Evolution and Dependencies of Haskell Packages
Haskell Infrastructure

Hackage: Haskell community central repository
Cabal: Package installation tool

Other Languages:
● Maven / Maven Central
● CPAN
● RubyGems
● Pip / PyPI
Package Versioning Policy (PVP)

Similar to Semantic Versioning (SemVer)
Increase major version on breaking change
Upper bounds on every dependency
Selected Quotes

“[…] upper bounds should be specified only when there is a known problem with a new version of a depended-upon package.”

“Those upper bounds are not worth the pain.”
Selected Quotes

“We've had several occasions in which our production builds broke due to the lack of proper upper bounds in one of our dependencies.”

“My plea to you: please follow the policy and put upper bounds on your version dependencies.”
Selected Quotes

“The real problem is that Hackage is maintaining conflicting packages!”
Selected Quotes

“The real problem is that Hackage is maintaining conflicting packages!”

Stackage (Stable Hackage)
Every Package compatible with every other Package
“Why are we relying on numeric identifiers for dependencies when we have so much richer information available?”
Selected Quotes

“[…] it's good form to follow the Package Versioning Policy and add a speculative upper bound […]. This is great, but in many cases the next version will /not/ break compatibility with your package.”
Example

```haskell
module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn number
```

```haskell
module Favourite where

number :: String
number = "fortytwo"

colour :: String
colour = "red"
```
Example cabal file

build-depends:    base >=4.6 && <4.7,
                   favourites >=0.1 && <=0.1

default-language: Haskell2010
module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn number

data awesomeapp-0.7

module Favourite where

number :: String
number = "fortytwo"

colour :: String
colour = "red"
Example

awesomeapp-0.7

module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn number

favourites-0.1

module Favourite where

number :: String
number = "fortytwo"

colour :: String
colour = "red"
Example

```
module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn number
```

```
module Favourite where

number :: String
number = "fortytwo"

colour :: String
colour = "green"
```
Example

```
module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn (number)
```

AwesomeApp-0.7

```
module Favourite where

number :: String
number = "fortyfour"

colour :: String
colour = "green"
```

Favourites-0.3
Safe updates

An update is a pair of versions
An update is safe for a package if it does not change any mentioned symbol
An update changes a symbol if it is not declared anymore
An update changes a symbol if it is declared with a different abstract syntax tree
When is a change breaking?

(Transitive) Syntax
Operational Semantics
Denotational Semantics
Intended Semantics
Tested Semantics
(Type) Signature
Query Example: uses

\texttt{uses(UsingDecl,Symb,UsedDecl) :-}
\begin{align*}
\text{dependency}(Pack,Dep), \\
\text{declaration}(Pack,UsingDecl), \\
\text{mentionedsymbol}(UsingDecl,Symb), \\
\text{declaration}(Dep,UsedDecl), \\
\text{declaredsymbol}(UsedDecl,Symb).}
\end{align*}
Query Example: uses
module Main where

import Favourite (number)

main :: IO ()
main = do
  putStrLn number

module Favourite where

number :: String
number = "fortyfour"

colour :: String
colour = "green"
Interchangeable Example

```
module Main where

import Favourite (number)

main :: IO ()
main = do
    putStrLn number
```

```
module Favourite where

number :: String
number = "forty" ++ "four"

colour :: String
colour = "green"
```
Query Example: interchangeable

\[\text{interchangeable}(\text{UsedDecl1}, \text{UsedDecl2}) : - \]
\[
\text{uses}(\text{Decl}, \text{Symb}, \text{UsedDecl1}), \\
\text{uses}(\text{Decl}, \text{Symb}, \text{UsedDecl2}), \\
\text{ast}(\text{UsedDecl1}, \text{Ast1}), \text{ast}(\text{UsedDecl2}, \text{Ast2}), \\
\text{Ast1} \neq \text{Ast2}.\]
Numbers

Packages: 78
Versions: 1149
Declarations: 217007
Distinct abstract syntax trees: 19167
Symbols: 12407
Interchangeable Example 1

\[
\text{insert } k \times x \cdot t \\
\quad = k \ `\text{seq}` \\
\quad \text{case } t \text{ of} \\
\quad \quad - \ (\text{Bin } p \ m \ l \ r) | \text{nomatch } k \ p \ m \rightarrow \text{join } k \ (\text{Tip } k \ x) \ p \ t \\
\quad \quad + \ (\text{Bin } p \ m \ l \ r) | \text{nomatch } k \ p \ m \rightarrow \text{link } k \ (\text{Tip } k \ x) \ p \ t \\
\quad \quad \quad | \text{zero } k \ m \rightarrow \text{Bin } p \ m \ (\text{insert } k \ x \ l) \ r \\
\quad \quad \quad | \text{otherwise} \rightarrow \text{Bin } p \ m \ l \ (\text{insert } k \ x \ r) \\
\quad \quad \text{Tip } ky \ _ | k \ == \ ky \rightarrow \text{Tip } k \ x \\
\quad - \ | \text{otherwise} \rightarrow \text{join } k \ (\text{Tip } k \ x) \ ky \ t \\
\quad + \ | \text{otherwise} \rightarrow \text{link } k \ (\text{Tip } k \ x) \ ky \ t \\
\quad \text{Nil} \rightarrow \text{Tip } k \ x
\]
Interchangeable Example 1

\[
\text{insert } k x t = k \ `\text{seq}`
\]

\[
\text{case } t \text{ of }
\]

- \text{Bin } p m l r | \text{nomatch } k p m \rightarrow \text{join } k (\text{Tip } k x) p t

+ \text{Bin } p m l r | \text{nomatch } k p m \rightarrow \text{link } k (\text{Tip } k x) p t

  | zero k m \rightarrow \text{Bin } p m (\text{insert } k x l) r

  | otherwise \rightarrow \text{Bin } p m l (\text{insert } k x r)

\text{Tip } ky _ | k == ky \rightarrow \text{Tip } k x

- | otherwise \rightarrow \text{join } k (\text{Tip } k x) ky t

+ | otherwise \rightarrow \text{link } k (\text{Tip } k x) ky t

\text{Nil} \rightarrow \text{Tip } k x
class (Monad m) => MonadReader r m | m -> r where

  ask :: m r

  local :: (r -> r) -> m a -> m a

  reader :: (r -> a) -> m a

  reader f
  = do r <- ask
       return (f r)
Interchangeable Example 2

class (Monad m) => MonadReader r m | m -> r where

    ask :: m r

    local :: (r -> r) -> m a -> m a

    reader :: (r -> a) -> m a
    reader f = do r <- ask
                   return (f r)
Interchangeable Example 3

```
newTerminal = do
  hFlush stdout
  hFlush stderr
  ref <- newIORef (return ()
  return (MkTerminal ref)
```

```
+newTerminal out err
+ = do ref <- newIORef (return ())
+    return (MkTerminal ref out err)
```
Interchangeable Example 3

newTerminal
- = do hFlush stdout
- hFlush stderr
- ref <- newIORef (return ()
- return (MkTerminal ref)

+ newTerminal out err
+ = do ref <- newIORef (return ()
+ return (MkTerminal ref out err)
Future Work

Find safe updates
Show that some version bounds are too tight
Get more data
Thanks