon \end{cases}$$

Other closure property

Given $N = (Q, \Sigma, \delta, q, F)$ the set of language recognized by N is A, then

- Complement: $\overline{A} = \Sigma^\star - A$
- Intersection: $A \cap B = \{x \mid x \in A \wedge x \in B\}$

Lemma

The class of regular languages is closed under complementation and intersection.

Proof.

- w.l.o.g, N is a DFA, then $\overline{N} = (Q, \Sigma, \delta, q, Q - F)$ will recognize $\overline{A}$.
- $A \cap B = \overline{\overline{A} \cup \overline{B}}$.

$\square$

Regular expression

Given alphabet Σ, we say that R is a regular expression if R is

1. a for some $a \in \Sigma$,
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,
6. $(R_1^\star)$, where R_1 is a regular expression.

Sometimes, we use $R_1 R_2$ instead of $(R_1 \circ R_2)$ if no confusion arises.

Language defined by regular expressions

regular expression R	language $L(R)$
a	$\{a\}$
ϵ	$\{\epsilon\}$
$\emptyset$	$\emptyset$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
$(R_1^\star)$	$L(R_1)^\star$

Equivalence with finite automata

Theorem

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

Proof.

- If: build the NFAs; (relatively easy)
- Only if: Automata $\Longrightarrow$ regular expressions.

  Sketch: (Dynamic programming)

  $$R(i,j,k) = R(i,j,k-1) \cup R(i,k,k-1)R(k,k,k-1)^{\star}R(k,j,k-1)$$

  $$L(M) = \bigcup \{R(1,j,n) \mid q_j \in F\}.$$

$\square$

Languages need counting

- $L_1 = \{\ell \in \{0,1\}^\star \mid \ell$ has an equal number of 0s and 1s$\}$.
- $L_2 = \{\ell \in \{0,1\}^\star \mid \ell$ has an equal number of occurrences of 01 and 10 as substrings$\}$.

- L_2 is regular;
- L_1 is or is not regular? It is not regular!

The pumping lemma for regular languages

Lemma

If A is a regular language, then there is a number p (i.e., the pumping length where if s is any string in A of length at least p, then s my be divided into three pieces, $s = xyz$, satisfying the following conditions:

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

Any string xyz in A can be pumped along y.

Proof

Pigeonhole principle

Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and $p = |Q|$. Let $s = s_1 s_2 \cdots s_n$ be a string in A with $n \geq p$. Let $r_1, \cdots, r_{n+1}$ be the sequence of states that M enters while processing s, i.e.,

$$r_{i+1} = \delta(r_i, s_i)$$

for $i \in [n]$.
Among the first $p+1$ states in the sequence, two must be the same, say r_j and r_k with $j < k \leq p+1$. Define

$$x = \underline{s_1 \cdots s_{j-1}}, \; y = \underline{s_j \cdots s_{k-1}}, \; z = \underline{s_k \cdots s_n}.$$

Example (1)

The language $L = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Proof.

If it is regular, choose p be the pumping length and consider $s = 0^p 1^p$. By the Pumping lemma, $s = xyz$ with $xy^i z \in L$ for all $i \geq 0$.

As $|xy| \leq p$ and $|y| > 0$, $y = 0^i$ for some $i > 0$.

But then $xz = 0^{n-i} 1^n \notin L$. Contradicting the lemma.

$\square$

Example (2)

The language $L = \{w \mid w \text{ has an equal number of 0s and 1s}\}$ is not regular.

Proof.

If it is regular, then $L \cap 0^\star 1^\star$ would also be regular.

However, this latter language is precisely the language in Example (1), which is not regular.

$\square$

Problems from formal language theory

Decision Problems

- Acceptance: does a given string belong to a given language?
- Emptiness: is a given language empty?
- Equality: are given two languages equal?

Language Problems concerning FA

Theorem

The following three problems:

- Acceptance: Given a DFA (NFA) A and a string w, does A accept w?
- Emptiness: Given a DFA (NFA) A is the language $L(A)$ empty?
- Equality: Given two DFA (NFA) A and B is $L(A)$ equal to $L(B)$?

The corresponding decision problems are all decidable.

Proof.

$\square$