# *Theory of Computation*

Course note based on *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, 2nd edition, authored by Martin Davis, Ron Sigal, and Elaine J. Weyuker.

course note prepared by

Tyng–Ruey Chuang

Week 11, Spring 2008

**About This Course Note**

- It is prepared for the course *Theory of Computation* taught at the National Taiwan University in Spring 2008.
- It follows very closely the book *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science*, 2nd edition, by Martin Davis, Ron Sigal, and Elaine J. Weyuker. Morgan Kaufmann Publishers. ISBN: 0-12-206382-1.
- It is available from Tyng-Ruey Chuang's web site:

  `http://www.iis.sinica.edu.tw/~trc/`

  

# 1 Regular Languages (9)

## 1.1 Closure Properties (9.4)

### Closure properties

- To show that the class of regular languages is closed under a large number of operations.
- To use deterministic or nondeterministic finite automata whenever necessary, as the two classes of automata are equivalent in expressiveness (Theorem 2.1).

**Nonrestarting dfa**

**Definition.** A dfa is called *nonrestarting* if there is no pair $q, s$ for which

$$\delta(q, s) = q_1$$

where $q_1$ is the initial state. **Theorem 4.1.** There is an algorithm that will transform a given dfa $\mathscr{M}$ into a nonrestarting dfa $\tilde{\mathscr{M}}$ such that $L(\tilde{\mathscr{M}}) = L(\mathscr{M})$.

**Constructing a nonrestarting dfa from a dfa**

*Proof of Theorem 4.1.* From a dfa $\mathscr{M}$, we can construct an equivalent nonrestarting dfa $\tilde{\mathscr{M}}$ by adding a new "returning initial" state $q_{n+1}$, and by redefining the transition function accordingly. That is, for $\tilde{\mathscr{M}}$, we define

- the set of states $\tilde{Q} = Q \cup \{q_{n+1}\}$
- the transition function $\tilde{\delta}$ by

$$\begin{aligned} \tilde{\delta}(q, s) &= \begin{cases} \delta(q, s) & \text{if } \; q \in Q \text{ and } \delta(q, s) \neq q_1 \\ q_{n+1} & \text{if } \; q \in Q \text{ and } \delta(q, s) = q_1 \end{cases} \\ \tilde{\delta}(q_{n+1}, s) &= \tilde{\delta}(q_1, s) \end{aligned}$$

- the set of final states $\tilde{F} = \begin{cases} F & \text{if } \; q_1 \notin F \\ F \cup \{q_{n+1}\} & \text{if } \; q_1 \in F \end{cases}$

To see that $L(\mathscr{M}) = L(\tilde{\mathscr{M}})$ we observe that $\tilde{\mathscr{M}}$ follows the same transitions as $\mathscr{M}$ except whenever $\mathscr{M}$ reenters $q_1$, $\tilde{\mathscr{M}}$ enters $q_{n+1}$. □

$L \cup \tilde{L}$

**Theorem 4.2.** If $L$ and $\tilde{L}$ are regular languages, then so is $L \cup \tilde{L}$. *Proof.* Let $\mathscr{M}$ and $\tilde{\mathscr{M}}$ be nonrestarting dfas that accept $L$ and $\tilde{L}$ respectively. $\mathscr{M}$ and $\tilde{\mathscr{M}}$ are distinct but use the same alphabet. We now construct a ndfa $\check{\mathscr{M}}$ by "merging" $\mathscr{M}$ and $\tilde{\mathscr{M}}$ but with a new initial state $\check{q}_1$. That is, we define $\check{\mathscr{M}}$ by

- the set of states $\check{Q} = Q \cup \tilde{Q} \cup \{\check{q}_1\} - \{q_1, \tilde{q}_1\}$
- the transition function $\check{\delta}$ by

$$\begin{aligned} \check{\delta}(q, s) &= \begin{cases} \{\delta(q, s)\} & \text{if } \; q \in Q - \{q_1\} \\ \{\tilde{\delta}(q, s)\} & \text{if } \; q \in \tilde{Q} - \{\tilde{q}_1\} \end{cases} \\ \check{\delta}(\check{q}_1, s) &= \{\delta(q_1, s)\} \cup \{\tilde{\delta}(\tilde{q}_1, s)\} \end{aligned}$$

- the set of final states

$$\check{F} = \begin{cases} F \cup \tilde{F} \cup \{\check{q}_1\} - \{q_1, \tilde{q}_1\} & \text{if } q_1 \in F \text{ or } \tilde{q}_1 \in \tilde{F} \\ F \cup \tilde{F} & \text{otherwise} \end{cases}$$

Note that once a first transition has been selected, $\check{\mathscr{M}}$ is locked into either $\mathscr{M}$ or $\tilde{\mathscr{M}}$. Hence $L(\check{\mathscr{M}}) = L \cup \tilde{L}$. □

$A^* - L$

**Theorem 4.3.** Let $L \subseteq A^*$ be a regular language. Then $A^* - L$ is regular. *Proof.* Let $\mathscr{M}$ be a dfa that accept $L$. Let dfa $\bar{\mathscr{M}}$ be exactly like $\mathscr{M}$ except that it accepts precisely when $\mathscr{M}$ rejects. That is, the set of accepting states of $\bar{\mathscr{M}}$ is $Q - F$. Then $L(\bar{\mathscr{M}}) = A^* - L$. □

$L_1 \cap L_2$

**Theorem 4.4.** If $L_1$ and $L_2$ are regular languages, then so is $L_1 \cap L_2$. *Proof.* Let $L_1, L_2 \subseteq A^*$. Then, by the De Morgan identity, we have

$$L_1 \cap L_2 = A^* - ((A^* - L_1) \cup (A^* - L_2))$$

Theorem 4.2 and 4.3 then give the result. □

**$\emptyset$ and $\{0\}$**

**Theorem 4.5.** $\emptyset$ and $\{0\}$ are regular languages. *Proof.* $\emptyset$ is clearly the language accepted by any automaton whose set of accepting states is empty. For $\{0\}$, we can construct a two-state dfa such that $F = \{q_1\}$ and $\delta(q_1, a) = \delta(q_2, a) = q_2$ for every symbol $a \in A$, the alphabet. Clearly this dfa accepts $\{0\}$. □

**Every finite subset of $A^*$ is regular**

**Theorem 4.5.** Let $u \in A^*$. Then $\{u\}$ is a regular language. *Proof.* Theorem 4.4 proves the case for $u = 0$. For the other case, let $u = a_1 a_2 \ldots a_l$ where $l \geq 1, a_1, a_2, \ldots a_l \in A$. We now construct a $(l+1)$–state ndfa $\mathscr{M}$ with initial state $q_1$, accepting state $q_{l+1}$, and the transition function $\delta$ given by

$$\begin{aligned} \delta(q_i, a_i) &= \{q_{i+1}\}, \quad i = 1, \ldots, l \\ \delta(q_i, a) &= \emptyset \quad \text{for } a \in A - \{a_i\}, \quad i = 1, \ldots, l \end{aligned}$$

Clearly $L(\mathscr{M}) = \{u\}$. □ **Corollary 4.7.** Every finite subset of $A^*$ is regular. □

## 1.2 Kleene's Theorem (9.5)

**Characterizations of Regular Languages**

We now show that the class of regular languages can be characterized as the class of all languages obtained from finite languages using the operations $\cup, \cdot, ^*$ a finite number of times. We will see that there are other characterizations of regular languages as well.

**Definitions of $L_1 \cdot L_2$ and $L^*$**

**Definition.** Let $L_1, L_2 \subseteq A^*$. Then we write

$$L_1 \cdot L_2 = L_1 L_2 = \{uv \mid u \in L_1 \text{ and } v \in L_2\}.$$

$\Box$. **Definition.** Let $L \subseteq A^*$. Then we write

$$L^* = \{u_1 u_2 \dots u_n \mid n \geq 0, u_1, u_2, \dots, u_n \in L\}.$$

$\Box$. Note that, for $L^*$,

- $0 \in L^*$
- The notation of $A^*$ is consistent with the definition of $L^*$.

$L \cdot \tilde{L}$

**Theorem 5.1.** If $L, \tilde{L}$ are regular languages, then $L \cdot \tilde{L}$ is regular. *Proof.* Let $\mathscr{M}$ and $\tilde{\mathscr{M}}$ be dfas that accept $L$ and $\tilde{L}$ respectively. The two are distinct but use the same alphabet. We now construct a ndfa $\dot{\mathscr{M}}$ by "gluing together" the two dfas. We define

- the set of states $\dot{Q} = Q \cup \tilde{Q}$
- the transition function $\dot{\delta}$ by

$$\dot{\delta}(q, s) = \begin{cases} \{\delta(q, s)\} & \text{if} \quad q \in Q - F \\ \{\delta(q, s)\} \cup \{\tilde{\delta}(\tilde{q}_1, s)\} & \text{if} \quad q \in F \\ \{\tilde{\delta}(q, s)\} & \text{if} \quad q \in \tilde{Q} \end{cases}$$

- the set of final states

$$\dot{F} = \begin{cases} F \cup \tilde{F} & \text{if } 0 \in \tilde{L} \\ \tilde{F} & \text{if } 0 \notin \tilde{L} \end{cases}$$

Clearly, $L \cdot \tilde{L} = L(\dot{\mathscr{M}})$, so that $L \cdot \tilde{L}$ is regular. $\Box$.

$L^*$

**Theorem 5.2.** If $L$ is a regular languages, then so is $L^*$. *Proof.* Let $\mathscr{M}$ be a nonrestarting dfa that accept $L$. We now construct a "looping" ndfa $\tilde{\mathscr{M}}$ with the same states and initial state as $\mathscr{M}$, and accepting state $q_1$. The transition function $\tilde{\delta}$ is defined as follows:

$$\tilde{\delta}(q, s) = \begin{cases} \{\delta(q, s)\} & \text{if} \quad \delta(q, s) \notin F \\ \{\delta(q, s)\} \cup \{q_1\} & \text{if} \quad \delta(q, s) \in F \end{cases}$$

That is, whenever $\mathscr{M}$ would enter an accepting state, $\tilde{\mathscr{M}}$ will enter either the corresponding accepting state or the initial state. Clearly, $L^* = L(\tilde{\mathscr{M}})$, so that $L^*$ is a regular language. $\Box$.

**Kleene's Theorem**

**Theorem 5.3.** A language is regular if and only if it can be obtained from finite languages by applying the three operators $\cup, \cdot, ^*$ a finite number of times. *Proof.* ($\Longleftarrow$) Every finite language is regular. The three operators build regular languages from regular languages. Therefore, by induction on the number of applications of $\cup, \cdot, ^*$, any language obtained from finite languages by applying these operators a finite number of times is regular. ($\Longrightarrow$) Let $L = L(\mathscr{M})$ where $\mathscr{M}$ is a dfa with states $q_1, \ldots, q_n$. As usual, $q_1$ is the initial state, $F$ the set of accepting states, $\delta$ the transition function, and $A = \{s_1, \ldots, s_m\}$ the alphabet.

We define the sets $R^k_{i,j}$, for all $i, j > 0, k \geq 0$, as follows:

$$\begin{aligned} R^k_{i,j} = \{x \in A^* \quad | \quad & \delta^*(q_i, x) = q_j \text{ and, as it moves across } x, \\ & \mathscr{M} \text{ passes through no state } q_l \text{ with } l > k\} \end{aligned}$$

**Kleene's Theorem, Continued**

*Proof (continued).* We observe that

$$\begin{aligned} R^0_{i,i} &= \{0\} \\ R^0_{i,j} &= \{a \in A \mid \delta(q_i, a) = q_j\}, \text{ for } i \neq j \end{aligned}$$

Now, to process any string of length $> 1$, $\mathscr{M}$ will pass through some intermediate state $q_l, l \geq 1$. We can write

$$R^{k+1}_{i,j} = R^k_{i,j} \cup (R^k_{i,k+1} \cdot (R^k_{k+1,k+1})^* \cdot R^k_{k+1,j})$$

In addition, $R^k_{i,j}$ is regular for for all $i, j, k$. This is proved by an induction on $k$. For $k = 0$, $R^0_{i,j}$ is finite hence regular. Assuming the result known for $k$, ($\Longleftarrow$) yields the result for $k+1$. Finally, we note that

$$L(\mathscr{M}) = \bigcup_{q_j \in F} R^n_{1,j}$$

and we conclude the proof. □

**Regular Expressions**

For an alphabet $A = \{\ s_1, s_2, \ldots, s_k\ \}$, we define the corresponding alphabet

$$\mathbf{A} = \{\ \mathbf{s_1}, \mathbf{s_2}, \ldots \mathbf{s_k}, \mathbf{0}, \boldsymbol{\emptyset}, \cup, \cdot, ^*, (, )\ \}.$$

The class of *regular expressions* on $A$ is then defined to be the subset of $\mathbf{A}^*$ determined by the following:

1. $\boldsymbol{\emptyset}, \mathbf{0}, \mathbf{s_1}, \mathbf{s_2}, \ldots \mathbf{s_k}$ are regular expressions.
2. If $\alpha$ and $\beta$ are regular expressions, then so is $(\alpha \cup \beta)$.

3. If $\alpha$ and $\beta$ are regular expressions, then so is $(\alpha \cdot \beta)$.

4. If $\alpha$ is a regular expression, then so is $\alpha^*$.

5. No expression is regular unless it can be generated using a finite number of applications of 1–4.

**Semantics of Regular Expressions**

For each regular expression $\gamma$, we define a corresponding regular language $\langle\gamma\rangle$ by recursion according to the following rules:

$$\begin{aligned} \langle \mathbf{s_i} \rangle &= \{s_i\} \\ \langle \mathbf{0} \rangle &= \{0\} \\ \langle \boldsymbol{\emptyset} \rangle &= \emptyset \\ \langle (\alpha \cup \beta) \rangle &= \langle\alpha\rangle \cup \langle\beta\rangle \\ \langle (\alpha \cdot \beta) \rangle &= \langle\alpha\rangle \cdot \langle\beta\rangle \\ \langle \alpha^* \rangle &= \langle\alpha\rangle^* \end{aligned}$$

When $\langle\gamma\rangle = L$, we say that the regular expression $\gamma$ *represents* $L$.

**Regular Expressions, Examples**

$$\begin{aligned} \langle(\boldsymbol{a} \cdot (\boldsymbol{b}^* \cup \boldsymbol{c}^*))\rangle &= \{ab^{[n]} \mid n \geq 0\} \cup \{ac^{[m]} \mid m \geq 0\} \\ \langle(\mathbf{0} \cup (\boldsymbol{a} \cdot \boldsymbol{b})^*)\rangle &= \{(ab)^{[n]} \mid n \geq 0\} \\ \langle((\boldsymbol{c}^* \cdot \boldsymbol{b}^*))\rangle &= \{c^{[m]}b^{[n]} \mid m, n \geq 0\} \end{aligned}$$

**Finite Subsets of $A^*$**

**Theorem 5.4.** For every finite subset $L$ of $A^*$, there is a regular expressions $\gamma$ on $A$ such that $\langle\gamma\rangle = L$. *Proof.* We need only to consider the following:

- If $L = \emptyset$, then $L = \langle\boldsymbol{\emptyset}\rangle$.
- If $L = 0$, then $L = \langle\mathbf{0}\rangle$.
- If $L = \{x\}$, where $x = s_{i_1}s_{i_2}\ldots s_{i_l}$, then

$$L = \langle(\boldsymbol{s_{i_1}} \cdot (\boldsymbol{s_{i_2}} \cdot (\boldsymbol{s_{i_3}} \ldots \boldsymbol{s_{i_l}}) \ldots))\rangle.$$

- If $L$ has more than one elements. Assuming the result is known for languages of $k$ elements, let $L$ have $k+1$ elements. Then we can write $L = L_1 \cup \{x\}$, where $x \in A^*$ and $L_1$ contains $k$ elements. By induction hypothesis, there is a regular expression $\alpha$ such that $\langle \alpha \rangle = L_1$. By the above, there is regular expression $\beta$ such that $\langle \beta \rangle = \{x\}$. Then we have
$$\langle (\alpha \cup \beta) \rangle = L_1 \cup \{x\} = L$$

$\square$

**Kleene's Theorem — Second Version**

**Theorem 5.5.** A language $L \subseteq A^*$ is regular if and only if there is a regular expression $\gamma$ on $A$ such that $\langle \gamma \rangle = L$. *Proof.* ($\Longleftarrow$) For any regular expression $\gamma$, the regular language $\langle \gamma \rangle$ is built up from finite languages by applying $\cup, \cdot, ^*$ a finite number of times, so $\langle \gamma \rangle$ is regular by the Kleene's theorem. ($\Longrightarrow$) If a regular language $L$ is finite, then by Theorem 5.4, there is a regular expression $\gamma$ such that $\langle \gamma \rangle = L$. Otherwise, by Kleene's theorem, $L$ can be obtained from certain finite languages by a finite of applications of $\cup, \cdot, ^*$.

Starting with regular expressions representing these finite languages, we then build up a regular expression representing $L$ by simply indicating each use of the operations $\cup, \cdot, ^*$ by writing $\cup$, $\boldsymbol{\cdot}$, $\boldsymbol{*}$, respectively, and punctuating with ( and ). $\square$

## 1.3 The Pumping Lemma and Its Application (9.6)

**Pigeon-Hole Principle**

*Pigeon-Hole Principle.* If $n+1$ objects are distributed among $n$ sets, then at least one of the sets must contain at least two objects. $\square$

**Pumping Lemma**

**Theorem 6.1.** Let $L = L(\mathscr{M})$, where $\mathscr{M}$ is a dfa with $n$ states. Let $x \in L$, where $|x| \geq n$. Then we can write $x = uvw$, where $v \neq 0$ and $uv^{[i]}w \in L$ for all $i = 0, 1, 2, 3, \ldots$. *Proof.* Since $x$ has at least $n$ symbols, $\mathscr{M}$ must go through at least $n$ state transitions. Including the initial state, this requires $\mathscr{M}$ to visit at least $n+1$ states. We conclude that $\mathscr{M}$ must visit at least one state $q$ more than once. Then we can write $x = uvw$, where

$$\begin{aligned} \delta^*(q_1, u) &= q, \\ \delta^*(q, v) &= q, \\ \delta^*(q, w) &\in F. \end{aligned}$$

However, the loop starting and ending at $q$ can be repeated any number of times and $\mathscr{M}$ still reachs the accepting states. It is clear that

$$\delta^*(q_1, uv^{[i]}w) = \delta^*(q_1, uvw) \in F.$$

Hence $uv^{[i]}w \in L$. $\square$

**Applications of The Pumping Lemma, I**

**Theorem 6.2.** Let $\mathscr{M}$ be a dfa with $n$ states. Then, if $L(\mathscr{M}) \neq \emptyset$, there is a string $x \in L(\mathscr{M})$ such that $|x| < n$. *Proof.* Let $x$ be a string in $L(\mathscr{M})$ of the shortest possible length. Suppose $|x| \geq n$. By the pumping lemma, $x = uvw$, where $v \neq 0$ and $uw \in L(\mathscr{M})$. Since $|uw| < |x|$, this is a contradiction. Thus $|x| < n$. $\square$ This theorem shows how to test a given dfa $\mathscr{M}$ to see whether the language it accepts is empty! We need only "run" $\mathscr{M}$ on all strings of length less than the number of states of $\mathscr{M}$. If none is accepted, we then conclude $L(\mathscr{M}) = \emptyset$.

**Applications of The Pumping Lemma, II**

**Theorem 6.4.** Let $\mathscr{M}$ be a dfa with $n$ states. Then, $L(\mathscr{M})$ is infinite if and only if $L(\mathscr{M})$ contains a string $x$ such that $n \leq |x| < 2n$. *Proof.* ($\Longrightarrow$) Let $L(\mathscr{M})$ be infinite. Then $L(\mathscr{M})$ must contain strings of length $\geq 2n$. Let $x \in L(\mathscr{M})$, where $x$ has the shortest possible length $\geq 2n$. We write $x = x_1x_2$, where $|x_1| = n$ and $|x_2| \geq n$. By using the pigeon-hole principle, we can write $x_1 = uvw$, where

$$\begin{aligned} \delta^*(q_1, u) &= q, \\ \delta^*(q, v) &= q, \quad \text{with } 1 \leq |v| \leq n, \\ \delta^*(q, wx_2) &\in F. \end{aligned}$$

Thus $uwx_2 \in L(\mathscr{M})$, and $|x| > |uwx_2| \geq |x_2| \geq n$. Since $x$ is a shortest string of $L(\mathscr{M})$ with length at least $2n$, we conclude $n \leq |uwx_2| < 2n$.

**Applications of The Pumping Lemma, II, Continued**

*Proof (Theorem 6.4).* ($\Longleftarrow$) Let $x \in L(\mathscr{M})$ with $n \leq |x| < 2n$. By the pumping lemma, we can write $x = uvw$, where $v \neq 0$ and $uv^{[i]}w \in L(\mathscr{M})$ for all $i$. This shows that $L(\mathscr{M})$ is infinite. $\square$ Theorem 6.4 shows how to test a given dfa $\mathscr{M}$ to see whether the language it accepts is finite! We need only run $\mathscr{M}$ on all strings $x$ such that $n \leq |x| < 2n$, where $\mathscr{M}$ has $n$ states. $L(\mathscr{M})$ is infinite just in case $\mathscr{M}$ accepts at least one of these strings.

**Applications of The Pumping Lemma, III**

The pumping lemma also provides us a technique for showing that given languages are not regular. For example, $L = \{a^{[n]}b^{[n]} \mid n > 0\}$ is not regular. Suppose it is, then $L = L(\mathscr{M})$, where $\mathscr{M}$ is a dfa and has $m$ states. We will derive a contradiction by showing that there is a word $x \in L$, with $|x| > m$, such that there is no way of writing $x = uvw$, with $v \neq 0$, so that $\{uv^{[i]}w \mid i \geq 0\} \subseteq L$. Let $x = a^{[m]}b^{[m]}$. If we write $x = uvw$, with $v \neq 0$, then either $v = a^{[l_1]}$, or $v = a^{[l_1]}b^{[l_2]}$, or $v = b^{[l_2]}$, with $l_1, l_2 \leq m$. However, in each case, $uvvw \notin L$, contradicting the pumping lemma, so there can be no such dfa $\mathscr{M}$. We just show that $L$ is not regular.