Hanus: Embedding Janus in Haskell

Joris ten Tusscher, Joris Burgers, Ivo Gabe de Wolff,
Cas van der Rest, Orestis Melkonian

Faculty of Science, Utrecht University

Introduction

- DSL description
- Reversible (Janus)

Reverse your program

- Division example
- Show inverse side-by-side

Syntactic Checking

- By using *QuasiQuotation*, the programmer gets notified of syntactic errors at compile-time!

Code

```
1 [hanus|procedure main() {
2   local n : Int = 10;
3     n += 10;
4   delocal n == 20;
5 }|]
```

Error

```
Exception when trying to run compile-time code:
  Parsing of Janus code failed in file ....
  First error:
--  Expecting "::" at position LineCol 2 10
```

Semantic Checking (Janus side)

- *Hanus* also reports semantic errors, such as violating Janus-specific constraints for expressions.

Code

```
1 count :: BinaryTree a -> Int
2 [hanus|
3     a :: BinaryTree Int;
4     procedure main() {
5         createNode a;
6         a.value += (count a.left) + a.value
7     }
8 |]
```

Error

```
Semantic Error (line 6, col 23):
    Assigned variable appears on the left-hand side.
```

Haskell Power

- The programmer can add additional operators by defining functions for forward and backward execution.
- We can define an operator that works on all Functors:

Definition

```
1 (=$$) :: Functor f => Operator (f a) (Operator a b, b)
2 (=$$) = Operator forward backward
3   where
4     forward  f (Operator fwd _, x) = fmap (`fwd` x) f
5     backward f (Operator _ bwd, x) = fmap (`bwd` x) f
```

Usage

```
1 procedure increase(tree :: BinaryTree Int) {
2   tree =$$ (+=, 42);
3 }
```

- Besides operators, the programmer can also define field and array indexers which allow you to use `tree.leftChild` and `array[x]` on the left hand side.

https://github.com/joristt/hanus