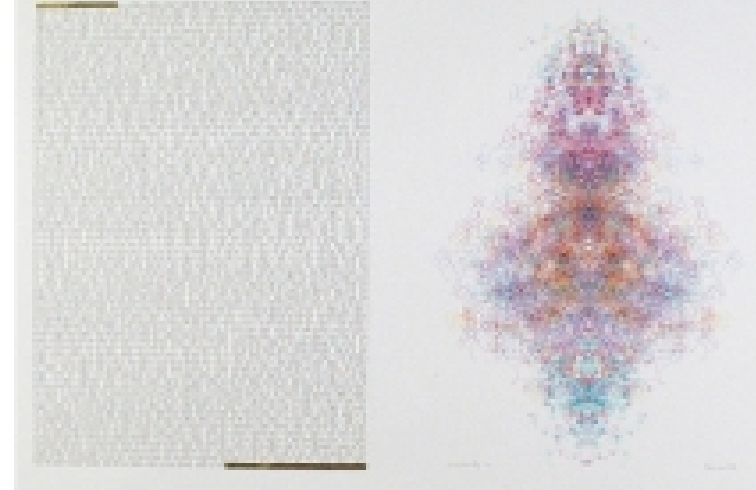




## Yes!

- 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



Manchester Illuminated Universal Turing Machine, #9  
from <http://www.vorosiko.com/manchester/manchester.html>

## Church-Turing Thesis

- Any mechanical computation can be performed by a Turing Machine
- There is a TM- $n$  corresponding to every decidable problem
- We can simulate one step on any "normal" (classical mechanics) computer with a constant number of steps on a TM:
  - If a problem is in P on a TM, it is in P on an iMac, CM5, Cray, Palm, etc.
  - But maybe not a quantum computer! (later class)

## Universal Language

- Is Scheme as powerful as a Universal Turing Machine?
- Is a Universal Turing Machine as powerful as Scheme?

## Complexity in Scheme

- Special Forms
    - if, cond, define, etc.
  - Primitives
    - Numbers (infinitely many)
    - Booleans: #t, #f
    - Functions (+, -, and, or, etc.)
  - Evaluation Complexity
    - Environments (more than 1/2 of our eval code)
- Can we get rid of all this and still have a useful language?

If we have lazy evaluation and don't care about abstraction, we don't need these.

Hard to get rid of?

## $\lambda$ -calculus

Alonzo Church, 1940

(LISP was developed from  $\lambda$ -calculus, not the other way round.)

$term = variable$

|  $term term$

|  $(term)$

|  $\lambda variable . term$

## 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...

## Real Definition

- A *calculus* is just a bunch of rules for manipulating symbols.
- 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:

$$\text{term} = \text{variable}$$

$$| \text{term term}$$

$$| (\text{term})$$

$$| \lambda \text{variable} . \text{term}$$

- 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

$\alpha$ -reduction (renaming)

$$\lambda y. M \Rightarrow_{\alpha} \lambda v. (M [y \alpha v])$$

where  $v$  does not occur in  $M$ .

$\beta$ -reduction (substitution)

$$(\lambda x. M)N \Rightarrow_{\beta} M [x \alpha N]$$

## Reduction (Uninteresting Rules)

$$\lambda y. M \rightarrow \lambda v. (M [y \alpha v])$$

where  $v$  does not occur in  $M$ .

$$M \rightarrow M$$

$$M \rightarrow N \Rightarrow PM \rightarrow PN$$

$$M \rightarrow N \Rightarrow MP \rightarrow NP$$

$$M \rightarrow N \Rightarrow \lambda x. M \rightarrow \lambda x. N$$

$$M \rightarrow N \text{ and } N \rightarrow P \Rightarrow M \rightarrow P$$