Lambda Calculus

Have you heard of_

(A) Yes

(B) Huh?

Your Favorite Language

Probably has lots of features:

- Assignment ($x = x + 1$)
- Booleans, integers, characters, strings,
- Conditionals
  - Loops
  - return, break, continue
- Functions
- Recursion
- References / pointers
- Objects and classes
- Inheritance
- ...

Which ones can we do without?

What is the smallest universal language?
What is computable?

Before 1930s

Informal notion of an effectively calculable function:

can be computed by a human with pen and paper, following an algorithm

1936: Formalization
What is the **smallest universal language**?

Alan Turing

1930s

Alonzo Church
The Next 700 Languages

Alonzo Church

Peter Landin
Whatever the next 700 languages turn out to be, they will surely be variants of lambda calculus.

Peter Landin, 1966

The Lambda Calculus

Has one feature:

- Functions
No, really

- Assignment: \( x = x + 1 \)
- Booleans, integers, characters, strings, ...
- Conditionals
- Loops
- `return`, `break`, `continue`
- Functions
- Recursion
- References/pointers
- Objects and classes
- Inheritance
- Reflection

More precisely, only thing you can do is:

- **Define** a function
- **Call** a function
Describing a Programming Language

- Syntax: what do programs look like?
- Semantics: what do programs mean?
  - Operational semantics: how do programs execute step-by-step?

Syntax: What Programs Look Like
Programs are expressions $e$ (also called \(\lambda\)-terms) of one of three kinds:

- **Variable**
  - $x$, $y$, $z$

- **Abstraction** (aka nameless function definition)
  - $\lambda x \rightarrow e$
  - $x$ is the formal parameter, $e$ is the body
  - “for any $x$ compute $e$”

- **Application** (aka function call)
  - $e_1 e_2$
  - $e_1$ is the function, $e_2$ is the argument
  - in your favorite language: $e_1(e_2)$

(Here each of $e$, $e_1$, $e_2$ can itself be a variable, abstraction, or application)
Examples

- The identity function (id)
- "for any x compute x"

- A function that returns (id)

- A function that applies its argument to id

QUIZ

Which of the following terms are syntactically incorrect?

A. $(\lambda x \to x) \to y$
B. $x \to (x \times x)$
C. $x \times y \times (y x)$
D. A and C
E. All of the above
E. all of the above

\[(\lambda x \to x) x \text{ valid}\]
\[(\lambda x \to (x x)) \text{ valid}\]

---

**Examples**

\(\lambda x \to x\)  
--- *The identity function*  
--- *("for any x compute x")*

\(\lambda x \to (\lambda y \to y)\)  
--- *A function that returns the identity function*

\(\lambda f \to f (\lambda x \to x)\)  
--- *A function that applies its argument*  
--- *to the identity function*

---

How do I define a function with two arguments?

- e.g. a function that takes \(x\) and \(y\) and returns \(y\)?
A function that returns the identity function

OR: a function that takes two arguments

and returns the second one!

How do I apply a function to two arguments?

- e.g. apply \( \lambda x \rightarrow (\lambda y \rightarrow y) \) to apple and banana?