Exploring Opaque Type Inference

Mike Furr

furr@cs.umd.edu
Modules are a powerful system for splitting up large programs.

- allow separate compilation
- break program components into logical pieces
- enforce interfaces between these components
Modules

Modules are made up of two parts:

- **signatures**: exported interface of the module
- **structures**: the actual implementation
Consider a module implementing complex numbers:

```ocaml
module Complex = struct
  type t = float * float
  let add c1 c2 =
    (fst c1 + fst c2, snd c1 + snd c2)
end

what should the signature be?
```
Signatures

it could have a transparent signature:

```plaintext
module Complex : sig
  type t = float * float
  val add: (float*float) -> (float*float)
  -> (float*float)
end

let add c1 c2 =
  (fst c1 + fst c2, snd c1 + snd c2)
end
```
Signatures

or perhaps an opaque one:

module Complex : sig
  type t
  val add : t -> t -> t
end = struct
  type t = float * float
  let add c1 c2 =
    (fst c1 + fst c2, snd c1 + snd c2)
end
which is better?
Opaque Interfaces

For data structures, this is often clear:

```
module IntSet =
struct
  type t
  val empty: t
  val add: int -> t -> t
  [...]
end
```

But what about larger, more complex modules? Not always obvious how opaque a module should be
Opaque Interfaces

Harper and Lillibridge remark:

• “fully opaque interfaces are awkward to use in practice since too much information is hidden”
• “fully transparent interfaces lead to excessive interdependencies.”

Where’s a good middle ground?
Inference to the Rescue!

My project is to develop a tool that,

• given a module implementation and code which uses that module, what is the most opaque signature possible?

• If the signature is not fully opaque, where in the invoking code is it used transparently?

• However, usefulness depends on the invoking code
Summary

Use opaque inference to

• make explicit the dependencies between modules.

• help programmers organize their code
Use opaque inference to

- make explicit the dependencies between modules.
- help programmers organize their code
- do I really need an excuse to hack on ML?
Questions?