OCamlScope: a New OCaml API Search

Jun Furuse - Standard Chartered Bank
Who am I?

OCaml hacker using Haskell at work
What did helped me most in Haskell industry?

- Type class?
- Purity?
- Laziness?
It's Hoole.

### API Search Engine for Haskell [Mitchell]

**Hoole**

```haskell
m a -> (a -> m b) -> m b
```

### Packages

- **base**
- **template-haskell**
- **QuickCheck**
- **HTTP**
- **mtl**

### Examples

<table>
<thead>
<tr>
<th>Function</th>
<th>Signature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>(&gt;&gt;=)</code></td>
<td><code>Monad m =&gt; m a -&gt; (a -&gt; m b) -&gt; m b</code></td>
<td>Same as <code>&gt;&gt;=</code>, but with the arguments interchanged.</td>
</tr>
<tr>
<td><code>(&lt;&lt;&lt;)</code></td>
<td><code>Monad m =&gt; (a -&gt; m b) -&gt; m a -&gt; m b</code></td>
<td></td>
</tr>
<tr>
<td><code>bindQ</code></td>
<td><code>Q a -&gt; (a -&gt; Q b) -&gt; Q b</code></td>
<td>template-haskell Language.Haskell.TH.Syntax</td>
</tr>
<tr>
<td><code>concatMap</code></td>
<td><code>(a -&gt; [b]) -&gt; [a] -&gt; [b]</code></td>
<td>base Prelude, base Data.List</td>
</tr>
<tr>
<td><code>bracket</code></td>
<td><code>IO a -&gt; (a -&gt; IO b) -&gt; (a -&gt; IO c) -&gt; IO c</code></td>
<td></td>
</tr>
</tbody>
</table>
API Search Engine

- **By Name:**
  - `List.concat`
  - `Array.concat`
  - `String.concat` ...

- **By Type:**
  - `'a t -> (a -> 'b t) -> 'b t`
  - `(>>=)`
  - `Core.Std.List.concat_map` ...

- **Or Both:**
  - `val search : regexp -> _`
    - `Regexp.search : regexp -> string -> int -> (int * result) option`

Theoretical foundations: [Rittri], [Runciman], [Di Cosmo]
Equivalent in OCaml?

I use Hoogle 30 times a day sometimes.

- Does OCaml have something equivalent? There are, but limited:
  - OCamlBrowser
  - OCaml API Search

- So I built OCamlScope
OCamlBrowser

GUI Source browsing + API search: https://forge.ocamlcore.org/projects/labltk/

- Only for **locally compiled source**
- Uses OCaml typing code; it is OCaml badly:
  - Need to give `--I dir` and things can be shadowed:
    ```
    $ ls */*.cmi
dir1/m.cmi  dir2/m.cmi
    $ ocamlbrowser --I dir1 --I dir2  # dir2/m.cmi is shadowed
    ```
  - `cmi`s are memory hungry
  - Search is too exact:
    ```
    ('a, 'b) t -> 'a -> 'b does not find Hashtbl.find.
    Requires ('a, 'b) Hashtbl.t -> 'a -> 'b
    ```
OCaml API Search

- Remote search server
- Search stdlib, otherlibs and Extlib
- Based on OCamlBrowser + CamlGI
  - Same characteristics with OCamlBrowser
- Discontinued
Difficulties existed in OCaml

- **cmi** file is less informative (no location, no docs)
- **ml/mli** require proper options (\(-I\), \(-pp\), ...) to re-analyze

```bash
ocamlfind ocamlc
   -package spotlib,findlib,treeprint,orakuda,xml_conv,levenshtein
   -thread -I +ocamldoc -I .
   -syntax camlp4o -package meta_conv.syntax,orakuda.syntax,pa_ounit.syntax
   -c stat.ml
```

- No unified installation: hard to get these options

  configure / make / omake / ...
They are now gone!

- **cmt/cmti** files gives you:
  - Compiled AST with locations
  - Contains arguments to re-process to run OCamlDoc

\[\text{stat.cmt} \Rightarrow\]

```
ocamlfind ocamlc
  -package spotlib,findlib,treeprint,orakuda,xml_conv,levenshtein
  -thread -I +ocamlloc -I .
  -syntax camlp4o -package meta_conv.syntax,orakuda.syntax,pa_ouini.syntax
  -c stat.ml
```

- OPAM unified installations
  - compiler-libs: easier access to OCaml internals
OCaml® Scope: Hoogle for OCaml

Ah, yes... mostly.

- Remote search server by Ocsigen/Eliom
- Edit distance based
- On memory DB
Search by edit distance

Too exact search is not very useful:

- `? finalize`
  - `Gc.finalise`

- `? val concat : string list -> string`
  - `val concat : sep:string -> string list -> string`

Search done around 3 secs at worst so far in a small cheap VPS.
On memory DB

Special Paths and Types with Hashconsing

Some numbers:

- Major 115 OPAM packages / 185 OCamlFind packages
- 525k entries (values, types, constructors...)
- 39Mb of the final data file
- 170Mb in Memory (1/2 of naive cmi loading)
OCaml specific challenges

- Scrapers have to deal with 2 package systems (OCamlFind and OPAM)
- Search result regrouping
Scraping and 2 package systems

- Scraping `cmt/cmtis` per OPAM package
  
  ```bash
  export OPAMKEEPBUILDDIR=yes
  ```

- Module hierarchy by OCamlFind packages:
  
  ```ocaml
  {batteries}.BatList.iter
  ```

- Detect OPAM ⇔ OCamlFind package relationships
Too many search results

OCaml specific problem:

? (+)
+260

? 'a t -> ('a -> 'b t) -> 'b t
+500

? map
+5000!
Why so many?

- Things aliased by module aliases and inclusions
  - `module List = BatList`
  - `include Core_kernel.Std_kernel`

- No type class
  - `Not (>>=) :: Monad m => m a -> (a -> m b) -> m b`
  - But,
    - `Option.(>>=)`
    - `List.(>>=)`
    - `Lwt.(>>=)`
    - `...`
Workaround

- Grouping results by "short looks"
  
  - \( L w t. (\gg\gg) : \ 'a \ L w t. t \to (\ 'a \to \ 'b \ L w t. t ) \to \ 'b \ L w t. t ) \)
  
  - \( (\gg\gg) : \ 'a \ t \to (\ 'a \to \ 'b \ t ) \to \ 'b \ t ) \)

- Results
  
  - +500 ⇒ 8 groups: ? \( 'a \ t \to (\ 'a \to \ 'b \ t ) \to \ 'b \ t ) \)
  
  - +260 ⇒ 30 groups: ? \( (+) \)
  
  - +5000 ⇒ 880 groups: ? map
Future work: Real alias analysis

One group, but with 69 results of \(? (+) : \text{int} \rightarrow \text{int} \rightarrow \text{int}\)

This should be improved like:

\(? (+) : \text{int} \rightarrow \text{int} \rightarrow \text{int}\)

Found 1 group of 1 result

\{\text{stdlib}.\text{Pervasives}.(+) : \text{int} \rightarrow \text{int} \rightarrow \text{int}\}

with 63 aliases (see details)

It would improve search performance too
So many things to do!

- Better Web GUI
- Remote query API
- Repository of scraped data
- Better match: ex. `snakeCase` should match with `snake_case`
- Bugs, bugs, bugs...

https://github.com/camlspotter/ocamloscope/issues
OCamlScope: a New OCaml API Search

- API Search by Name and/or Types for OCaml
- Already searchable +100 top OPAM packages
- Any ideas, reports and contributions are welcome!

URL: http://ocamloscope.herokuapp.com

Source: https://github.com/camlspotter/ocamloscope