International Journal of Computer Applications (0975 – 8887)
Volume 60– No.10, December 2012

To Develop an Efficient Algorithm that Generalize the Method of Design of Finite Automata that Accept "N" base Number such that when "N" is Divided by "M" Leaves Reminder "X"

Danish Ather
Teerthanker Mahaveer Univerisity,
Moradabad, India

Raghuraj Singh
Computer Science & Engineering Department, H.B.T.I.,
Kanpur, India

Vinodani Katiyar
SRCET Computer Science & Engineering Department,
Lucknow, India

ABSTRACT

Theory of computation is always been an issue for the students to understand. So there is a research gap which will ease the process of teaching learning. Our research objective is to develop method to make teaching learning process of theory of computation easier, simpler and understandable. In this paper we develop an algorithm and a tool based on the same algorithm which will generalize the design of finite automata that accept "N" base number such that when "N" is divided by "M" leaves reminder "X" i.e. "X" MOD "M".

KEYWORD DFA, Transition Table, MOD

1 INTRODUCTION

Objective of this paper is to develop an efficient algorithm that generalize the method of design of finite automata that accept "N" base number such that when "N" is divided by "M" leaves reminder "X" i.e. "X" MOD "M". In addition we will design a tool that will simulate the behavior of finite automata accepting "N" base number such that when "N" is divided by "M" leaves reminder "X" i.e. "X" MOD "M"

Where

X is the remainder when "N" base number is divided by M. For Example if N base number is divisible by M then X is 0(Zero).

2 METHODOLOGY

In automata theory, a branch of theoretical computer science, a deterministic finite automaton (DFA)—also known as deterministic finite state machine—is a finite state machine that accepts/rejects finite strings of symbols and only produces a unique computation (or run) of the automaton for each input string.[1] 'Deterministic' refers to the uniqueness of the computation. In search of simplest models to capture the finite state machines, McCulloch and Pitts were among the first researchers to introduce a concept similar to finite automaton in 1943.[2][3]

A DFA is defined as an abstract mathematical concept, but due to the deterministic nature of a DFA, it is implementable in hardware and software for solving various specific problems. For example, a DFA can model software that decides whether or not online user-input such as email addresses are valid.[4]

Finite Automata (M) is defined as a set of five tuples (Q, ∑, δ, Q_0, F)

Where

Q= a finite, non-empty set of states

∑= a finite, non-empty set of inputs

δ is the state-transition function:

δ:QX∑→Q

Q_0 is the initial state

F is the set of final states, a (possibly empty) subset of Q.

δ can be represents using either of three approach given below

- *Transition Graph.*
- *Transition Function.*
- *Transition Table.*

We had used the transition table as the approach to represent δ.

3 ALGORITHM

3.1 N base number allowed digits 0, 1, 2...... N-1 (INPUT SYMBOL)

3.2 If a number is divisible by M then possible remainder values as 0, 1, 2 …M-1(STATES)

3.3 Design a transition table in which columns represent the input symbol and row represent the states of finite automata.

3.4 Total No. of columns =N labeled 0, 1, 2…. N-1 and Total No. of rows =M labeled Q_0 , Q_1 , Q_2 , ….Q_M

3.5 Each cell of Transition Table is represented as: Tij i=0toN-1 and J=0to M-1.

3.6 Fill $T_{ij} = Q_k$ such that:

3.7 For i= 1 to N-1

3.8 do

3.9 For j= 1 to M-1

3.10 do

3.11 $T_{ij} = Q_k$ for k=0to M-1

3.12 if k=m-1 then k=0 else k=k+1

3.13 done inner loop

3.14 done outer loop

3.15 Q_0 being the initial state

3.16 Q_x is the final state where x is the remainder obtained when N base number is divided by M. x ε 0 to M-1.

3.17 DFA will accept a string if all the input is consumed and halting state is the final state.

4 IMPLEMENTATION

4.1 Design a DFA that accept N base number divisible by M.

Let the resultant DFA is M`= (Q`, ∑`, δ`, Q_0`, F`)

Q` = { Q_0 , Q_1 , Q_2 , ….Q_{M-1}}

∑` = {0, 1, 2 …N-1}

δ` is given by

Table 1

Q_0` = {Q_0}

F` = {Q_0} (if a number is divisible by M will yield reminder 0(Zero)).

Table 1 Transition Table of DFA that accept N base number divisible by M.

INPUTS /STATES	0	1	2	…….	N-1
Q_0	Q_0	Q_1	…	Q_{M-1}	…
Q_1	…	Q_0	Q_1	Q_2	…
Q_2	…	…	…	Q_{M-1}	…
…….	Q_0	Q_1	…	…….	…
……		Q_{M-1}	Q_0	Q_1	
Q_{M-1}	…	…	…	…….	Q_{M-1}

4.2 Design a DFA that accept base 8 number divisible by 5.

M= (Q, ∑, δ, Q_0, F)

Q= { Q_0, Q_1 , Q_2 , Q_3 , Q_4 }

∑ = {0, 1, 2, 3, 4, 5, 6, 7}

δ is given by TABLE 2

Q_0 = {Q_0}

F = {Q_0} (if a number is divisible by M will yield reminder 0(Zero)).

TABLE 2 Transition Table of DFA that accept base 8 number divisible by 5.

INPUTS /STATES	0	1	2	3	4	5	6	7
Q_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2
Q_1	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0
Q_2	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3
Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1
Q_4	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4

4.3 Design a DFA that accept decimal number such that when a decimal number is divided by 5 leaves remainder 2 i.e. 2 MOD 5

M= (Q, ∑, δ, Q_0, F)

Q= { Q_0 , Q_1 , Q_2 , Q_3 , Q_4 }

∑ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

δ is given by Table 3

Q_0 = {Q_0}

F = {Q_2}

INPUTS /STATES	0	1	2	3	4	5	6	7	8	9
Q_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4
Q_1	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4
Q_2	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4
Q_3	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4
Q_4	Q_0	Q_1	Q_2	Q_3	Q_4	Q_0	Q_1	Q_2	Q_3	Q_4

TABLE 3 Transition Table of DFA that accept decimal number such that when a decimal number is divided by 5 leaves remainder 2

4.4 Design a DFA that accept base 16 number divisible by 6.

M= (Q, ∑, δ, Q_0, F)

Q= { Q_0 , Q_1 , Q_2 , Q_3 , Q_4 , Q_5}

∑ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

δ is given by TABLE 4

Q_0 = {Q_0}

F = {Q_0} (if a number is divisible by 6 will yield reminder 0(Zero)).

TABLE 4 Transition Table of DFA that accept base 16 number divisible by 6.

INPUTS /STATES	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Q_0	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3
Q_1	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1
Q_2	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5
Q_3	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3
Q_4	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1
Q_5	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5

4.5 Design a DFA that accept Binary number such that when a Binary number is divided by 3 leaves remainder 2 i.e. 2 MOD 3

M= (Q, ∑, δ, Q_0, F)

Q= { Q_0 , Q_1 , Q_2 , Q_3 , Q_4 }

∑ = {0, 1}

δ is given by TABLE 5

Q_0 = {Q_0}

F = {Q_2}

TABLE 5 Transition Table of DFA that accept Binary number such that when a Binary number is divided by 3 leaves remainder 2.

INPUTS /STATES	0	1
Q_0	Q_0	Q_1
Q_1	Q_2	Q_0
Q_2	Q_1	Q_2

5 CONCLUSION

In this paper we have developed an efficient algorithm which will generalize the design of finite automata that accept "N" base number such that when "N" is divided by "M" leaves reminder "X" i.e. "X" MOD "M". Presented algorithm will help students in better understanding of the design of the finite automata that accept "N" base number such that when "N" is divided by "M" leaves reminder "X". But the algorithm works with a limitation for unary base Number.

ACKNOWLEDGMENT

This research is a part of my research entitled "Optimized simplified machine design model of computational research". We also thank R. K. Dwivedi for his help and for the discussions on some of the topics in this paper.

REFERENCES

[1.] Hopcroft, John E.; Motwani, Rajeev; Ullman, Jeffrey D. (2001). Introduction to Automata Theory, Languages, and Computation (2 ed.). Addison Wesley. ISBN 0-201-44124-1. Retrieved 19 November 2012.

[2.] Lawson, Mark V. (2004). Finite automata. Chapman and Hall/CRC. ISBN 1-58488-255-7. Zbl 1086.68074.

[3.] McCulloch, W. S.; Pitts, E. (1943). "A logical calculus of the ideas imminent in nervous activity". Bulletin of Mathematical Biophysics: 541–544.

[4.] Rabin, M. O.; Scott, D. (1959). "Finite automata and their decision problems.". IBM J. Res. Develop.: 114–125.

[5.] Sakarovitch, Jacques (2009). Elements of automata theory. Translated from the French by Reuben Thomas. Cambridge: Cambridge University Press. ISBN 978-0-521-84425-3. Zbl 1188.68177.

[6.] Sipser, Michael (1997). Introduction to the Theory of Computation. Boston: PWS. ISBN 0-534-94728-X.. Section 1.1: Finite Automata, pp. 31–47. Subsection "Decidable Problems Concerning Regular Languages" of section 4.1: Decidable Languages, pp. 152–155.4.4 DFA can accept only regular language

[7.] C. Allauzen and M. Mohri, Finitely subsequential transducers, International J. Foundations Comp. Sci. 14 (2003), 983-994

[8.] M.-P. Béal and O. Carton, Determinization of transducers over finite and infinite words, Theoret. Comput. Sci. 289 (2002), 225-251

[9.] Bruggemann-Klein, Regular expressions into finite automata, Lecture Notes in Computer Science 583 (1992), 87-98

[10.] J. Carroll and D. Long, Theory of Finite Automata, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[11.] M. V. Lawson, Finite Automata, CRC Press, Boca Raton, FL, 2003.

[12.] D. Perrin, Finite automata, in Handbook of Theoretical Computer Science, Volume B (editor J. Van Leeuwen), Elsevier, Amsterdam, 1990, 1-57.

[13.] D. Perrin and J. E. Pin, Infinite Words, Elsevier, Amsterdam, 2004.

[14.] Ch. Reutenauer, Subsequential functions: characterizations, minimization, examples, Lecture Notes in Computer Science 464 (1990), 62-79.

[15.] J. Sakarovitch, Kleene's theorem revisited, Lecture Notes in Computer Science 281(1987), 39-50.

[16.] E. Roche and Y. Schabes (editors), Finite-State Language Processing, The MIT Press, 1997.

[17.] B. W. Watson, Implementing and using finite automata toolkits, in Extended Finite State Models of Language (editor A. Kornai), Cambridge University Press, London, 1999, 19-36.