A Universal Computer

One-Slide Summary

- The **Turing machine** is a fundamental model of computation. It models input, output, processing and memory. A Turing machine has a finite state machine controller as well as an infinite tape. At each step it reads the current tape symbol, writes a new tape symbol, moves the tape head left or right one square, and moves to a new state in the finite state machine controller. Turing machines are universal: they are just as powerful as Scheme, Python, C, or Java.

- The **lambda calculus** is also a universal, fundamental model of computation. You can view it as “the essence of Scheme”.

Thursday December 6

- Presentations in OLS 009 at 5pm
  - Extra credit for attending.
  - I will provide pizza and soda.
  - Rough head-count?
- Lecture on December 6th
  - Officially: Reading Day
  - Optional Class, you Vote
    - Quantum Computing?
    - Romance Novels?
    - Aspect-Oriented Programming?
    - Free-form Question and Answer? Why car?

Where Are We?

- Last Time: Passwords, Security, etc.

Finite State Machine

- There are lots of things we can’t compute with only a finite number of states
- Solutions:
  - Infinite State Machine
    - Hard to describe and draw
  - Add an infinite tape to the Finite State Machine
    - We’ll do this instead.

Turing’s Explanation

“We have said that the computable numbers are those whose decimals are calculable by finite means. … For the present I shall only say that the justification lies in the fact that the human memory is necessarily limited.”
FSM + Infinite Tape

- Start:
  - FSM in Start State
  - Input on Infinite Tape
  - Pointer (= read/write head) to start of input
- Step (4 sub-steps each time):
  - Read one input symbol from tape
  - Write symbol on tape, and move L or R one square
  - Follow transition rule from current state
- Finish:
  - Transition to halt state

Turing Machine Hardware

- Infinite (“as much as you want”) Paper Tape
  - can read from and write to
  - but only finite time ...
- Finite Brain (set of rules)
  - modeling “Computers” (People)

Matching Parentheses

1: look for )
- If you don’t find one, the parentheses match, write a 1 at the tape head and halt.
2: look for ( (X, X, R (X, X, L)
- If you find it, replace it with an X (they matched)
- If you don’t find it, the parentheses didn’t match - end write a 0 at the tape head and halt
Matching Parentheses

Turing Machine

Turing Machine: FSM + Infinite Tape

Liberal Arts Trivia: Politics

Liberal Arts Trivia: Chemistry

A function \( f(w) \) has:

\[
\text{Domain} \quad D \quad w \in D
\]

\[
\text{Result Region} \quad S \quad f(w) \in S
\]
Integer Domain:
- Unary: 11111
- Binary: 101
- Decimal: 5

We prefer Unary representation:
- Easier to manipulate

A function may have many parameters:

Example:
- Addition function
  \[ f(x, y) = x + y \]
  \( x, y \) are integers

Definition:
A function \( f \) is computable if there is a Turing Machine \( M \) such that:

\[
\begin{array}{c|c|c}
\text{Initial Configuration} & \text{Final configuration} \\
\hline
\text{Input} w & \text{Output} f(w) \\
q_0 & q_f \text{ final state}
\end{array}
\]

\( w \in D \) Domain

Example
The function \( f(x, y) = x + y \) is computable

Turing Machine:
- Input string: \( x0y \) unary
- Output string: \( xy0 \) unary

Turing machine for function \( f(x, y) = x + y \):
Execution Example:

\[ x = 11 \quad \text{(2)} \]

\[ y = 11 \quad \text{(2)} \]

Final Result

\[ x + y \]

\[ \begin{array}{c|c|c|c|c} 
    & x & y & 0 & 1 \\
\hline
q_0 & 1 & 1 & 0 & 1 \\
q_1 & 1 & 1 & 1 & 1 \\
q_2 & 1 & 1 & 1 & 1 \\
q_3 & 1 & 1 & 1 & 0 \\
q_4 & 1 & 1 & 1 & 1 \\
\end{array} \]
Liberal Arts Trivia: American Lit

• This early-20th-century American author invited and wrote about cosmic horror, the idea that life is incomprehensible to human minds and that the universe is fundamentally alien. Thus, those who genuinely reason gamble with insanity. He was little read during life but is now regarded with Edgar Allen Poe as one of the most influential horror writers of the 20\textsuperscript{th} century.

Liberal Arts Trivia: Chemistry

• This highly flammable compound is formed by nitrating cellulose through exposure to nitric acid. It can be used as a propellant or low-order explosive. French chemist Paul Vieille invited the first practical smokeless powder for firearms and artillery ammunition from this; is has also been used as a film base for medical X-rays. It is commonly produced in introductory chemistry classes by treating cotton balls with nitric acid.

Liberal Arts Trivia: Taoism

• Name the group of legendary, undying xian from Chinese mythology. Each member’s power can be transferred to a tool that can give life or destroy evil. They are revered by Taoists, and include Royal Uncle Cao, Iron Crutch Li, Elder Zhang Guo, and Lu Tung-Pin. Many of them were said to practice neidan, or internal alchemy.
Describing Finite State Machines

TuringMachine ::= < Alphabet, Tape, FSM >
FSM ::= < States, TransitionRules, InitialState, HaltingStates >
States ::= { StateName* }  
InitialState ::= StateName  must be element of States
HaltingStates ::= { StateName* } all must be elements of States
TransitionRules ::= { TransitionRule* }
TransitionRule ::= < StateName, OneSquare, StateName, OneSquare, Direction >
Direction ::= L, R, #

Enumerating Turing Machines

• Now that we’ve decided how to describe Turing Machines, we can number them
• TM-5023582376 = balancing parens
• TM-57239683 = even number of 1s
• TM-3523796834721038296738259873 = Photomosaic Program
• TM-3672349872381692309875823987609823712347823 = Windows 7

Universal Turing Machine

P
Number of TM
Universal Turing Machine
I
Input Tape
Output Tape for running TM-P in tape I

Yes!

• People have designed Universal Turing Machines with
  - 4 symbols, 7 states (Marvin Minsky)
  - 4 symbols, 5 states
  - 2 symbols, 22 states
  - 18 symbols, 2 states
  - 2 states, 5 symbols (Stephen Wolfram)
• No one knows what the smallest possible UTM is

Church-Turing Thesis

• Any mechanical computation can be performed by a Turing Machine
• There is a TM-$n$ corresponding to every computable problem
• We can simulate any “normal” (classical mechanics) computer with a TM
  - If a problem is in polynomial time on a TM, it is in polynomial time on a PC, iPhone, etc.
  - But maybe not a quantum computer! (later class?)

Universal Language

• Is Java/Python/C as powerful as a Universal Turing Machine?
• Is a Universal Turing Machine as powerful as Java/Python/C?

Universal Language

• Is Java/Python/C as powerful as a Universal Turing Machine?
• Is a Universal Turing Machine as powerful as Java/Python/C?

Complexity in Python

• Special Forms
  - if, return, define, etc.
• Primitives
  - Numbers (infinitely many)
  - Booleans: True, False
  - Functions (+, -, and, or, etc.)
• Evaluation Complexity
  - Environments (more than ½ of our eval code)

λ-calculus

Alonzo Church, 1940
(LISP was developed from λ-calculus, not the other way round.)

\[
\text{term} = \text{variable} \\
| \text{term term} \\
| (\text{term}) \\
| \lambda \text{ variable . term}
\]

What is Calculus?

• In High School:
  \[
d/dx x^n = nx^{n-1} \quad \text{[Power Rule]} \\
d/dx (f + g) = d/dx f + d/dx g \quad \text{[Sum Rule]}
\]

  Calculus is a branch of mathematics that deals with limits and the differentiation and integration of functions of one or more variables...
Surprise Liberal Arts Trivia

- This branch of mathematics involving symbolic expressions manipulated according to fixed rules takes its name from the diminutive form of calx/calcis, the Latin word for rock or limestone. The diminutive word thus means “pebble”: in ancient times pebbles were placed in sand and used for counting using techniques akin to those of the abacus.

Real Definition

- A calculus is just a bunch of rules for manipulating symbols.
  - Latin word calx meaning pebble ...
- People can give meaning to those symbols, but that’s not part of the calculus.
- Differential calculus is a bunch of rules for manipulating symbols. There is an interpretation of those symbols corresponds with physics, slopes, etc.

Lambda Calculus

- Rules for manipulating strings of symbols in the language:
  \[ \text{term} = \text{variable} \]
  \[ \mid \text{term} \text{term} \]
  \[ \mid (\text{term}) \]
  \[ \mid \lambda \text{ variable} . \text{ term} \]
- Humans can give meaning to those symbols in a way that corresponds to computations.

Why?

- Once we have precise and formal rules for manipulating symbols, we can use it to reason with.
- Since we can interpret the symbols as representing computations, we can use it to reason about programs.

Evaluation Rules

\[ \alpha \text{-reduction (renaming)} \]
\[ \lambda y. M \Rightarrow_\alpha \lambda v. (M \text{ [each } y \text{ replaced by } v]) \]
where \( v \) does not occur in \( M \).

\[ \beta \text{-reduction (substitution)} \]
\[ (\lambda x. M)N \Rightarrow_\beta M \text{ [each } x \text{ replaced by } N \]