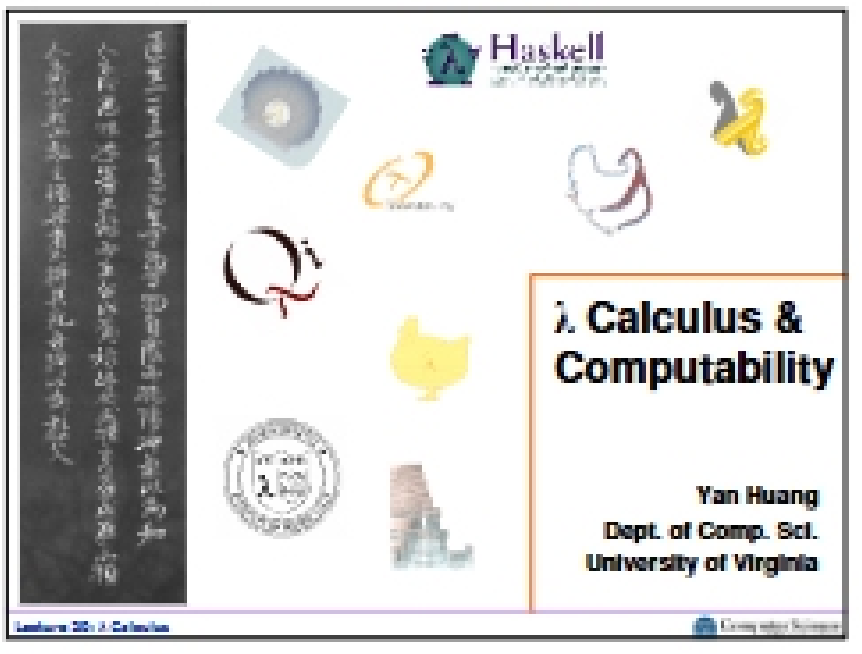


Review of the Turing Machine

- Formalism $(Q, \Gamma, \Sigma, \delta, q_{start}, q_{accept}, q_{reject})$
- Abstract Problems
- Language Problems
- Computation
- Computability vs. Decidability

Today we are looking at a completely different formal computation model – the λ-Calculus!

Calculus

- What is calculus?
  - **Calculus** is a branch of mathematics that includes the study of limits, derivatives, integrals, and infinite series.
- Examples

$d(uv) = u(du) + u(dv)$ The product rule

$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx}$ The chain rule

Real Definition

- ***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, geometry, etc.

λ Calculus Formalism (Grammar)

- Key words: λ . ( ) **terminals**
- *term* → *variable*
  | (*term*)
  | λ *variable* . *term*
  | *term term*

Humans can give meaning to those symbols in a way that corresponds to computations.

λ Calculus Formalism (rules)

- Rules

α-reduction (renaming)

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

where v does not occur in M.

β-reduction (substitution)

$(\lambda x.\ M)N \Rightarrow_\beta M\ [\ x \mapsto N\ ]$

Try Example 1, 2, & 3 on the notes now!

Replace all x's in M with N

Free and Bound variables

- In λ Calculus all *variables* are local to function definitions
- Examples
  - $\lambda x.xy$

    x is bound, while y is free;
  - $(\lambda x.x)(\lambda y.yx)$

    x is bound in the first function, but free in the second function
  - $\lambda x.(\lambda y.yx)$

    x and y are both bound variables. (it can be abbreviated as $\lambda xy.yx$)

Be careful about β-Reduction

- $(\lambda x.\ M)N \Rightarrow M\ [\ x \mapsto N\ ]$

Try Example 4 on the notes now!

Replace all x's in M with N

If the substitution would bring a free variable of N in an expression where this variable occurs bound, we rename the bound variable before the substitution.

Computing Model for λ Calculus

- ***redex***: a *term* of the form $(\lambda x.\ M)N$

  Something that can be β-reduced
- An expression is in ***normal form*** if it contains no redexes (*redices*).
- To evaluate a lambda expression, keep doing reductions until you get to *normal form*.

β-Reduction represents all the computation capability of Lambda calculus.

Another exercise

$(\lambda f.\ ((\lambda x.\ f\ (xx))\ (\lambda x.\ f\ (xx))))\ (\lambda z.z)$

Possible Answer

$(\lambda f.\ ((\lambda x.f\ (xx))\ (\lambda x.\ f\ (xx))))\ (\lambda z.z)$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.\ (\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda z.z)\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda z.z)\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta$ ...

Alternate Answer

$(\lambda f.\ ((\lambda x.f\ (xx))\ (\lambda x.\ f\ (xx))))\ (\lambda z.z)$

$\rightarrow_\beta\ (\lambda x.(\lambda z.z)(xx))\ (\lambda x.\ (\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda x.xx)\ (\lambda x.(\lambda z.z)(xx))$

$\rightarrow_\beta\ (\lambda x.xx)\ (\lambda x.xx)$

$\rightarrow_\beta\ (\lambda x.xx)\ (\lambda x.xx)$

$\rightarrow_\beta$ ...

Be Very Afraid!

- Some λ-calculus terms can be β-reduced forever!
- The order in which you choose to do the reductions might change the result!

Take on Faith

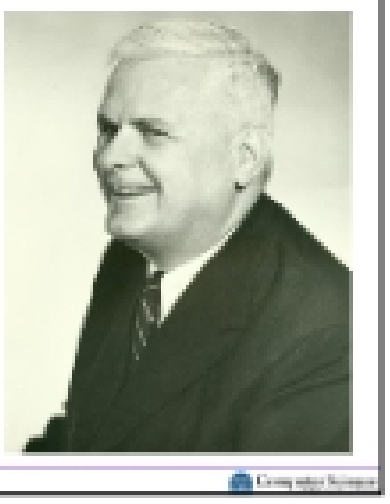
- All ways of choosing reductions that reduce a lambda expression to normal form will produce the same normal form (but some might never produce a normal form).
- If we always *apply the outermost lambda first*, we will find the normal form if there is one.
  - This is *normal order reduction* – corresponds to normal order (lazy) evaluation

Alonzo Church (1903~1995)

Lambda Calculus

Church-Turing thesis

If an algorithm (a procedure that terminates) exists then there is an equivalent Turing Machine or applicable λ-function for that algorithm.



Alan M. Turing (1912~1954)

- Turing Machine
- Turing Test
- Head of Hut 8

Advisor:
Alonzo Church

Equivalence in Computability

- λ Calculus ↔ Turing Machine
  - (1) Everything computable by λ Calculus can be computed using the Turing Machine.
  - (2) Everything computable by the Turing Machine can be computed with λ Calculus.

Simulate λ Calculus with TM

- The initial tape is filled with the initial λ *expression*
- Finite number of reduction rules can be implemented by the finite state automata in the Turing Machine
- Start the Turing Machine; it either stops – ending with the λ *expression* on tape in normal form, or continues forever – the β-reductions never ends.