CS3110 Fall 2013 Lecture 13: Modules, Functors, Reals (10/10)

Robert Constable

1 Lecture Plan

- announcements
- reading
  Hickey book
  Kozen Lecture 9
  PS4 write up
- more on modules
- functors / parameterized modules
- illustrating with $\mathbb{R}$
- computing with $\mathbb{R}$ and proofs about $\mathbb{R}$
- constructing an irrational number

2 Examples from Michael George's Lecture 12

Modules are collections of types and functions that are useful together for specifying and implementing commonly needed functionality in a system.

They are especially important for organizing a program or system with clear subcomponents. They could be used to organize course material as well. They are key to software engineering.

The OCaml modules are called *structures* in SML, and that name was inspired by the notion of **algebraic structures** in mathematics.

Proof assistants such as MetaPRL use theories to capture the structure idea. MetaPRL is implemented in OCaml.

As Mike noted in his lecture, modules are key for specifying tasks. Recall his quote:

> "Write specs for somebody else because:
> tomorrow you will be somebody else."

3 Mathematical Structures vs OCaml Modules

The ambitious "encyclopedia of mathematics," ***The Elements of Mathematics***, written by several eminent French mathematicians under the pseudonym Nicolas Bourbaki, is organized around mathematical structures, some are *algebraic* structures some *topological* structures, all based on set theory.

A Real Analysis module, as presented in H.L. Royden's *Real Analysis* could look like this module.

```
module type Reals_Type = sig
  type reals
  val 0
  val 1
  val add : reals -> reals -> reals
  val mult :
  val sub :
  val div :
end
```

CONTENTS

OF

THE ELEMENTS OF MATHEMATICS SERIES

4 Implementing the Reals with a module

```
module Reals : Reals_Type = struct
  type reals = int -> int * int
  let 0 = (0, 1)
  let 1 = (1, 1)
  let add = fun n -> fun m -> ...
  let mult = fun n -> fun m -> ...
end
```

There are other implementations we can build

```
module Reals2 : Reals_Type = struct
  type reals = int stream
  let 0 = [0; 0; ... ]
  let 1 = [1; 0; ... ]
end
```

```
module Reals3 : Reals_Type = struct
  type reals = int * int
  (* Dedekind Cuts *)
  let 0 = ...
end
```

5 Using functors (parameterized modules) to implement reals

A functor is a module **parameterized** by a module. We could parameterize the Reals by the ways of creating the elements of *real*.

```
module MakeReal (Values : Real_Sig)
  type Values
  val add : Values -> ...
  .
  .
  .
end
```

6 Diagonalization over $\mathbb{N} \to \mathbb{B}$

Suppose $e : \mathbb{N} \to (\mathbb{N} \to \mathbb{B})$ and for all f in $\mathbb{N} \to \mathbb{B}$ there is an $n_f \in \mathbb{N}$ such that $e(n_f) = f$, i.e., the map e is *onto* $\mathbb{N} \to \mathbb{B}$ or *surjective.*

Then define the diagonal function

$d(n) = \overline{e(n)(n)}$, where $\overline{b}$ is the complement of the Boolean, e.g. $\overline{true} = false$.

This d is a function $\mathbb{N} \to \mathbb{B}$, so there is an n_d such that $e(n_d) = d$

BUT $d(d) = \overline{e(d)(d)} = \overline{d(d)}$. This is a contradiction.

$$
\begin{array}{llll}
\overline{f_0(0)} & f_0(1) & & \\
f_1(0) & \overline{f_1(1)} & & \\
f_2(0) & f_2(1) & \overline{f_2(2)} & \\
 & & & \\
f_d(0) & \cdots\cdots\cdots & & \overline{f_d(d)} \\
 & & & \ddots
\end{array}
$$

7 Constructing an irrational number by diagonalization

Theorem 1 (Bishop '67)

> Let $\{a_n\}$ be a sequence of real numbers. Let x_0 and y_0 be real numbers, $x_0 < y_0$. Then we can construct a real number x such that
>
> (a) $x_0 \leq x \leq y_0$ and
> (b) $x \neq a_n$ for any n in $\mathbb{Z}^+$.

Example

> Take $x_0 = 0$ and $y_0 = 1$. Let $\{a_n\}$ be the rational numbers (injected into $\mathbb{R}$) in the interval $[0, 1]$.
>
> In this case x is an irrational number in $[0, 1]$.

How to do this? We construct $\{x_n\}, \{y_n\}$ to "miss every rational" and satisfy

$*$ $x_0 \leq x_n \leq x_m < y_m \leq y_n \leq y_0$ and

$**$ $x_n > a_n$ OR $y_n < a_n$ for all $n \geq 1$

$***$ $y_n - x_n < 1/n$

Construct $\{x_n\}, \{y_n\}$ inductively (e.g. define the numbers recursively).

assume $x_0, x_1, \ldots, x_{n-1}$ and $y_0, y_1, \ldots, y_{n-1}$ are constructed.

Notice, either $a_n > x_{n-1}$ or $a_n < y_{n-1}$ by $**$.

> If $a_n > x_{n-1}$, pick a rational x_n such that
> $x_{n-1} < x_n < min\{a_n, y_{n-1}\}$ and y_n such that
> $x_n < y_n < min\{a_n, y_{n-1}, x_n + 1/n\}$

If $a_n < y_{n-1}$, then let y_n be any rational such that $max\{a_n, x_{n-1}\} < y_n < y_{n-1}$ and x_n such that $max\{a_n, x_{n-1}, y_n - 1/n\} < x_n < y_n$.

From $*$ and $***$ we have

$|x_m - x_n| = x_m - x_n < y_n - x_n < 1/n \quad (m \geq n)$

Likewise $|y_m - y_n| < 1/n$ for $m \geq n$.

Thus $\{x_n\}$, $\{y_n\}$ are reals, by $***$ they are equal.

If $a_n < x_n$, then $a_n < x$

If $a_n > y_n$, then $a_n > y$. Hence $a_n \neq x$ for all n.