Turing Machines

Lecture 13

Binghui Peng

We Need a Stronger Model!

Finite automata is simple to describe, but it can not describe complex language ($\{0^n1^n : n \geq 0\}$)

- Intuitive reason: Finite automaton only has constant memory.

We Need a Stronger Model!

Finite automata is simple to describe, but it can not describe complex language ($\{0^n 1^n : n \geq 0\}$)

- Intuitive reason: Finite automaton only has constant memory.

Push down automata is (still) simple to describe, but it also can not describe complex language ($\{a^n b^n c^n : n \geq 0\}$)

- Intuitive reason: The stack has "one shot" memory.

We Need a Stronger Model!

Finite automata is simple to describe, but it can not describe complex language ($\{0^n1^n : n \geq 0\}$)

- Intuitive reason: Finite automaton only has constant memory.

Push down automata is (still) simple to describe, but it also can not describe complex language ($\{a^nb^nc^n : n \geq 0\}$)

- Intuitive reason: The stack has "one shot" memory.

Turing machine is powerful!
A Turing machine has a finite control plus **an unbounded tape**.

Informal Picture of a Turing Machine

A Turing machine consists of:

1. a finite set of states,
2. an infinite tape divided into cells,
3. a head that reads one cell at a time,
4. rules telling the machine what to write, where to move, and which state to enter next.

On each step the machine can read, write, move left or right, and change state.

Formal Definition

A **Turing machine** is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_s, q_{\text{acc}}, q_{\text{rej}})$$

where:

1. Q is a finite set of states.
2. Σ is the input alphabet.
3. Γ is the tape alphabet with $\Sigma \subseteq \Gamma$ and blank symbol $\sqcup \in \Gamma$.
4. $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ is the transition function. It tells us what to do after reading one tape symbol.
5. q_s is the start state.
6. q_{acc} is the accept state.
7. q_{rej} is the reject state.

What Does the Transition Mean?

Suppose

$$\delta(q, a) = (p, b, R).$$

This means:

1. the machine is currently in state q,
2. the head is currently reading the tape symbol a,
3. it overwrites that cell with b,
4. it changes its state to p,
5. it moves the head one cell to the **right**.

Similarly,

$$\delta(q, a) = (p, b, L)$$

means the same thing except the head moves one cell to the **left**.

A Tiny Example of One Step

Suppose the current tape looks like

$$\cdots \sqcup 0\ 1\ 1 \sqcup \cdots$$

and the head is scanning the first 1.

If

$$\delta(q, 1) = (r, X, L),$$

then in one step the machine:

1. changes that 1 into X,
2. changes the state from q to r,
3. moves the head one cell left.

So after the step, the tape looks like

$$\cdots \sqcup 0\ X\ 1 \sqcup \cdots$$

and the head is now on the symbol 0.

Configurations and Computations

A **configuration** records:

1. the current state,
2. the current tape contents,
3. the current head position.

We write $C \vdash_M C'$ if machine M can go from configuration C to C' in one step. A computation is a sequence

$$C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \cdots$$

starting from the input configuration.

Example Language: $\{0^n1^n : n \geq 0\}$

A Turing machine can decide this language by repeatedly matching one 0 with one 1. High-level algorithm:

1. Find the leftmost unmarked 0 and change it to X.
2. Move right to find the leftmost unmarked 1 and change it to Y.
3. Move back to the left end and repeat.
4. If no unmarked 0 remains, check that no unmarked 1 remains either.

Trace on Input 0011

One possible run looks like this:

$$\begin{aligned} 0011 &\Rightarrow X011 \\ &\Rightarrow X01Y \\ &\Rightarrow XX1Y \\ &\Rightarrow XXYY. \end{aligned}$$

At that point all 0's and 1's are matched, so the machine accepts.

Recognizable and Decidable Languages

Definition: A language L is **Turing-recognizable** if some Turing machine recognizes it.

- If $x \in L$, the machine eventually accepts.
- If $x \notin L$, the machine may reject or loop forever.

Definition: A language L is **decidable** if some Turing machine decides it.

- On every input x, the machine halts.
- It accepts when $x \in L$ and rejects when $x \notin L$.

Examples of Decidable Languages

The following problems are decidable.

1. Given a DFA M and string w, determine whether M accepts w.
2. Given a CFG G and string w, determine whether $w \in L(G)$.
3. Given two integers a, b, determine whether a divides b.
4. Given a graph G, determine whether G is connected.
5. Given the board status of Go, determine whether black player will win
6. There are many more ...

Church-Turing Thesis

Church-Turing Thesis: Every function that is effectively computable by an algorithm can be computed by a Turing machine.

Church-Turing Thesis

Church-Turing Thesis: Every function that is effectively computable by an algorithm can be computed by a Turing machine.

1. Many different machine models all give the same computable functions,
2. Every practical programming language can be simulated by a Turing machine,
3. No convincing counterexample has ever been found.

Universality

The exact tape model is not sacred. We can change many details and still get the same computability notion:

1. multi-tape machines,
2. random-access style machines

Variation 1: Multi-Tape Machines

A **multi-tape Turing machine** has several tapes, each with its own head.

For example, a 2-tape machine can do the following in one step:

1. read the current symbol on tape 1,
2. read the current symbol on tape 2,
3. write new symbols on those tapes,
4. move each head independently,
5. change to a new state.

This often makes algorithms much easier to describe. One tape can store the input, and another tape can be used for scratch work.

Variation 2: RAM Machines

The **RAM model** means **Random Access Machine**.

This model looks more like an actual computer:

1. it has registers or memory cells,
2. it supports instructions like load, store, add, compare, and jump,
3. it can access memory locations more directly than scanning one tape cell (move left or right) at a time.

So RAM programs are often closer to the algorithms we write in practice.

Important fact: RAM machines are also equivalent to Turing machines for computability.

Takeaways

1. Turing machines extend finite automata with unbounded tapes.
2. Church-Turing thesis asserts that TM gives a general mathematical model of computation/algorithm.