Computability: Turing Machines

1 The Main Result

- We'll focus on functions $f : \mathbb{N} \to \mathbb{N}$.
- For a computer program to **compute** f is for it to yield $f(n)$ as output whenever it is given n as input ($n \in \mathbb{N}$).
- *Theorem:* not every function is computable.

  (And I can give you examples!)

2 How We'll Get There

- Turing Machines are computers of an especially simple sort.
- We'll see that some functions are not Turing-computable.
- But: any function that can be computed using an ordinary computer is also computed by some Turing Machine.

3 Turing Machines

Hardware

Memory tape A long strip of paper, divided into cells:

(An assistant is ready to add paper on either end, as needed.)

Tape-reader At any given time, the reader is positioned on a cell of the memory tape and is able to perform any of the following functions:

- Read the symbol written on the cell
- Write a new symbol on the cell
- Move one cell to the left
- Move one cell to the right

Software

A finite list of **command lines**:

⟨current state⟩ ⟨current symbol⟩ ⟨new symbol⟩ ⟨direction⟩ ⟨new state⟩

Think of a command line as encoding the following instruction:

> If you are in ⟨current state⟩ and your reader sees ⟨current symbol⟩ written on the memory tape, replace ⟨current symbol⟩ with ⟨new symbol⟩. Then move one step in direction ⟨direction⟩, and go to ⟨new state⟩.

Operation

- Start out in state 0. Then carry out the following procedure, for as long as you are able:
  - Perform the instruction corresponding to the (first) command line that matches your current state and the symbol on which your reader is positioned.
  - Repeat.
- If you are unable carry out the procedure, halt.

4 A Turing Machine Simulator

http://morphett.info/turing/

5 Computing functions on a Turing Machine

Computability:

- For a computer program to **compute** f is for it to yield $f(n)$ as output whenever it is given n as input.

Turing Computabiity:

- M takes n ($n \in \mathbb{N}$) as **input** if it starts out with a tape that contains only a sequence of n ones (with the reader positioned at the left-most one, if $n > 0$).

- M delivers $f(n)$ as **output** if it halts with a tape that contains only a sequence of $f(n)$ ones (with the reader positioned at the left-most one, if $n > 0$).
- M **computes** a function $f(x)$ if and only if it delivers $f(n)$ as output whenever it is given n as input.

6 The Fundamental Theorem

The reason Turing Machines are so valuable is that it is possible to prove the following theorem:

Fundamental Theorem of Turing Machines A function from natural numbers to natural numbers is Turing-computable if and only if it can be computed by an ordinary computer, assuming unlimited memory and running time.

- One shows that every Turing-computable function is computable by an ordinary computer (given unlimited memory and running time) by showing that one can program an ordinary computer to simulate any given Turing Machine.
- One shows that every function computable by an ordinary computer (given unlimited memory and running time) is Turing-computable by showing that one can find a Turing Machine that simulates any given ordinary computer.

7 Church-Turing

Computer scientists tend to think that something stronger than the Fundamental Theorem is true:

Church-Turing Thesis A function is Turing-computable if and only if it can be computed algorithmically.

For a problem to be solvable **algorithmically** is for it to be possible to specify a finite list of instructions for solving the problem such that:

1. Following the instructions is guaranteed to yield a solution to the problem, in a finite amount of time.

2. The instructions are specified in such a way that carrying them out requires no ingenuity of any kind: they can be followed mechanistically.

3. No special resources are required to carry out the instructions: they could in principle be carried out by a machine built from transistors.

4. No special physical conditions are required for the computation to succeed (no need for faster-than-light travel, special solutions to Einstein's equations, etc).

8 Coding Turing Machines as Numbers

The Plan

Turing Machine → Sequence of symbols → Sequence of numbers → Unique number

Sequence of symbols → Sequence of numbers

$$\begin{array}{rcl} \text{“r”} & \rightarrow & 0 \\ \text{“}*\text{”} & \rightarrow & 1 \\ \text{“l”} & \rightarrow & 2 \end{array}$$

Sequence of numbers → Unique number

Codes the sequence $\langle n_1, n_2, \ldots, n_k \rangle$ as the number:

$$p_1^{n_1+1} \cdot p_2^{n_2+1} \cdot \ldots \cdot p_k^{n_k+1}$$

where p_i is the ith prime number.

(Treat any number that doesn't code a valid sequence of command lines as a code for the "empty" Turing Machine.)

24.118 Paradox and Infinity

Spring 2019