CS3110 Guest Lecture Tue Oct 28, 2014

by Robert Constable – taught Fall 2013 offering

## Programming Courses circa 2020

To predict what this course will be, it helps to know how it got here, to OCaml. It is a course with MIT imprint and taught by Cornell professors with "MIT genes" – Hottenbacher, Zabih, Myers, ....

They learned from Abelson & Sussman

Structure and Interpretation of
Computer Programs
MIT Press, 1985

The course was grounded in Lisp and Scheme, strongly functional programming. Then ML took over, it had Cornell genes in Bob Harper. OCaml and SML are two dialects of Classic ML from Edinburgh LCF.

John McCarthy

A Basis for a Mathematical Theory
of Computation
1963

Lisp Programmer's Manual MIT 1960

CS 3110 Tue Oct 28, 2014

Lisp is based on the Lambda Calculus of Alonzo Church.

Church was the first and perhaps greatest American logician. He discovered the first provably unsolvable mathematics problem, and for Church's Thesis he is widely known. Both done in 1935.

He attracted Turing to visit Princeton and Gödel to live in Princeton and work at the IAS, the "Institute."

He invented the Lambda Calculus as part of his foundational theory of mathematics, based on functions and types instead of sets. It seems that via computer science, he might prevail in due course.

OCaml has roots in this work, and the logic associated with OCaml and implemented in the Coq proof assistant is based on

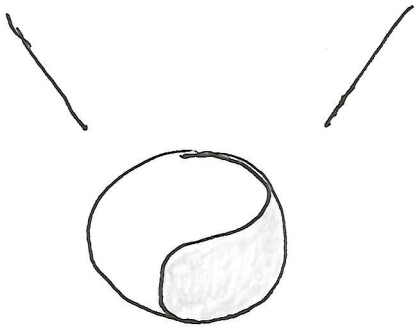
CONSTRUCTIVE TYPE THEORY

as is the logic implemented by the Cornell proof assistant, Nuprl.

CS3110 Guest Lecture Tue Oct 28, 2014

To see what the future might be, think of these pairs

| | |
|---|---|
| OCaml – language | ML – language |
| Coq – type theory | Nuprl – logic |



the yin and yang of 2020 PL

We will look at a small piece of the language/logic connection. The types look similar, but there is a critical difference

Base types

| Logic | | Language |
|---|---|---|
| $\mathbb{Z}$ | int | |
| $\mathbb{N}$ | nat | |
| $\mathbb{B}$ | bool | |
| Unit | unit | |
| A List | 'a list | |
| $A \to B$ | 'a -> 'b | |
| $A \times B$ | 'a * 'b | |
| $A \vee B$ | A \| B | |

CS 3110 Guest Lecture Tue Oct 28, 2014

The logical types are <u>total types</u>, all elements have canonical forms, e.g. values.

$$\mathbb{Z} = \{ 0, 1, -1, 2, -2, \cdots \}$$

In OCaml, the types are <u>partial</u>. We can make expressions of the type that fail to have a value. For example, define this function

```
let rec loop (x int) int =
    loop(x) = loop(x).
```

Then loop(0) belongs to int, but it diverges.

The logical types can be <u>dependent</u>, e.g.

$x : A \multimap B(x)$

$x : A * B(x)$

$\{ x : A \mid B(x) \}$

We can add these to an <u>OCaml specification language</u>. See CS 3110 Fall 13 Lectures 15, 17.

A neat dependent type is $(x : \alpha$ where $b(x))$ for $b : \alpha \multimap$ bool. We used this to define

```
(x: int where not x=0)   and
(x: int where 0 <= x)
```