Migrating HLint to the GHC API

Neil Mitchell

ndmitchell.com

What is HLint?

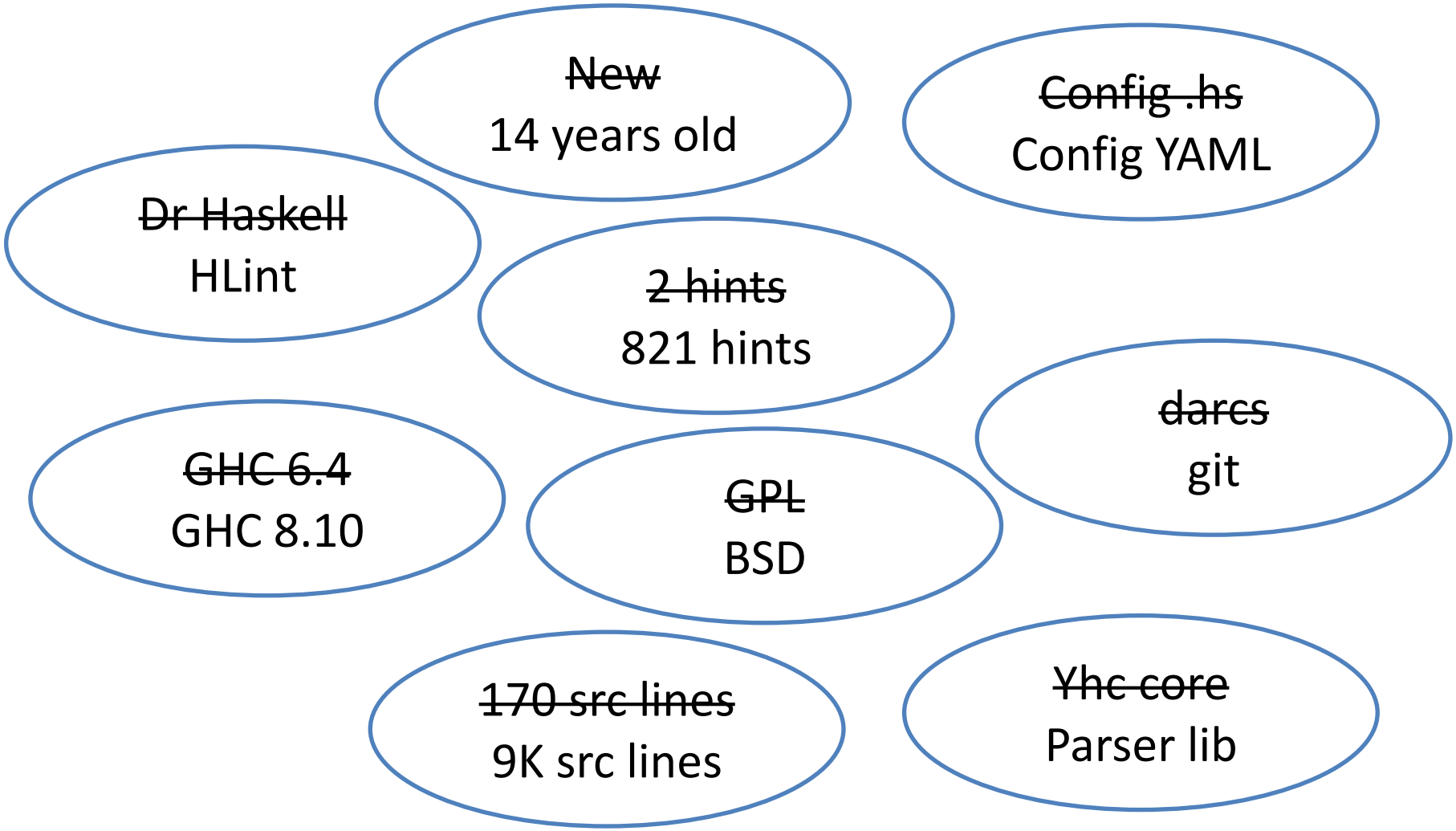
- A tool for suggesting possible improvements to Haskell code.
- https://github.com/ndmitchell/hlint

```
$ hlint darcs-2.1.2
darcs-2.1.2\src\CmdLine.lhs:94:1: Warning: Use concatMap
Found:
  concat $ map escapeC s
Perhaps:
  concatMap escapeC s
```

How HLint works

- For each file individually
  - Parse the file into an AST
  - Examine the AST with lots of possible “hints”
  - Report the ones that match, with suggestions

What changed since HLint v0?



PICKING A PARSER

Which parser to pick?

- HLint 2006-2008: Used Yhc Core
  - Dependency on compiling your code with Yhc
  - Mostly unused
- HLint 2008, wanted to pick a parser, options:
  - GHC API, the internals of GHC, exposed in 2006
  - haskell-src, forked the GHC parser in 2004 (or earlier?), standalone library
  - haskell-src-exts, forked the GHC parser in 2004, standalone library, added XML literals etc

Parser showdown (2008)

haskell-srcs-exts (HSE)

- Stable since 2004
- Easy to modify
- Fast release cycle
- Compatible with GHC
- Some other users
- Responsive maintainer (Niklas Broberg)

GHC API

- Existed since late 2006
- Significant changes in every release
- Hard to modify
- Long release cycle
- Compatible with GHC
- Slow link times (minutes)
- No real users

Haskell-src-exts worked well

- Simple API
- Good documentation
- Good printing of source
- Easy to pattern match against
- Later, added precise span information

Getting less compatible

- Lots of examples of it getting less compatible
- 64 issues tagged as HSE bugs, probably more

```
hgcastWith
  :: forall (a :: k) (b :: k') (r :: Type).
     (a :~~: b)
  -> (a ~~ b => r)
  -> r
```

```
getter (getIdent -> unIdent -> parent)  =
     TM.toCamel parent
```

```
module Data.HTree
   ( HTree(..), HShape, HL, type (:++:))
   where
```

Compatibility matters

- Every incompatibility means a “parse error” for a user
  - They report, I report upstream, wait for a fix (sometimes years)
  - They don’t report, might not use HLint anymore, definitely a bad experience

Parser showdown (2018)

haskell-srcs-exts (HSE)

- Slow release cycle
- Incompatible with GHC
- Multiple short-term maintainers
- HLint the biggest user

GHC API

- No need to modify
- Medium release cycle
- Compatible with GHC
- Big changes in every release
- Much worse as a library

Therefore…

- HLint should change to the GHC API

- But…
  - Big changes in every version
  - Much worse as a library

Version compatibility

- HLint is very tied to the AST, every minor AST change breaks something
- HLint supports GHC 8.6, 8.8, 8.10
- Forcing users to upgrade in lockstep would suck
- What to do:
  - CPP?
  - Something else?

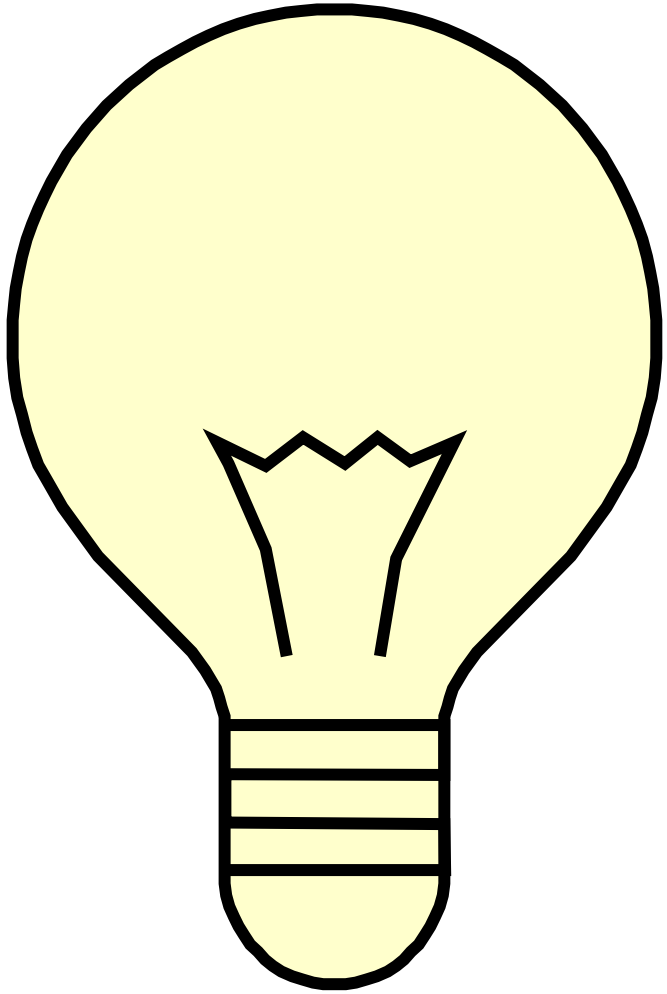
Is CPP infeasible?

- GHC 8.8 to 8.10:
  - 40 changed files
  - 416 additions, 386 deletions
- If we had been CPP based that would have been grim
  - An additional 1.5K lines? Per release ☹
  - No good IDE support
  - Every PR contribution in 3 flavours
  - Impossible to refactor

Smart solution!

- Use the GHC parser
- Copied from GHC repo to a standalone library
- Write a script to copy the right code in future

Had the idea about 4 years before implementation…



ghc-lib

- ghc-lib-parser is the GHC parser, 194 modules
- ghc-lib is everything else, 327 modules
  - Neither are fast to compile…
- v8.10.2.20200808 is GHC 8.10.2
- Supports 3 GHC versions, using GHC's bootstrap guarantee
- Doesn't have any libraries, e.g. base, so you need to find those yourself

ghc-lib implementation

- GHC has lots of generated code
- Also it builds with a custom build system

- So run the build system a bit
- Move the sources around
- Merge dependencies (e.g. template-haskell)
- Preprocess a bit
- Produce a .cabal library
- About 1000 lines of code

ghc-lib credits

Shayne Fletcher and Digital Asset – thanks!

Digital Asset

Why GHC is worse?

- `show` debugging doesn't work (use pretty-print)
- Lots of abstract types
- Lots of types of names: Id, Name, RdrName…
- Type families galore, for trees that grow
- Lots of code, poorly documented
- Lots of partial functions
- Pat/expr merging in some places
- Long compile times for ghc-lib-parser (e.g. CI)

```
asDo (view -> App2 bind lhs (Lambda _ [v] rhs)) =
   [Generator an v lhs, Qualifier an rhs]

asDo (view ->
      App2 bind lhs
       (L _ (HsLam _ MG {
           mg_origin=FromSource
         , mg_alts=L _ [
             L _ Match {  m_ctxt=LambdaExpr
                       , m_pats=[v@(L _ VarPat{})]
                       , m_grhss=GRHSs _
                                 [L _ (GRHS _ [] rhs)]
                                 (L _ (EmptyLocalBinds _))}]}))
    ) =
 [ noLoc $ BindStmt noExtField v lhs noSyntaxExpr noSyntaxExpr
 , noLoc $ BodyStmt noExtField rhs noSyntaxExpr noSyntaxExpr ]
```

Solution

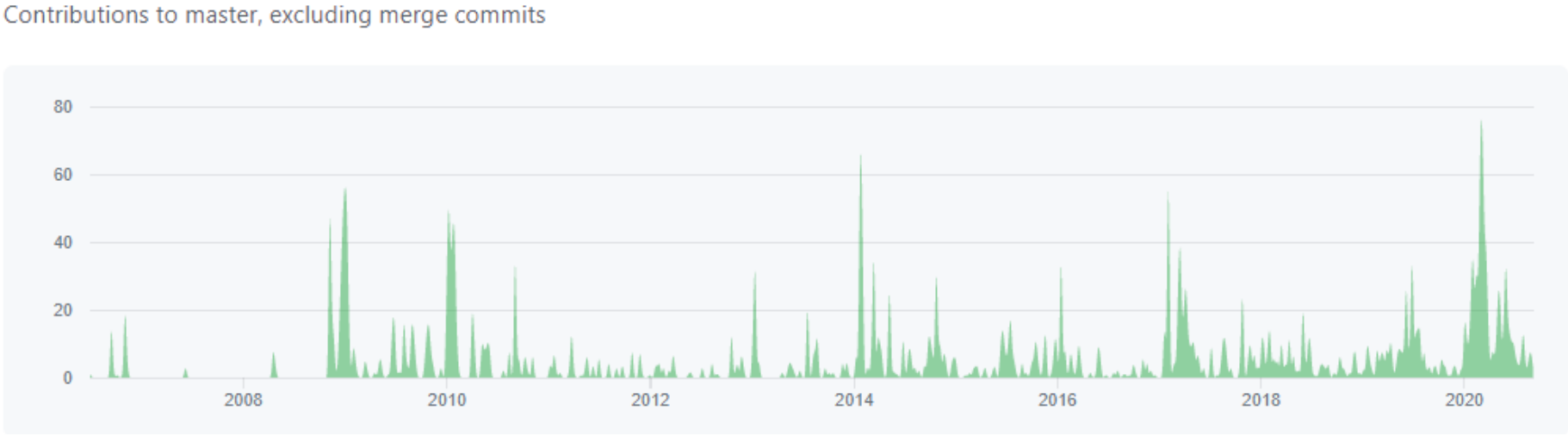
- Suck it up ☹

- Working on wrappers like ghc-lib-parser-ex
  - Again, credits to Shayne
- More abstractions tailored for GHC API

CHANGING PARSER

HLint is used and popular

- Lots of contributors, lots of users, 414 PRs
- Conversion could take a long time (months)
- Stop-the-world conversion was not feasible



Incremental conversion

- Preparation
  - Get us ready to support both at once
- Conversion
  - Convert module at a time
- Cleanup
  - Get rid of whatever we introduced in preparation

Make regular releases throughout, catch bugs

But! Minimize API incompatible 0.1 bumps

HLint architecture

Support

- CmdLine
- Testing
- Suggestion type
- Scope utils
- Parallelism
- Report writing

Hint groups (17)

- Match (754)
- Pragmas
- Comments
- Brackets
- Monads
- …

PREPARATION

Delete whatever we could

- Support for .hs config files (already supported .yaml)
- Support for QuickCheck hint generation (didn’t work since GHC 7.2)
- Anything marked deprecated
- Remove support for older GHC

Add the ghc-lib dependency

- Adding a huge dependency might break stuff
- And you have no idea what!

- First step, add a dependency on ghc-lib-parser
- Ensure ghc-lib-parser compiles for everyone
- Make a release (nothing broke – yay!)

Abstract the API

- HLint has an API, in terms of HSE types
- Make some of the fundamental ones abstract

```
parseModuleEx
    :: ParseFlags
    -> FilePath
    -> Maybe String
-   -> IO (Either ParseError (Module SrcSpanInfo, [Comment]))
+   -> IO (Either ParseError ModuleEx)
```

Parse twice

```
data ModuleEx = ModuleEx {
   hse  :: (Module SrcSpanInfo, [Comment]),
   ghc :: Located (HsModule GhcPs)
}
```

- Parse twice, propagate errors if either fail

Bugs

- v2.1.18, v2.1.19, accidentally changed API, reverted in v2.1.19, v2.1.20 (PVP violation)
- v2.1.21, realised it caused segfaults in haskell-ide-engine
  - getOrSetLibHSghc modifies a global variable
  - Representation of FastString table changed
  - GHC API and ghc-lib-parser were both using it
  - Moritz Kiefer figured it out

CONVERSION

Hint by Hint

- Change each hint from use the HSE AST, to the GHC AST
- As part of that, write any libraries/utils it required
- Go from easiest to hardest, as the utils are fleshed out

Hint 1: Newtype

Suggest newtype instead of data for type declarations that have only one field.

- data Foo = Foo Int -- newtype Foo = Foo Int
  - Plus 18 other test cases

4 files changed, 123 additions and 42 deletions.

Credit to Georgi Lyubenov

Hint 2: Naming

Should things be in CamelCase or not. 19 tests.

5 files changed, 104 additions, 57 deletions.

Starting to become a pattern…

Hint 11: Extensions

Are these extensions unused. 60 tests.

6 files changed, 246 additions, 148 deletions.

Credit to Shayne Fletcher

Example extension hint

```
used BangPatterns =
    hasS isPBangPat ||^ hasS isStrictMatch

isPBangPat :: LPat GhcPs -> Bool
isPBangPat (L _ BangPat{}) = True
isPBangPat _ = False

hasS :: (Data x, Data a) => (a -> Bool) -> x -> Bool
hasS test = any test . universeBi
```

Type level programming is untyped

- Uniplate universeBi relies on the target type
- GHC data types can be polymorphic

```
f :: GRHS a b -> Bool
f (GRHS _ xs _) = length xs > 1
```

```
a = GhcPs
b = LHsExpr GhcPs
```

Hint 17: Match

The match hint applies rules:

- warn: {lhs: concat (map f x), rhs: concatMap f x}

f/x are unification variables, match any expression

- For every sub expression
  - Match, check unification, check conditions, substitute

CLEANUP AND POLISH

Clean up technical debt

- Delete all unused old mini-libraries
- Move modules around
- Remove primes, e.g. Scope’ -> Scope
- Remove HSE entirely
- Remove all HSE types in API, breaking API
  - Fixities
  - Parse options
- v3.0 + release post (2.0 was 2017-04-06)

Final release

- Family caught COVID-19 in March, waited to recover before releasing

3.0, released 2020-05-02

… 52 lines …

Improve parse error context messages

… 11 API breaks …

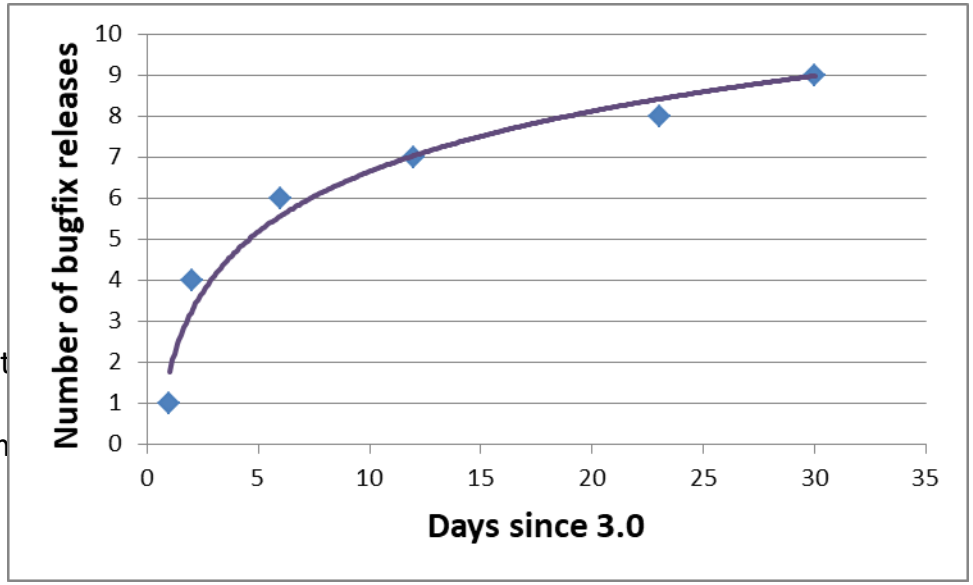
Merge ParseMode into ParseFlags

2.2.11, released 2020-02-09

Respond (quickly) to bugs

3.1.4, released 2020-05-31
  #1018, stop --cross being quadratic
3.1.2, released 2020-05-24
  #1014, don't error on empty do blocks
3.1.1, released 2020-05-13
  #993, deal with infix declarations in the module they occur
  #993, make createModuleEx use the default HLint fixities
3.1, released 2020-05-07
* #974, split ParseFlags.extensions into enabled/disabled
  #971, add support for -XNoFoo command line flags
  #971, add support for NoFoo language pragmas
3.0.4, released 2020-05-03
  #968, fail on all parse errors
  #967, enable TypeApplications by default
3.0.3, released 2020-05-03
  #965, fix incorrect avoid lambda suggestion
3.0.2, released 2020-05-03
  #963, don't generate use-section hints for tuples
  #745, fix up free variables for A{x}, fixes list comp hint
3.0.1, released 2020-05-02
  #961, don't crash on non-extension LANGUAGE pragn
3.0, released 2020-05-02

1 month
13 regressions
9 releases
1 API change



Fun bugs: Language pragmas

- {-# LANGUAGE Safe, GADTs #-}
  - Turns out “Safe” is not like other extensions
- {-# LANGUAGE NoGADTs #-}
  - We didn’t support negation
- Enable TypeApplications by default
  - Not really a bug, but wasn’t possible before because haskell-src-exts implemented it wrong

HSE /= GHC

Fun bugs: Infix declarations

- HLint has a “default” set of infix declarations
  - Those in base
  - Plus those that are “common” – lens, hspec, quickcheck, esqueleto, lattices
  - Plus those that are “tricky” - `on`
- Need to merge that, with user prefs, with infix declarations in the module
- Got it all kinds of wrong

Early work not revisited.
Lacking tests.

Fun bugs: Parse failed successfully

x = f (g @X), with –XNoTypeApplications

POk!

- But with errors
- Gives f (_) as the parse tree
- Those are redundant brackets!

GHC surprises.
Our expectations.

Fun bugs: Accidentally quadractic

- --cross became quadratic
- List comprehension gone wrong
  - [… | x <- xs, x <- xs]

https://neilmitchell.blogspot.com/2020/05/hlint-cross-was-accidentally-quadratic.html

Performance in corner cases

AFTERMATH

Conclusion

- It worked – took about 1 year
  - 2019-04-17 .. 2020-05-31, 31 releases
  - 229 PRs, 23 contributors (3 mostly on conversion)
  - 818 commits, 8685 lines added, 8426 deleted
  - I didn't personally convert a single hint
- Now we have few/no GHC incompatibilities
- Did HLint users notice?