Theory of Computation (I)

Yijia Chen

Instructor

Yijia Chen

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

Email: yijia.chen@cs.sjtu.edu.cn

Textbook

Introduction to the Theory of Computation

Michael Sipser, MIT

Third Edition, 2012.

Part One.
Automata and Languages

Regular Languages

Finite automata

Definition

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.

Formal definition of computation

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton 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 a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with:

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

We say that M recognizes A if

$$A = \{ w \mid M \text{ accepts } w \}.$$

Regular languages

Definition

A language is called regular if some finite automaton recognizes it.

The regular operators

Definition

Let A and B be languages. 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: $A^* = \{x_1 x_2 \ldots x_k \mid k \geq 0 \text{ and each } x_i \in A\}$.

Closure under union

Theorem

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 (1)

For $i \in [2]$ let $M_i = \big(Q_i, \Sigma_i, \delta_i, q_i, F_i\big)$ recognize A_i. We can assume without loss of generality $\Sigma_1 = \Sigma_2$:

- Let $a \in \Sigma_2 - \Sigma_1$.
- We add $\delta_1(r, a) = r_{\text{trap}}$, where r_{trap} is a new state with

$$\delta_1(r_{\text{trap}}, w) = r_{\text{trap}}$$

  for every w.

Proof (2)

We construct $M = \big(Q, \Sigma, \delta, q_0, F\big)$ to recognize $A_1 \cup A_2$:

1. $Q = Q_1 \times Q_2 = \big\{(r_1, r_2) \;\big|\; r_1 \in Q_1 \text{ and } r_2 \in Q_2\big\}$.
2. $\Sigma = \Sigma_1 = \Sigma_2$.
3. For each $(r_1, r_2) \in Q$ and $a \in \Sigma$ we let
$$\delta\big((r_1, r_2), a\big) = \big(\delta_1(r_1, a), \delta_2(r_2, a)\big).$$
4. $q_0 = (q_1, q_2)$.
5. $F = (F_1 \times Q_2) \cup (Q_1 \times F_2) = \big\{(r_1, r_2) \;\big|\; r_1 \in F_1 \text{ or } r_2 \in F_2\big\}$.

Closure under concatenation

Theorem

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 $A_1 \circ A_2$.

We prove the above theorem by nondeterministic finite automata.

Nondeterminism

Definition

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, where $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$,
4. $q_0 \in Q$ is the start state, and
5. $F \subseteq Q$ is the set of accept states.

Formal definition of computation

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

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

Let $N = \big(Q, \Sigma, \delta, q_0, F\big)$ be the NFA recognizing some language A. We construct a DFA $M = \big(Q', \Sigma, \delta', q_0', F'\big)$ recognizing the same A.

First assume N has no ε arrows.

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

$$\delta'(R, a) = \big\{q \in Q \;\big|\; q \in \delta(r, a) \text{ for some } r \in R\big\}.$$

3. $q_0' = \{q_0\}$.
4. $F' = \big\{R \in Q' \;\big|\; R \cap F \neq \emptyset\big\}$.

Proof (2)

Now we allow ε arrows.

For every $R \in Q'$, i.e., $R \subseteq Q$, let

$$E(R) = \big\{ q \in Q \;\big|\; q \text{ can be reached from } R \text{ by traveling along 0 and more } \varepsilon \text{ arrows} \big\}.$$

1. $Q' = \mathcal{P}(Q)$.
2. Let $R \in Q'$ and $a \in \Sigma$. Then we define
$$\delta'(R, a) = \big\{ q \in Q \;\big|\; q \in E\big(\delta(r, a)\big) \text{ for some } r \in R \big\}.$$
3. $q_0' = E\big(\{q_0\}\big)$.
4. $F' = \big\{ R \in Q' \;\big|\; R \cap F \neq \emptyset \big\}$.

Corollary

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

Second proof of the closure under union

For $i \in [2]$ let $N_i = \big(Q_i, \Sigma_i, \delta_i, q_i, F_i\big)$ recognize A_i. We construct an $N = \big(Q, \Sigma, \delta, q_0, F\big)$ to recognize $A_1 \cup A_2$:

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

Theorem

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

Proof

For $i \in [2]$ let $N_i = (Q_i, \Sigma_i, \delta_i, q_i, F_i)$ recognize A_i. We construct an $N = (Q, \Sigma, \delta, q_1, F_2)$ to recognize $A_1 \circ A_2$:

1. $Q = Q_1 \cup Q_2$.
2. The start state 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 $a \in \Sigma_\varepsilon$

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

Closure under star

Theorem

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

Proof

Let $N_1 = \big(Q_1, \Sigma, \delta_1, q_1, F_1\big)$ recognize A_i. We construct an $N = \big(Q, \Sigma, \delta, q_0, F\big)$ to recognize A_1^*:

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

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

Regular expression

Definition

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^*), where R_1 is a regular expressions.

We often write $R_1 R_2$ 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	$\{\varepsilon\}$
$\emptyset$	$\emptyset$
$(R_1 \cup R_2)$	$L(R_1) \cup L(R_2)$
$(R_1 \circ R_2)$	$L(R_1) \circ L(R_2)$
(R_1^*)	$L(R_1)^*$

Equivalence with finite automata

Theorem

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

The languages defined by regular expressions are regular

1. $R = a$: Let $N = \big(\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\}\big)$, where $\delta(q_1, a) = \{q_2\}$ and $\delta(r, b) = \emptyset$ for all $r \neq q_1$ or $b \neq a$.
2. $R = \varepsilon$: Let $N = \big(\{q_1\}, \Sigma, \delta, q_1, \{q_1\}\big)$, where $\delta(r, b) = \emptyset$ for all r and b.
3. $R = \emptyset$: Let $N = \big(\{q_1\}, \Sigma, \delta, q_1, \emptyset\big)$, where $\delta(r, b) = \emptyset$ for all r and b.
4. $R = R_1 \cup R_2$: $L(R) = L(R_1) \cup L(R_2)$.
5. $R = R_1 \circ R_2$: $L(R) = L(R_1) \circ L(R_2)$.
6. $R = R_1^*$: $L(R) = L(R_1)^*$.

Regular languages can be defined by regular expressions

We need generalized nondeterministic finite automata (GNFA) – nondeterministic finite automata wherein the transition arrows may have any regular expressions as labels.

1. The start state has transition arrows going to every other state but no arrows coming in from any other state.
2. There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
3. Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.

Generalized nondeterministic finite automata

Definition

A GNFA is a 5-tuple $(Q, \Sigma, \delta, q_{\text{start}}, q_{\text{accept}})$, where

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

Formal definition of computation

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

1. $q_0 = q_{\text{start}}$ is the start state,
2. $q_k = q_{\text{accept}}$ is the accept state, and
3. for each $i \in [k]$, we have $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$.

Regular languages can be defined by regular expressions

Let M be the DFA for language A.

- We convert M to a GNFA G by adding a new start state and a new accept state and additional transition arrows as necessary.
  1. The start state has transition arrows going to every other state but no arrows coming in from any other state.
  2. There is only a single accept state, and it has arrows coming in from every other state but no arrows going to any other state. Furthermore, the accept state is not the same as the start state.
  3. Except for the start and accept states, one arrow goes from every state to every other state and also from each state to itself.
- Then we use a procedure CONVERT on G to return an equivalent regular expression.

$\textsc{convert}(G)$:

1. Let k be the number of states of G.
2. If $k = 2$, then return the regular expression R labelling the arrow from q_{start} to q_{accept}.
3. If $k > 2$, we select any state $q_{\text{rip}} \in Q - \{q_{\text{start}}, q_{\text{accept}}\}$ and let $G' = \left(Q', \Sigma, \delta', q_{\text{start}}, q_{\text{accept}}\right)$ be the GNFA, where

   $$Q' = Q - \{q_{\text{rip}}\},$$

   and for any $q_i \in Q' - \{q_{\text{accept}}\}$ and $q_j \in Q' - \{q_{\text{start}}\}$, let

   $$\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4),$$

   for $R_1 = \delta(q_i, q_{\text{rip}})$, $R_2 = \delta(q_{\text{rip}}, q_{\text{rip}})$, $R_3 = \delta(q_{\text{rip}}, q_j)$, and $R_4 = \delta(q_i, q_j)$.
4. Compute $\textsc{convert}(G')$ and return this value.