GP Analysis of Hackage

Nikolaos Bezirgiannis, Christiaan Floor
Utrecht University

7 November, 2011

Outline

- Introduction
- Approach
- Results
- Implementation
- Conclusions & Future Work

Introduction

Goal of the Project

- Analysis of use of GP on Hackage
  - How much libraries use GP?
  - For what reason they use GP?
  - What are the most "popular" GP libraries?

Analysis

- Split the analysis

  - Automatic analysis
    - Deriving analysis
    - Function analysis

  - Manual Analysis

Deriving Analysis

- Count **deriving** occurences
  - In datatypes
  - Standalone
  - Newtype-deriving
- Count **overloading** occurences
  - Custom-written instances instead of deriving Haskell98 deriveable classes

Function Analysis

- Create a list with functions used in GP
  - [everywhere, everything, mkT, mkQ...]
- Count number of calls of each function
- Mark the context they are applied in
  - Module
  - Source Position

Manual Analysis

- Looking at the results of the automatic analysis
- Draw conclusions from these numbers
- Manually looking at some packages to understand the use of GP

Approach

Approach for D Analysis

1. Download the full Hackage repository
2. Parse every module
3. Apply the Deriving Analysis
4. Store the results for later consumption

Approach for F Analysis

1. Download the full Hackage repository
2. Parse the cabal file for each package
   - Figure out most popular GP libraries
   - SYB and Uniplate **(Our Focus)**
3. Parse every library that uses SYB or Uniplate
4. Apply the Function Analysis
5. Store the Results

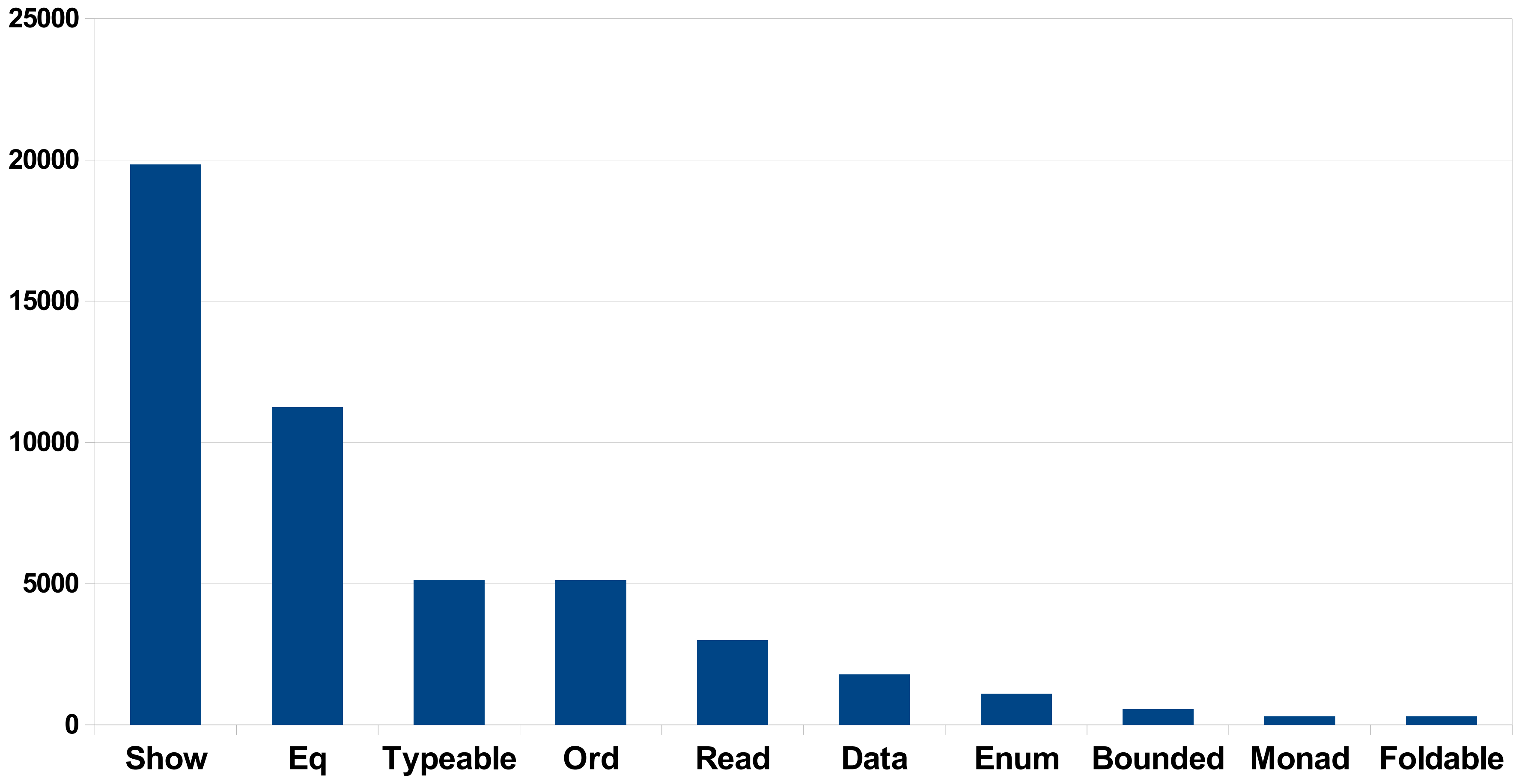
Results

Results of D Analysis

- Total number of instances: 51093
- Count of different styles:
  - Datatype deriving: 43415
  - Standalone deriving: 648
  - Overloading: 7030

- Count of 'non-regular' deriving: 2554
  - NewTypeDeriving is the reason

Top 10 D Analysis



Results of D Analysis (contd.)

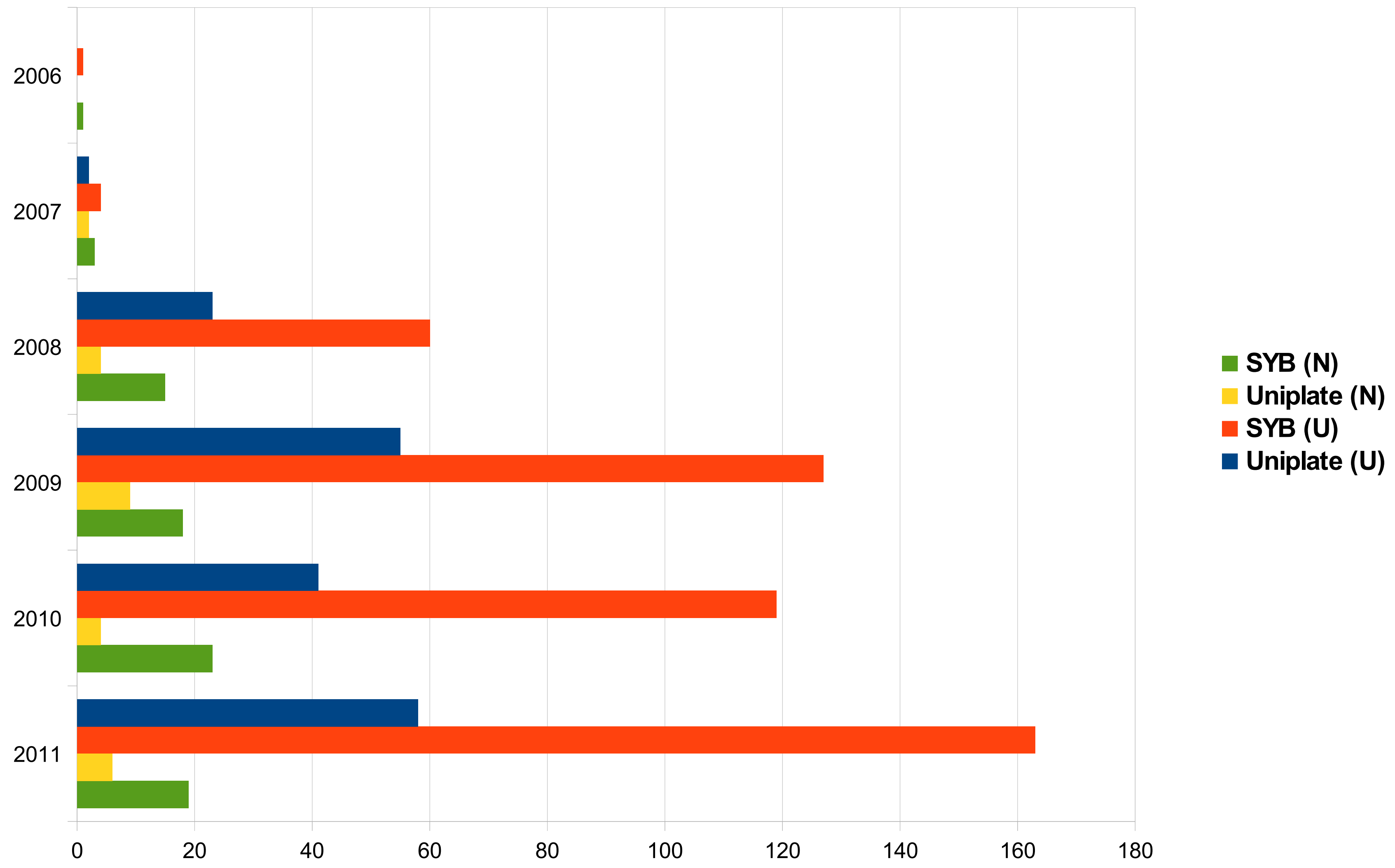
- Count of possible "genericity"

• Typeable instances:	5138
• Data instances:	1780
• Generic instances:	95
• Total:	7013

GP Libraries

- Out of 136 GP-dependent packages
  - 52 provide at least one executable
- We analysed 105 of them
  - Focus on SYB and Uniplate
  - 80 SYB-dependent packages
  - 25 Uniplate-dependent packages

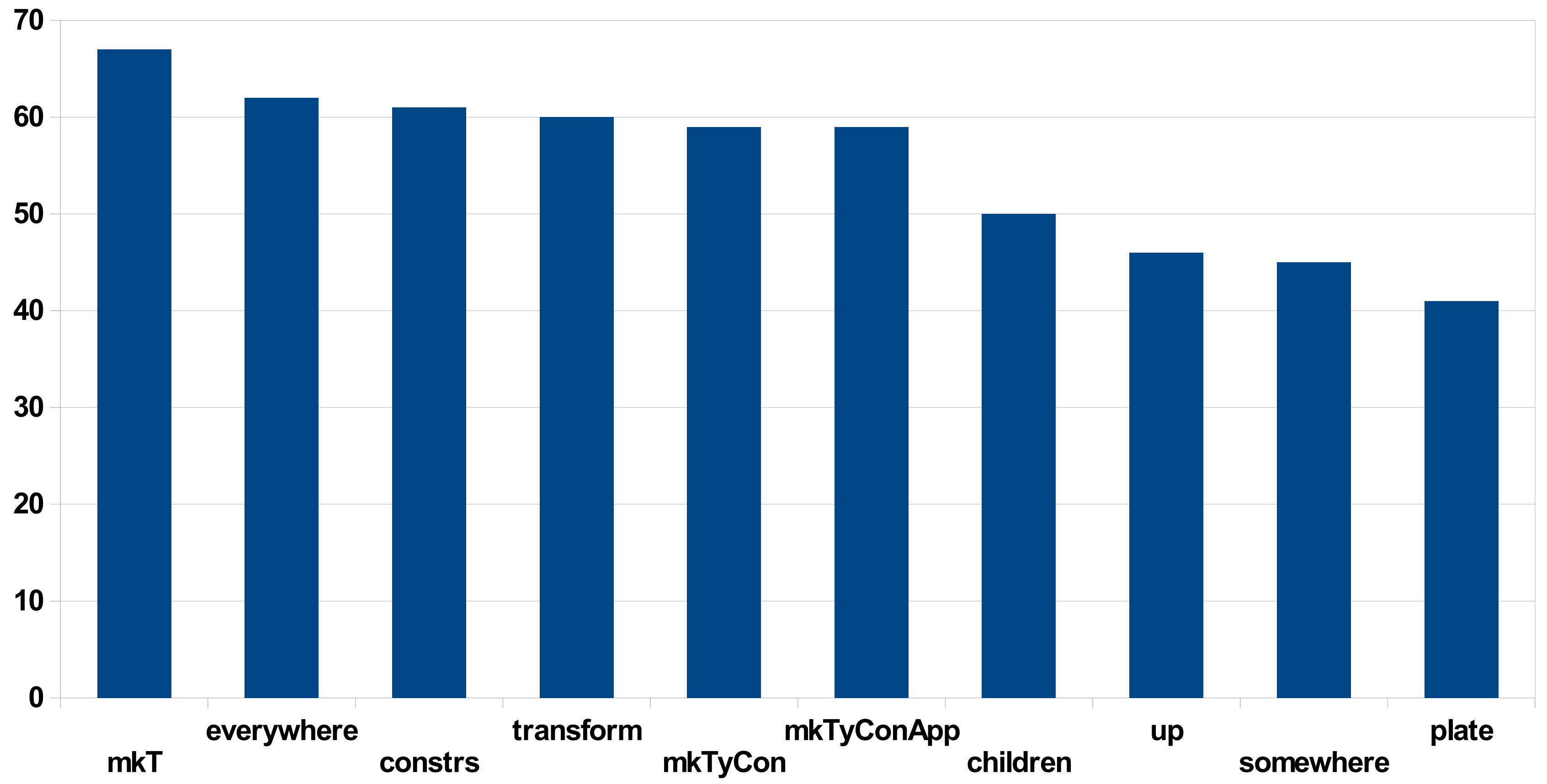
History of SYB and Uniplate



Deriving Uniplate

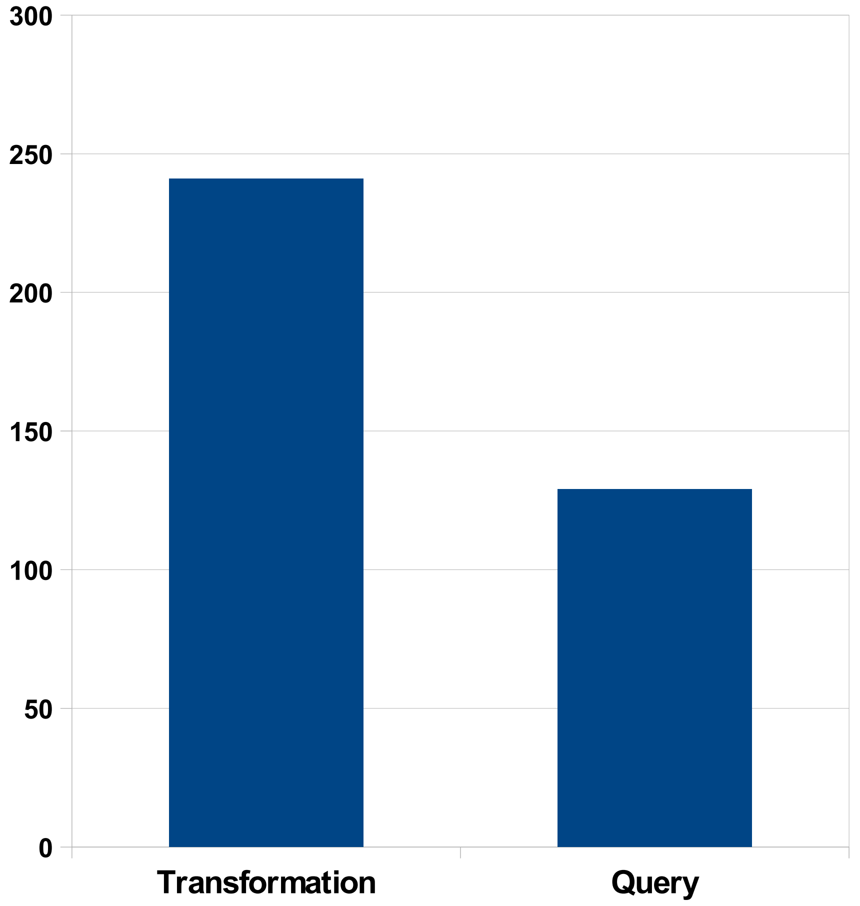
- Automatic: Count: 20
  - Easier to use
  - Just derive Data and Typeable
- Manual: Count: 7
  - Faster
  - Needs to write instances by hand
- Mixed: Count: 8
  - Allows to mix automatic deriving and manual instances within the same module

Top 10 F Analysis



Q/T Ratio

- **Queries** on datastructures used more than **Transformations**
- Sometimes both are used in a library
- But mostly one of the two



Manual Analysis

- SYB
  - prolog
  - preprocessor-tools
- Uniplate
  - hoogle
  - derive

Prolog

- Interpreter for Prolog written in Haskell
- Uses SYB for
  - Checking if a Prolog term occurs in another Prolog term (Q)
  - Traverse the AST and annotate the depth to each variable (T)

Preprocessor-tools

- Extend Haskell syntax using a custom preprocessor
- Uses SYB for
  - Replacing nodes with other nodes in the AST (T)
  - Setting a default value to all SrcPos of nodes (T)

Hoogle

- Search engine for the Haskell standard libraries
- Uses Uniplate for
    - Parsing and changing the AST (Q T)
    - Searching on Types
        - Follow Aliases (String ↔ [Char])
        - Create, show, search TypeGrahs (Q T)
    - Rendering the results for the user (T)

Derive

- A lib and a tool to derive instances for Haskell
- Uses Uniplate for
  - Languages
    - Lang ↔ Lang'
    - Construct, prettyprint, simplify internal DSL
    - Optimize TemplateHaskell code
  - Types
    - Substitutions
    - Restrictions on what types can be derived
  - Generating Instances
    - Traverse and transform datatypes

Implementation

Process

- We used both SYB and Uniplate for the analyses functions
- Parsed the modules using HaskellSrcExts
- Took about 2 hours to analyze whole Hackage
- Serialized the results using Data.Binary

Pitfalls

- Lazy IO: many open files problem
  - Solved with Control.Exception.evaluate
- Wrong Encoding format of modules
  - Solved with Codec.Text.Iconv
- The analysis was running out of memory
  - The results were lazily evaluated
  - Solved with: DeepSeq
- A lot of modules failed to parse

Conclusions & Future Work

Conclusions

- SYB and Uniplate most popular
  - SYB is bundled with GHC
- Main reason for using GP: Boilerplate removal
- Uniplate is mainly used for ease of use
- Strongly connected to Parsing and Manipulating ASTs

Future Work

- Add support for other GP-libraries
- Determine if a dependency on SYB can be replaced with Uniplate
- Add more search criteria to make numbers more accurate

Thank you