Tiny Idris Program Synthesis - 408 Progress Presentation

Scott Mora

01-2021

Structure

Introduce the structure of functional programs

Look at synthesis systems.

Look at the current progress.

Short demo

Functional Programming

Uses the structure of data to proceed using pattern matching definitions.

Lends itself to immutable data structures, limiting code with unintended side effects.

Functions can be composed allowing for more concise code.

Data Type Declarations

Booleans

```
data Bool : Type where
  True : Bool
  False : Bool
```

Lists

```
data IntegerList : Type where
   Nil : IntegerList
   Cons : Integer -> IntegerList -> IntegerList
```

Type Signatures

isEmpty

```
isEmpty : IntegerList -> Bool
```

max

```
max : IntegerList -> Integer
```

Function Definitions

isEmpty

```
isEmpty : IntegerList -> Bool
isEmpty Nil       = True
isEmpty Cons i is = False
```

Polymorphism

Lists

```
data List : Type -> Type where
    Nil  : List a
    Cons : a -> List a -> List a
```

Dependent Types

Numbers

```
data Num : Type where
  Zero : Num
  OnePlus : Num -> Num
```

Vect

```
data Vect : Num -> Type -> Type where
   Nil  : Vect Zero a
   Cons : a -> Vect n a -> Vect (OnePlus n) a
```

Synthesis

Similar structures can lead to repetitive code.

Boilerplate can lead to human error.

We want to search all constructable terms.

The strong type system can be used to reduce the possible terms to a searchable amount.

Previous Systems

Began using little type information.

Progressively use more, following similar patterns.

Specialised tools such as Leon, Myth, Synquid, ReSyn.

Proof search tools implemented in languages such as Agda, Idris and Coq.

Implementation

Checks if any local variables are suitable.

Checks if any data constructors will result in a valid term, and attempts to synthesise arguments.

Checks if any functions would result in a valid term and synthesises arguments if so.

Testing

Tests have been divided by common structures.

Lists, Vectors, Equality, Sorting and AVL trees.

Synthesis can be called individually or in batches based on files.

Future

Improve base functionallity.

Introduce case splitting definitions.

Introduce better heuriustics for choosing an acceptable solution.