BMS Institute of Technology and Management

(An Autonomous Institution Affiliated to VTU, Belagavi)

Report on

"Automated Teller Machine"

Submitted in the partial fulfillment for the requirements of the degree of

BACHELOR OF ENGINEERING IN
COMPUTER SCIENCE AND ENGINEERING

Submitted By

DIYA HARWALKAR	**1BY22CS067**
ARNAV MAHESH	**1BY22CS215**
CHANDANA GOWRI D A	**1BY23CS404**
INDUJJA R	**1BY23CS406**

Under the guidance of

DR. THIPPESWAMY G

PROFESSOR & HOD
Department of CSE, BMSIT&M

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
YELAHANKA, BENGALURU - 560064.
2024-202

ABSTRACT

Automated Teller Machines (ATMs) are a vital part of modern banking, enabling customers to access financial services securely and conveniently. In this context, computational models like regular expressions and finite automata play a crucial role in defining and validating the sequences of operations within an ATM system. An ATM's behavior can be modeled as a regular language, where valid transitions—such as authentication, transaction selection, and confirmation—are expressed through structured patterns.

This report explores the theoretical and practical aspects of computational models applied to ATM systems. A regular expression is a sequence of characters that defines a pattern for matching strings, making it a foundational tool for ensuring that ATM operations follow predefined sequences. For instance, the interaction sequence of PIN entry, transaction type selection, and confirmation can be expressed as a regular expression, ensuring the system processes inputs correctly and securely.

The report begins by reviewing the syntax and semantics of regular expressions, emphasizing key components such as literals, concatenation, and the Kleene star, which enable the precise definition of ATM operation sequences. It further examines the theoretical relationship between regular expressions and finite automata (DFA/NFA), providing a computational framework for validating and processing user interactions within the ATM.

Additionally, the report discusses practical implementation strategies for ATM operations, focusing on real-time validation of inputs, handling errors such as invalid PINs or insufficient funds, and integrating fault tolerance to ensure uninterrupted service. Optimized techniques, such as state minimization in automata and dynamic programming approaches, are highlighted to improve system efficiency and reliability.

TABLE OF CONTENTS

INTRODUCTION

Automated Teller Machines (ATMs) are a crucial aspect of modern banking, providing secure and convenient access to financial services for customers worldwide. Key operations such as PIN authentication, cash withdrawals, balance inquiries, and fund transfers can be modeled and validated using computational concepts like regular expressions and finite automata. These operations can be represented as regular languages, where regular expressions provide a concise and efficient way to describe valid sequences of interactions.

For example, a standard ATM transaction sequence—such as PIN validation, transaction type selection (withdrawal, deposit, or balance inquiry), and confirmation—can be expressed as a regular expression, ensuring that all user interactions adhere to predefined rules. Regular expressions are constructed using fundamental operations: union (a + b), concatenation (ab), and the Kleene star (a*), allowing for the definition of complex patterns that describe various ATM workflows. These expressions ensure that transitions between states (e.g., from PIN entry to transaction selection) occur in a secure and predictable manner, preventing invalid or unsafe operations such as unauthorized access or incomplete transactions.

By modeling ATM processes as regular languages, finite automata (DFA/NFA) can be employed to recognize and validate user inputs, providing a mathematical framework for secure and reliable ATM operations. Real-world applications of computational models in ATMs include ensuring accurate PIN validation, error detection (e.g., insufficient funds or incorrect PINs), and fault tolerance to handle system failures. This computational approach enhances the security and efficiency of ATM systems by preventing unauthorized access, ensuring error-free operations, and optimizing user interactions.

Thus, the integration of regular expressions and finite automata bridges theoretical computation and practical banking applications, enabling ATMs to operate as secure, efficient, and user-friendly systems for financial transactions.

PROBLEM STATEMENT

The problem focuses on designing an efficient and secure Automated Teller Machine (ATM) system that facilitates seamless financial transactions for users while maintaining strict security standards. The system should model ATM operations, including user authentication, transaction selection, and confirmation, ensuring that each step adheres to predefined security protocols and logical workflows.

The challenge is to develop a system that reliably handles multiple transaction types—such as withdrawals, deposits, balance inquiries, and fund transfers—while preventing unauthorized access, mitigating errors like insufficient funds or invalid PINs, and ensuring smooth user experience.

A key aspect of the solution involves using Finite State Machines (FSMs) to model ATM operations, with each state transition representing a critical step in the transaction process. Regular expressions can be utilized to define valid input and transaction patterns (e.g., PIN_VALID_TXN_CONFIRM), ensuring that the system operates within the bounds of secure and predictable sequences.

The system must handle potential errors, such as incorrect PIN entries, exceeded retry limits, or communication failures with the banking server, ensuring robust fault tolerance and secure recovery mechanisms. Additionally, measures must be implemented to detect and prevent fraudulent activities, such as card skimming or unauthorized access attempts.

The ultimate goal is to develop a robust ATM system that optimizes transaction efficiency while ensuring user safety, data security, and compliance with banking regulations, leveraging computational models such as regular expressions and FSMs for enhanced reliability and accuracy.

USE CASE DESCRIPTION

Problem Statement:

Design a finite automaton or regular expression to validate user interactions in an ATM system, including PIN authentication, transaction selection, and confirmation.

Actors:

- **Input:** Strings representing ATM transaction sequences (e.g., "1234W100CONFIRM").
- **System:** A regular expression or deterministic finite automaton (DFA) for validating transaction sequences.

Formal Definition:

- **Alphabet (Σ):** {0, 1, 2, ..., 9, W, D, B, C}, where:
  - Digits (0-9) represent PIN or transaction amounts.
  - W represents withdrawal.
  - D represents deposit.
  - B represents balance inquiry.
  - C represents cancel.
- **Language (L):** The set of strings over Σ that represent valid ATM operations, such as entering a valid PIN followed by a transaction type and confirmation.

Regular Expression:

The regular expression to validate a complete transaction sequence can be:

(PIN_VALID)(W|D|B)(\d+)?(CONFIRM)

Explanation:

- **PIN_VALID:** Matches a valid 4-digit PIN (e.g., 1234).
- **W:** Matches a withdrawal request.
- **D:** Matches a deposit request.
- **B:** Matches a balance inquiry request.
- **\d+:** Matches transaction amounts (e.g., 100 for withdrawal or deposit).
- **CONFIRM:** Ensures the user confirms the transaction.

Steps in Theory of Computation:

Finite Automaton Representation:

Convert the regular expression into a DFA (Deterministic Finite Automaton):

- **States:** Represent the progression of matching the input string, with each state corresponding to a user action (PIN entry, transaction selection, amount entry, confirmation).
- **Transitions:** Follow the logical sequence: PIN → Transaction Type → Amount (if required) → Confirmation

Example:

1. **Start State (S):** Accepts PIN entry.
2. **State (P):** Verifies if the PIN is valid.
3. **State (T):** Accepts transaction type (W, D, B).
4. **State (A):** Accepts transaction amount (for W or D).
5. **Final State (F):** Confirms the transaction

LANGUAGE

The language LLL in the context of the ATM System consists of valid strings representing the sequence of states the ATM goes through during a transaction. Each string in LLL is chosen from the alphabet set defined by the ATM states. The language will define all valid sequences of states that the ATM transitions through during a transaction.

Alphabet (Σ):

The alphabet for the ATM System includes the following states:

- S: Start state (Accepts PIN entry)
- P: PIN validation state (Validates PIN)
- TTT: Transaction type selection state (Withdraw, Deposit, Balance Inquiry)
- A: Transaction amount state (For Withdraw or Deposit)
- F: Final state (Confirms the transaction)

Formal Definition:

Language (L):

The set of valid strings over ΣΣΣ that represent valid ATM transaction flows. The key requirement is that the states must follow a specific, predictable sequence, such as S→P→T→(A)→FS \to P \to T \to (A) \to FS→P→T→(A)→F.

L={w ∣ w∈Σ and w matches a valid ATM transaction sequence format}L = \{ w \ | \ w \in Σ \text{ and } } w \text{ matches a valid ATM transaction sequence format} \}L={w ∣ w∈Σ and w matches a valid ATM transaction sequence format}

Regular Expression:

The regular expression for valid ATM sequences can be:
SP(TA ∣ T)FSP(TA|T)FSP(TA ∣ T)F .This ensures that:

1. S: The ATM starts by accepting the PIN entry.

2. P: The ATM validates the PIN.
3. T: The user selects a transaction type.
4. A: If the transaction type requires an amount (e.g., Withdraw or Deposit), this state is mandatory.
5. FFF: The transaction is finalized.

Example Sequences for ATM System:

1. **SPTF:**Represents a valid sequence for a Balance Inquiry transaction: Start → PIN Validation → Transaction Type (Balance Inquiry) → Finalize.
2. **SPTAF:**Represents a valid sequence for a Withdraw/Deposit transaction: Start → PIN Validation → Transaction Type (Withdraw/Deposit) → Amount → Finalize.
3. **SPF:**Invalid sequence because the transaction type is missing.This would be rejected by the FSM.
4. **SPTA:**Invalid sequence because the transaction is not finalized.This would also be rejected by the FSM.

REGULAR EXPRESSION

This regular expression ensures that the ATM transaction sequence follows the defined order:

- S: Start → P: PIN Validation → T: Transaction Type (Withdraw, Deposit, or Balance Inquiry) → (Optional A: Amount for Withdraw/Deposit) → F: Finalize.
  SP(TAT)F

Formal Definition for Traffic Signal Control System:

- **Alphabet (Σ):** { SSS, PPP, TTT, AAA, FFF }
- **Language (L):** A set of strings over Σ that follow the valid ATM transaction sequence.L={w ∣ w∈Σ∗ and w matches the valid ATM transaction sequence format}L = \{ w \| \ w \in Σ^* \text{ and } } w \text{ matches the valid ATM transaction sequence format} \}L={w ∣ w∈Σ∗ and w matches the valid ATM transaction sequence format

Example Valid Sequences:

1. **SPTF:**Represents a Balance Inquiry transaction: Start → PIN Validation → Transaction Type (Balance Inquiry) → Finalize.
2. **SPTAF:**Represents a Withdraw or Deposit transaction: Start → PIN Validation → Transaction Type (Withdraw/Deposit) → Amount → Finalize.
3. **SPTFAF:**Represents multiple transactions in sequence, e.g., Start → PIN Validation → Transaction Type → Finalize → Another Transaction (Withdraw/Deposit) → Amount → Finalize.

Invalid Sequences:

1) **SPF:**Invalid because the transaction type is missing after PIN validation.
2) **SPT:**Invalid because the transaction is not finalized.
3) **SPTA:**Invalid because the transaction does not reach the final state.

GRAMMAR

Non-Terminals:

R: Represents a regular expression or a sequence of states in the ATM system.

E: Represents a specific ATM state, such as S, P, T, A, or F

Terminals: S,P,T,A,F: Represent the ATM states:

- S: Start (Accepts PIN entry).
- P: PIN Validation.
- T: Transaction Type selection (Withdraw, Deposit, or Balance Inquiry).
- A: Amount entry for Withdraw/Deposit.
- F: Finalize the transaction.

a. ϵ\epsilonϵ: Represents the empty string.

b. +: Alternation/Union (OR operator), used to represent choice between different states.

c. *: leene star (zero or more repetitions), used to denote the possibility of repeating sequences.

d. .: Concatenation, used to combine states in sequence.

e. (): Parentheses used for grouping sequences together.

Production Rules

1. R→ER \to ER→E:A regular expression can be a single element, corresponding to an ATM state like S, P, T, A, or F.
2. R→R+RR \to R + RR→R+R:A regular expression can be the union (alternation) of two other regular expressions. For example, T+AT + AT+A means either Transaction Type (T) or Amount (A) can be valid.
3. R→R.RR \to R . RR→R.R:A regular expression can be the concatenation of two other regular expressions. For example, P.TP . TP.T means PIN Validation (PPP) followed by Transaction Type (T).

4. R→R∗ R \to R*R→R∗ :A regular expression can be followed by a Kleene star to indicate zero or more repetitions. For example, (P.T)∗ (P .T)*(P.T)∗ allows multiple transactions after PIN validation.
5. E→S｜P｜T｜A｜FE \to S | P | T | A | FE→S｜P｜T｜A｜F:A regular expression can represent any state from the set of ATM states.
6. E→(R)E \to (R)E→(R):A regular expression can represent a group, which is another regular expression enclosed in parentheses. For example, (T.A)(T . A)(T.A) would group Transaction Type followed by Amount.
7. E→ϵE \to \epsilonE→ϵ:The empty string (ϵ\epsilonϵ) is also a valid regular expression, allowing for the possibility of no operations.

Example Breakdown:

Literal States:

- S,P,T,A,FS, P, T, A, FS,P,T,A,F: These are the simplest forms of regular expressions that represent the ATM states.

Concatenation:

- S.PS . PS.P: Matches Start followed by PIN Validation.
- S.P.T.FS . P . T . FS.P.T.F: Matches Start → PIN Validation → Transaction Type → Finalize.

  S.P.T.A.FS . P . T . A . FS.P.T.A.F: Matches Start → PIN Validation → Transaction Type → Amount → Finalize.

Alternation (Union):

- T+AT + AT+A: Matches either Transaction Type or Amount.
- T+A+FT + A + FT+A+F: Matches Transaction Type, Amount, or Finalize.

Kleene Star:

- (P.T)∗ (P . T)*(P.T)∗ : Matches zero or more repetitions of PIN Validation and Transaction
- (T.A)∗ (T . A)*(T.A)∗ : Matches zero or more repetitions of TransactionType and Amount.

Grouping:

- (P+T).F(P + T) . F(P+T).F: Matches either PIN Validation or Transaction Type and Finalize.
- S.(T+A).FS . (T + A) . FS.(T+A).F: Matches Start → either Transaction Type or Amount → Finalize.

ACCEPTED STRINGS

Allowed Example Sequences for ATM System:

S.P.T.F: This is a valid ATM transaction sequence. The user enters the PIN, selects a transaction type (e.g., Balance Inquiry), and finalizes the transaction.

S.P.T.A.F: This sequence is valid for a transaction requiring an amount, such as Withdraw or Deposit. It starts with PIN entry, proceeds to transaction type selection, then amount entry, and finally confirmation

S.P.T.F.S.P.T.A.F: This sequence represents multiple consecutive transactions. The user starts with one transaction (e.g., Balance Inquiry) and then performs another transaction (e.g., Withdraw or Deposit).

S.P.T.A.F.T.F: This is a valid sequence where the user performs a transaction requiring an amount (e.g., Withdraw) and then another without an amount (e.g., Balance Inquiry).

Explanation of Sequences:

S.P.T.F (Start → PIN → Transaction → Finalize): Represents a simple transaction flow, such as Balance Inquiry, where no amount input is needed.

S.P.T.A.F (Start → PIN → Transaction → Amount → Finalize): Represents a transaction like Withdraw or Deposit, where an amount is required before finalizing.

S.P.T.F.S.P.T.A.F (Start → PIN → Transaction → Finalize → Start → PIN → Transaction → Amount → Finalize): Demonstrates multiple transactions within a single

session, where the user performs one transaction (e.g., Balance Inquiry) and then initiates another (e.g., Withdraw).

S.P.T.A.F.T.F (Start → PIN → Transaction → Amount → Finalize → Transaction → Finalize): Illustrates a mixed sequence, where the user first performs a Withdraw or Deposit and then proceeds to a simple Balance Inquiry.

Key Rules:

1. **Mandatory Start State:**Every transaction sequence must begin with S (Start) for PIN entry.
2. **PIN Validation:**The state P(PIN Validation) must follow S. Without P, the transaction is invalid.
3. **Transaction Type Selection:**T(Transaction Type) is required after P. The user must choose a transaction type (e.g., Balance Inquiry, Withdraw, or Deposit).
4. **Amount Entry for Certain Transactions:**A (Amount) is mandatory if the transaction type involves monetary input (e.g., Withdraw or Deposit).
5. **Finalization:**All transactions must end with F (Finalize) to confirm the operation.
6. **Logical Sequence of States:**States must follow the prescribed logical order: S→P→T→(A)→FS → P → T → (A) → FS→P→T→(A)→F

UNACCEPTED STRINGS

Invalid Example Sequences:

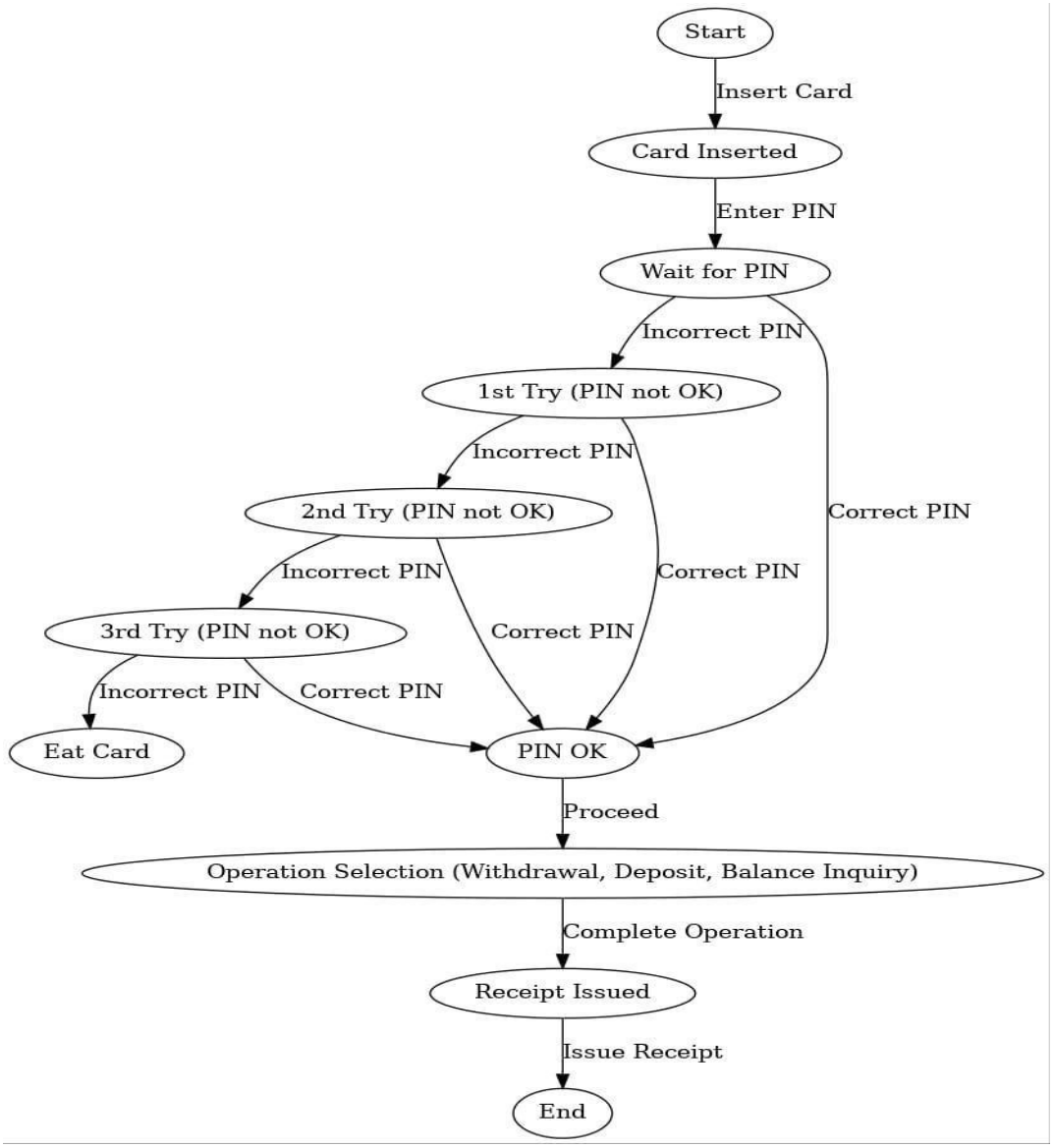
S.P.A.F: Invalid because TTT (Transaction Type) is missing after PIN validation.

S.F: Invalid because PPP (PIN Validation) and TTT (Transaction Type) are skipped.

P.T.A.F: Invalid because the sequence does not start with SSS (Start).

S.P.T.A: Invalid because the sequence is incomplete without FFF (Finalize).

STATE TRANSITION DIAGRAM :



Definition of State Transition for ATM System

A state transition in the ATM system is the process of moving from one operational state to another based on user inputs and system validations. The system transitions through a series of states representing each step in the transaction process, ensuring secure and accurate processing of user requests.

States for ATM System

1. **Initial State (S):**The starting state where the ATM is ready to accept user input, typically the PIN entry.
2. **PIN Validation State (P):**The state where the system verifies the entered PIN for correctness and authenticity.
3. **Transaction Selection State (T):**The state where the user selects the type of transaction, such as Withdraw, Deposit, or Balance Inquiry.
4. **Amount Entry State (A):**The state where the user enters the transaction amount (for Withdraw or Deposit).
5. **Transaction Confirmation State (C):**The state where the system processes the selected transaction and confirms its success or failure.
6. **Final State (F):**The state where the system finalizes the transaction, updates the balance, and resets to the initial state for the next user.

Transition Rules for ATM System

The following transition rules can be defined for the traffic signal control system:

S → P: On user input of a PIN, transition from the initial state to the PIN validation state.

P → T: On successful PIN validation, transition to the transaction selection state.

T → A: On selection of a transaction requiring an amount (Withdraw/Deposit), transition to the amount entry state.

A → C: On entering a valid amount, transition to the transaction confirmation state.

T → C: On selection of a transaction not requiring an amount (Balance Inquiry), transition directly to the transaction confirmation state.

C → F: On successful transaction processing, transition to the final state.

F → S: After finalization, reset to the initial state, ready for the next transaction.

Example of State Transitions for an ATM Transaction

1. Start in the initial state (S):The ATM is ready to accept the user's PIN.
2. Input PIN and transition to PIN validation state (P):The user enters the PIN, and the system validates it.
3. Successful PIN validation transitions to transaction selection state (T):The user selects a transaction type.
4. After amount entry, transition to transaction confirmation state (C):The system processes the entered amount and confirms whether the transaction was successful.
5. If Balance Inquiry is selected, transition directly to confirmation state (C):The system displays the user's balance.
6. Successful transaction transitions to final state (F):The transaction is finalized, and the system is reset for the next user.
7. Transition from final state (F) back to initial state (S):The ATM is ready for the next transaction.
8. h as Withdraw, Deposit, or Balance Inquiry.
9. If Withdraw or Deposit is selected, transition to amount entry state (A):The user enters the amount to withdraw or deposit.

Key Observations for State Transitions in ATM System

1) Sequential Flow:The ATM system transitions through states in a logical sequence, ensuring that each step (PIN validation, transaction type selection, amount entry, etc.) is completed before moving forward.
2) Validation at Each Step:Every state transition is contingent on successful validation (e.g., PIN correctness, amount availability).
3) Cyclic Nature**:**After completing a transaction, the system resets to the initial state, ready for a new session.
4) Flexibility in Transitions:Certain transactions like Balance Inquiry skip the amount entry state, streamlining the process.
5) Security and Accuracy:Each state ensures secure handling of user input and accurate processing of the transaction