REPORT ON TOC PROJECT

BTech/ II Year CSE/ IV Semester

19CSE214/ Theory of Computation

Final Project Review (Report)

Roll No	Name
CB.EN.U4CSE21628	KAKOLLU KUSUMA
CB.EN.U4CSE2146	PHD SURYA SHANMUK
CB.EN.U4CSE21651	HASANTHI RAYABARAM
CB.EN.U4CSE21669	YEJNAKSHARI MEGHANA K
CB.EN.U4CSE21670	YELLINA SRIBHARGAV

Problem statement:

Design a finite state machine for a vending machine that sells beverages.Design the state transitions and define the states necessary to implement the behaviour of the vending machine Consider various user interactions, potential errors, and the appropriate feedback/messages to provide a smooth and user-friendly experience.

Approach:

Initially, the vending machine is in the idle state, waiting for user interaction.

When a user approaches the vending machine, it transitions to the welcome state

In the welcome state, the vending machine presents the available beverage options to the user, such as coffee, tea, milk, or hot chocolate.

The user can select one of the available beverage options.

After the user selects a beverage, the vending machine transitions to the payment state.

In the payment state, the vending machine provides payment options to the user, such as cash or UPI.

If the user selects the cash option, the vending machine transitions to the cash payment state, awaiting the user to insert the required amount of cash(5$).

If the user selects the card option, the vending machine transitions to the card payment state, prompting the user to give a 4 bit pin for the machine.

Once the payment is successfully processed, the vending machine transitions to the dispensing state.

In the dispensing state, the vending machine prepares and dispenses the selected beverage to the user.

After dispensing the beverage, the vending machine transitions to the transaction complete state and displays a transaction completion message.

If the selected beverage is out of stock, the vending machine transitions to the unavailable state and prompts the user to the select state again.

If the payment fails or is insufficient, the vending machine transitions to the payment failure state and prompts the user to idle.

Once the transition reaches complete state the process is successfully completed.

Non-Deterministic Automata (NDPA) :

An NPDA (Nondeterministic Pushdown Automaton) is a computational model that extends the capabilities of a deterministic pushdown automaton (PDA). It is a theoretical device used in the field of automata theory and formal languages to recognize and generate context-free languages.

Similar to a PDA, an NPDA consists of:

An input tape: It provides the input symbols to the automaton.

A stack: It serves as a memory structure and allows the automaton to store and retrieve symbols.

States: The automaton operates in a finite set of states.

Transitions: The rules that define the behaviour of the automaton, specifying how it moves between states based on the input and stack symbols.

Alphabet: Σ = {a, b} (Input symbols)

Stack Alphabet: Γ = {A, B} (Stack symbols)

States: Q = {q0, q1, q2, q3} (States)

Start State: q0 (Initial state)

Accepting States: F = {q3} (Accepting state)

However, unlike a deterministic PDA, an NPDA allows for nondeterminism. This means that in a given state, with a specific input symbol, and a symbol on top of the stack, there can be multiple possible transitions that the automaton can choose from. This nondeterminism allows the NPDA to explore multiple paths of computation simultaneously, potentially leading to greater expressive power.

NPDAs are typically used to recognize context-free languages, which are a class of languages defined by context-free grammars. They are more powerful than finite-state machines and can recognize more complex languages. NPDAs can also be used for parsing, validating and generating strings based on context-free grammars.

To simulate an NPDA, various algorithms and techniques are used, such as the Nondeterministic Finite Automaton (NFA) to Deterministic Finite Automaton (DFA) conversion or the use of backtracking and recursion.

It's important to note that while NPDAs are a theoretical concept and not directly implemented in practical programming languages, they provide insights into the theoretical foundations of computation and serve as a basis for designing and analysing more complex computational systems.

States :

1. Idle: The initial state of the vending machine, waiting for user input.

2. Select : The user has pressed a button to select a product.

3. Dispensing: The machine is dispensing the selected product.

4. Unavailable:The given input of selected product is unavailable.

5. InsufficientFunds: The user has not inserted enough money to purchase the selected product.

6.Complete: The user has successfully purchased the product.

7. Refund: The machine is refunding the user's money due to a cancellation or error.

8. Maintenance: The machine is undergoing maintenance and is temporarily out of service.

9. Coffee: The machine is currently dispensing coffee.

10. Tea: The machine is currently dispensing tea.

11. Milk: The machine is currently dispensing milk.

12. Hot chocolate : The machine is currently dispensing cocoa.

13. Payment: The user is selecting the payment method.

14. Cash: The user has selected to pay with cash.

- 1$: Amount received is 1$
- 2$: Amount received is 2$
- 3$: Amount received is 3$
- 4$: Amount received is 4$
- 5$: Amount received is 5$

15. UPI: The user has selected to pay with UPI.

- 1 : If the first digit of the pin is 1 this state is reached.
- 0 : If the second digit of the pin is 1 this state is reached.
- 1 : If the third digit of the pin is 1 this state is reached.
- 0 : If the fourth digit of the pin is 1 this state is reached.

Input Symbols:

- ButtonPress : B

- Maintenance : M

- ProductSelected : S

- Tea : T

- Coffee : C

- Milk : M

- Coca : H

- Not Available : N

- Available : A

- Payment Failed : F

- UPI : U

- Cash : G

- Processing : P

- Insufficient Funds : I

Stack Symbols:

- λ (empty symbol)

- UserInput

Initial State : Idle

Final State : Dispensing

Transitions:

1. (Idle, B,L) → (Maintenance,B)
2. (Idle, S, S) → (Select, e)
3. (Select, T, S) → (Tea, AS)
4. (Select, C, S) → (Coffee, AS)

5. (Select, M, S) → (Milk,AS)
6. (Select, H, S) → (Hot Chocolate, NS)
7. (Select, a, X) → (q2, Y)
8. (Tea,L, N) → (Unavailable, N)
9. (Coffee, L, N) → (Unavailable, N)
10. (Milk, L, N) → (Unavailable, N)
11. (Hot Chocolate, L, N) → (Unavailable, N)
12. (Unavailable, L, N) → (Maintenance, L)
13. (Tea, L, A) → (Payment, L)
14. (Coffee, L, A) → (Payment, L)
15. (Milk, L, A) → (Payment, L)
16. (Hot Chocolate, L, A) → (Payment, L)
17. (Payment, U, S) → (UPI, US)
18. (Payment, G, S) → (Cash, GS)
19. (UPI, 0, S) → (PaymentFail, S)
20. (PaymentFail, L, S) → (Idle, L)
21. (Payment, U, S) → (UPI, US)
22. (Payment, G, S) → (Cash, GS)
23. (UPI, 0, S) → (PaymentFail, S)
24. (UPI, 1, S) → (1, 1S)
25. (1, 0, 1) → (0, 01)
26. (0, 0, 1) → (PaymentFail, 01)
27. (0, 1, 0) → (1, 10)
28. (1, 1, 1) → (PaymentFail, L)
29. (1, 0, 1) → (0, 01)
30. (0, L, L) → (Successful, L)
31. (Successful, L, S) → (Dispensing,S)
32. (Dispensing,L,S)→(Complete,L)
33. (Dispensing, D, S) → (Dispenser Failed,DS)
34. (Dispenser Failed,L,D)→(Refund,L)
35. (Payment, G, S) → (Cash, GS)
36. (Cash, X, L) → (1$, XG)
37. (1$, X, L) → (2$, X)
38. (2$, X, L) → (3$, X)
39. (3$, X, L) → (4$,, X)
40. (4$, X, L) → (5$, X)
41. (1$, L, L) → (Insufficient Fund, L)
42. (2$, L, L) → (Insufficient Fund, L)
43. (3$, L, L) → (Insufficient Fund, L)
44. (4$, L, L) → (Insufficient Fund, L)
45. (Insufficient Fund, L,1) →(Refund,L)
46. (5$, L, L) → (Sufficient, L)
47. (Sufficient, L, L) → (Successful, L)
48. (Successful, L, X) → (Refund, X)

49. (Refund, L, S) → (Complete, L)

Context Free Language:-

S=>T/C/M/H

B=>M

H=>N

N=>M

T=>AP

C=>AP

M=>AP

P=>U/G

G=>X/S1

U=>F/0/1/S1

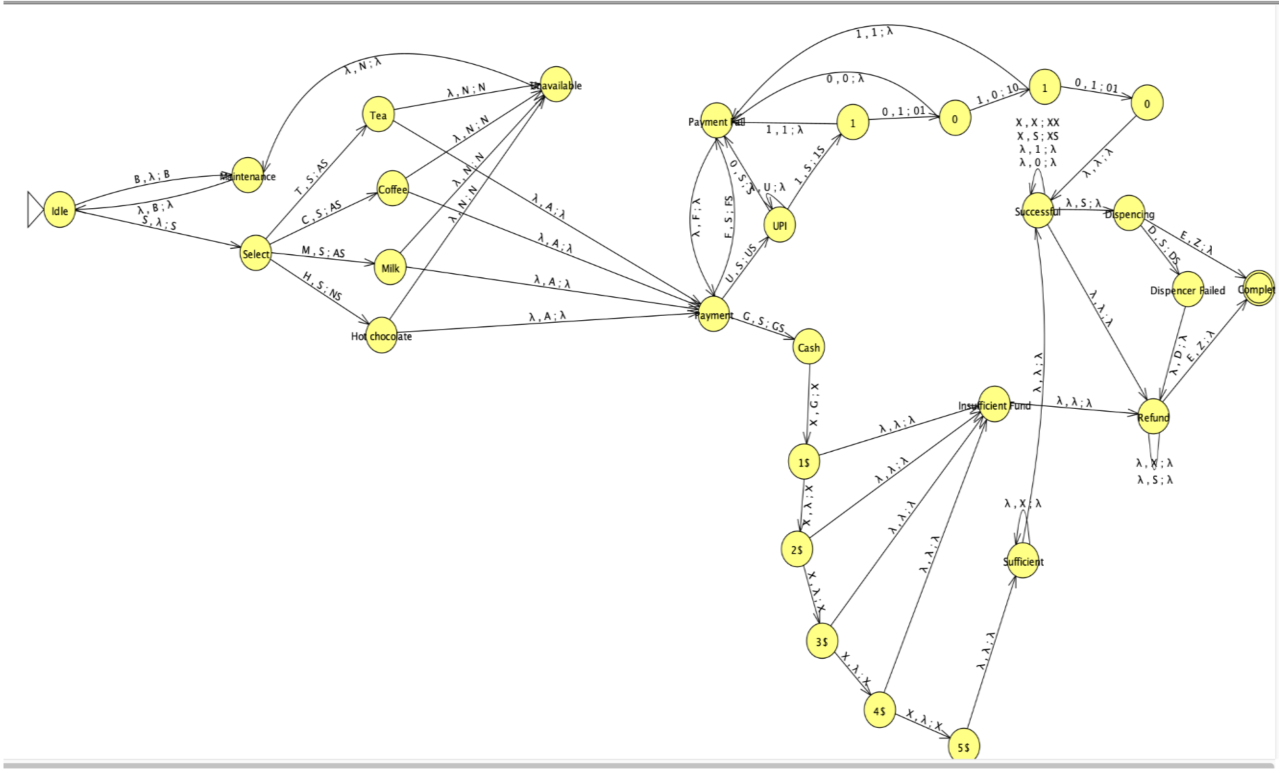
F=>P

S1=>D

D=>C

R=>C

NPDA MODEL WITH JFLAP :



Turing Machine:

A Turing Machine (TM) is a mathematical abstraction used to model computation. It consists of an infinite tape divided into cells, each capable of storing symbols. The TM has a head that can read and write symbols on the tape. It also has a state register that keeps track of its internal state.

The TM operates by reading a symbol from the current cell, replacing it with another symbol, changing its internal state, and moving the head either to the left or right along the tape. The movement direction is determined by the transition function, which takes into account the current state and the symbol being read.

If the TM reaches a final state after a series of transitions, it accepts the input string. Otherwise, if it enters a non-final state or gets stuck in an infinite loop, the input string is rejected.

Formally, a Turing Machine can be represented by a 7-tuple (Q, X, ∑, δ, q0, B, F), where:
- Q is a finite set of states the TM can be in.
- X is the set of symbols that can be written on the tape.
- ∑ is the set of symbols that can be used as input.
- δ is the transition function, which defines how the TM moves between states based on the current state and symbol being read, and also specifies the symbol to be written on the tape and the direction to move the head.
- q0 is the initial state of the TM.
- B is the blank symbol, representing empty cells on the tape.
- F is the set of final states, indicating when the TM should accept an input string.

Turing Machine uses **Unrestricted grammar** which generates recursively **enumerable language**

Unrestricted grammar:
Unrestricted grammar is type-0 grammar.
In automata theory, the class of unrestricted grammars (also called semi-Thue, type-0 or phrase structure grammars) is the most general class of grammars in the Chomsky hierarchy. No restrictions are made on the productions of an unrestricted grammar, other than each of their left-hand sides being non-empty.

Enumerable language:
RE languages or type-0 languages are generated by type-0 grammars. An RE language can be accepted or recognized by the Turing machine which means it will enter into the final state for the strings of language and may or may not enter into the rejecting state for the strings which are not part of the language. It means TM can loop forever for the strings which are not a part of the language. RE languages are also called Turing recognizable languages.

STATES:

1. Idle: The initial state of the vending machine, waiting for user input.

2. Selecting: The user has pressed a button to select a product.

3. Processing: The machine is processing the user's selection and preparing to dispense the product.

4. Dispensing: The machine is dispensing the selected product.

5. InsufficientFunds: The user has not inserted enough money to purchase the selected product.

6. TransactionComplete: The user has successfully purchased the product.

7. Refunding: The machine is refunding the user's money due to a cancellation or error.

8. Maintenance: The machine is undergoing maintenance and is temporarily out of service.

9. Coffee: The machine is currently dispensing coffee.

10. Tea: The machine is currently dispensing tea.

11. Milk: The machine is currently dispensing milk.

12. Cocoa: The machine is currently dispensing cocoa.

13. PaymentMethod: The user is selecting the payment method.

14. Cash: The user has selected to pay with cash.

15. UPI: The user has selected to pay with UPI.

16. 1$ ->Accepts one dollars(for cash payment method)

17. 2$ ->Accepts two dollars(for cash payment method)

18. 3$ ->Accepts three dollars(for cash payment method)

19. 4$ ->Accepts four dollars(for cash payment method)

20. 5$ -> Accepts five dollars(for cash payment method)

21. 1 -> It accepts binary number 1 (for UPI payment method to check UPI)

22. 0 -> It accepts binary number 0 (for UPI payment method to check UPI)

INPUT SYMBOLS:

B-Button Press: This input is used to trigger the vending machine to enter the Maintenance state, where it can be serviced or repaired.

S-Selecting: This input changes the state of the vending machine from idle to the select state, indicating that a beverage selection is in progress.

T-Tea: This input changes the state of the vending machine from the select state to the Tea state, indicating that the customer has selected tea as their beverage choice.

C-Coffee: This input changes the state of the vending machine from the select state to the Coffee state, indicating that the customer has selected coffee as their beverage choice.

M-Milk: This input changes the state of the vending machine from the select state to the Milk state, indicating that the customer has selected milk as their beverage choice.

H-Cocoa: This input changes the state of the vending machine from the select state to the Cocoa state, indicating that the customer has selected cocoa as their beverage choice.

N-Not Available: This state indicates that the selected beverage is not available or out of stock in the vending machine.

A-Available: This state indicates that the selected beverage is available and can be dispensed.

Z-Payment Failed: This state indicates that the payment for the selected beverage has failed.

U-UPI: This input represents the selection of UPI (Unified Payments Interface) as the payment method.

G-Cash: This input represents the selection of cash as the payment method.

P-Successful: This state indicates that the payment for the selected beverage has been successfully processed.

I-Insufficient Fund: This state indicates that the selected payment method does not have sufficient funds to complete the transaction.

D-Dispensing: This state indicates that the vending machine is currently dispensing the selected beverage.

Y-Dispenser Failed: This state indicates that the dispenser of the vending machine has encountered a failure during the dispensing process.

F-Processing Failed: This state indicates that there has been a failure in processing the selected beverage or payment.

1- For validating UPI Pin (Binary): This input represents a binary signal used to validate the UPI PIN during the payment process.

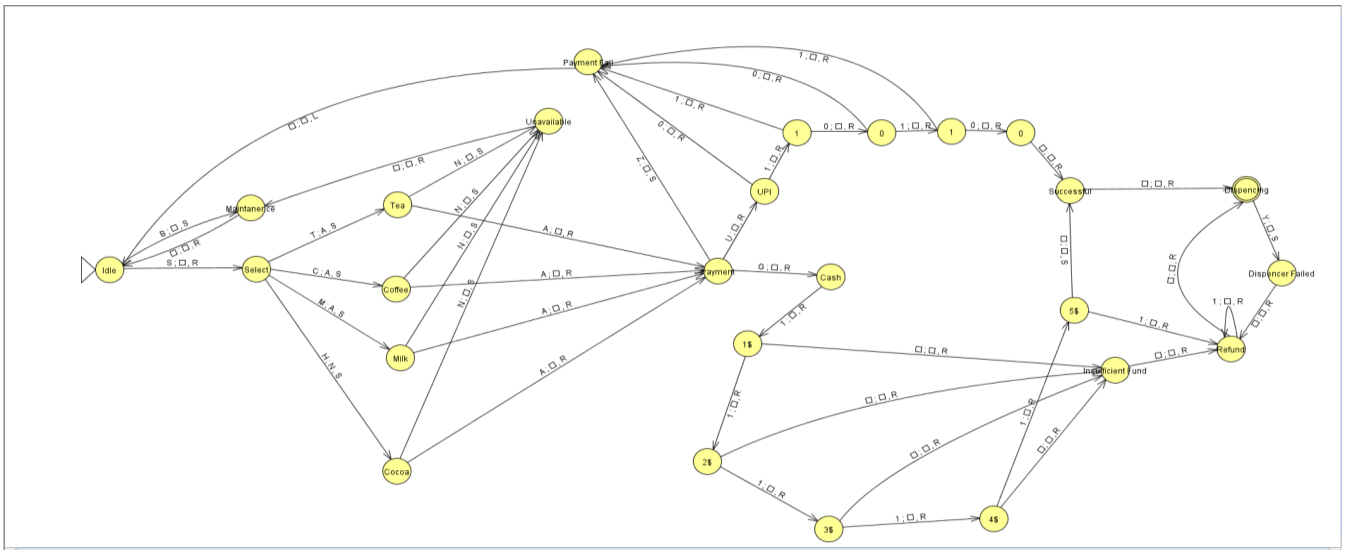
0- For validating UPI Pin (Binary): This input represents a binary signal used to validate the UPI PIN during the payment process.

Transitions:

1. (Idle,B,[]) = (Maintenance,S)
2. (Maintenance,[],[]) =(Idle,R)
3. (Idle,S,[]) = (Select,R)
4. (Select,T,[]) = (Tea ,R)
5. (Select,C,[]) = (Coffee,R)
6. (Select,M,[]) = (Milk,R)
7. (Select,H,[]) = (Cocoa,R)
8. (Tea,N,[]) = (Unavailable,S)
9. (Coffee,N,[]) = (Unavailable,S)
10. (Milk,N,[]) = (Unavailable,S)
11. (Cocoa,N,[]) = (Unavailable,S)
12. (Unavailable,[],[]) = (Maintenance,R)
13. (Tea,A,[]) = (Payment,R)
14. (Coffee,A,[]) = (Payment,R)
15. (Milk,A,[]) = (Payment,R)
16. (Cocoa,A,[]) = (Payment,R)
17. (Payment,Z,[]) = (Payment Fail,S)
18. (Payment Fail,[],[]) =(Idle,L)
19. (Payment,U,[]) = (UPI,S)
20. (Payment,G,[]) = (Cash,S)
21. (Payment,Z,[]) = (Payment Fail,S)
22. (UPI,0,[]) = (Payment Fail,R)
23. (UPI,1,[]) = (1,R)
24. (1,1,[]) = (Payment fail,R)
25. (1,0,[]) = (0,R)
26. (0,0,[]) = (Payment fail,R)
27. (0,1,[]) = (1,R)
28. (1,1,[]) =(Payment fail,R)
29. (1,0,[]) = (0,R)
30. (0,[],[]) = (Successful,R)
31. (Successful,D,[]) = (Dispensing,R)
32. (Dispensing,Y,[]) = (Dispenser Failed,S)
33. (Dispenser Failed,[],[]) = (Refund,R)
34. (Cash,1,[]) = (1$,R)
35. (1$,1,[]) = (2$,R)
36. (2$,1,[]) = (3$,R)
37. (3$,1,[]) = (4$,R)
38. (4$,1,[]) = (5$,R)

39. (5$,[],[]) = (Successful,S)
40. (1$,[],[]) = (Insufficient Fund,R)
41. (2$,[],[]) = (Insufficient Fund,R)
42. (3$,[],[]) = (Insufficient Fund,R)
43. (4$,[],[]) = (Insufficient Fund,R)
44. (5$,1,[]) = (Refund, R)
45. (Refund,1,[]) = (Refund,R)
46. (Refund,[],[]) = (Dispensing,R)
47. (Insufficient Fund,[],[]) = (Refund,R)

OUTPUT:



Tool we used to generate NPDA and TM is JFLAP(software).

JFLAP (Java Formal Languages and Automata Package):

JFLAP (Java Formal Languages and Automata Package) is software developed by the University of Arizona. It provides a user-friendly interface for designing, simulating, and experimenting with various languages, automata, and other computational models.

Key features of JFLAP:

JFLAP (Java Formal Language and Automata Package) provides many tools and functions for working with formal languages and automata. Below are some of the key tools available in JFLAP:

1. Automata Editor: JFLAP includes an interactive graphical editor that allows users to create, modify, and visualise various types of automata, such as finite automata, pull-down automata, and Turing. machines.

The editor provides tools for adding, modifying and accepting states, and functionality for editing and modifying content.

2. Grammar editor: JFLAP, regular grammar, non-grammatical content, etc. It provides a grammar editor for creating and editing grammar, including Users can define production codes, specify initialization, and use terminals and non-terminals. The editor supports syntax highlighting and error checking for correct syntax.

3. Vending Simulator: JFLAP provides a simulation tool that allows users to interact with vending machines and observe their behaviour. Users can enter the string and complete the calculation step by step, seeing the changes in the state and the result of each step. The simulator provides options to control the shutter speed and calculate the trace.

4.Formal language operations: JFLAP, intersection, integration, integration, aggregation, etc. It provides tools for performing various official language operations, such as These functions allow users to connect, compare and analyse different words defined by automata or grammar.

5. Conversion tools: JFLAP includes conversions between different automata types and grammars. Users can convert regular expressions to finite automatons, finite automatons to grammar and similar conversions of other language constructs.

This transform tool helps explore the relationship and balance between different representations.

6. Testing tools: JFLAP provides tools to test language membership and equivalence. Users can input strings to check for automaton-recognized or grammar-generated words. In addition, JFLAP also provides automata or grammatical equation evaluation functions to help users verify the correctness of their construction.

7. Testing and visualisation: JFLAP allows users to test and visualise features of formal languages and automata. It provides functions for creating and visualising language patterns such as random sequences, split trees, and variable shapes. These features are useful for understanding and analysing the behaviour of formal language constructs.

JFLAP is written in Java and is compatible with many operating systems, including Windows, macOS, and Linux.

It is free to download and use, making it accessible to students, teachers, and researchers interested in language and automata theory.

INDIVIDUAL CONTRIBUTION:

CB.EN.U4CSE21628 (K Kusuma) -Turing machine implementation and word document.

CB.EN.U4CSE21646(PhD Surya Shanmuk) - Turing machine implementation and word document.

CB.EN.U4CSE21651(R Hasanthi) - Non deterministic pushdown automata implementation and word document.

CB.EN.U4CSE21669(Kurasala Yejnakshari Meghana)- Turing Machine implementation Turing Machine Idea,NPDA idea,preparation of Word Document

CB.EN.U4CSE21670(Yellina SriBhargav) -Turing Machine implementation and PDA Machine Implementation, preparation of Word Document.