CS:4330 Theory of Computation

Spring 2018

Regular Languages

Finite Automata and Regular Expressions

Haniel Barbosa

The University of Iowa

Readings for this lecture

Chapter 1 of [Sipser 1996], 3rd edition. Sections 1.1 and 1.3.

Abstraction of Problems

- *Data*: abstracted as a word in a given alphabet
  - Σ: alphabet, a finite, non-empty set of symbols
  - Σ^*: all the words of finite length built up using Σ
- *Conditions*: abstracted as a set of words, called *language*
  - Any subset $L \subseteq \Sigma^*$
- *Unknown*: Implicitly a Boolean variable: *true* if a word is in the language, *false* otherwise
  - Given $w \in \Sigma^*$ and $L \subseteq \Sigma^*$, does $w \in L$?

Finite Automata

- The simplest computational model is called a *finite state machine* or a *finite automaton*

- Representations:
  - Graphical
  - Tabular
  - Mathematical

Computation of a Finite Automaton

- ▷ The automaton receives the input symbols one by one from left to right, changing the “active” state
- ▷ After reading each symbol, the “active state” moves from one state to another along the transition that has that symbol as its label
- ▷ When the last symbol of the input is read the automaton produces the output: *accept* if the “active state” is in an accept state, or *reject* otherwise

Applications

- ▷ Finite automata are popular in parser construction of compilers
- ▷ Finite automata and their probabilistic counterparts, *Markov chains*, are useful tools for pattern recognition
  Example: speech processing and optical character recognition
- ▷ Markov chains have been used to model and predict price changes in financial applications

Formal Definition of Finite Automata

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ in which:

- Q is a finite set called the *states*
- Σ is a finite set called the *alphabet*
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the *start state*, also called *initial stat*
- $F \subseteq Q$ is the set of *accepted states*, also called the *final states*

Language Recognized by an Automaton

- If L is the set of all strings that a finite automaton M accepts, we say that L is the *language of the machine M* and write $\mathscr{L}(M) = L$

- An automaton may accept several strings, but it always recognizes only one language

- If a machine accepts no strings, it still recognizes one language, namely the empty language $\varnothing$

Formal Definition of Acceptance

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton and $w = \mathrm{a}_0 \ldots \mathrm{a}_n$ be a string over Σ

Then M *accepts* w if a sequence of states $r_0, \ldots, r_n$ exists in Q such that:

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

Condition (1) says where the machine starts

Condition (2) says that the machine goes from state to state according to its transition function δ

Condition (3) says when the machine accepts its input: if it ends up in an accept state

Regular Languages

We say that a finite automaton M recognizes the language L if
$L = \{w \mid M \text{ accepts } w\}$

Definition: A language is called *regular language* if there exists a finite automaton that recognizes it.

Designing Finite Automata

▷ Whether it be of automaton or artwork, design is a creative process. Consequently it cannot be reduced to a simple recipe or formula.

▷ The approach:

- Identify the finite pieces of information you need to solve the problem. These are the *states*
- Identify the condition (alphabet) to change from one state to another
- Identify the start and final states
- Add missing transitions

Operations on Regular Languages

Let A and B be languages. We define regular operations *union*, *concatenation*, and *start* as follows

- *Union:* $A \cup B = \{x \mid x \in A \vee x \in B\}$
- *Concatenation:* $A \circ B = \{xy \mid x \in A \wedge y \in B\}$
- *Star:* $A^* = \{x_1 \ldots x_k \mid k \geq 0 \wedge x_i \in A,\ 0 \leq i \leq k\}$

Note:

1. $\epsilon \in A^*$, no matter what A is
2. A^+ denotes $A \circ A^*$

Regular Expressions

A *regular expression* (RE in short) is a string of symbols that describes a regular language.

- Three base cases:
  - For any $\mathrm{a} \in \Sigma$, a is a regular expression denoting the language $\{\mathrm{a}\}$
  - ϵ is a regular expression denoting the language $\{\epsilon\}$
  - $\varnothing$ is a regular expression denoting the language $\varnothing$;

- Three recursive cases: If r_1 and r_2 are regular expressions denoting languages L_1 and L_2, respectively, then
  - *Union:* $r_1 \cup r_2$ denotes $L_1 \cup L_2$
  - *Concatenation:* $r_1 r_2$ denotes $L_1 \circ L_2$
  - *Star:* r_1^* denotes L_1^*

Some useful notation

Let r be a regular expression:

- The string r^+ represents rr^*, and it also holds that $r^+ \cup \{\epsilon\} = r^*$
- The string r^k represents $\underbrace{rr\ldots r}_{k \text{ times}}$
- Recall that the symbol Σ represents the alphabet $\{\mathrm{a}_1, \ldots, \mathrm{a}_k\}$
- As with automata, the language represented by r is denoted $\mathscr{L}(r)$

Precedence Rules

- ▷ The star (*) operation has the highest precedence
- ▷ The concatenation ($\circ$) operation is second on precedence order
- ▷ The union ($\cup$) or $(+)$ operation is the least preferred
- ▷ Parenthesis can be omitted using these rules

Examples

$0^*10^* \quad =$

Examples

$$0^*10^* = \{w \mid w \text{ contains a single } 1\}$$

$$\Sigma^*1\Sigma^* =$$

Examples

$$
\begin{aligned}
0^*10^* &= \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* &= \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* &=
\end{aligned}
$$

Examples

$$
\begin{aligned}
0^*10^* &= \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* &= \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* &= \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* &=
\end{aligned}
$$

Examples

$$
\begin{aligned}
0^*10^* &= \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* &= \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* &= \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* &= \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* &=
\end{aligned}
$$

Examples

$$
\begin{array}{lcl}
0^*10^* & = & \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* & = & \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* & = & \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* & = & \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* & = & \{w \mid w \text{ is of even length}\}
\end{array}
$$

$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ —

Examples

$$
\begin{array}{lcl}
0^*10^* & = & \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* & = & \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* & = & \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* & = & \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* & = & \{w \mid w \text{ is of even length}\}
\end{array}
$$

$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ – all words starting and ending with the same letter

$(0 \cup \epsilon)1^* = 01^* \cup 1^*$ –

Examples

$$
\begin{array}{lcl}
0^*10^* & = & \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* & = & \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* & = & \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* & = & \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* & = & \{w \mid w \text{ is of even length}\}
\end{array}
$$

$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ – all words starting and ending with the same letter

$(0 \cup \epsilon)1^* = 01^* \cup 1^*$ – all strings of forms $1, 11, 1 \ldots 1$ and $0, 1, 11, 1 \ldots 1$

$R\varnothing$ –

Examples

$$
\begin{array}{lcl}
0^*10^* & = & \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* & = & \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* & = & \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* & = & \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* & = & \{w \mid w \text{ is of even length}\}
\end{array}
$$

$0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$ – all words starting and ending with the same letter

$(0 \cup \epsilon)1^* = 01^* \cup 1^*$ – all strings of forms $1, 11, 1 \ldots 1$ and $0, 1, 11, 1 \ldots 1$

$R\varnothing$ – $\varnothing$

$\varnothing^*$ –

Examples

$$
\begin{array}{lcl}
0^*10^* & = & \{w \mid w \text{ contains a single } 1\} \\
\Sigma^*1\Sigma^* & = & \{w \mid w \text{ has at least a single } 1\} \\
\Sigma^*(101)\Sigma^* & = & \{w \mid w \text{ contains 101 as a substring}\} \\
1^*(01^+)^* & = & \{w \mid \text{ every 0 in } w \text{ is followed by at least a single 1}\} \\
(\Sigma\Sigma)^* & = & \{w \mid w \text{ is of even length}\}
\end{array}
$$

$$
\begin{array}{lcl}
0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1 & - & \text{all words starting and ending with the same letter} \\
(0 \cup \epsilon)1^* = 01^* \cup 1^* & - & \text{all strings of forms } 1, 11, 1\ldots1 \text{ and } 0, 1, 11, 1\ldots1 \\
R\varnothing & - & \varnothing \\
\varnothing^* & - & \{\epsilon\}
\end{array}
$$