Recap: Minimality

Distinguishability: state ~ fate

distinguishability of strings: Two strings are distinguishable (with respect to a given language L) if we can append a common suffix z, causing a DFA for L to accept one string and reject the other.

pairwise-distinguishability of a set of strings: A set of strings is pairwise-distinguishable (with respect to a given language L) if every unique pair of strings in the set is distinguishable with respect to L.

distinguishability theorem: If a set of n strings is pairwise-distinguishable (with respect to a given language L), then a DFA that accepts L must contain at least n states.

Let’s practice

Use the distinguishability theorem to prove that any DFA that accepts the following language requires at least 5 states: \( L = \{ w \mid \text{the third bit in } w \text{ is a 1} \} \).

Irregularity: proving that a language is not regular

Myhill-Nerode theorem: A language L is regular if and only if we can define a set S of pairwise distinguishable strings such that |S| (i.e., the size of S) is finite.

Proof by contradiction: Assume that opposite of what you want to prove is true. Then show that this assumption leads to a contradiction (and therefore, the assumption is false).

Turing Machines

A Turing machine \( M \) consists of:

- an alphabet \( \Sigma \)
- a finite set of states, including an initial state and accepting state(s)
- transitions between states (described by a finite-state controller)
- an infinitely large tape, which can be read or written
- a current location on the tape

Given a string \( w \), \( M \) accepts \( w \) if consuming \( w \) causes \( M \) to terminate in an accepting state.

Transitions

Each transition has three parts:

- The input character to match (i.e., what’s under the “read-head”).
- The output character to overwrite at the current location.
- A direction to move (left or right).
**Example**

This simple Turing machine transforms “CS 60” into “CS 42”.

![Turing machine diagram](image)

**Let’s practice**

What does this machine do for the input 11?
What does it do in general?

![Turing machine diagram](image)

Bonus practice problem (we won’t do this in class, but it’ll be in the online slides): Construct a Turing machine that accepts the language $L = \{a^n b^m\}$.

---

**Undecidability**

There are some problems that computers can’t solve. We call these problems *undecidable*.

**The Halting Problem**

The Halting Problem asks: can you build a Turing machine that takes two inputs: (1) a program (represented as another Turing machine) and (2) some input to that program, then it determines whether the given program will halt on the given input.

The Halting Problem is undecidable: it is not possible to create a Turing machine that can give the correct (or even any) answer for all possible programs and all possible inputs.

**The superpower and superweakness of Computer Science:**

There is no difference between behavior (programs) and data

*Next time: gates and circuits*