www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 3 Issue 1, Jan 2014 Page No. 3700-3703

A Novel Approach to Construct Deterministic Finite State Automata

Amrita Bhattacharjee[1], Bipul Syam Purkayastha[2]

[1]Department of Computer Science,
Assam University Silchar 788011, India
amritabhattacharjee10@gmail.com

[2]Department of Computer Science,
Assam University Silchar 788011, India
bipulsyam@gmail.com

Abstract: ***The finite-state automaton is one of the most significant tools of computational linguistics. It is divided into two parts Deterministic and Nondeterministic. In this paper description of how automata are used to describe regular languages is given. Some illustrations about finite state automaton and state-transition table and their contribution to represent regular languages are given in this paper. An algorithm to construct deterministic finite state automata is presented. This algorithm gives finite state automaton of a word over regular languages instantly.***

Keywords: Finite state automaton, State transition table, Context free grammar, Regular languages

1. Introduction

The finite-state automaton is one of the most significant tools of computational linguistics. It is a mathematical object that takes a word as input and decides either to accept it or reject it. Since all computational problems are reducible into accept or reject query on words, automata theory plays a vital role in computational theory. The automata are in only one state at a time. They can change from one state to another when initiated by a triggering event or condition, this is called a transition. The automaton is represented as a directed graph known as state graph which consists of a finite set of vertices known as nodes, together with a set of directed links between pairs of vertices called arcs. Vertices are represented by circles and arcs by arrows. We can also represent an automaton with a state-transition table. As in the graph notation, the state-transition table represents the start state, the accepting states, and what transitions leave each state with which symbols.

Finite state automata is divided into two parts one whose behavior during recognition is fully determined by the state it is in and there is one and only one state to which the automaton can transition from its current state is called "Deterministic". In contrast, a "Nondeterministic" finite automaton has the power to be in several states at one time. This ability is often expressed as an ability to guess something about its input.

Finite-state automata have been used in many areas of computational linguistics. Their use can be justified by both linguistic and computational arguments. The usefulness of an automaton for defining a language is that it can express an infinite set in a closed form. Linguistically, finite automata are suitable since they permit one to express without difficulty most of the relevant phenomena encountered in the empirical study of language. They often lead to a compact representation of lexical rules, or idioms and words that appears as natural to linguists. Graphic tools also allow one to visualize and modify automata. This helps in correcting and completing a grammar. From the computational point of view, the use of finite-state machines is mainly motivated by considerations of time and space efficiency. Time efficiency is usually achieved by using deterministic automata. The output of deterministic machines depends, in general linearly, only on the input size and can therefore be considered as optimal from this point of view. Space efficiency is achieved with classical minimization algorithms for deterministic automata.

Finite automata now also constitute an affluent section of theoretical computer science. Many books [1, 2] and research papers [3, 4, 5] are written on the subject. A few examples are cited in this context. Finite automata are used in text processing, compilers, and hardware design. Finite-state automata are very constructive mathematical models of computation used to design both computer programs and sequential logic circuits.

There are numbers of papers which have investigated on the development of algorithms to construct finite state automata. A few examples [6, 7 …11] are cited in this context.

Organization of this paper is as follows. The following section 2 recapitulates the basic preliminaries of Finite State Automata. Section 3 goes toward description of how automata are used to describe regular languages. Section 4, an algorithm to construct deterministic finite state automata is proposed. Section 5 gives result analysis of the above algorithm. We conclude the paper in Section 6 by summarizing the observations.

2. Preliminaries

First of all we recall some definitions.

Regular expression: A regular expression, first developed by Kleene in 1956 is a formula in a language that is used for specifying simple classes of strings. A string is any sequence of alphanumeric characters like letters, numbers, spaces, tabs, and punctuation.

Regular language: The class of languages that are definable by regular expressions are regular languages. Given a finite alphabet Σ, the following constants are defined as regular languages:

i. Empty set: ∅ denoting the set ∅.
ii. Empty string: ε denoting the set containing only the "empty" string, which has no characters at all.
iii. Literal character: "*a*" in Σ denoting the set containing only the character *a*.

The following operations over given regular languages R and S are defined to produce more regular languages:

- Concatenation: *RS* denoting the set {αβ | α in set described by language *R* and β in set described by *S*}. For example $\{abc, d\}\{e, fg\} = \{abce, abcfg, de, dfg\}$.
- Alternation: *R* | *S* denoting the union of sets described by *R* and *S*. For example, if R describes $\{ab, cd\}$ and S describes $\{cd, fg, h\}$, language *R*|*S* describes $\{ab, cd, fg, h\}$.
- Kleene star: This is the set of all strings that can be made by concatenating any finite number (including zero) of strings from set described by *R*. For example, {"0","1"}* is the set of all finite binary strings (including the empty string),and $\{ab, c\}^* = \{\varepsilon, ab, c, abc, abab, cab, cc, ababab, abcab, \cdots\}$

Context-free grammar: A context-free grammar *G* is defined by the 4-tuple: $G = \{V, \Sigma, R, S\}$ where

1. V is a finite set; each element $v \in V$ is called *a non-terminal character* or a *variable*. Each variable represents a different type of phrase or clause in the sentence. Variables are also sometimes called syntactic categories. Each variable defines a sub-language of the language defined by G.
2. Σ is a finite set of *terminal*s, disjoint from V, which make up the actual content of the sentence. The set of terminals is the alphabet of the language defined by the grammar G.
3. R is a finite relation from V to $(V \cup \Sigma)^*$. The members of R are called the *rule*s or *production*s of the grammar. S is the start variable, used to represent the whole sentence (or program). It must be an element of V.

The asterisk represents the Kleene star operation.

Automata theory definition: An automaton is represented formally by a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of *states*.
- Σ is a finite set of symbols, called the alphabet of the automaton.
- δ is the transition function, that is, δ: Q × Σ → Q.
- q_0 is the start state, that is, the state of the automaton before any input has been processed, where $q_0 \in Q$.
- F is a set of states of Q (i.e. $F \subseteq Q$) called accept states.

State transition table: We can also represent an automaton with a state-transition table. Like finite state automata the state-transition table represents the start state, the accepting states, and what transitions leave each state with which symbols.

Input word: An automaton reads a finite string of symbols $a_1, a_2, ..., a_n$, where $a_i \in \Sigma$, which is called an input word. The set of all words is denoted by Σ*.

Run: A run of the automaton on an input word $w = a_1, a_2, ..., a_n \in \Sigma^*$, is a sequence of states $q_0, q_1, q_2, ..., q_n$, where $q_i \in Q$ such that q_0 is the start state and $q_i = \delta(q_{i-1}, a_i)$ for $0 < i \leq n$. In words, at first the automaton is at the start state q_0, and then the automaton read symbols of the input word in sequence. When the automaton reads symbol a_i it jumps to state $q_i = \delta(q_{i-1}, a_i)$. q_n is said to be the final state of the run.

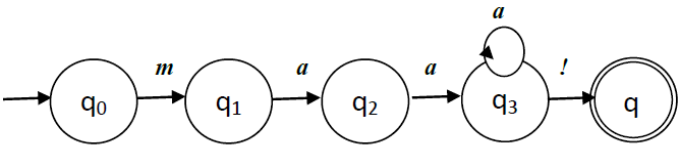
Deterministic & Nondeterministic Finite state automata: Finite state automata is divided into two parts one whose behavior during recognition is fully determined by the state it is in and there is one and only one state to which the automaton can transition from its current state is called "Deterministic". In contrast, a "Nondeterministic" finite automaton has the power to be in several states at one time. This ability is often expressed as an ability to guess something about its input.

3. Use of finite state automata to describe regular languages

A finite state automaton is a very important notion for the study of regular languages. Some illustrations about finite state automaton and state-transition table and their contribution to represent regular languages are given in this section. Using an FSA to recognize the language of a child calling his mother is given below:

We define the language a child calling his mother as any string from the following (infinite) set:

maa!
maaa!
maaaa!
maaaaa!
maaaaaa!
… .



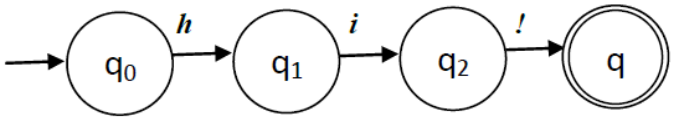
We can also represent an automaton with a state-transition table. As in the graph notation, the state-transition table represents the start state, the accepting states, and what transitions leave each state with which symbols. Here's the state-transition table for the FSA of the above Figure:

	Input		
State	**m**	**a**	**!**
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	3	Ø
3	Ø	3	4
4:	Ø	Ø	Ø

More examples:

Example 1

Let us consider the word 'Hi' A Finite State Automaton for this word is

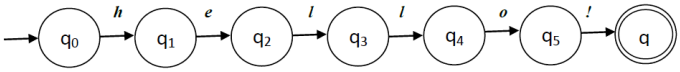


The State transition table is

	Input		
State	**h**	**i**	**!**
0	1	Ø	Ø
1	Ø	2	Ø
2	Ø	Ø	3
3:	Ø	Ø	Ø

Example 2

Let us consider another word 'Hello' A Finite State automaton for this word is



The state Transition table is

	Input				
State	**h**	**e**	**l**	**o**	**!**
0	1	Ø	Ø	Ø	Ø
1	Ø	2	Ø	Ø	Ø
2	Ø	Ø	3	Ø	Ø
3	Ø	Ø	4	Ø	Ø
4	Ø	Ø	Ø	5	Ø
5	Ø	Ø	Ø	Ø	6
6:	Ø	Ø	Ø	Ø	Ø

4. An Algorithm to Construct Deterministic Finite State Automata

The following pseudo code gives instantly the Deterministic Finite State Automata of a string.

Algorithm:

01 Initialize a word = ' w '

02 Set len = length of w

03 ***For*** $i = 0$ to len do

04 Draw an Arrow mark $\longrightarrow$

05 ***If*** $(i = \text{len})$ *// finite state*

06 Draw two circles [one is inner & name it q_i *, another is outer]*

07 ***else***

08 Draw a circle and name it q_i

09 ***If*** $(i \neq 0)$

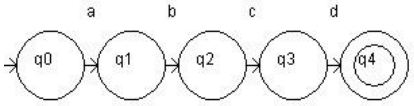
10 Label the Arrow mark with character of the word at i^{th} *position*

11 ***End***

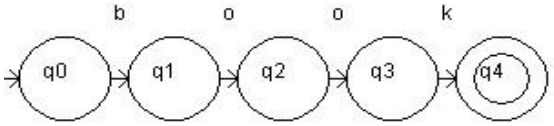
5. Result Analysis

The algorithm introduced gives finite state automaton of a word over regular languages instantly. Initial and final stages are distinct. The stages are named as q_i. The stages are separated by arrow mark. The character of the word at the i^{th} position is marked above the arrow mark. Using the above algorithm finite state automata of some words are given below.

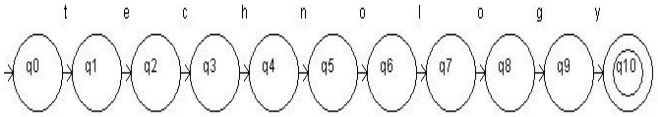
Example (i): The FSA of the word "abcd" is



Example (ii): The FSA of the word "book" is



Example (iii): The FSA of the word "technology" is



6. Conclusion

A finite state automaton is a very important notion for the study of regular languages. In this paper finite state automata and state transition table are discussed in the context of regular languages. An algorithm to construct deterministic finite state automata is introduced. This algorithm gives finite state automaton of a word over regular languages instantly. Initial and final stages are distinct. The stages are named as q_i. The stages are separated by arrow mark. The character of the word at the i^{th} position is marked above the arrow mark. With this algorithm further investigations can be done on the field of FSA. The extension of this algorithm towards non-deterministic finite state automata is an open problem.

References

[1] Daniel Jurafsky and James H. Martin, Speech and Language Processing, Prentice-Hall, 2007 (book style)

[2] J. E. Hopcroft, Rajeev Motwani and J. D. Ullman, "Introduction to Automata Theory Languages and Computation," Pearson Education Publishing Company, Second Edition (book style)

[3] M. O. Rabin, and D. Scott, "Finite automata and their decision problems", IBM Journal of Research and Development, 1959, 3(2), 114–125

[4] Josh Bongard Hod Lipson "Active Co evolutionary Learning of Deterministic Finite Automata" Journal of Machine Learning Research 6 (2005) 1651–1678

[5] Boris Melnikov, Alexandra Melnikova, "A New Algorithm of Constructing the Basis Finite Automaton" Informatica, Institute of Mathematics and Informatics, Vilnius, 2002, Vol. 13, No. 3, 299–310

[6] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, "The Design and Analysis of Computer Algorithms," Addison-Wesley Publishing Company, 1974

[7] Jan Daciuk, Stoyan Mihov, Bruce Watson, and Richard Watson, " Incremental construction of minimal acyclic finite state automata," Computational Linguistics, 26(1):3–16, April 2000

[8] John E. Hopcroft. "An n log n algorithm for minimizing the states in a finite automaton" In Z. Kohavi, editor, The Theory of Machines and Computations, pages189–196. Academic Press, 1971

[9] Dominique Revuz. "Dictionnaires et lexiques: m´ethodes et algorithms" PhD thesis, Institut Blaise Pascal, Paris, France, 1991. LITP 91.44.

[10] Dominique Revuz. "Minimisation of acyclic deterministic automata in linear time" Theoretical Computer Science, 92(1):181–189, 1992

[11] Bruce Watson. A fast new (semi-incremental) algorithm for the construction of minimal acyclic DFAs. In Third Workshop on Implementing Automata, pages 91–98, Rouen, France, September 1998. Lecture Notes in Computer Science, Springer

Author Profile

<Author Photo>

Amrita Bhattacharjee received the M.Sc. degree in Mathematics from Gauhati University in 2009.