CSCI 340: Computational Models

Variations of Turing Machines

Chapter 22

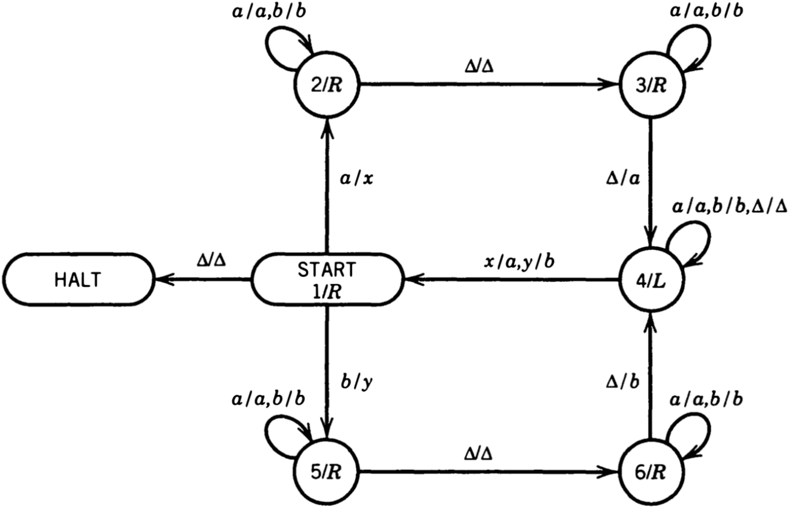
Department of Computer Science

Variations of Turing Machines

- The Move-in-State Machine
- The Stay-Option Machine
- The k-track Turing Machine
- The Two-Way Infinite Tape Turing Machine
- The Nondeterministic Turing Machine
- The Read-Only Turing Machine
- The Transition Turing Machine

The Move-in-State Machine

Combination of Mealy and Moore machine



The Move-in-State Machine

- The TAPE HEAD moves upon entering a state
- Transitions have the READ and WRITE actions

Theorem

For every move-in-state machine M, there is a TM (T) which accepts the same language. If M crashes, so does T. If M loops, so does T. If M crashes, so does T. M and T will always have the ***exact*** *same tape*

Proof.

- One-by-one take every single edge in M and change its labels
- If the next state tells the TAPE HEAD to move:
  - Right, change X/Y to (X, Y, R).
  - Left, change X/Y to (X, Y, L).
- Any edge going to HALT shall move the TAPE HEAD right.
- Once all edges are converted, remove movements from all states

□

The Move-in-State Machine

Theorem

For every TM (T), there exists a move-in-state machine M which accepts the same language and operates in exactly the same way on all inputs (and will always result in the ***exact*** *same tape.*

Proof.

- We cannot just "do the reverse" — see Mealy ↔ Moore machines
- If edges with different TAPE HEAD movements feed into the same state, we must "replicate" the state
- If the START has to split, only one of the clones can be called START — it doesn't matter which one
- If a state split loops back to itself, carefully decide which copy to loop back to

□

The Stay-Option Machine

- Instead of moving left or right, we introduce a third option to stay where we are
- This is a bit ridiculous — but is still possible

Definition

A Turing Machine with a stay option is called a **stay-option machine**.

Theorem

stay-option machine = TM

Proof.

- All (X, Y, S) transitions can be split into:
  (X, Y, R) followed by $(any, =, L)$
- $(any, =, L)$ is shorthand for (a, a, L), (b, b, L), (Δ, Δ, L) ...

□

The k-track Turing Machine

Definition

A **k-track TM** or kTM – has k normal TM TAPES and one TAPE HEAD which reads corresponding cells on all TAPES simultaneously and can write onto all TAPES at once. There is still an input alphabet Σ and tape alphabet Γ.

To operate on a kTM, the input initially lives only on TAPE 1. The output is the content on **all** TAPES

Theorem

- P1 *Given any TM and any k, there is a kTM that acts on all inputs exactly as the TM does*
- P2 *Given any kTM for any k, there is a TM that acts on all inputs exactly as the kTM does*

In other words, as an acceptor or transducer, TM = kTM

The k-track Turing Machine

We say that the 3TM TAPE status

a	d	g	...
b	e	h	...
c	f	i	...

corresponds to the one-TAPE TM status

a	b	c	d	e	f	g	h	i	...

Given this representation/conversion — the proofs (albeit long and omitted in these slides) show they are equivalent in power

The Two-Way Infinite Tape Turing Machine

- Two-Way Infinite Tapes were a part of Turing's original model
- The input string is placed *somewhere* on the TAPE — and the TAPE HEAD points to the first character of input
  1. We do not have to worry about crashing as we move to the left
  2. We now have two work areas to perform "calculations"

Theorem

TMs with two-way TAPES are exactly as powerful as TMs with a one-way TAPES as both language-acceptors and transducers

Proof Part 1 — Run a one-way TM on a two-way TM.

- Introduce a special symbol Ψ to the left of the first input character on the TAPE
- Simulate the one-way TM on the two-way TM
- If Ψ is read, the machine will crash □

Part 2 Proof omitted (emulating two-way TM as 2-tape TM)

Nondeterministic Turing Machine

Definition

A **nondeterministic TM**, or **NTM**, is defined like a TM but allows more than one edge leaving any state with the same “read” character.

An input string is accepted by an NTM if there exists *some* path through the program that leads to HALT, even if some paths loop or crash

Two NTMs (T_1 and T_2) are deemed equivalent if-and-only-if

$$\text{Accept}(T_1) = \text{Accept}(T_2)$$

Theorem

$$NTM = TM$$

Proof (Part 1).

The deterministic TM is by definition a NTM $\square$

Nondeterministic Turing Machine

Proof (Part 2).

General Idea: Simulate on a 3TM where the three tracks:

1. Run the input using “parent’s” advice
2. Generate “parent’s” advice
3. Keep a copy of the original input string

This allows us to try all paths of non-determinism

- This method emulates backtracking and rewinding
- Deterministically evaluates all options
- Since we can convert a 3TM to a TM, a TM can do everything a NTM can

□

Context-Free Languages on Turing Machines

Theorem

Every CFL can be accepted by some TM

Proof.

- Every CFL can be accepted by some PDA (perhaps NPDA)
- Every (N)PDA can be written as a (N)PM
- Every NPM can be written as a NTM
- Every NTM can be written as a 3TM
- Every 3TM can be written as a TM

□

Read-Only Turing Machine

Definition

A **read-only TM** is a TM with the property that for every edge the READ and WRITE fields are the same. Because of this restriction, the contents of the TAPE cannot be altered.

- We can refer to a read-only TM as a two-way FA
- As a transducer, a read-only TM is easy to describe:

$$\text{input} = \text{output}$$

Theorem

A read-only TM accepts exclusively regular languages

The Transition Turing Machine

Definition

A **transition Turing machine** is a nondeterministic read-only TM which allows *transition edges* similar to a transition graph.

Essentially, apply the bypass algorithm to a right-only transition Turing machine