How Do Statically-Typed Functional Programmers Author Code?

Justin Lubin

Why care?

Evidence-Based Tools | Eased Community Onboarding | Understudied Audience

(a)
```
let
  hardBit =
    Debug.todo “”
in
« body »
```

(b)
```
let
  hardBit =
    Debug.todo “”
in
« body, using hardBit »
```

(c)
```
let
  hardBit : Int -> Maybe Int
  hardBit =
    Debug.todo “”
in
« body, using hardBit »
```

(d)
```
let
  hardBit : Int -> Maybe Int
  hardBit =
    « implementation of hardBit »
in
« body, using hardBit »
```

① Type Construction

They start by ***iteratively constructing types*** to model their problem domain and ***encode design decisions***.

Feeling Odd When Types Are Amiss | Iteratively Constructing Types and Expressions

```
-- Set (Position, Position)
-- Set (Position, Tile, Maybe Color, Position)
-- Board, Set (Maybe Color, Position)
-- Board
```

② Focusing Techniques

They leverage types to help themselves ***focus*** by relying on the ***compiler as an assistant*** and using these types to help ***decompose their tasks***.

Relying on Compiler as Assistant | Using Types to Reduce Context

> *I kind of understand, maybe, what I've got, so I can do some **bottom-up exploration**.*
>
> *And I pretty much know where I want to be (which is the type signatures), and it allows me to do some **top-down programming**.*
>
> *And when it's not clear to me how to connect the two, and … I'm not feeling super productive or I feel stuck trying to think from one end,*
>
> ***I just switch to the other to try to glean some more context.***

③ Hierarchical and Opportunistic Programming

When faced with difficult or unknown problem domain, they ***complement*** this ***systematic*** style of programming with an ***exploratory*** one, the details of which are ***highly varied***.

Opportunistic Strategies | Interplay of Hierarchical and Opportunistic Programming

```
type Exp = Add Exp Exp | ... | Div Exp Exp

eval : Exp -> Int
eval e =
  case e of
    Add left right ->
      eval left + eval right
    ...
    Div numerator [0] ->
      raise DivisionByZero
```

④ Mental Models and Expressing Intent

No matter the style of programming, they have ***diverse mental models*** and express their intent in many ways, ***not all of which produce valid code***.

Diversity of Mental Models | Reasoning About Code Essence | Signaling and Executing Intent