Oberlin College
CSCI 383
Fall 2026
Lecture 22. Capabilities of Turing Machines
Kozen - Section 29

Last time: Defined Turing machines.

Today: What can Turing machines do? Two languages.

Recall definition from last time:

A Turing machine is a 9-touple $M = (Q, \Sigma, \Gamma, \vdash, \sqcup, \delta, s, t, r)$

- Q - states
- Σ - input alphabet (stuff that starts on tape)
- Γ - tape alphabet (stuff you're allowed to write)
- δ - transitions.

Special stuff.

- $\vdash \in \Gamma \setminus \Sigma$ - left endmarker
- $\sqcup \in \Gamma \setminus \Sigma$ - blank tape symbol
  Initial tape: $\vdash x \sqcup^\omega$
- $s \in Q$ - start state
- $t \in Q$ - accept state (sink)
- $r \in Q$ - reject state (sink)

Transitions: $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$.

Doubled Strings

Consider the language $L = \{ww \mid w \in \{a, b\}^*\}$.

Regular? Context-free?

Goal: Build a Turing machine that accepts L.

Running example: *aabbaaabba*

Relevant machine data. Easy stuff:

- Σ - $\{a, b\}$
- $\Gamma = \{a, b, \sqcup, \vdash, \dashv, \tilde{a}, \tilde{b}, \hat{a}, \hat{b}\}$

  Intuition: "Marking" characters on tape.

Trickier Stuff:

- States $Q = \{s, t, r, ...\}$ we'll leave others implicit.
- Transitions δ: implicit below.

First phase: Check for even length.

- Scan through the string, keeping track of length. (not quite! track parity)
- When you reach end:
  - If length odd, reject.
  - If length even, write $\dashv$.

Second phase: Find the middle of the string.

Thing that doesn't work: Count the number of characters in string, divide by two.

Thing that does work:

- Move left: Mark the first character with $\tilde{}$. Proceed to $\vdash$.
- Move right: Mark the first character with $\hat{}$. Proceed to $\dashv$.
- Move left: Mark Mark the first unmarked character with $\tilde{}$. Proceed to $\vdash$.
- Move right: Mark the first unmarked character with $\hat{}$. Proceed to $\dashv$.
- ..and so on.

When have we found the middle? When you can make a full pass without encountering an unmarked character.

Third phase: Match characters from the first and second halves.

Thing that doesn't work: Remember first half. Check against second half.

Thing that does:

- Delete first character in first half. Remember it.
- Delete first character in second half. If it doesn't match: reject.

Repeat.

Once tape is blank, accept.

Primes

$\{a^p \mid p \text{ prime}\}$

Let's say you had to determine by hand whether 637 was prime w/ pencil, paper.

Algorithm: Sieve of Eratosthenes.

- Cross out multiples of 2.
- Cross out multiples of 3.
- Cross out multiples of 5.
- ...

Generally: To determine if p is prime.

- Find earliest non-crossed out number k
- Cross out all multiples of k.

If p is crossed out, return false. If you reach $k = p$, return true.

Idea: Delete all as corresponding to multiples of 2, 3, 5.

(Show on example.)

Phase 1: Formatting/preproccessing.

- Check if $p = 0, 1$.
- Delete first a, replace pth a with \$.

Phase 2: Sieving

- Count by 2s, deleting every other a.
- Count by 3s, deleting every third a.
- Count by 4s, etc.

Not quite! Can't count with finite control.
Thing that works: Pebble trick.

- Mark kth spot, delete a.
- Put pebbles on spots $1, \ldots, k-1$.
- Move pebbles one at a time.
- When out of pebbles, you've counted k spots to the right.
- Repeat.

Repeat until you reach \$.

If you delete \$, reject.

TM for Primes

- Phase 1:
  - Check if $p = 0, 1$.
  - Delete first a, replace pth a with \$.
- Phase 2:
  - while there are as left:
  - find first a, at position k
  - erase as at multiples of k using pebble trick
  - if you erase \$, reject
- Accept.