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 the 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

Selected Quotes

“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

awesomeapp-0.7

```
module Main where

import Favourite (number)

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

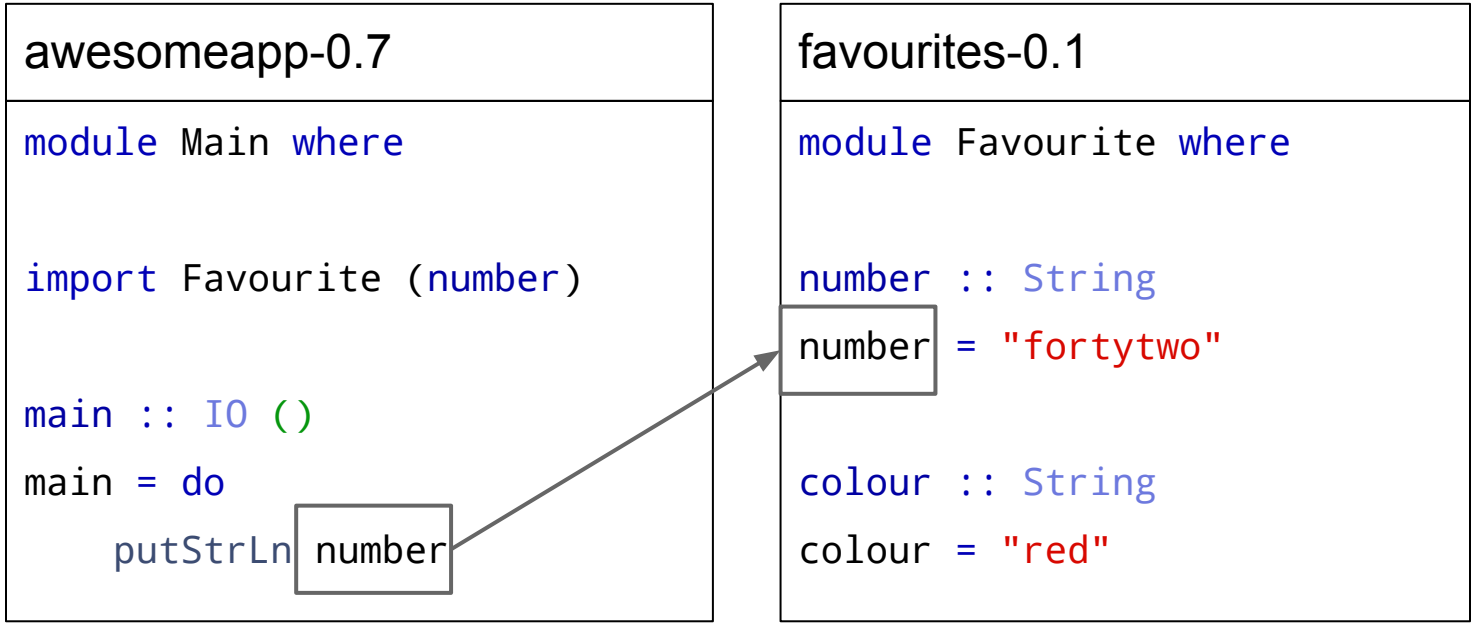
favourites-0.1

```
module Favourite where

number :: String
number = "fortytwo"

colour :: String
colour = "red"
```

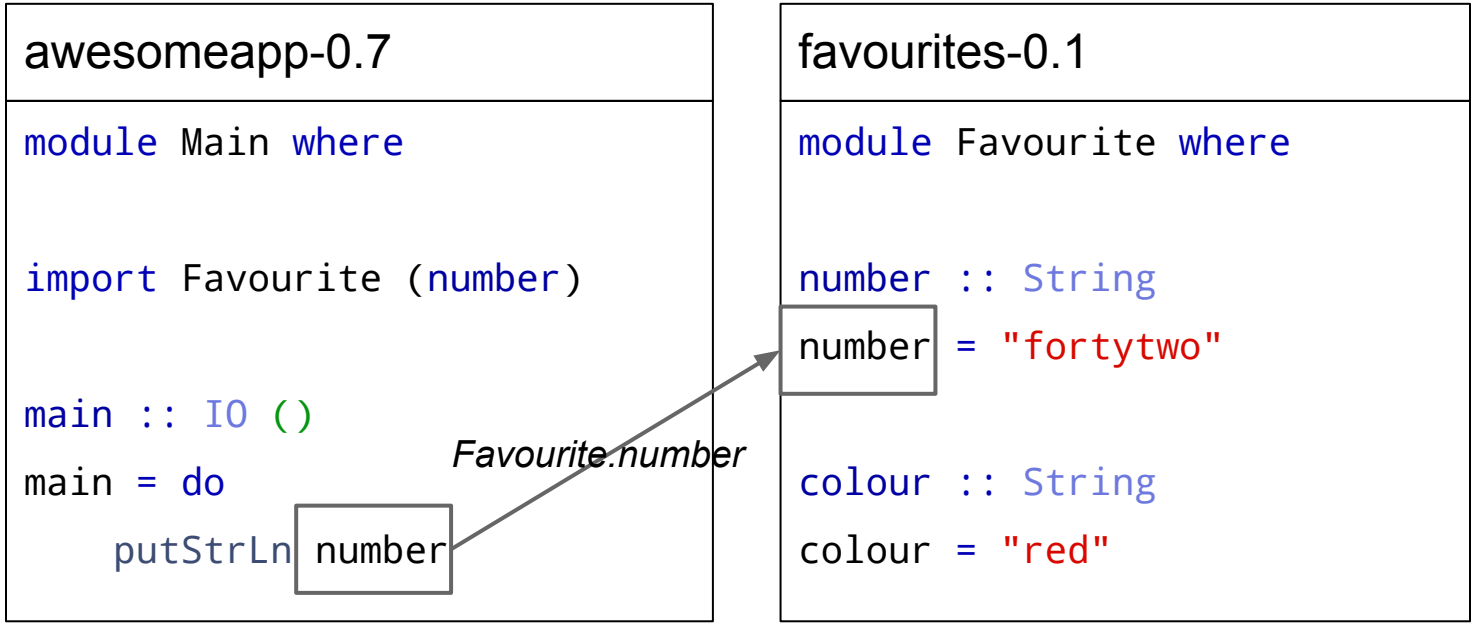
Example cabal file

```
build-depends:       base >=4.6 && <4.7,
                     favourites >=0.1 && <=0.1
default-language:    Haskell2010
```

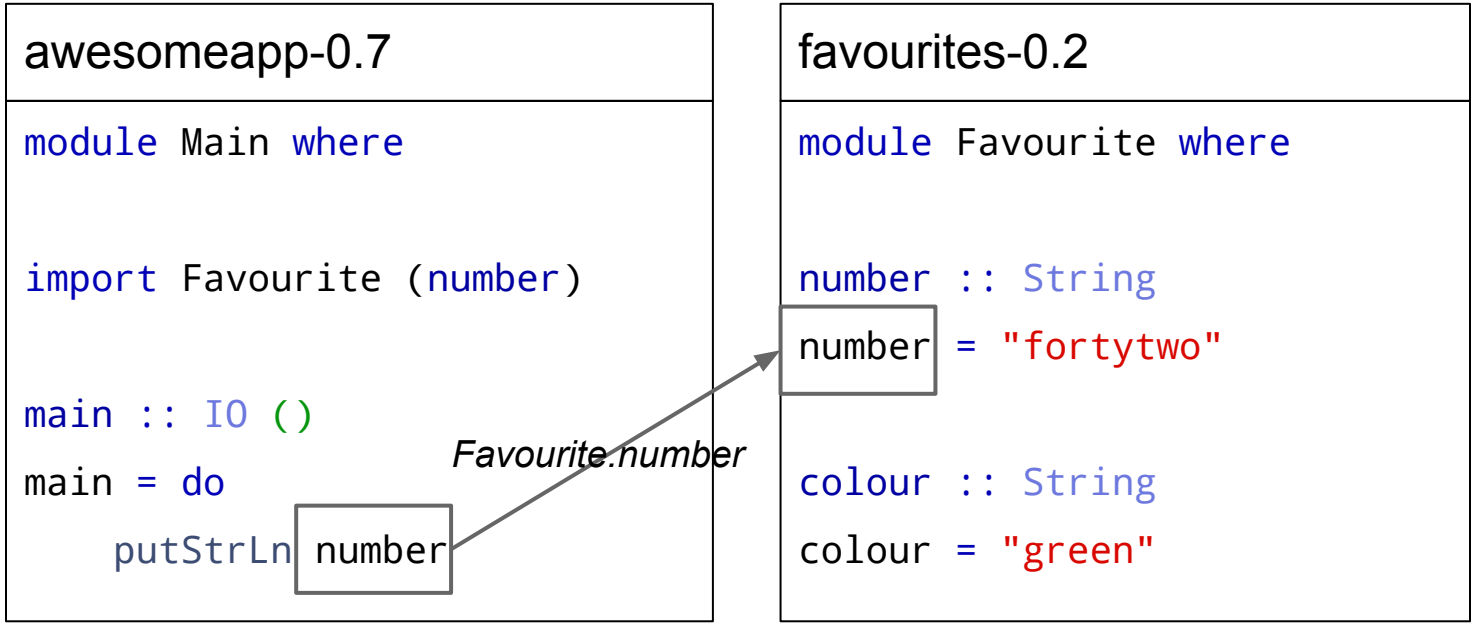
Example


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"

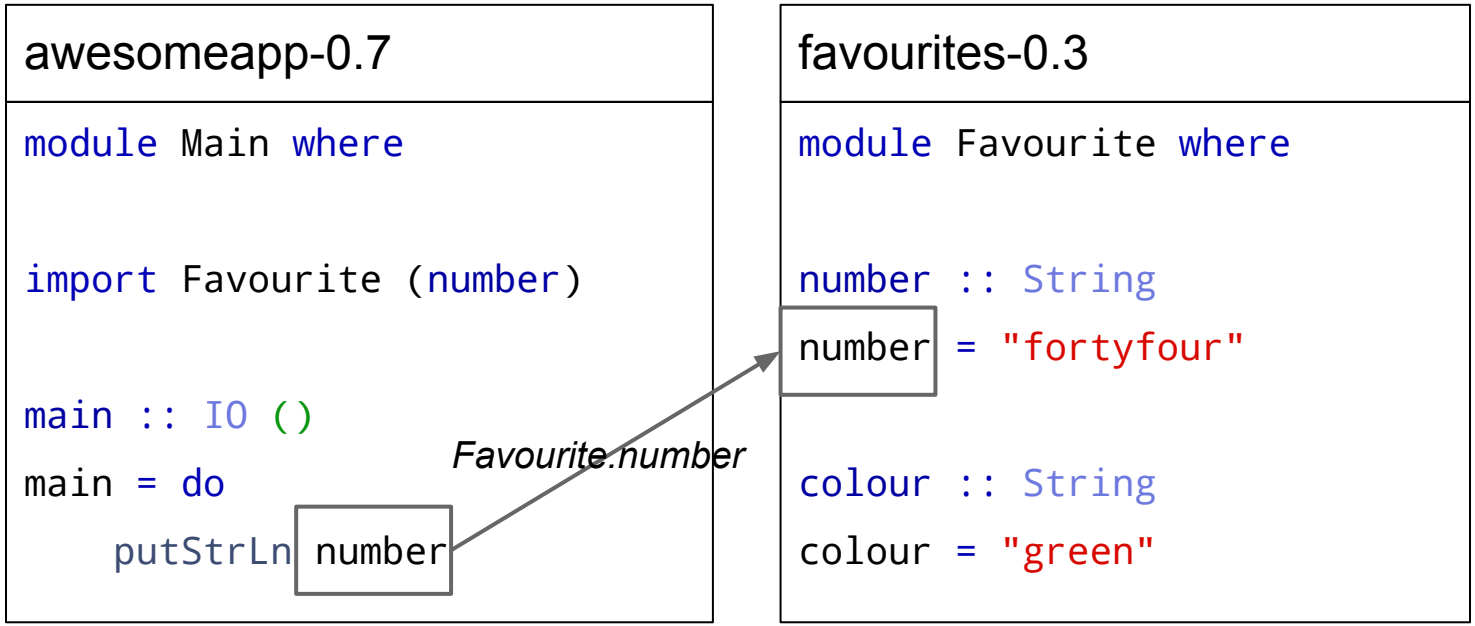
Example


awesomeapp-0.7
module Main where
import Favourite (number)
main :: IO ()
main = do
putStrLn number
Favourite.number
favourites-0.1
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
Favourite.number
favourites-0.2
module Favourite where
number :: String
number = "fortytwo"
colour :: String
colour = "green"

Example


awesomeapp-0.7
module Main where
import Favourite (number)
main :: IO ()
main = do
putStrLn number
Favourite.number
favourites-0.3
module Favourite where
number :: String
number = "fortyfour"
colour :: String
colour = "green"

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 is declared with a different abstract syntax tree

When is a change breaking?

(Transitive) Syntax

Operational Semantics

Denotational Semantics

Intended Semantics

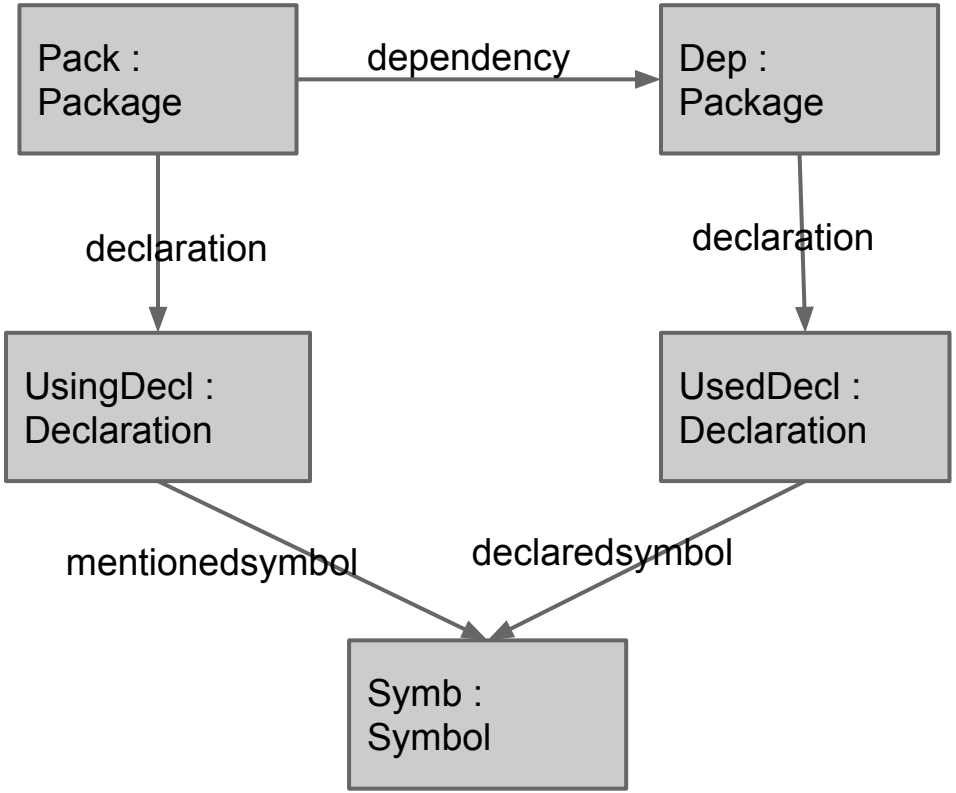
Tested Semantics

(Type) Signature

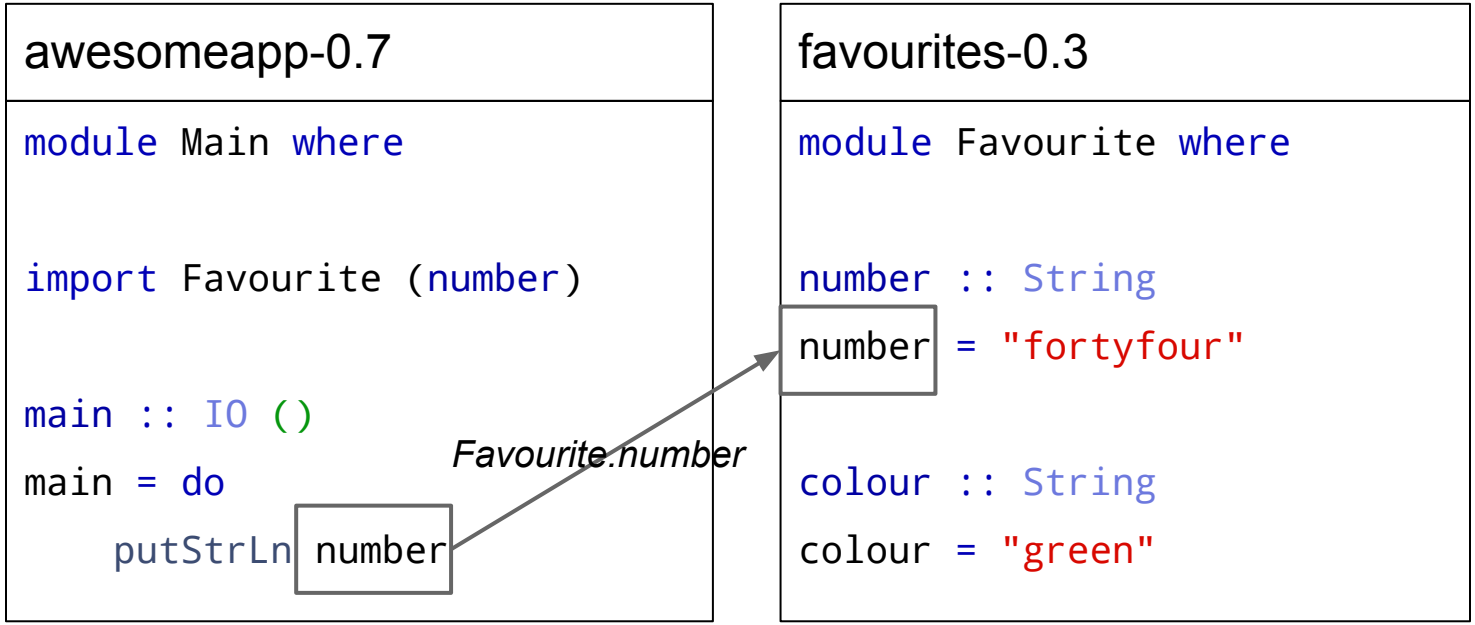
Query Example: uses

```
uses(UsingDecl,Symb,UsedDecl) :-
    dependency(Pack,Dep),
    declaration(Pack,UsingDecl),
    mentionedsymbol(UsingDecl,Symb),
    declaration(Dep,UsedDecl),
    declaredsymbol(UsedDecl,Symb).
```

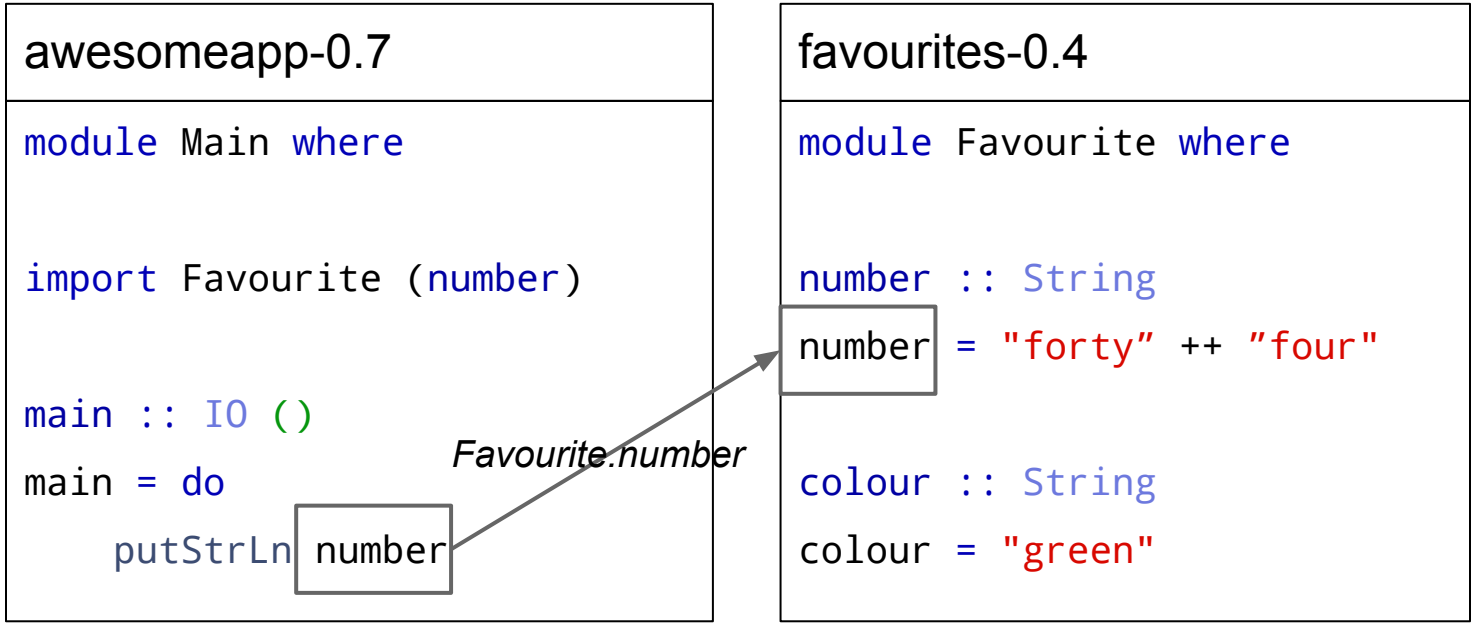
Query Example: uses



Interchangeable Example



Interchangeable Example



Query Example: interchangeable

```
interchangeable(UsedDecl1,UsedDecl2) :-
    uses(Decl,Symb,UsedDecl1),
    uses(Decl,Symb,UsedDecl2),
    ast(UsedDecl1,Ast1),ast(UsedDecl2,Ast2),
    Ast1 ≠ Ast2.
```

Numbers

Packages: 78

Versions: 1149

Declarations: 217007

Distinct abstract syntax trees: 19167

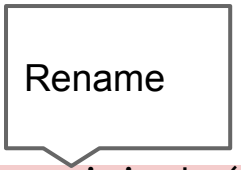
Symbols: 12407

Interchangeable Example 1

```
insert k x t
   = k `seq`
       case t of
-          Bin p m l r | nomatch k p m -> join k (Tip k x) p t
+          Bin p m l r | nomatch k p m -> link k (Tip k x) p t
                      | zero k m -> Bin p m (insert k x l) r
                      | otherwise -> Bin p m l (insert k x r)
           Tip ky _ | k == ky -> Tip k x
-                   | otherwise -> join k (Tip k x) ky t
+                   | otherwise -> link k (Tip k x) ky t
           Nil -> Tip k x
```

Interchangeable Example 1

```
insert k x t
   = k `seq`
       case t of
-          Bin p m l r | nomatch k p m -> join k (Tip k x) p t
+          Bin p m l r | nomatch k p m -> link k (Tip k x) p t
                      | zero k m -> Bin p m (insert k x l) r
                      | otherwise -> Bin p m l (insert k x r)
           Tip ky _ | k == ky -> Tip k x
-                   | otherwise -> join k (Tip k x) ky t
+                   | otherwise -> link k (Tip k x) ky t
           Nil -> Tip k x
```



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 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)
```

Default

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