NANDHA COLLEGE OF TECHNOLOGY

ERODE- 638 052

DEPARTMENT OF INFORMATION TECHNOLOGY

CS345 – THEORY OF COMPUTATION

(Regulations 2021)

FOURTH SEMESTER

(ACADEMIC YEAR 2022-23)

ASSIGNMENT / CASE STUDY REPORT – I

PROBLEM: CONVERT ε -NFA INTO NFA

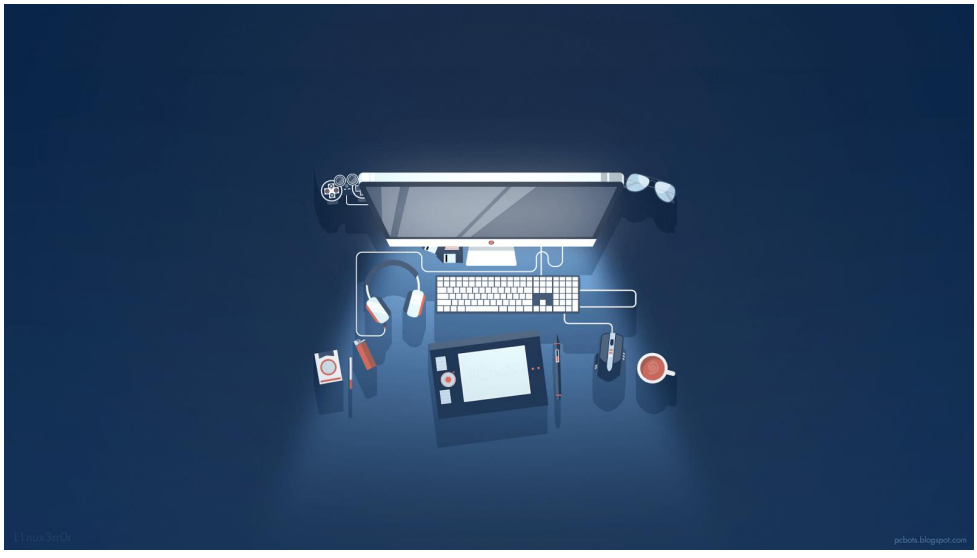
REGISTER NUMBER	**732121205016**
NAME OF THE STUDENT	**C.K.DHARUN**
SUBMITTED ON	
MARKS OBTAINED	
STAFF SIGN WITH DATE	

PROBLEM: CONVERT

ε -NFA INTO NFA

CONTENT:

1. **Introduction**
2. **Question**
3. **NFA(Non-Deterministic finite automata)**
4. **ε -NFA(Non-Deterministic finite automata)**
5. **Problem solution**
6. **Conclusion**

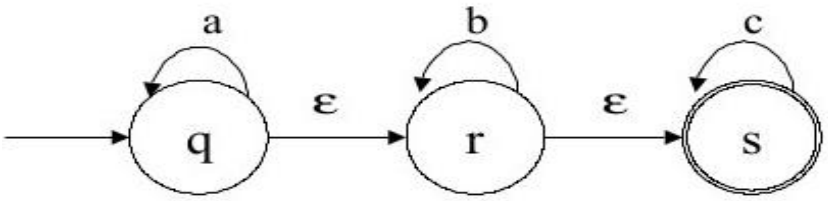


1.Introduction:

ε-NFA (Nondeterministic Finite Automaton with ε transitions) is a variant of NFA that allows for empty or ε transitions, which can move the automaton from one state to another without consuming any input symbol. While ε-NFA is a powerful tool for describing regular languages, it is often necessary to convert it to a standard NFA (Non-deterministic Finite Automaton) to make it more suitable for implementation. However, the conversion process from ε-NFA to NFA is not straightforward, and it poses several challenges and problems that must be addressed. In this context, this article will provide an overview of some of the common problems encountered during the conversion process and how they can be addressed to obtain an equivalent NFA

2. Question:

4.Convert ε-NFA into NFA:



3.NFA(Non-Deterministic finite automata):

- NFA stands for non-deterministic finite automata. 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.
- Every NFA is not DFA, but each NFA can be translated into DFA.
- NFA is defined in the same way as DFA but with the following two exceptions, it contains multiple next states, and it contains ε transition.

4. ε-NFA(Non-Deterministic finite automata):

An epsilon nondeterministic finite automaton (NFA) has **null or epsilon transitions from one state to another**. Epsilon NFA is also called a null NFA or an NFA lambda. A regular expression for a language forms an epsilon NFA. This epsilon NFA then converts to a simple NFA.

Null Closure Method

Suppose an NFA$<Q,\Sigma,q0,\delta, F>$ and $S \subseteq Q$ is a defined set of states. where

- Q is the finite set of states.
- Σ is the input symbols.
- $q0$ is the start state.
- δ is the transition function.
- F is the final state.

The null closure of S will be the set $\Lambda(S)$ and can be defined recursively as follows.

1. $S \subseteq \Lambda(S)$
2. For every $q \in \Lambda(S)$, $\delta(q,\Lambda) \subseteq \Lambda(S)$

To convert an epsilon NFA to NFA, the null closure method makes use of the following general set of rules.

i. Find the null closures for each state.
ii. For each state, check for the transitions by the null closures obtained, the given input, and then the null closures again. This eliminates any potential null closures that can occur.
iii. Make an NFA by the transitions obtained in the previous step. The final state will be all those states that have F in them.

5.**Problem Solution:**

Step1: We will first obtain ε-closure of each state i.e. we will find out ε-reachable states from current state.

Hence

- ε-closure(q)={q,r,s}
- ε-closure(r)={r,s}
- ε-closure(s)={s}

As ε-closure(q) means with null input we can reach to q,r or s.In a similar manner for r and s ε-closure are obtained.

Step 2: Now we will obtain δ transitions for each state on each input symbol.

$\delta(q,a) = \varepsilon\text{-closure}(\delta(\delta(q,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(q),a))$

$=\varepsilon\text{-closure}(\delta(q,r,s),a)$

$=\varepsilon\text{-closure}(\delta(q,a)\cup\delta(r,a)\cup\delta(s,a))$

$=\varepsilon\text{-closure}(q\cup\phi\cup\phi)$

$=\varepsilon\text{-closure}(q)=\{q,r,s\}$

$\delta(q,b) = \varepsilon\text{-closure}(\delta(\delta(q,\varepsilon),b))$

$= \varepsilon\text{-closure}(\delta(q,r,s,b)$

$= \varepsilon\text{-closure}(\delta(q,b)\cup\delta(r,b)\cup\delta(s,b)$

$=\varepsilon\text{-closure}(\phi\cup r\cup\phi)$

$=\varepsilon\text{-closure}(r)$

$\delta(q,b)=\{r,s\}$

$\delta(r,a)= \varepsilon\text{-closure}(\delta(\delta(r,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(r),a))$

$= \varepsilon\text{-closure}(\delta(r,s),a)$

$=\varepsilon\text{-closure}(\delta(r,a) \cup \delta(s,a))$

$= \varepsilon\text{-closure}(\phi \cup \phi)$

$= \varepsilon\text{-closure}(\phi)$

$=\phi$

$\delta(r,b)= \varepsilon\text{-closure}(\delta(\delta(r,\varepsilon),b))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(r),b))$

$= \varepsilon\text{-closure}(\delta(r,s),b)$

$= \varepsilon\text{-closure}(\delta(r,b) \cup \delta(s,b))$

$=\varepsilon\text{-closure}(r \cup \phi)$

$= \varepsilon\text{-closure}(r)$

$=\{r,s\}$

$\delta(s,a)= \varepsilon\text{-closure}(\delta(\delta(s,\varepsilon),a))$

$= \varepsilon\text{-closure}(\delta(\varepsilon\text{-closure}(s),a))$

$= \varepsilon\text{-closure}(\delta(s,a))$

$= \varepsilon\text{-closure}(\phi)$

$=\phi$

$\delta(s,b)= \varepsilon\text{-closure}(\delta(\delta(s,\varepsilon),b))$

= ε-closure(δ(ε-closure(s),b))

= ε-closure(δ(s,b))

= ε-closure(ϕ)

δ(r,b)=ϕ

δ(q,c)= ε-closure(δ(δ(q,ε),c))

= ε-closure(δ(ε-closure(q),c))

= ε-closure(δ(q,r,s),c))

= ε-closure(δ(q,c)∪δ(r,c)∪δ(s,c))

= ε-closure(ϕ∪ϕ∪s)

= ε-closure(s)

={s}

δ(r,c)= ε-closure(δ(δ(r,ε),c))

= ε-closure(δ(ε-closure(r),c))

= ε-closure(δ(r,s),c)

= ε-closure(δ(r,c)∪δ(s,c))

= ε-closure(ϕ∪s)

={s}

δ(s,c)= ε-closure(δ(δ(s,ε),c))

= ε-closure(δ(ε-closure(s),c))

= ε-closure(δ(s,c))

= ε-closure(s)

={s}

Now we will summarize all the computed δ transitions

δ(q.a)={q,r,s}, δ(q,b)={r,s}, δ(q,c)={s}

δ(r,a)=ϕ δ(r,b)={r,s}, δ(r,c)={s}

δ(s,a)=ϕ δ(s,b)=ϕ δ(s,c)={s}

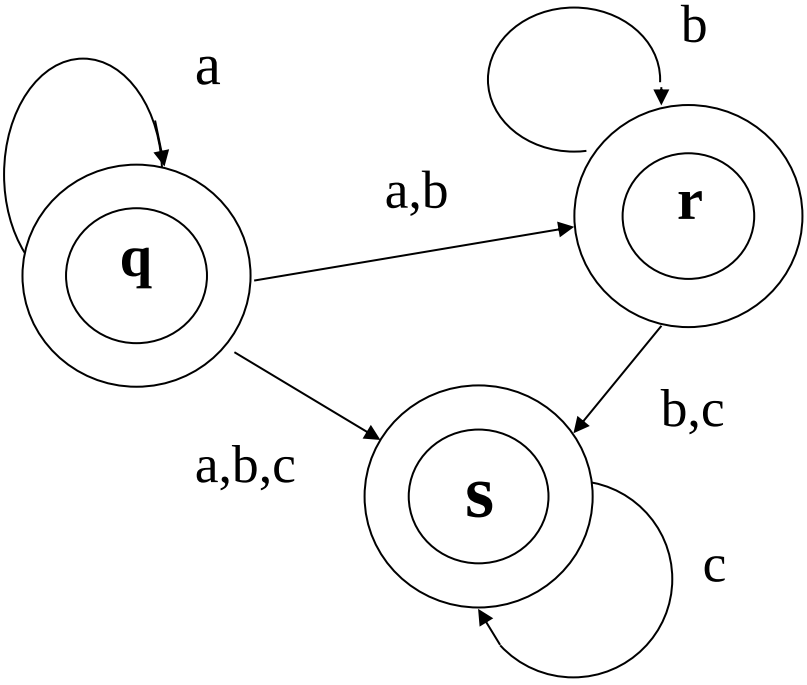
Step3:

From this we can write the transition table as:

State/Input	A	b	c
q	{q,r,s}	{r,s}	{s}
r	Φ	{r,s}	{s}
s	ϕ	ϕ	{s}

Step 4: The NFA will be

,



5.Conclusion:

In conclusion, converting ε-NFA to NFA is an important process in automata theory and has several practical applications in computer science, including pattern matching, lexical analysis, and parsing. However, the conversion process is not always straightforward, and several challenges and problems can arise during the conversion process. These include the generation of multiple transitions for each input symbol, the creation of new states to handle empty transitions, and the need to eliminate inaccessible states and unproductive states. To address these challenges and obtain an equivalent NFA, several techniques can be used, including the subset construction method, which involves generating a new NFA based on the powerset of the ε-NFA states. By understanding the common problems and solutions involved in the conversion process, researchers and practitioners can effectively implement regular expressions and other formal languages in software systems, contributing to the development of efficient and robust applications in various domains