<Type Inference Rules>

1. Data constructor is a function too.

```
data Maybe a
  = Nothing
  | Just a
```

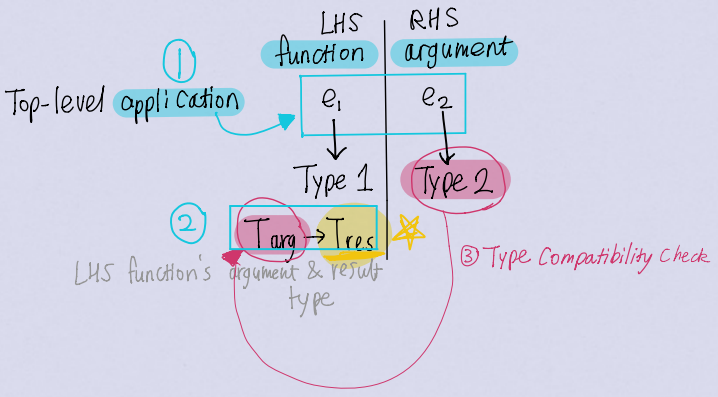
`Just :: a → Maybe a`

'Just' turns 'a' into 'Maybe a'

2-1. Split the top-level app. into LHS & RHS.
(function) (argument)

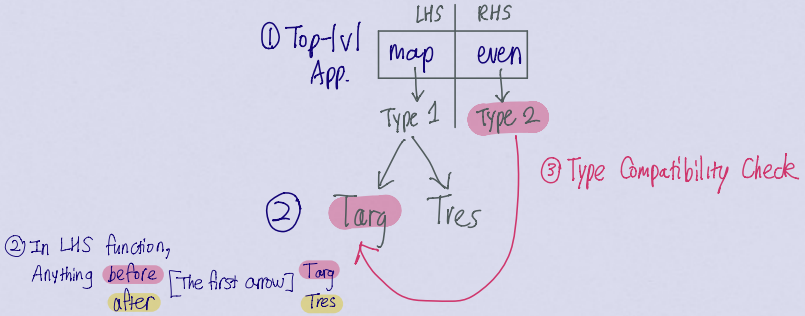
2-2. Split the function in LHS into Targ → Tres.

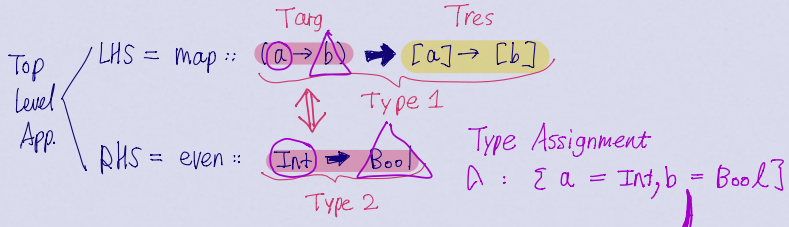
2-3. Check if the argument in RHS (Type 2) is compatible w/ the function in LHS.



2-4. Function type is determined by Tres.
The type of its result in LHS

ex> `:t map even`

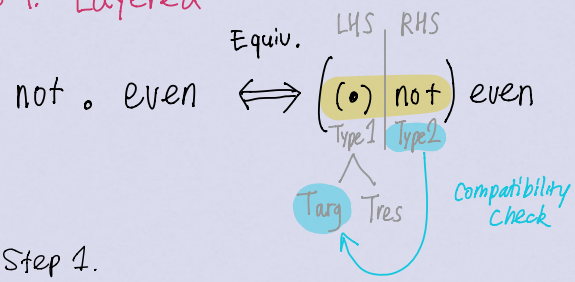




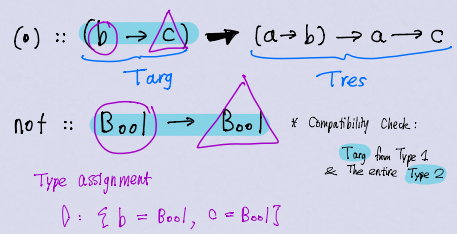
∴ The type of (map even) is Tres = `[a] → [b]`.
By type assignment, map even :: `[Int] → [Bool]`

3. Function Composition

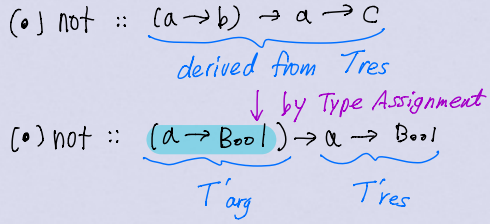
3-1. Layered



Step 1.

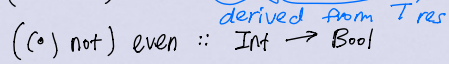


Step 2.

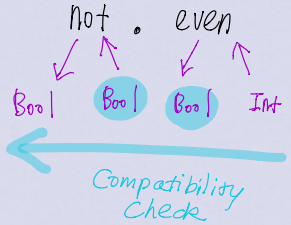


Type assignment : { a = Int }

Step 3.

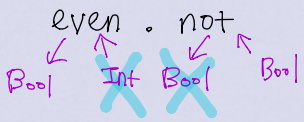


3-2. Pipeline Approach



∴ not . even :: Int → Bool

cf)



Type Error

3-3. Function composition result is also a function w/ arrow!