NFA to DFA Converter

Preeti Bailke, Tejas Adsare, Tejas Dharmik, Neeraj Agrawal, Dhananjay Deore, Shubhankar Gupta

Vishwakarma Institute of Technology, Pune-411037, Maharashtra, India

Abstract *- The Theory of Computation is the branch of computer science and mathematics that deals with whether and how efficiently problems can be solved on a model of computation using an algorithm. In theoretical computer science, automata theory is the study of abstract machines and the computational problems that can be solved using these abstract machines. These abstract machines are called automata. Finite automata can be deterministic and non-deterministic. Every regular language that is described by non-deterministic finite automata can also be described by deterministic finite automata. So, in this paper we described about how we can convert a non-deterministic finite automaton (NFA) into deterministic finite automata (DFA).*

Keywords - DFA, Finite Automata, NFA, Transition Table, Pandas, Python.

I. INTRODUCTION

Automata theory's modern uses go well beyond compiler algorithms and hardware verification. Automata are frequently used in software, distributed systems, real-time systems, and structured data modelling and verification. They also contain qualities that allow them to model time and probabilities. Computer programs generally need to know all possible transitions and states for a given state machine. A non-deterministic finite automaton can have a transition that goes to any number of states for a given input and state. This is a problem for a computer program because it needs precisely one transition for a given input from a given state. The process of converting NFA to DFA eliminates this ambiguity and allows a program to be made.

NFA also suffers from combinatorial explosion. Reducing to a DFA can reduce state and transitions.

II. LITERATURE REVIEW

Various methodologies and ideas had been proposed to categorize the behavioral and emotions of the user.

1. M. Davoudi - Monfared, et al contributed in “Converting an NFA to a DFA with programming C++”. In this paper, they used engrafted lists dynamically and define some functions in different classes and grafted list is used in classes. In grafter list, there is description about the structure and the nodes type. Then ceases before and after nodes are evaluated. The final result is shown in array.

2. Indu and Jyoti contributed in “Technique for Conversion of Regular Expression to and from Finite Automata”. In this paper it provides an insight into the various approaches used for conversion of DFA to RE and vice versa.The most time-consuming partof the project was coding the parser for the regular expression. This is because while regular expressions define regular languages, they themselves are not regular and must be described by context-free grammars.

3. Ho Ngoc Vinh, et al contributed in “Nfa To Dfa Conversion: A New Approach Using Languages Of Bounded Words” In this paper, concepts of bounded words, ◊-languages and monoid ◊-morphisms are introduced. These allow defining an extension of the automaton on the set of bounded words. Hence, a new perspective of mathematical model of sets of states, edges, languages recognized by the finite automaton according to the length of languages is given and a new checking algorithm is proposed.

III. Basic Terms And Definitions

1.Deterministic Finite Automaton (DFA) :

A Deterministic Finite Automaton (DFA) is defined as a 5-tuple (Q, Σ, δ, s, F) consisting of. A finite set Q (the set of states) A finite set of symbols Σ (the input alphabet) A transition function δ: Q × Σ → Q mapping the current state q ∈ Q and input symbol a ∈ Σ to a new state δ(q, a) ∈ Q.

2. **Non-deterministic finite automata** (NFA) :

A Non-deterministic Finite Automaton(NFA) is also defined as 5-tuple (Q, Σ, δ, s, F) consisting of. A finite set Q (the set of states) A finite set of symbols Σ (the input alphabet) A transition function δ: Q × Σ → Q mapping the current state q ∈ Q and input symbol a ∈ Σ to a new state δ(q, a) ∈ Q.

It is easy to construct an NFA than DFA for a given regular language. The finite automata are called NFA when there exist many paths for specific input from the current state to the next state.

3. **Transition Table :**

Transition table is basically a tabular representation of the transition function. It takes two arguments a state and a symbol and returns a state i.e. the next state.
A transition table is represented by the following things:

1. Columns correspond to input symbols.
2. Rows correspond to states.
3. Entries correspond to the next state.
4. The start state is denoted by an arrow with no source.
5. The accept state is denoted by a star.

IV. Methodology/Experimental

The methodology of this proposed project includes two steps. First is implementation of NFA which is taken as input from user and secondly, conversion of NFA to DFA.

A. Implementation of NFA as Input

The system basically accepts the number of states ,no of input symbols (e.x (a,b),(0,1))from the user For each state the system takes the multiple paths.
If transition ends at more place we used input().split () for accepting more states at once.
Similarly, we have taken the final states for the given NFA by user .
For showing the NFA table we have basically imported the pandas module.

B. Conversion Of NFA To DFA

We have implemented a python language for conversion of nfa to dfa. An NFA can have zero, one or more than one move from a given state on a given input symbol. An NFA can also have NULL moves (moves without input symbol). On the other hand, DFA has one and only one move from a given state on a given input symbol.

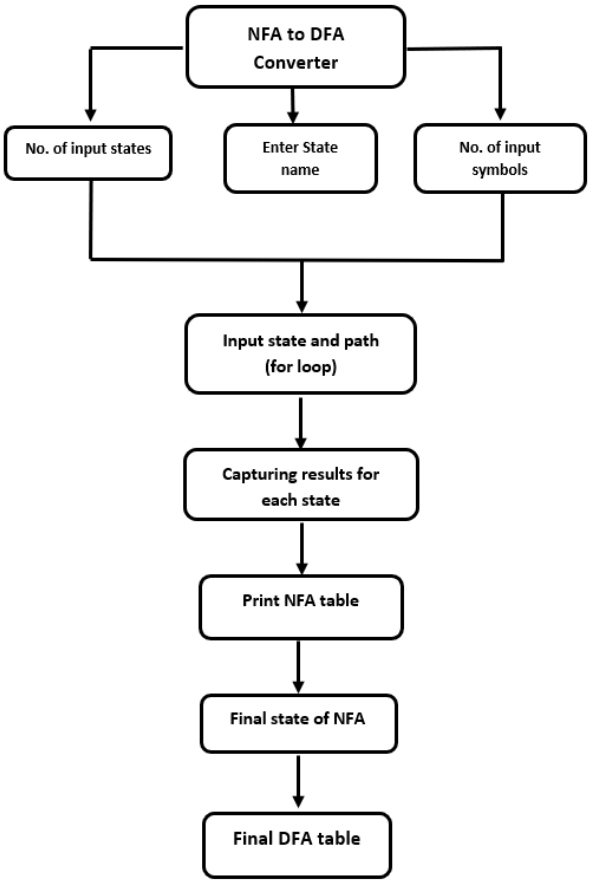

Fig. 1. Flowchart of NFA to DFA Converter

Steps for coversion:
Suppose there is an NFA N < Q, ∑, q0, δ, F > which recognizes a language L. Then the DFA D < Q', ∑, q0, δ', F' > can be constructed for language L as:

Step 1: Initially Q' = ϕ.
Step 2: Add q0 to Q'.
Step 3: For each state in Q', find the possible set of states for each input symbol using transition function of NFA. If this set of states is not in Q', add it to Q'.
Step 4: Final state of DFA will be all states with contain F (final states of NFA)

We have created an empty dictionary for key value pair for each state of NFA and DFA. After taking the inputs such as no. of states, their names and no. of input symbols. Using pandas which is a python library helps to represent the data in proper format table.

Firstly, for creating a dictionary of DFA it accepts the new states from NFA table.
The DFA ends until any new states left in NFA after converting. It's basically check for each new state and compare with the key value of dictionary If no new states are left it prints the DFA table in output and for that ,we have imported the pandas module.

After that it displays the final states of DFA by checking the final states of NFA table. By displaying the final states of DFA our program terminates.

V. Results and Discussions

We have successfully converted NFA to its corresponding DFA and also construct the transition table of NFA and DFA in the output by taking states and symbols from the user, as mentioned below in following points.

Let us consider one example-

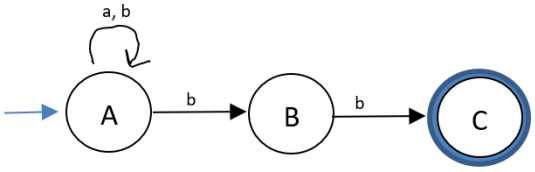

Fig. 2. Example for NFA to DFA conversion

Results-

1. Taking no. of states and input symbols from the user -

```
Enter the following inputs to form NFA table :


No. of states : 3
No. of input symbols : 2
```

Fig. 3. Example for NFA to DFA conversion

2. Printing NFA table which will acts as our input-

```
Printing NFA table :-
     a       b
A  [A]  [A, B]
B   []     [C]
C   []      []
```

Fig. 4. NFA table as input

3. Then by entering the state name, path and end state we will get the final DFA table and its final states-

```
Printing DFA table :-
     a    b
A    A   AB
AB   A  ABC
ABC  A  ABC
```

Fig. 5. Final DFA table

```
Final states of the DFA are :  ['ABC']
```

Fig. 6. Final states of DFA

VI. Scope of Project

For future work we try to simulate Turing machine. The Turing machine works as a computer and can solve many computational algorithms. DFA and NFA are weak than the Turing machine. If it is possible, then any DFA and NFA simulate simply.

VII. Conclusion

This paper work provides an insight into the approach used for conversion of non-deterministic finite automata to deterministic finite automata. For any given number of states and input symbols we are able to construct a NFA which converts to DFA. This program can answers for any NFA included large NFA, simple NFA without empty alphabet and etc.

ACKNOWLEDGMENT

It would be our utmost pleasure to express our sincere thanks to our guide Prof. Preeti Bailke who gave us the opportunity to do this project on the topic “NFA to DFA Converter”, which also helped us in doing a plenty of Research and that we came to understand about such a lot of new things. As well as we would like to thanks friends and seniors who helped us in completing this project.

REFERENCES

[1] M. Davoudi-Monfared, R. shafiezadehgarousi, E. S. Haghi , S. Zeinali , S.Mohebali, “Converting an NFA to a DFA with programming C++” presented in International Journal of Advanced Computer Research ISSN ,Volume-5 ,21 December-2015

[2] Indu, Jyoti, “Technique for Conversion of Regular Expression to and from Finite Automata”, Indu et al. International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 3, Issue 2, June 2016, pp. 62-64

[3] M. Sipser, “An Introduction to the Theory of Computation”, Second Edition, Thomson Course Tecnology, 2006.

[4] P. Linz, “An Introduction to formal languages and Automata", Second Edition, D. C. Heath and Company, 1996.

[5] K. Salomma, S. Yu, NFA to DFA transformation for finite languages, Lecture Notes in Computer Science,1260, 1997, 149-158.

[6] Ho Ngoc Vinh1 and Nguyen Thi Thu Ha, “Nfa To Dfa Conversion: A New Approach Using Languages Of Bounded Words”, ICIC International 2018 ISSN 1881-803X.