Functors

Prof. Clarkson
Fall 2018

Today’s music: "Uptown Funk"
by Mark Ronson feat. Bruno Mars
Attendance question

Fill in the blank to complete the analogy:

A(n) ___________ is to a structure
as
a type is to a value

A. abstract type
B. module
C. signature
D. compilation unit
Review

Previously in 3110:
• modules, structures, signatures, abstract types
• aspects of modularity: namespaces, abstraction

Today:
• code reuse: functors and includes
Review

Encapsulation: hide parts of module from clients

```plaintext
module type Stack = sig
  type 'a t
  val push : 'a -> 'a t -> 'a t
end

module ListStack : Stack = struct
  type 'a t = 'a list
  let push x s = x::s
end
```

type constructor t is abstract: clients of this signature know the type exists but not what it is
Review

Encapsulation: hide parts of module from clients

```ocaml
module type Stack = sig
  type 'a t
  val push : 'a -> 'a t -> 'a t
end

module ListStack : Stack = struct
  type 'a t = 'a list
  let push x s = x::s
end
```

module is sealed: all definitions in it except those given in signature Stack are hidden from clients
Consider this code:

```ocaml
module type Stack =
  sig
    type 'a t
    val empty : 'a t
    val push : 'a -> 'a t -> 'a t
  end

module ListStack : Stack =
  struct
    type 'a t = 'a list
    let empty = []
    let push x s = x :: s
  end
```

Which of the following expressions will type check?

A. `Stack.empty`
B. `ListStack.push 1 []`
C. `fun (s:ListStack) -> ListStack.push 1 s`
D. All of the above
E. None of the above
FUNCTORS

(funk you up?)

Cornell (CS) funk you up:
https://www.youtube.com/watch?v=Au56Ah92Ulk
Functors are "functions" on structures
Matching

A structure **Struct** matches a signature **Sig** if:

1. **Struct** defines every declaration in **Sig**

2. The type of each definition in **Struct** is the same as or **more general** than the declaration in **Sig**
Question

Which structure matches?

A. struct
   let c = [42]
   end

B. struct
   type t = int
   let c = [42]
   end

C. struct
   type t = int
   let c = []
   end

D. two of the above
E. all three of the above
Re-using code

PARAMETERIZED MODULE: TEST SUITE
Re-using code

PARAMETERIZED MODULE: MAP

INCLUDES
Code reuse from includes

• Interface inheritance
• Implementation inheritance
Upcoming events

• N/A

This is higher-order funk.

THIS IS 3110