CS304: Automata and Formal Languages

Lec 21

Turing Machines

Rachit Nimavat

October 9, 2025

Outline

Turing Machines

So far...

Finite Automata are great for simple pattern matching, like checking for valid identifiers or keywords. They have a finite memory (their states).

Pushdown Automata are more powerful. By adding a stack, they gain infinite memory, allowing them to recognize context-free languages, like balanced parentheses or palindromes.

Even PDAs have limits.

They can't handle languages that require more complex memory like $L = \{a^n b^n c^n \mid n \geq 0\}$.

A single stack isn't enough to handle this "double-matching" requirement. We need a machine with unrestricted access to memory.

Turing Machines

Conceived by Alan Turing in 1936

Can simulate any algorithmic process

A simple, abstract model of computation powerful enough to **capture the essence* of what it means to compute**

Any problem that can be solved by any computer (no matter how futuristic or how advanced*) can be solved by a Turing Machine

Forms the basis for modern computing theory

How does a human compute?

- Write input on a paper
- Do the computation (think and write the intermediate results on the paper)
- Write output on the paper

How did Turing formalize human computation?

- Turing = Named after Alan Mathison Turing
- Machine = Computing machine

Human computation	Machine computation
	Tape
	Tape head
	Transition function

Turing Machine: Informal Defn

Imagine a machine with a **read/write head** that can move along an infinitely long piece of **tape**. This tape is divided into **cells**, each capable of holding a single symbol.
This simple model has three fundamental abilities:

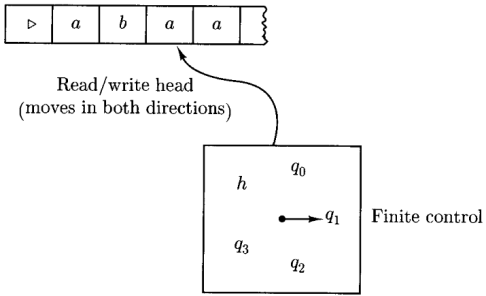
Read the symbol on the tape cell under the head.

Write a new symbol to that cell, erasing what was there.

Move the head one cell to the left or one cell to the right. **(Two-Way Head)**

That's it! The machine's behavior is directed by a finite set of states, just like an Finite Automaton.

Diagram of a Turing machine (TM)



Source: Lewis and Papadimitriou. Elements of the Theory of Computation.

Concept	Meaning
Tape	Simulates unlimited sheets of paper for computation.
Tape head	Read/write onto a tape cell. Moves left/right.
States	Simulates states of a human mind.
Input	Finite number of symbols initially on the tape.
Output	Finite number of symbols finally on the tape.
Computation	State transitions based on rules and input symbols.

First Example

Consider the language $L = \{a^n b^n c^n \mid n \geq 0\}$.

A string w is given on the tape, and the head starts at the leftmost symbol of w.

Infinitely many blank symbols (denoted by $\square$) follow w.

Turing Machine naturally mimics our intuition for algorithms:

1. Scan right until first $\square$ to check whether w is of the form $a^*b^*c^*$. If not, reject.
2. Return head to the leftmost symbol.
3. Scan right, crossing off single a, b, and c (replace them with $\times$) in each pass.
4. If all symbols are crossed off, accept.
5. If a symbol is missing, reject.
6. Go to step 2.

What is a Turing machine (TM)?

Definition

A Turing machine (TM) M is a 6-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, H)$, where,

1. Q: A finite set (set of states).
2. Σ: A finite set (input alphabet). Σ excludes $\triangleright, \square, \leftarrow, \rightarrow$.
3. Γ: A finite set (tape alphabet). $\Sigma \cup \{\triangleright, \square\} \subseteq \Gamma$. Γ excludes $\leftarrow, \rightarrow$.
4. $\delta : (Q - H) \times \Gamma$ to $Q \times (\Gamma \cup \{\leftarrow, \rightarrow\})$ is the transition function such that the tape head never falls off or erases $\triangleright$ symbol
   $\triangleright$ Time (computation)
5. q_0: The start state (belongs to Q).
6. $H = \{q_{\text{acc}}, q_{\text{rej}}\}$: The set of halting states (subset of Q).

Some notes on the Turing machine

Symbols

- $\triangleright$: Left end symbol
- $\square$: Blank symbol
- $\leftarrow, \rightarrow$: Left and right movement symbols
- Σ : Represents input/output/special symbols
- Γ : Represents symbols that can be present on the tape

Transition

- M never falls off the left end of the tape i.e.,
  when the current symbol is $\triangleright$, the tape head has to move right
- M stops when it reaches an accept or a reject state i.e.,
  δ is not defined for states in H

Language of TM

Language of a TM M, denoted $L(M)$, is the set of strings that M accepts

That is, for each string $w \in L(M)$, M **halts** in the accept state when started with w on its tape

A language L is called **Recursively Enumerable (RE)** (not to be confused with Regular Expressions) if there exists a TM M such that $L = L(M)$

Construct TM to erase the input string

Problem

- Construct a TM to erase the input string

Solution

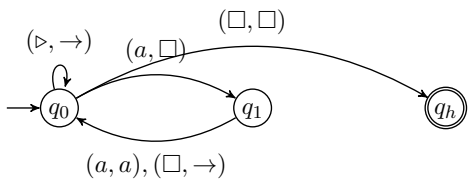
- Language recognizers such as DFA's cannot perform computational tasks such as erasing strings.
  So, no DFA can be used for erasing strings.
- Language generators such as CFG's cannot perform computational tasks such as erasing strings.
  So, no CFG can be used for erasing strings.
- TM's are more powerful than language recognizers and language generators.
  A TM can be used for erasing strings.

Construct TM to erase the input string

Problem

- Construct a TM to erase the input string

Solution (continued)



	Current symbol (Γ)		
Current state ($Q - H$)	$\triangleright$	a	$\square$
q_0	$(q_0, \rightarrow)$	$(q_1, \square)$	$(q_h, \square)$
q_1	–	(q_0, a)	$(q_0, \rightarrow)$

Construct TM to erase the input string

Problem

- Construct a TM to erase the input string

Solution (continued)

Time	State	Tape					
0	q_0	$\triangleright$	a	a	a	$\square$	$\cdots$
1	q_0	$\triangleright$	a	a	a	$\square$	$\cdots$
2	q_1	$\triangleright$	$\square$	a	a	$\square$	$\cdots$
3	q_0	$\triangleright$	$\square$	a	a	$\square$	$\cdots$
4	q_1	$\triangleright$	$\square$	$\square$	a	$\square$	$\cdots$
5	q_0	$\triangleright$	$\square$	$\square$	a	$\square$	$\cdots$
6	q_1	$\triangleright$	$\square$	$\square$	$\square$	$\square$	$\cdots$
7	q_0	$\triangleright$	$\square$	$\square$	$\square$	$\square$	$\cdots$
8	q_h	$\triangleright$	$\square$	$\square$	$\square$	$\square$	$\cdots$