Theory of Computation

Prof. Michael Mascagni

Florida State University
Department of Computer Science

Alphabets and Strings

- An *alphabet* is a finite nonempty set A of *symbols*.
- An n-tuple of symbols of A is called a *word* or a *string* on A. In stead of writing a word as $(a_1, a_2, \ldots, a_n)$ we write simply $a_1 a_2 \ldots a_n$.
- If $u = a_1 a_2 \ldots a_n$, then we say that n is the length of u and we write $|u| = n$.
- We allow a unique null word, written 0, of length 0.
- The set of all words on the alphabet A is written as A^*.
- Any subset of A^* is called a *language on A* or a *language with alphabet A*.

Alphabets and Strings, More

- If $u, v \in A^*$, then we write $\widehat{uv}$ for the word obtained by placing the string v after the string u. For example, if $A = \{a, b, c\}$, $u = bab$, and $v = caa$, then $\widehat{uv} = babcaa$.
- Where no confusion can result, we write uv instead of $\widehat{uv}$.
- It is obvious that, for all u, $u0 = 0u = u$, and that, for all u, v, w, $u(vw) = (uv)w$.
- If u is a string, and $n \in N, n > 0$, we write

$$u^{[n]} = \underbrace{uu \ldots u}_{n}$$

  We also write $n^{[0]} = 0$.
- If $u \in A^*$, we write u^R for u written backward; i.e., if $u = a_1 a_2 \ldots a_n$, then $u^R = a_n \ldots a_2 a_1$. Clearly, $0^R = 0$, and $(uv)^R = v^R u^R$ for $u, v \in A^*$.

The Concept of Finite Automata

- A finite automaton has a finite number of internal *states* that control its behavior. The states function as memory in the sense that the current state keeps track of the progress of the computation.
- The automaton begins by reading the leftmost symbol on a finite input tape, in a specific state called the *initial state*.
- If at a given time, the automaton is in a state q_i, reading a given symbol s_j on the input tape, the machine moves one square to the right on the tape and enters a state q_k.
- The current state plus the symbol being read from the tape completely determine the automaton's next state.
- When all symbols have been read, the automaton either stops at an accepting state or a non-accepting state.

Definition of Finite Automaton

Definition. A *finite automaton* $\mathscr{M}$ consists of

- an *alphabet* $A = \{s_1, s_2, \ldots, s_n\}$,
- a set of *states* $Q = \{q_1, q_2, \ldots, q_m\}$,
- a *transition function* δ that maps each pair $(q_i, s_j), 1 \leq i \leq m, 1 \leq j \leq n$, into a state q_k,
- a set $F \subseteq Q$ of *final* or *accepting* states, and
- an *initial* state $q_1 \in Q$.

We can represent the transition function δ using a state versus symbol table.

What Does This Automaton Do?

The finite automaton $\mathscr{M}$ has

- *alphabet* $A = \{a, b\}$,
- the set of *states* $Q = \{q_1, q_2, q_3, q_4\}$,
- the *transition function* δ defined by the following table:

δ	a	b
q_1	q_2	q_4
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_4	q_4

- the set $F = \{q_3\}$ as the accepting states, and
- q_1 as the initial state.

What Does Automaton $\mathscr{M}$ Do?

For strings *aabbb*, *baba*, *aaba*, and *abbb*, the finite automaton $\mathscr{M}$

- accepts *aabbb* as $\mathscr{M}$ terminates in state q_3, which is an accepting state;
- rejects *baba* as $\mathscr{M}$ terminates in state q_4, which is not an accepting state;
- rejects *aaba* as $\mathscr{M}$ terminates in state q_4, which is not an accepting state;
- accepts *abbb* as $\mathscr{M}$ terminates in state q_3, which is an accepting state.

Function $\delta^*(q_i, u)$

If q_i is any state of $\mathscr{M}$ and $u \in A^*$, we shall write $\delta^*(q_i, u)$ for the state which $\mathscr{M}$ will enter if it begins in state q_i at the left end of the string u and moves across u until the entire string has been processed.

- $\delta^*(q_1, aabbb) = q_3$,
- $\delta^*(q_1, baba) = q_4$,
- $\delta^*(q_1, aaba) = q_4$,
- $\delta^*(q_1, abbb) = q_3$.

Definition of Function $\delta^*(q_i, u)$

A formal definition of function $\delta^*(q_i, u)$ is by the following recursion:

$$\begin{aligned} \delta^*(q_i, 0) &= q_i, \\ \delta^*(q_i, us_j) &= \delta(\delta^*(q_i, u), s_j). \end{aligned}$$

Obviously, $\delta^*(q_i, s_j) = \delta(q_i, s_j)$.

We say that $\mathscr{M}$ *accepts* a word u provided that $\delta^*(q_1, u) \in F$. $\mathscr{M}$ *rejects* a word u means that $\delta^*(q_1, u) \in Q - F$.

Regular Languages

The language accepted by a finite automaton $\mathscr{M}$, written $L(\mathscr{M})$, is the set of all $u \in A^*$ accepted by $\mathscr{M}$:

$$L(\mathscr{M}) = \{u \in A^* \mid \delta^*(q_1, u) \in F\}.$$

A language is called *regular* if there exists a finite automaton that accepts it.

What Language Does This Automaton Accept?

The finite automaton $\mathscr{M}$ has

- the alphabet $A = \{a, b\}$,
- the set of states $Q = \{q_1, q_2, q_3, q_4\}$,
- the transition function δ defined by the following table:

δ	a	b
q_1	q_2	q_4
q_2	q_2	q_3
q_3	q_4	q_3
q_4	q_4	q_4

- the set $F = \{q_3\}$ as the accepting states, and
- q_1 as the initial state.

What Language Does Automaton $\mathscr{M}$ Accept?

The language it accepts is

$$\{a^{[n]}b^{[m]} \mid n, m > 0\}.$$

As the above language is accepted by a finite automaton, we say it is a regular language.

State Transition Diagram

- Another way to represent the transition function δ is to draw a graph in which each state is represented by a *vertex*.
- The fact that $\delta(q_i, s_j) = q_k$ is represented by drawing an *arrow* from vertex q_i to vertex q_k and labeling it s_j.
- The diagram thus obtained is called the *state transition diagram* for the given automaton.
- See Fig. 1.1 in the textbook (p. 240) for the state transition diagram for the finite automaton we just showed in the previous two slides.

Nondeterministic Finite Automata

- We modify the definition of a finite automaton to permit transitions at each stage to either zero, one, or more than one states.
- That is, we make the the values of the transition function δ be *sets of states*, i.e., *sets of elements of* Q (rather than members of Q).
- The devices so obtained are called *nondeterministic finite automata* (ndfa).
- Sometimes the ordinary finite automata are then called *deterministic finite automata* (dfa).

Definition of Nondeterministic Finite Automaton

Definition. A *nondeterministic finite automaton* $\mathscr{M}$ consists of

- an alphabet $A = \{s_1, s_2, \ldots, s_n\}$,
- a set of states $Q = \{q_1, q_2, \ldots, q_m\}$,
- a transition function δ that maps each pair $(q_i, s_j), 1 \leq i \leq m, 1 \leq j \leq n$, into a subset of states $Q_k \subseteq Q$,
- a set $F \subseteq Q$ of final or accepting states, and
- an initial state $q_1 \in Q$.

Definition of Function $\delta^*(q_i, u)$

The formal definition of function $\delta^*(q_i, u)$ is now by:

$$\begin{aligned} \delta^*(q_i, 0) &= \{q_i\}, \\ \delta^*(q_i, us_j) &= \bigcup_{q \in \delta^*(q_i, u)} \delta(q, s_j). \end{aligned}$$

- A ndfa $\mathscr{M}$ with initial state q_1 *accepts* $u \in A^*$ if $\delta^*(q_1, u) \cap F \neq \emptyset$.
- That is, at least one of the states at which $\mathscr{M}$ ultimately arrives belongs to F.
- $L(\mathscr{M})$, *the language accepted by* $\mathscr{M}$, is the set of all strings accepted by $\mathscr{M}$.

What Does This Automaton Do?

The nondeterministic finite automaton $\mathscr{M}$ has

- the alphabet $A = \{a, b\}$,
- the set of states $Q = \{q_1, q_2, q_3, q_4\}$,
- the transition function δ defined by the following table:

δ	a	b
q_1	$\{q_1, q_2\}$	$\{q_1, q_3\}$
q_2	$\{q_4\}$	$\emptyset$
q_3	$\emptyset$	$\{q_4\}$
q_4	$\{q_4\}$	$\{q_4\}$

- the set $F = \{q_4\}$ as the accepting states, and
- q_1 as the initial state.
- For the state transition diagram of $\mathscr{M}$, see Fig. 2.1 in the textbook (p. 243).

What Strings Does Automaton $\mathscr{M}$ Accept?

M accepts a string on the alphabet $\{a, b\}$ just in case at least one of the symbols has two successive occurrence in the string.

Why?

Viewing dfa as ndfa

- Strictly speaking, a dfa is *not* just a special kind of ndfa.
- This is because for a dfa, $\delta(q, s)$ is a state, where for a ndfa it is a set of states.
- But it is natural to identify a dfa $\mathscr{M}$ with transition function δ, with the closely related ndfa $\bar{\mathscr{M}}$ whose transition function $\bar{\delta}$ is given by

  $$\bar{\delta}(q, s) = \{\delta(q, s)\},$$

  and which has the same final states as $\mathscr{M}$.
- It is obviously that $L(\mathscr{M}) = L(\bar{\mathscr{M}})$.

dfa is as expressive as ndfa

Theorem 2.1. A language is accepted by a ndfa if and only if it is regular. Equivalently, a language is accepted by an ndfa if and only if it is accepted by a dfa.

Proof Outline. As we have seen, a language accepted by a dfa is also accepted by an ndfa.

Conversely, let $L = L(\mathscr{M})$, where $\mathscr{M}$ is an ndfa with transition function δ, set of states $Q = \{q_1, \ldots, q_m\}$, and set of final states F. We will construct a dfa $\tilde{\mathscr{M}}$ such that $L(\tilde{\mathscr{M}}) = L(\mathscr{M}) = L$.

The idea of the construction is that the individual states of $\tilde{\mathscr{M}}$ will be sets of states of $\mathscr{M}$.

Constructing $\tilde{\mathscr{M}}$

The dfa $\tilde{\mathscr{M}}$ consists of

- the same alphabet $A = \{s_1, s_2, \ldots, s_n\}$ of the ndfa $\mathscr{M}$,
- the set of states $\tilde{Q} = \{Q_1, Q_2, \ldots, Q_{2^m}\}$ which consists of all the 2^m subsets of the set of states of the ndfa $\mathscr{M}$,
- the transition function $\tilde{\delta}$ defined by

$$\tilde{\delta}(Q_i, s) = \bigcup_{q \in Q_i} \delta(q, s),$$

- the set $\mathscr{F}$ of final states given by

$$\mathscr{F} = \{Q_i \mid Q_i \cap F \neq \emptyset\},$$

- the initial state $Q_1 = \{q_1\}$, where q_1 is the initial state of $\mathscr{M}$.

Lemma 1. Let $R \subseteq \tilde{Q}$. Then

$$\tilde{\delta}\Big(\bigcup_{Q_i \in R} Q_i,\ s\Big) = \bigcup_{Q_i \in R} \tilde{\delta}(Q_i, s).$$

Proof. Let $\bigcup_{Q_i \in R} Q_i = Q$. Then by definition,

$$\begin{aligned}
\tilde{\delta}(Q, s) &= \bigcup_{q \in Q} \delta(q, s) \\
&= \bigcup_{Q_i \in R} \bigcup_{q \in Q_i} \delta(q, s) \\
&= \bigcup_{Q_i \in R} \tilde{\delta}(Q_i, s).
\end{aligned}$$

□

Lemma 2. For any string u,

$$\tilde{\delta}^*(Q_i, u) = \bigcup_{q \in Q_i} \delta^*(q, u).$$

Proof. The proof is by induction on $|u|$. If $|u| = 0$, then $u = 0$ and

$$\tilde{\delta}^*(Q_i, 0) = Q_i = \bigcup_{q \in Q_i} \{q\} = \bigcup_{q \in Q_i} \delta^*(q, 0)$$

Proof. (Continued) If $|u| = l + 1$ and the result is known for $|u| = l$, we write $u = vs$, where $|v| = l$, and observe that, using Lemma 1 and the induction hypothesis,

$$\begin{aligned}
\tilde{\delta}^*(Q_i, u) = \tilde{\delta}^*(Q_i, vs) &= \tilde{\delta}(\tilde{\delta}^*(Q_i, v), s) \\
&= \tilde{\delta}\big(\bigcup_{q \in Q_i} \delta^*(q, v), s\big) \\
&= \bigcup_{q \in Q_i} \tilde{\delta}(\delta^*(q, v), s) \\
&= \bigcup_{q \in Q_i} \bigcup_{r \in \delta^*(q,v)} \delta(r, s) \\
&= \bigcup_{q \in Q_i} \delta^*(q, vs) = \bigcup_{q \in Q_i} \delta^*(q, u).
\end{aligned}$$

□

Lemma 3. $L(\mathscr{M}) = L(\tilde{\mathscr{M}})$.

Proof. $u \in L(\tilde{\mathscr{M}})$ if and only if $\tilde{\delta}^*(Q_1, u) \in \mathscr{F}$. But, by Lemma 2,

$$\tilde{\delta}^*(Q_1, u) = \tilde{\delta}^*(\{q_1\}, u) = \delta^*(q_1, u).$$

Hence,

$$\begin{aligned} u \in L(\tilde{\mathscr{M}}) \quad & \text{if and only if} \quad \delta^*(q_1, u) \in \mathscr{F} \\ & \text{if and only if} \quad \delta^*(q_1, u) \cap F \neq \emptyset \\ & \text{if and only if} \quad u \in L(\mathscr{M}) \end{aligned}$$

□

Note that Theorem 2.1 is an immediate consequence of Lemma 3.

Additional Examples

- Construct a dfa that accepts the language:

$$\{(11)^{[n]} \mid n \geq 0\}$$

- The vendor machine example. (Fig. 3.2 in textbook, p. 248)
- Construct an ndfa that accepts all and only strings which end in *bab* or *aaba*.
- Construct an ndfa that accepts the language:

$$\{a^{[n_1]}b^{[m_1]} \dots a^{[n_k]}b^{[m_k]} \mid n_1, m_1, \dots, n_k, m_k > 0\}.$$

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

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

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

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. 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

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

- 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 δ 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. □