KARNATAK LAW SOCIETY'S

GOGTE INSTITUTE OF TECHNOLOGY

UDYAMBAG, BELGAUM-590008

(An Autonomous Institution under Visvesvaraya Technological University, Belgaum)

(APPROVED BY AICTE, NEW DELHI)

Course Activity Report

Subset construction by Lazy evaluation scheme to convert NFA to DFA

Submitted in the partial fulfillment for the academic requirement of

FIFTH Semester B.E.

in

FORMAL LANGUAGES AND AUTOMATA THEORY

Submitted by :

Batch Number-5

1)	Vivek R Jadhav	2GI19CS185
2)	Venkatesh G Dhongadi	2GI19CS175
3)	Vinayak S Bhandage	2GI19CS179
4)	Suyash Rawool	2GI19CS161

Under the Guidance of :

Prof. Sudha V.S

Department of Computer Science and Engineering

KLS GIT Belgaum.

2021

CERTIFICATE

This is to certify that the Seminar entitled "Subset construction by Lazy evaluation scheme to convert NFA to DFA" is a bona fide record of the Seminar work done by ***Vivek R Jadhav*** *(2GI19CS185),* ***Venkatesh G Dhongadi****(2GI19C175),* ***Vinayak S Bhandage****(2GI19CS179) and* ***Suyash Rawool*** *(2GI19CS161) under my supervision and guidance, in partial fulfillment of the requirements for the Outcome Based Education Paradigm in Computer science and Engineering from Gogte Institute of Technology for the academic year 2020-21*

Mrs. Sudha V.S
Asst. Professor

Dr. Vijay S. Rajpurohit
Professor & Head
Dept. of Computer Science and Engineering

Place: KLS Gogte Institute of Technology Belgaum.

Date: 20-01-2022

COURSE REPORT

Marks allocation:

	Batch No. : 5					
1.	Title: Subset Construction By Lazy Evaluation To Convert NFA To DFA	Marks Range	USN			
			2GI19CS161	2GI19CS175	2GI19CS179	2GI19CS161
2.	Abstract (PO2)	0-2				
3.	Application of topic to the course (PO2)	0-3				
4.	Literature survey and its findings (PO2)	0-4				
5.	Methodology, Results and Conclusion (PO1,PO3,PO4)	0-6				
6.	Report and Oral presentation skill (PO9 PO10)	0-5				
	Total	20				

*** 20 marks is converted to 10 marks for CGPA calculation**

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

2. Problem Analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and Engineering sciences.

3. Design/Development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. **Individual and team work:** Function effectively as an individual and as a member or leader in diverse teams, and in multidisciplinary settings.

10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

11. **Project management and finance:** Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

12. **Life-long learning:** Recognize the need for and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

ACKNOWLEDGEMENT

This group feels greatly indebted to Computer Science and Engineering Department, for the opportunity given us to undertake this "Subset construction by Lazy evaluation scheme to convert NFA to DFA" Project. This Project includes thoughts and contribution of many individuals. And we wish to express our sincere appreciation and gratitude to them.

First and foremost, we want to extend entirely our gratitude to our lecturer Mrs. Sudha V.S for sharing her knowledge and profound wisdom with us. We appreciate all her comments and suggestions, which are incorporated into this project.

We would also like to express our gratitude towards and group members. Without their help, support, and encouragement, this project would never had been completed.

In our respect, this project is an outcome of the learning experience we have Shared with our fellow students. We dedicate this Project to all our fellow engineering students.

VIVEK R JADHAV
VENKATESH DHONGADI
VINAYAK BHANDAGE
SUYASH RAWOOL

CONTENTS

ABSTRACT

This report consists a detailed study on Subset construction by Lazy evaluation scheme to convert NFA to DFA.
This report initially deals with a brief introduction to the method of the construction.
It deals with an in-depth construction of dfa with an example.
The report gives the advantages of the lazy evaluation method over subset construction.

INTRODUCTION

DFAs and NFAs represent the same class of languages: the regular languages. This means that each language we can represent using an NFA can also be represented using a DFA and vice versa. All deterministic automata are trivially nondeterministic (the transition function only returns singleton sets of states), but showing that the equivalence holds the other way around is somewhat trickier.

Lazy evaluation is an evaluation strategy which holds the evaluation of an expression until its value is needed. It avoids repeated evaluation. **Haskell** is a good example of such a functional programming language whose fundamentals are based on Lazy Evaluation.

Lazy evaluation is used in Unix map functions to improve their performance by loading only required pages from the disk. No memory will be allocated for the remaining pages.

METHOD

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

Conversion from NFA to DFA
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' = ɸ.
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)

Example
Consider the following NFA shown in Figure 1.

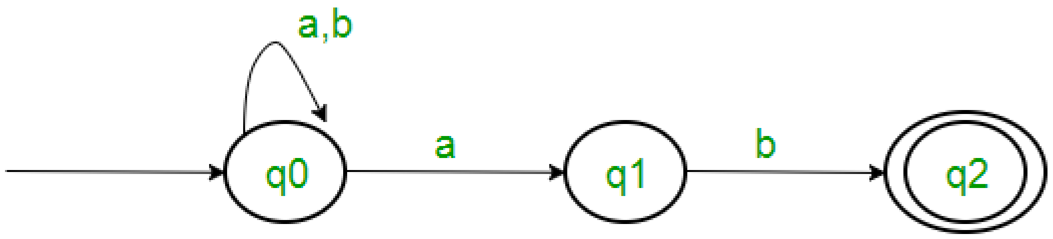
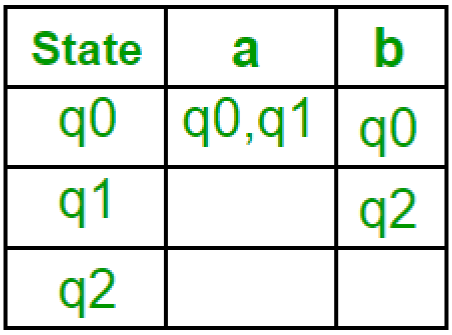


Figure 1

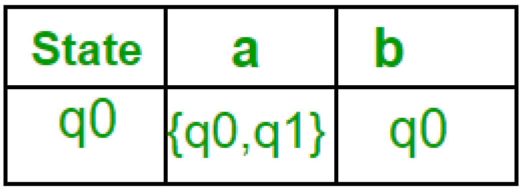
Following are the various parameters for NFA.
Q = { q0, q1, q2 }
∑ = (a, b)
F = { q2 }
δ (Transition Function of NFA)



State	a	b
q0	q0,q1	q0
q1		q2
q2		

Step 1: Q’ = ϕ
Step 2: Q’ = {q0}
Step 3: For each state in Q’, find the states for each input symbol.
Currently, state in Q’ is q0, find moves from q0 on input symbol a and b using transition function of NFA and update the transition table of DFA.

δ’ (Transition Function of DFA)

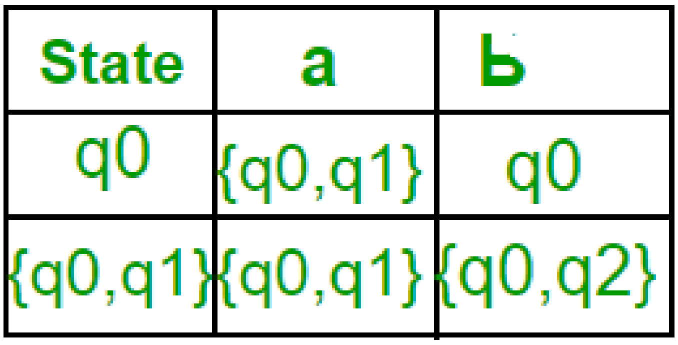


State	a	b
q0	{q0,q1}	q0

Now { q0, q1 } will be considered as a single state. As its entry is not in Q’, add it to Q’.
So Q’ = { q0, { q0, q1 } }

Now, moves from state { q0, q1 } on different input symbols are not present in transition table of DFA, we will calculate it like:
δ’ ({ q0, q1 }, a) = δ (q0, a) ∪ δ (q1, a) = { q0, q1 }
δ’ ({ q0, q1 }, b) = δ (q0, b) ∪ δ (q1, b) = { q0, q2 }
Now we will update the transition table of DFA.

δ’ (Transition Function of DFA)

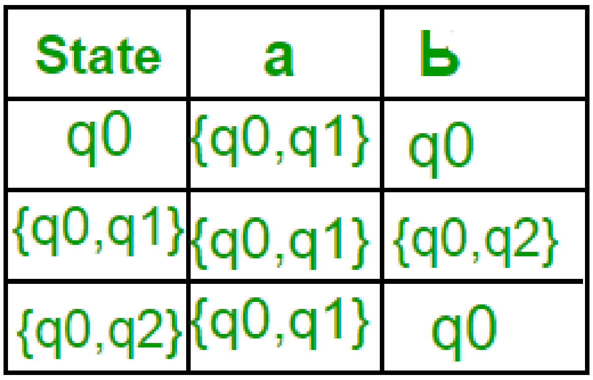


State	a	b
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}

Now { q0, q2 } will be considered as a single state. As its entry is not in Q’, add it to Q’.
So Q’ = { q0, { q0, q1 }, { q0, q2 } }

Now, moves from state {q0, q2} on different input symbols are not present in transition table of DFA, we will calculate it like:
δ’ ({ q0, q2 }, a) = δ (q0, a) ∪ δ (q2, a) = { q0, q1 }
δ’ ({ q0, q2 }, b) = δ (q0, b) ∪ δ (q2, b) = { q0 }
Now we will update the transition table of DFA.

δ' (Transition Function of DFA)



State	a	b
q0	{q0,q1}	q0
{q0,q1}	{q0,q1}	{q0,q2}
{q0,q2}	{q0,q1}	q0

As there is no new state generated, we are done with the conversion. Final state of DFA will be state which has q2 as its component i.e., { q0, q2 }

Following are the various parameters for DFA.
Q' = { q0, { q0, q1 }, { q0, q2 } }
∑ = (a, b)
F = { { q0, q2 } } and transition function δ' as shown above. The final DFA for above NFA has been shown in Figure 2.

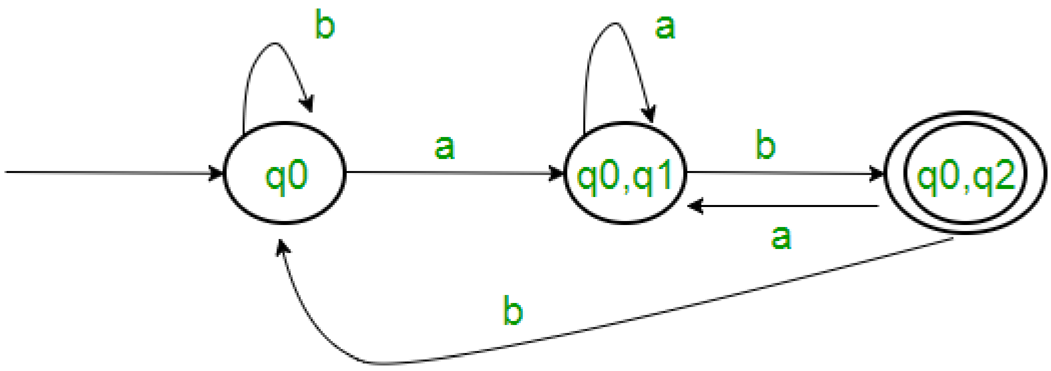


Figure 2

Note : Sometimes, it is not easy to convert regular expression to DFA. First you can convert regular expression to NFA and then NFA to DFA.

Question : The number of states in the DFA corresponding to the regular expression (0+1)*(10) are _____.

Solution : First, we will make an NFA for the above expression. To make an NFA for (0 + 1)*, NFA will be in same state q0 on input symbol 0 or 1. Then for concatenation, we will add two moves (q0 to q1 for 1 and q1 to q2 for 0) as shown in Figure 3.

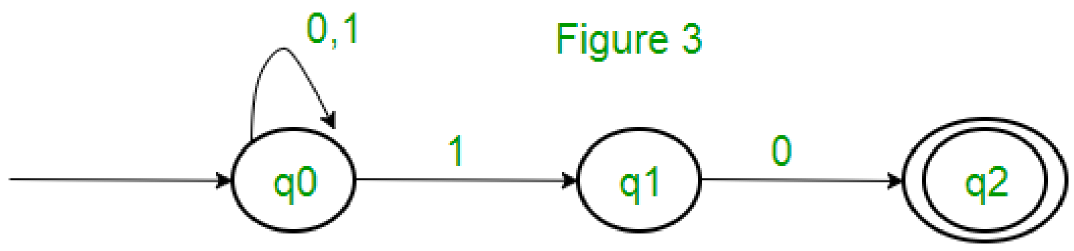
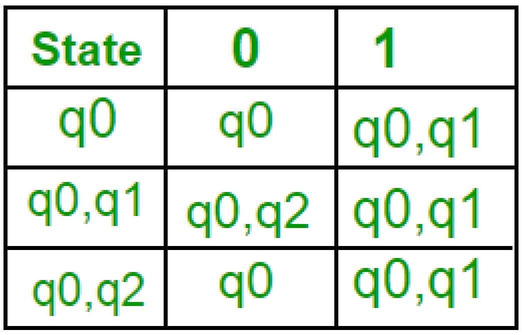
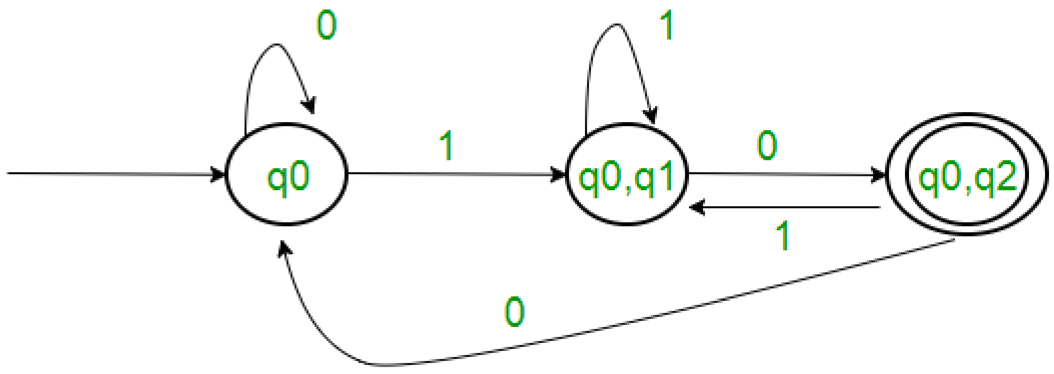


Figure 3



State	0	1
q0	q0	q0,q1
q0,q1	q0,q2	q0,q1
q0,q2	q0	q0,q1



Hence, the number of states in the minimal deterministic finite automaton corresponding to the regular expression (0 + 1)* (10) is 3

ADVANTAGES

- It allows the language runtime to discard sub-expressions that are not directly linked to the final result of the expression.
- It reduces the time complexity of an algorithm by discarding the temporary computations and conditionals.
- It allows the programmer to access components of data structures out-of-order after initializing them, as long as they are free from any circular dependencies.
- It is best suited for loading data which will be infrequently accessed.
- It has the ability to define control flow (structures) as abstractions instead of primitives.
- It has the ability to define potentially infinite data structures. This allows for more straightforward implementation of some algorithms.

DRAWBACKS

- It forces the language runtime to hold the evaluation of sub-expressions until it is required in the final result by creating **thunks** (delayed objects).
- Sometimes it increases space complexity of an algorithm.
- It is very difficult to find its performance because it contains thunks of expressions before their execution.

CONCLUSION

Lazy evaluation is much more efficient than subset construction in terms of both number of states in final DFA and also number of steps required to convert NFA to DFA.

REFERENCES

Introduction to Automata Theory, Languages, and Computation, 3rd edition, by John E.Hopcroft, Rajeev Motwani and Jeffrey D. Ullman.

M. O. Rabin and D. Scott, "Finite Automata and their Decision Problems", *IBM Journal of Research and Development*, **3**:2 (1959) pp. 115–125.

Michael Sipser, *Introduction to the Theory of Computation*. PWS, Boston. 1997. ISBN 0-534-94728-X.