

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*, we notify the programmer of possible 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)

- We also report 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.
```

Semantic Checking (Haskell side)

- Since we generate Haskell programs, users are also notified for error made in anti-quoted Haskell expressions.

CODE

```
1 [hanus|
2   init :: Int;
3   a :: BinaryTree Int;
4   procedure main() {
5     createNode a;
6     a.nodeValue += map (+ 1) init;
7 }|]
```

ERROR

```
- Could not match expected type Int with actual
  type [Integer]
- In the expression: map (+ 1) i
```

Haskell Power

- Functor example