CS 42—Computability

Thursday, September 13, 2018

Summary

Today, we finish our deepest dive into the world of theoretical Computer Science. Remember that the big question we've been asking is: **What kinds of problems can computers solve?**

To answer this question, we asked ourselves two additional questions and posited some answers:

> **What do we mean by "problem"?** Currently, our definition is: "Decision problems on finite, bitstring inputs."
>
> **What do we mean by "computer"?** Currently, our definitions are: "DFAs, NFAs, and regular expressions (from the assignment)."

Today, we'll evaluate whether the definitions we've developed for computers correspond to what we usually think of as computers (spoiler alert: no!).

Most of today is "bonus", in the sense that you'll never need to reproduce this knowledge on a CS 42 exam. We're talking about it because it reveals something about the essence of Computer Science—the thing that is CS's superpower and superweakness. These themes and ideas will recur over and over again this semester, and—if you study more CS in the future—you will no doubt see them again.

Recap: Distinguishability

Given a language L, can we *prove* that a DFA for L requires at least n states, for some n?

Definition: Two strings w_1, w_2 are **distinguishable** if there is some other string z such that $w_1 z \in$ L and $w_2 z \notin$ L.

Definition: A set of strings S is **pairwise-distinguishable** for a language L if every pair of strings $w_i \neq w_j$ is distinguishable.

Theorem: If a set of strings S = $\{w_1, w_2, \ldots, w_n\}$ is pairwise distinguishable for a language L, then any DFA that accepts L must have at least n states.

Regularity

Given a language L, if we can construct a DFA, NFA, or regular expression for L, then we say that L is a **regular** language.

Proving that a language is *not* regular

Given a language L, if there exists a set of strings S that:

- has infinite size
- is pairwise-distinguishable with respect to L

then the DFA for L has an infinite number of states, which is a contradiction (by definition, a DFA must have a finite number of states). Therefore, we cannot create a DFA for L, which means that L is not regular.

Note: When we do proofs like this, often the strings in S are *not* strings in L. However, the strings in S are *distinguishable with respect to* L.

Turing Machines (TM)

A TM has a finite alphabet (Σ) of valid symbols.

A TM has a finite set of states—one initial state and some accepting states.

A TM has an infinitely large "tape", which can be read or written.

A TM's *configuration* is its current location on the tape and the input symbol at that location.

A TM has a transition function. Each transition has three parts:

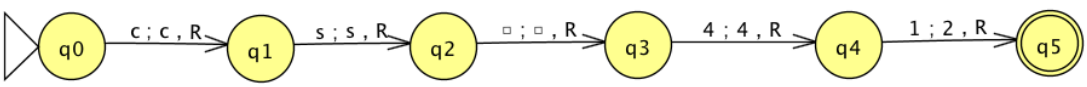
- The input symbol to match (i.e., what's under the "read head".
- The output symbol to overwrite at the current location.
- A direction to move (left or right).

Given an input string, a TM operates as follows:

> The machine starts in the initial state.
>
> The machine follows its transition function.
>
> If consuming the input causes the machine to terminate in an accepting state, the machine accepts the input. If consuming the input causes the machine to terminate in a rejecting state, the machine rejects the input. If the machine enters a configuration for which there is no transition defined, then the machine halts and rejects.

Example

The simple Turing machine below transforms "CS ❐ 41" into "CS ❐ 42". (The character ❐ is a blank character.)



Next time: Physical machines—gates and circuits