

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

Institute of Information Science, Academia Sinica

Department of Information Management, National Taiwan University

Week 9, Spring 2010

About This Course Note

- ▶ It is prepared for the course *Theory of Computation* taught at the National Taiwan University in Spring 2010.
- ▶ 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/>

and released under a Creative Commons
"Attribution-ShareAlike 3.0 Taiwan" license:

<http://creativecommons.org/licenses/by-sa/3.0/tw/>

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

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

□.

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

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

□.

Note that, for L^* ,

- ▶ $\epsilon \in L^*$
- ▶ 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.

$L \cdot \tilde{L}$

Theorem 5.1. If L, \tilde{L} are regular languages, then $L \cdot \tilde{L}$ is regular.

Proof. Let \mathcal{M} and $\tilde{\mathcal{M}}$ be dfas that accept L and \tilde{L} respectively.

The two are distinct but use the same alphabet. We now construct a ndfa $\tilde{\mathcal{M}}$ by “gluing together” the two dfas.

$L \cdot \tilde{L}$

Theorem 5.1. If L, \tilde{L} are regular languages, then $L \cdot \tilde{L}$ is regular.

Proof. Let \mathcal{M} and $\tilde{\mathcal{M}}$ be dfas that accept L and \tilde{L} respectively.

The two are distinct but use the same alphabet. We now construct a ndfa $\dot{\mathcal{M}}$ by “gluing together” the two dfas. We define

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

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

- ▶ the set of final states

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

$L \cdot \tilde{L}$

Theorem 5.1. If L, \tilde{L} are regular languages, then $L \cdot \tilde{L}$ is regular.

Proof. Let \mathcal{M} and $\tilde{\mathcal{M}}$ be dfas that accept L and \tilde{L} respectively.

The two are distinct but use the same alphabet. We now construct a ndfa $\dot{\mathcal{M}}$ by “gluing together” the two dfas. We define

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

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

- ▶ 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, $L \cdot \tilde{L} = L(\dot{\mathcal{M}})$, so that $L \cdot \tilde{L}$ is regular. □

L^*

Theorem 5.2. If L is a regular languages, then so is L^* .

L^*

Theorem 5.2. If L is a regular languages, then so is L^* .

Proof. Let \mathcal{M} be a nonrestarting dfa that accept L . We now construct a “looping” ndfa $\tilde{\mathcal{M}}$ with the same states and initial state as \mathcal{M} , and accepting state q_1 . The transition function $\tilde{\delta}$ is defined as follows:

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

L^*

Theorem 5.2. If L is a regular languages, then so is L^* .

Proof. Let \mathcal{M} be a nonrestarting dfa that accept L . We now construct a “looping” ndfa $\tilde{\mathcal{M}}$ with the same states and initial state as \mathcal{M} , and accepting state q_1 . The transition function $\tilde{\delta}$ is defined as follows:

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

That is, whenever \mathcal{M} would enter an accepting state, $\tilde{\mathcal{M}}$ will enter either the corresponding accepting state or the initial state.

Clearly, $L^* = L(\tilde{\mathcal{M}})$, so that L^* is a regular language. \square .

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.

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. (\Leftarrow) 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.

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. (\Leftarrow) 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.

(\Rightarrow) Let $L = L(\mathcal{M})$ where \mathcal{M} is a dfa with states q_1, \dots, q_n . As usual, q_1 is the initial state, F the set of accepting states, δ the transition function, and $A = \{s_1, \dots, s_m\}$ the alphabet.

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. (\Leftarrow) 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.

(\Rightarrow) Let $L = L(\mathcal{M})$ where \mathcal{M} is a dfa with states q_1, \dots, q_n . As usual, q_1 is the initial state, F the set of accepting states, δ the transition function, and $A = \{s_1, \dots, s_m\}$ the alphabet.

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

$$R_{i,j}^k = \{x \in A^* \mid \delta^*(q_i, x) = q_j \text{ and, as it moves across } x, \\ \mathcal{M} \text{ passes through no state } q_l \text{ with } l > k\}$$

Kleene's Theorem, Continued

Proof (continued). We observe that

$$R_{i,j}^0 = \{0\}$$

$$R_{i,j}^0 = \{a \in A \mid \delta(q_i, a) = q_j\}, \text{ for } i \neq j$$

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

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

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

$$L(\mathcal{M}) = \bigcup_{q_j \in F} R_{1,j}^n$$

and we conclude the proof.

Regular Expressions

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

$$\mathbf{A} = \{ \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k, \mathbf{0}, \emptyset, \cup, \cdot, *, (,) \}.$$

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

1. $\emptyset, \mathbf{0}, \mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_k$ are regular expressions.
2. If α and β are regular expressions, then so is $(\alpha \cup \beta)$.
3. If α and β are regular expressions, then so is $(\alpha \cdot \beta)$.
4. If α is a regular expression, then so is α^* .
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 γ , we define a corresponding regular language $\langle \gamma \rangle$ by recursion according to the following rules:

$$\langle s_i \rangle = \{s_i\}$$

$$\langle 0 \rangle = \{0\}$$

$$\langle \emptyset \rangle = \emptyset$$

$$\langle (\alpha \cup \beta) \rangle = \langle \alpha \rangle \cup \langle \beta \rangle$$

$$\langle (\alpha \cdot \beta) \rangle = \langle \alpha \rangle \cdot \langle \beta \rangle$$

$$\langle \alpha^* \rangle = \langle \alpha \rangle^*$$

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

Regular Expressions, Examples

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

Finite Subsets of A^*

Theorem 5.4. For every finite subset L of A^* , there is a regular expressions γ on A such that $\langle \gamma \rangle = L$.

Finite Subsets of A^*

Theorem 5.4. For every finite subset L of A^* , there is a regular expressions γ on A such that $\langle \gamma \rangle = L$.

Proof. We need only to consider the following:

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

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

- ▶ 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 α such that $\langle \alpha \rangle = L_1$. By the above, there is regular expression β such that $\langle \beta \rangle = \{x\}$. Then we have

$$\langle (\alpha \cup \beta) \rangle = L_1 \cup \{x\} = L$$

Kleene's Theorem — Second Version

Theorem 5.5. A language $L \subseteq A^*$ is regular if and only if there is a regular expression γ on A such that $\langle \gamma \rangle = L$.

Kleene's Theorem — Second Version

Theorem 5.5. A language $L \subseteq A^*$ is regular if and only if there is a regular expression γ on A such that $\langle \gamma \rangle = L$.

Proof. (\Leftarrow) For any regular expression γ , 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.

Kleene's Theorem — Second Version

Theorem 5.5. A language $L \subseteq A^*$ is regular if and only if there is a regular expression γ on A such that $\langle \gamma \rangle = L$.

Proof. (\Leftarrow) For any regular expression γ , 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.

(\Rightarrow) If a regular language L is finite, then by Theorem 5.4, there is a regular expression γ 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 $\cup, \cdot, *$, respectively, and punctuating with (and). \square

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. □

Pumping Lemma

Theorem 6.1. Let $L = L(\mathcal{M})$, where \mathcal{M} is a dfa with n states. Let $x \in L$, where $|x| \geq n$. Then we can write $x = uvw$, where $v \neq \epsilon$ and $uv^i w \in L$ for all $i = 0, 1, 2, 3, \dots$

Proof. Since x has at least n symbols, \mathcal{M} must go through at least n state transitions. Including the initial state, this requires \mathcal{M} to visit at least $n + 1$ states. We conclude that \mathcal{M} must visit at least one state q more than once. Then we can write $x = uvw$, where

$$\delta^*(q_1, u) = q,$$

$$\delta^*(q, v) = q,$$

$$\delta^*(q, w) \in F.$$

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

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

Hence $uv^i w \in L$.

Applications of The Pumping Lemma, I

Theorem 6.2. Let \mathcal{M} be a dfa with n states. Then, if $L(\mathcal{M}) \neq \emptyset$, there is a string $x \in L(\mathcal{M})$ such that $|x| < n$.

Proof. Let x be a string in $L(\mathcal{M})$ of the shortest possible length. Suppose $|x| \geq n$. By the pumping lemma, $x = uvw$, where $v \neq \epsilon$ and $uw \in L(\mathcal{M})$. Since $|uw| < |x|$, this is a contradiction. Thus $|x| < n$. □

Applications of The Pumping Lemma, I

Theorem 6.2. Let \mathcal{M} be a dfa with n states. Then, if $L(\mathcal{M}) \neq \emptyset$, there is a string $x \in L(\mathcal{M})$ such that $|x| < n$.

Proof. Let x be a string in $L(\mathcal{M})$ of the shortest possible length. Suppose $|x| \geq n$. By the pumping lemma, $x = uvw$, where $v \neq \epsilon$ and $uw \in L(\mathcal{M})$. Since $|uw| < |x|$, this is a contradiction. Thus $|x| < n$. □

This theorem shows how to test a given dfa \mathcal{M} to see whether the language it accepts is empty! We need only “run” \mathcal{M} on all strings of length less than the number of states of \mathcal{M} . If none is accepted, we then conclude $L(\mathcal{M}) = \emptyset$.

Applications of The Pumping Lemma, II

Theorem 6.4. Let \mathcal{M} be a dfa with n states. Then, $L(\mathcal{M})$ is infinite if and only if $L(\mathcal{M})$ contains a string x such that $n \leq |x| < 2n$.

Proof. (\implies) Let $L(\mathcal{M})$ be infinite. Then $L(\mathcal{M})$ must contain strings of length $\geq 2n$. Let $x \in L(\mathcal{M})$, 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 $uwx_2 \in L(\mathcal{M})$, and $|x| > |uwx_2| \geq |x_2| \geq n$.

Applications of The Pumping Lemma, II, Continued

Proof (Theorem 6.4). Recall that we assume x is a shortest string of $L(\mathcal{M})$ with length at least $2n$. If $|x| = 2n$, then $|uwx_2| < |x| = 2n$. If $|x| > 2n$, then either uwx_2 becomes the shortest string of length at least $2n$ (which is a contradiction), or $|uwx_2| < 2n$. we conclude $n \leq |uwx_2| < 2n$.

(\Leftarrow) Let $x \in L(\mathcal{M})$ with $n \leq |x| < 2n$. By the pumping lemma, we can write $x = uvw$, where $v \neq \epsilon$ and $uv^i w \in L(\mathcal{M})$ for all i . This shows that $L(\mathcal{M})$ is infinite. \square

Theorem 6.4 shows how to test a given dfa \mathcal{M} to see whether the language it accepts is finite! We need only run \mathcal{M} on all strings x such that $n \leq |x| < 2n$, where \mathcal{M} has n states. $L(\mathcal{M})$ is infinite just in case \mathcal{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 $L = L(\mathcal{M})$, where \mathcal{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 $v \neq \epsilon$, so that $\{uv^i w \mid i \geq 0\} \subseteq L$.

Let $x = a^{[m]}b^{[m]}$. If we write $x = uvw$, with $v \neq \epsilon$, then either $v = a^{[h_1]}$, or $v = a^{[h_1]}b^{[h_2]}$, or $v = b^{[h_2]}$, with $h_1, h_2 \leq m$. However, in each case, $uv^2w \notin L$, contradicting the pumping lemma, so there can be no such dfa \mathcal{M} . We just show that L is not regular.