

Idris

A Programming Language with Dependent Types

Sumner Evans and Sam Sartor
2018-03-22

Mines Linux Users Group
<https://lug.mines.edu>

What is Idris?

"Idris is a general purpose pure functional programming language with dependent types."

The Idris Website

- **Version 0.1.3** of Idris was released in December of 2009.
- **Version 1.2.0** is the latest stable release and was released on January 9, 2018.
- Idris was named after the singing dragon in the 1970s UK children's television program *Ivor the Engine*.
- Idris development is led by Edwin Brady at the University of St. Andrews.

The Obligatory Picture of This Madman



Properties of Idris

- Idris can be **interpreted, transpiled, or compiled**.
- Idris is **statically typed**.
- Idris is **strongly typed**.
- Idris has **first class functions**, much like Haskell.
- Idris has **first class types**. This means that types can be treated as data. In fact, types are the *only* type of data.

Idris Features

Idris is a general purpose language, and thus it has a lot of features. We will focus on the following aspects of the language.

- Haskell-like Syntax
- Dependent Types
- Proof Assistant

Dependent Types

Idris Syntax: Function Signatures

The Idris function signature syntax is *very* similar to the Haskell function signature syntax. Here are a few examples of Idris function signatures:

```
even : Nat -> Bool
add  : Nat -> Nat -> Nat
foo  : (a:Nat) -> (b:Nat) -> a = b
bar  : (a:Nat) -> (b:Nat) -> LTE a b
```

If you are familiar with Haskell, you will note the use of `:` rather than `::`. This makes it look a bit more like a mathematical function definition:

$$f : \mathbb{N} \rightarrow \mathbb{N}.$$

You will also note that instead of the `(Type x) => x` syntax, it uses a more concise `(x:Type)` syntax.

Idris Syntax: Currying and Pattern Matching

Because of its foundation in Lambda Calculus, all functions only take a single argument. We can still handle multiple arguments using *currying*. For example, the `plus` operator is defined as follows:

```
plus : Nat -> Nat -> Nat
plus  Z      y  = y
plus  (S k)  y  = S (plus k y)
```

Like Haskell, functions are implemented using *pattern matching*.

Idris Syntax: Pattern Matching

Idris Syntax: Type Definition Syntax

Idris Syntax: Holes

Using Idris as a Proof Assistant

Quotes From Our Exploration

"The concept of a programming language in which the possibility of inline assembly is an entirely foreign concept hurts my brain."

"Where do I put it? Do I put it in the type?"

"When your Rust program compiles, you know it won't seg-fault, or give you any undefined behavior at runtime. When your Idris program compiles, you throw away your executable, and publish your dissertation."

Questions?