The Pony Project

Andrew Hirsch

GWU

2012-10-10 Wed

What Pony Is

- Pony is an extensible compiler
- For C99
- Written in Haskell
    - A purely functional language

What Already Exists

- Parts of Pony already exist
    - Transformation data type
    - C Abstract Syntax Tree
    - Application of transformations
- What doesn't exist
    - An extensible parser
      Can't parse anything that doesn't look like C!
    - Collision Detection
      When do two transformations cause a problem?
    - A Transformation Language
      Currently, a transformation writer must know Haskell

The Parser

- Extensible parsers are old hat
  . . . in Object-Oriented languages
- Lack of composable data types hampers Haskell
- But we need such a parser

Parsec

- A parser combinator library
- Small pieces of parsers
- Ways to compose them
- Compose parsers defined on-the-fly?

Data Types a la Carte

- Wouter Sweirstra's Functional Pearl
- Subtyping for Haskell
- Solves Wadler's "Expression Problem"
- Implemented in Data.Comp library

Semantic Collision

- When do two transformations "collide"?
  Act on one another, causing issues
- In general, undecidable
  - For syntax, same as asking "Is a Context-Free Grammar ambiguous"?
  - For semantics, much harder question

  Same as "What is the dependent type"?

Logic Programming

- For the weaker case, we ask “do these collide on this code”?
- This may be decidable!
    - View AST as a database
    - Find logical equivalent of code
    - solve for constraints!
- This is what the Logic Programming paradigm is designed for
- Curry language

Transformation Language

- We need a language for defining transformation
- Lots of work in this area:
    - XOC
    - William Cook's Grammars
        - Define parser and pretty-printer for grammars
        - Composition of Grammars
        - Can we extend with "Mixins"?

Conclusion

- There's a lot of interesting work to be done!
- Three big areas:
  - Detecting collision
  - Parsing
  - A language for user interaction