Welcome to CS 704

The greatest class on the principles of programming languages since 2015

Aws Albarghouthi
PhD from UToronto

Does program P satisfy some property φ?

deadlock-freed
program equivalence
memory safety issues
termination
space [$O(n^2)$ / constant time
fairness constraints
privacy

How do we enforce P satisfies φ?

How can we check that P satisfies φ?

(1) By programming language design + types (A)

(2) By static analysis (B)
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

λ calculus

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

two constructs : function application
" definition

Turing showed that λ calculus ≡ 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

term

$t = x$ variable

$\quad | \ \lambda x . \underline{t}$ abstraction

$\quad | \ t \ t$ applying t to t

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

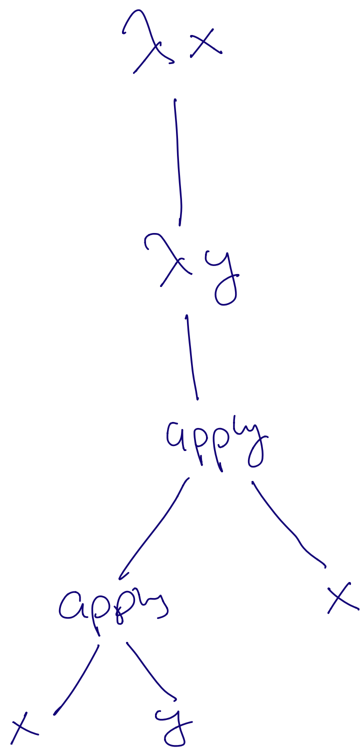
```
def f(x)
    return x
```

$$\underbrace{\lambda x}\ .\ \underbrace{\lambda y\ .\ \underline{x}\ \ y\ \ \underline{x}}$$

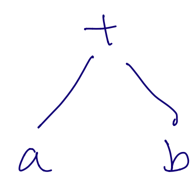
$$\lambda x\ .\left(\lambda y\ .\ \underbrace{(x\ \ y)}\ \underbrace{x}\right)$$

Bodies of abstractions extend to the right

Application is left associative



a + b



Semantics

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

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

simplification = computation

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

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

def f(x)
return x+1

$$\xrightarrow{f(10)} \quad 10+1 \longrightarrow 11$$

$$[x \mapsto 10]$$

$$(\lambda x . x)\ y \longrightarrow y$$

$$(\lambda x . x (\lambda x . x)) \ (u\ r)$$

$$x_u y$$

$$u$$

$$\longrightarrow (u\ r)\ (\lambda x . x)$$

$\boxed{\beta\text{-reduction}}$

$$(\lambda x.\, x\ (\lambda x.\, x))\quad (u\ r)$$

apply

apply

apply