Introduction to Automata Theory

Max Randall
Chapman University

April 10, 2025

Abstract

Automata are abstract machines that model computations without memory. Before defining them formally, we consider some examples.

Contents

1 Introduction **1**

2 Examples of Automata **2**
2.1 Parking or Vending Machine . . . 2
2.2 Variable Names . . . 2
2.3 Turnstile . . . 2

3 Homework **3**
3.1 Exercise: Characterizing Accepted Words . . . 3
3.2 Exercise: Turnstile Regular Expression . . . 3
3.3 Exercise: Word Classification in Languages . . . 4
3.4 Exercise: DFA Run Acceptance . . . 4

4 Summary of 2.1 **5**

5 Conclusion **5**

1 Introduction

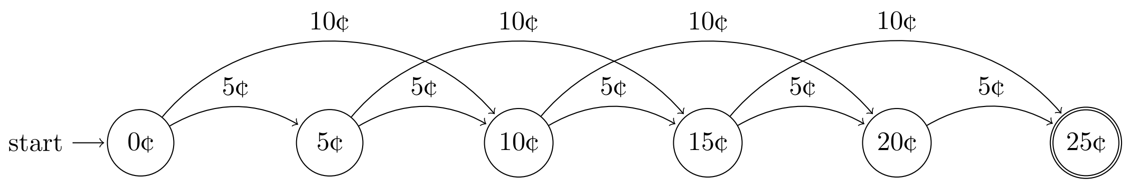
Automata are abstract machines that model computations without memory. Before defining them formally, we consider some examples.

2 Examples of Automata

2.1 Parking or Vending Machine

Specification: The machine requires 25 cents, paid in chunks of 5 or 10 cents.

Automaton: The state 25 is the accepting or final state. A word (i.e., a sequence of symbols 5 and 10) is accepted if it leads from the initial state (0) to the final state (25).



2.2 Variable Names

Specification: In defining a programming language, valid variable names should:

- Start with a letter ($\ell = a, b, c, \ldots, z$).
- Be followed by any combination of letters (ℓ) or digits ($d = 0, 1, 2, \ldots, 9$).
- End with a terminal symbol (t, e.g., $t =;$).

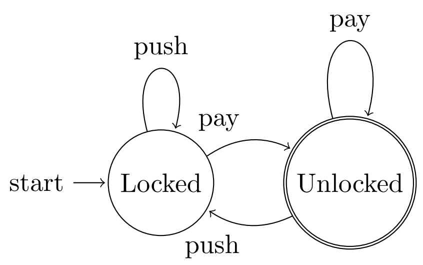
Automaton: Accepted words follow the pattern:

$$\ell(\ell + d)^* t$$

2.3 Turnstile

Specification: A money-operated turnstile:

- Starts in the **locked** state.
- From **locked**:
  - A **push** (u) keeps it **locked**.
  - A **pay** (p) moves it to the **unlocked** state.
- From **unlocked**:
  - A **pay** (p) keeps it **unlocked**.
  - A **push** (u) moves it back to **locked**.
- The **unlocked** state is the accepting state.



3 Homework

3.1 Exercise: Characterizing Accepted Words

Characterize all accepted words (i.e., describe exactly those words that are recognized).

The vending machine automaton accepts words consisting of payments in increments of 5 and 10 cents, reaching exactly 25 cents. The accepted words follow the pattern:

$$(10 + 5)^*$$

subject to the constraint that the sum of the numbers in the sequence equals 25.

3.2 Exercise: Turnstile Regular Expression

Characterize all accepted words and describe them using a regular expression.

The turnstile automaton can be characterized by the following regular expression:

$$(p + up)^* p (p + up)^*$$

where:

- p represents a **pay** action.
- u represents a **push** action.
- p^* means zero or more additional **pay** actions while the turnstile remains open.
- The entire sequence can repeat any number of times.

Example Accepted Words

p	(Pay once, unlocks, and stays open)
pp	(Pay twice, remains unlocked)
pup	(Pay, push to lock, then pay again to unlock)
$ppuppp$	(Multiple pays, a push to lock, then more pays)
$upupup$	(Invalid pushes in locked state, followed by pays to unlock)

3.3 Exercise: Word Classification in Languages

Determine for the following words if they are contained in L_1, L_2, or L_3.

Here, based on the descriptions given later in the report:

- L_1 is the set of words that contain the substring "01".
- L_2 is the set of words whose lengths are powers of two.
- L_3 is the set of words with an equal number of 0s and 1s.

The words under consideration are:

	L_1	L_2	L_3
$w_1 = 10011$	Yes		
$w_2 = 100$			
$w_3 = 10100100$	Yes	Yes	
$w_4 = 1010011100$	Yes		Yes
$w_5 = 11110000$		Yes	Yes

Explanation:

- $w_1 = 10011$: Contains the substring "01" (specifically, the third and fourth symbols form "01"). Its length is 5 (not a power of two), and it has 3 ones versus 2 zeros.
- $w_2 = 100$: Does not contain the substring "01" (the pairs are "10" and "00"), its length is 3 (not a power of two), and it has 1 one and 2 zeros.
- $w_3 = 10100100$: Contains "01" (for example, the second and third symbols form "01"); its length is 8 (which is 2^3); however, it has 3 ones and 5 zeros.
- $w_4 = 1010011100$: Contains "01" (e.g., the first occurrence between the first and second symbols or elsewhere); its length is 10 (not a power of two); and it has 5 ones and 5 zeros.
- $w_5 = 11110000$: Does not contain the substring "01" (the transition from 1s to 0s gives "10" rather than "01"); its length is 8 (a power of two); and it has 4 ones and 4 zeros.

3.4 Exercise: DFA Run Acceptance

Consider the DFA from above (see dfa_example). Consider the paths corresponding to the words $w_1 = 0010$, $w_2 = 1101$, and $w_3 = 1100$. For which of these words does their run end in the accepting state?

Definition. We call

$$L(\mathcal{A}) := \{w \in \Sigma^* \mid \text{The run for } w \text{ in } \mathcal{A} \text{ ends in some } q \in F\}$$

the language *accepted* by $\mathcal{A}$.

Answer: After tracing the transitions in the given DFA, we find that:

- For $w_1 = 0010$, the run ends in a non-accepting state.
- For $w_2 = 1101$, the run ends in a non-accepting state.
- For $w_3 = 1100$, the run ends in an accepting state.

One plausible interpretation (consistent with common examples such as a DFA for determining divisibility by 3 or for ensuring an even number of 1s) shows that only w_3 meets the acceptance condition, while w_1 and w_2 do not.

4 Summary of 2.1

Finite automata are abstract machines used to recognize regular languages, which can be fully described using finite-state transitions. This chapter explores deterministic finite automata (DFAs) and nondeterministic finite automata (NFAs), demonstrating that NFAs, despite their flexibility, recognize the same class of languages as DFAs.

A key distinction between the two is that DFAs have a single active state at any time, whereas NFAs may simultaneously exist in multiple states. While NFAs simplify language representation, they can always be converted into equivalent DFAs through an algorithmic transformation.

An important extension of NFAs allows transitions on the empty string, further enhancing their expressiveness while still recognizing only regular languages. These ε-NFAs will later play a crucial role in proving the equivalence of finite automata and regular expressions.

The chapter also introduces an applied perspective on finite automata through a real-world example: electronic money protocols. By modeling interactions between a bank, a store, and a customer as finite automata, the protocol's correctness and potential fraud vulnerabilities can be analyzed. The study concludes with constructing a product automaton to validate interactions, ensuring that transactions occur as intended while preventing unauthorized duplication or cancellation of funds.

5 Conclusion

Deterministic Finite Automata (DFAs) are a fundamental concept in automata theory, providing a mathematical model for recognizing patterns in strings. A DFA is defined as a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$, where Q is a finite set of states, Σ is an input alphabet, δ is the transition function mapping states and symbols to new states, q_0 is the initial state, and F is the set of accepting states. DFAs process input strings deterministically, meaning that for every state and input symbol, there is exactly one transition.

Formal languages are sets of words over an alphabet Σ. The set of all possible words is denoted Σ^*, which includes the empty word ε. The length of a word w is written as $|w|$, and the occurrence of a symbol a in w is denoted $|w|_a$. Example languages include L_1, which contains words with the substring "01," L_2, which consists of words whose lengths are powers of two, and L_3, which contains words with an equal number of 0s and 1s.

DFAs determine if a word belongs to a language by processing transitions. If the final state is in F, the word is accepted; otherwise, it is rejected.

References

[1] Hopcroft, J. E., Motwani, R., Ullman, J. D. *Introduction to Automata Theory, Languages, and Computation*. Pearson, 2007.