

Welcome to CS 704

The greatest class on the principles of programming languages since 2015

Aws Albarghouti
PhD from UToronto

Does program P satisfy some property Q ?

How do we enforce
 P satisfies Q ?

How can we check that P
satisfies Q ?

deadlock-free
program equivalence
memory safety issues

space $\left[\begin{array}{l} \text{termination} \\ O(n^2) \end{array} \right] / \text{constant time}$

fairness constraints

privacy

① By programming language design + types

② By static analysis

take a program and analyze it

Ⓐ

Ⓑ

Systems / FS

Robotics / robotics / visualization

ML / ML for code / ML for verification /
verifying ML

Theory / complexity of certain verification
problems

DB / many connections

Optim / used in static analysis

Net / network verification + programming

Calculus

a model of computation developed by Alonzo Church in the 1920s

two constructs : function application
// definition

Turing showed that λ calculus \equiv Turing machines

In mid 1960s, Peter Landin

"The next 700 programming languages"

- Correspondence between ALGOL 60 and Church's λ calculus

McCarthy developed LISP

① a simple, stripped-down FPL

② we can reason about such programs

Syntax

Everything is a function

$t = x$ variable

term

$\lambda x. t$ abstraction

$t \ t$ applying t to t

$\text{def } f(x) \{t\}$

Given $\lambda x. t$

we say x is bound in t

λx is a binder whose scope is t

a variable that is not bound is free

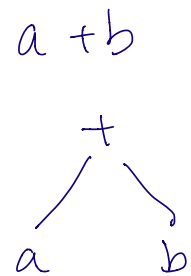
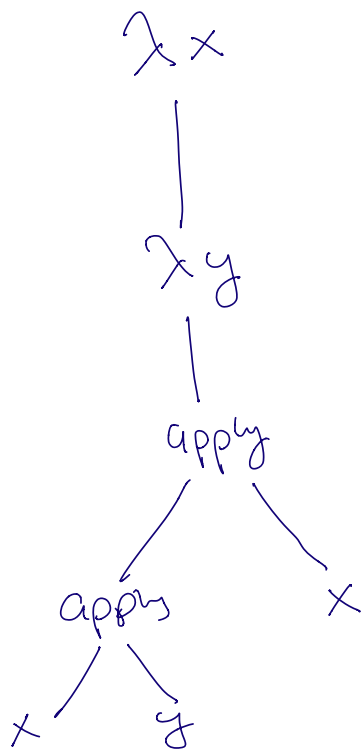
$(\lambda x. x)$ identity combinator

$\text{def } f(x)$

return x

$$\frac{\lambda x. \lambda y. \underline{x} \ \underline{y} \ \underline{x}}{\lambda x. (\lambda y. (\underline{x} \ y) \ \underline{x})}$$

Bodies of abstractions extend to the right
Application is left associative



Semantics

abstraction $\lambda x. t$ as a one argument function

application $M N$ applying M to "data" N

Simplification = computation

$$1 + 2x + 3x \\ = 1 + 5x$$

$$\underbrace{(\lambda x. t)}_{\text{function}} \underbrace{t_2}_{\text{input}} \xrightarrow{\text{simplification}} [x \mapsto t_2] t$$

$$\begin{array}{ccc} \text{def } f(x) & \xrightarrow{f(10)} & 10+1 \rightarrow 11 \\ \text{return } \underbrace{x+1} & & \underbrace{[x \mapsto 10]} \end{array}$$

$$\begin{array}{ccc} (\lambda x. \underline{x}) y & \longrightarrow & y \\ \downarrow & & \downarrow \\ (\lambda \underline{x}. \underline{x} (\lambda \underline{x}. \underline{x})) (\underline{u} \underline{r}) & & x \cup y \\ \uparrow & & \cup \\ \longrightarrow & (\underline{u} \underline{r}) (\lambda x. x) & \end{array}$$

β -reduction

