Hoogle
Fast Type Searching

Neil Mitchell
www.cs.york.ac.uk/~ndm/
Hoogle Synopsis

“Hoogle is a Haskell API search engine, which allows you to search many standard Haskell libraries by either function name, or by approximate type signature.”

Or, Google for Haskell
Hooble Background

• Over 4 years old
• 4 major versions (each a complete rewrite)
  – Version 1 in Javascript, 2-4 in Haskell
• Over half a million queries with Hooble 3

• I am current working full-time on Hooble thanks to Google Summer of Code and haskell.org (2 weeks left!)
Exact Searching

• You ask, Hoogle responds:
  – map Prelude.map
  – Map module Data.Map
  – (a → b) → [a] → [b] Prelude.map
  – Ord a ⇒ [a] → [a] Data.List.sort

• Exact searching is easy!
Inexact Text Searching

• Exact text matching is really easy (Trie)
• Substring matching is really easy (Trie with different entries)
• Can use Levenshtein/edit distance (harder to implement)

• Hoogle (1-4) all use substring matching
  – Hoogle 4 uses a Trie, 1-3 use linear search
Inexact Type Searching

- Most study has been on type isomorphisms (useless for searching)

- Want to “read the users mind”

- The game: I put up some type signatures, you guess the best answer
Human Search Engine

- $a \rightarrow [(a,b)] \rightarrow b$
- $\text{Int} \rightarrow \text{Int} \rightarrow \text{Int}$
- $[a] \rightarrow [b]$
- $[\text{Int}] \rightarrow \text{String}$
- $[a] \rightarrow (a \rightarrow b) \rightarrow [b]$
- $a \rightarrow \text{Maybe}$
- $a \rightarrow \text{Just a}$
- $\text{float} \rightarrow \text{float}$
Ranking

• Hoogle ranks results using a multiset of costs (about 14 in Hoogle 4)
  – You missed an argument (badarg)
  – You missed an instance (badinst)

• match :: Query → Result → Maybe [Cost]
  – Do not need to worry about ordering marks
Brainstorm

• match :: Query $\rightarrow$ Result $\rightarrow$ Maybe [Cost]

What is Cost?
How are they calculated?
Ideas

• Alpha equality (Hoogle 1)
• Isomorphism (Rittri, Runciman – 1980’s)
• Textual type searching (Hayoo!)
• Unification (Hoogle 2)
• Edit distance (Hoogle 3)
• Full edit distance (Hoogle 3.5)
• Structural edit distance (Hoogle 4)
Alpha equality

• Take a type signature, and “normalise” it
• Rename variables to be sequential
• The do an exact text match

• $k \rightarrow v \rightarrow \text{Map } k \; v$
• $a \rightarrow b \rightarrow \text{Map } a \; b$
Isomorphism

- Only match types which are isomorphic
  - Long before instances/type aliases
- Ismorphism is about equal structure
  - \( a \rightarrow b \rightarrow c \equiv (a, b) \rightarrow c \)

- uncurry :: \((a \rightarrow b \rightarrow c) \rightarrow (a, b) \rightarrow c\)
- :: \((a \rightarrow b \rightarrow c) \rightarrow a \rightarrow b \rightarrow c\)
Textual Type Searching

• Alpha normalise + strength reduced alpha normalisation

• $k \rightarrow v \rightarrow \text{Map } k \ v$

• $a \rightarrow b \rightarrow \text{Map } a \ b \ & \ a \rightarrow b \rightarrow c \ a \ b$

• Plus substring searching
Unification

• Unify against each result, like a compiler
• The lookup problem:
  – $a \rightarrow [(a,b)] \rightarrow b \neq a \rightarrow [(a,b)] \rightarrow \text{Maybe } b$

• Works OK, but not great, in practice
  – Gives more general answers, but not less general
• People are too fuzzy in their requests
Edit Distance

• What changes do I need to make to equalise these types
• Each change has a cost
  – a → [(a,b)] → b
  – a → [(a,b)] → Maybe b
  – Eq a ⇒ a → [(a,b)] → Maybe b
• The same idea in Hoogle 3.5 and 4, but different implementations
Hoogle 3 Edit Distance

- database :: [Type], length database \(\equiv n\)
- match :: Type \(\rightarrow\) Maybe [Cost]

- \([t \mid t \leftarrow \text{database}, \text{Just } c \leftarrow [\text{match } t],\text{ order by } c]\)
  - \(O(n)\) to search all items
  - \(O(n)\) to find the first result
Hoogle 3.5/4 Costs

- Alias following (String ↔ [Char])
- Instances (Ord a ⇒ a ↔ a)
- Boxing (a ↔ m a , a ↔ [a])
- Free variable duplication ((a,b) ↔ (a,a))
- Restriction ([a] ↔ m a , Bool ↔ a)
- Argument deletion (a → b ↔ b)
Per Argument Searching

• The idea: Search for each argument separately, combine the results
  – Some costs are applied in combination

• i.e. Search a → b → c

• combine $ search arguments a `merge`
  search arguments b `merge`
  search results c
Combine/Search

- search returns results for a particular type within a set of types in order of rank

- combine takes a list of results for arguments, and combines them into results matching an entire signature, removes duplicates, checks each argument is present etc.
Combine Notes

- Combine is fiddly
- Needs to apply costs such as instances, variable renaming, argument deletion
- As soon as it knows no result will rank lower, it returns a result

- Fast to search for the best matches
Hooble 3.5 Search

- Have type graphs, annotated with costs
  - Dijkstra’s graph search algorithm
The Problem

- Finds the first result very quick
- Graphs may be really big
- But a particular search may match many results in many ways
  - Finding all results can take some time
  - ~5 secs with 5000 functions

- Need to be more restrictive with matching!
Hoogle 4 structure matching

• We can decompose any type into a structure and a list of terms
• Either (Maybe a) (b,c)
• (?) (?) (?) (?) + Either Maybe a (,) b c

• Searching for a type involves finding an exact structure match and then a binding to the list of terms
Hoogle 4 additional costs

• Structure matching ignores a number of costs
  – Aliases – fully expand all aliases initially, combine has a heuristic to pay for them
  – Box/Unbox – allow one box/unbox at the top level, just perform 3 structure searches

• The base libraries have at most 22 different term sequences for a structure
Hoogle 4 results

- Fast to find the first result, fast to find all results, ~0.5sec on the base libraries
- Fast enough to develop and debug using Hugs on all the base libraries
  - Very helpful to me!

- Hoogle 4 demo, network connection permitting...
Ranking Costs

- Given a multiset of costs, need to order the results
- Solution: Assign each cost an integer, sum the costs, compare these numbers
- Initial attempt: Make up numbers manually
  - Did not scale at all, hard to get right, like solving a large constraint problem in your head
Hoole 3/4 Ranking

- Hoole has a ranking file, a list of searches with the desired order of results
- When someone complains, I add their complaint to this list

- Generates a set of constraints, then solves
  - Hoole 3 used ECLiPSe constraint solver
  - Hoole 4 uses a custom finite domain search
Hoogte Statistics

• 560,000 searches with Hoogle 3
• About 1 in 6 searches are type searches
  – I never do type search withHoogle!
  – Type searches decreasing with time

• Becoming an essential part of Haskell hacking for me
Future Work

• Hoogle 4 final release
• Integration with Cabal/Hackage (search your packages and all packages)
• AJAX style interface
• Ranking/search tweaks

• Hoogle 4 is substantially faster and gives pretty good search results
Conclusions

• Type and Name search are useful for learning and developing
  – Type search is a lot harder to do

• Having a practical online search engine is a real bonus
Funny Searches

- eastenders
- california public schools portable classes
- Bondage
- diem chuan truong dai hoc su pham ha noi 2008
- Messenger freak
- ebay consistency version
- Simon Peyton Jones Genius
- free erotic storeis
- videos pornos gratis
- gia savores de BARILOCHE
- name of Peanuts carton bird
- Colin Runciman