Mónadas – Modificación 1

- Reescribimos y dibujamos los recuadros

```
eval (Cte n) = Just n
eval (Div e1 e2) =
  case (eval e1) of
       Nothing  -> Nothing
       Just v1'  ->
            (\v1 -> case (eval e2)  of
                      Nothing -> Nothing
                      Just v2' -> (\v2 -> if v2 == 0
                                          then  Nothing
                                          else  Just  (v1 / v2)
                                  )  v2'
            )  v1'
```

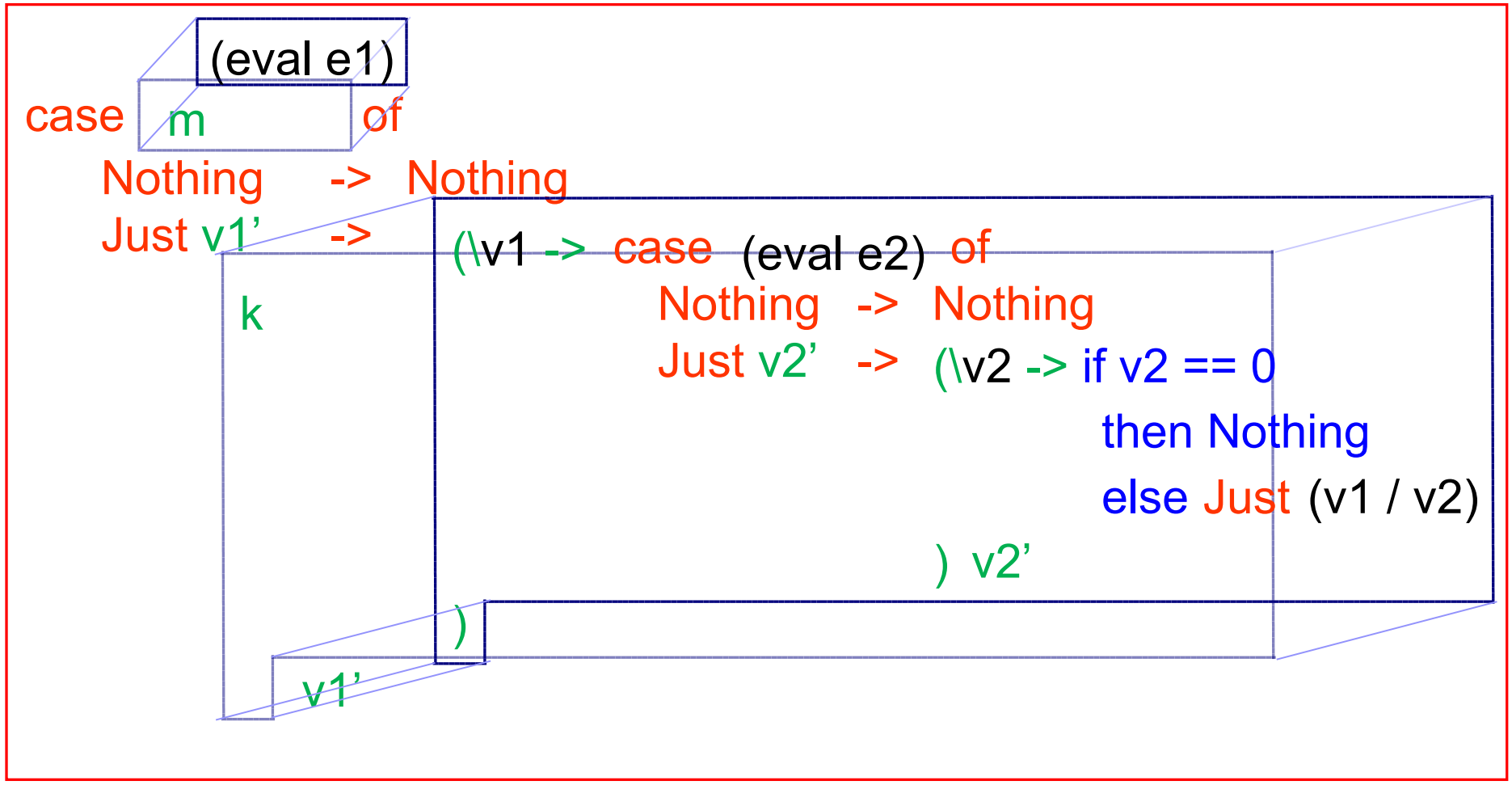
Mónadas – Modificación 1

- Reescribimos y dibujamos los recuadros

```
eval (Cte n) = Just n
eval (Div e1 e2) =
  case (eval e1) of
       Nothing  -> Nothing
       Just v1'  ->
            (\v1 -> case (eval e2) of
                         Nothing -> Nothing
                         Just v2' -> (\v2 -> if v2 == 0
                                             then  Nothing
                                             else  Just (v1 / v2)
                                     ) v2'
            ) v1'
```

Mónadas – Modificación 1

- Separamos algunos recuadros...



Mónadas – Modificación 1

- ...y los ponemos como parámetros

```
eval (Div e1 e2) =
(\m k -> case m of
     Nothing    ->  Nothing
     Just v1'   ->  k v1')
    (eval e1)
(\v1 -> case (eval e2) of
     Nothing   ->  Nothing
     Just v2'  ->  (\v2 -> if v2 == 0
                           then Nothing
                           else Just (v1 / v2)
                   ) v2'
)
```

Mónadas – Modificación 1

- ...y los ponemos como parámetros

```
eval (Div e1 e2) =
(\m k -> case m of
     Nothing    ->  Nothing
     Just v1'   ->  k v1')
    (eval e1)
    (\v1 -> (\m k -> case m of
                 Nothing   ->  Nothing
                 Just  v2' ->  k v2')
               (eval e2)
               (\v2 -> if v2 == 0
                       then Nothing
                       else Just (v1 / v2))
    )
```

Mónadas – Modificación 1

- Damos nombre a los recuadros

```
returnM :: ??
returnM x   = Just x
bindM :: ??
bindM m k = case m of  Nothing ->  Nothing
                       Just v   ->  k v
raiseError  = Nothing

eval (Cte n)       = returnM n
eval (Div e1 e2) =
    bindM (eval e1)
          (\v1 -> bindM (eval e2)
                        (\v2 -> if v2 == 0
                                then raiseError
                                else returnM (v1 / v2)))
```

Mónadas – Modificación 1

- Reescribimos la sintaxis por comodidad

```
returnM :: ??
returnM x   = Just x
bindM :: ??
bindM m k  = case m of  Nothing -> Nothing
                        Just v   -> k v
raiseError  = Nothing

eval (Cte n)       = returnM n
eval (Div e1 e2) =
        eval e1   `bindM` \v1 ->
        eval e2   `bindM` \v2 ->
        if v2 == 0
         then raiseError
         else returnM (v1 / v2)
```

Mónadas – Modificación 1

- Reescribimos la sintaxis por comodidad

```
returnM :: a -> Maybe a
returnM x   = Just x
bindM :: Maybe a -> (a -> Maybe b) -> Maybe b
bindM m k  = case m of  Nothing -> Nothing
                        Just v    -> k v
raiseError   = Nothing

eval (Cte n)       = returnM n
eval (Div e1 e2) =
        eval e1   `bindM` \v1 ->
        eval e2   `bindM` \v2 ->
        if v2 == 0
         then raiseError
         else returnM (v1 / v2)
```

Mónadas – Modificación 2

- Reescribimos el código y dibujamos los recuadros

```
eval (Cte n) = (\x s -> (x, s)) n
eval (Div e1 e2) =
\s -> let (v1', s1') = (eval e1) s
      in (\v1 ->
        \s1 -> let (v2', s2') = (eval e2) s1
               in (\v2 ->
                 \s2 -> let (vd, s3') = inc "div" s2
                        in (\_ ->
                          (\x s3 -> (x, s3)) (v1 / v2)
                        ) vd s3'
               ) v2' s2'
      ) v1' s1'
```

Mónadas – Modificación 2

- Rearmamos los recuadros

```
eval (Div e1 e2) =
  (\m k -> \s -> let (v1', s1') = m s
                 in k v1' s1')
    (eval e1)
    (\v1 -> (\m k -> \s1 -> let (v2', ks2')= m s1
                            in k v2' s2')
      (eval e2)
      (\v2 -> (\m k -> \s2 -> let (vd, s3') = m s2
                              in k vd s3')
        inc "div"
        (\_  ->  (\x s3 -> (x, s3))  (v1 / v2))))
```

Mónadas – Modificación 2

- Damos nombre a los recuadros

```
returnS :: ??
returnS x = \s -> (x, s)
bindS :: ??
bindS m k = \s -> let (v, s') = m s in k v s'
inc “div” (“div”, v) = ((), (“div”, v + 1))

eval (Cte n)       = returnS n
eval (Div e1 e2) =
    bindS  (eval e1)
           (\v1 -> bindS (eval e2)
                         (\v2 -> bindS inc “div”
                                       (\_ ->  returnS (v1 / v2))))
```

Mónadas – Modificación 2

- Reescribimos la sintaxis por comodidad

```
returnS :: ??
returnS x = \s -> (x, s)
bindS :: ??
bindS m k = \s -> let (v, s') = m s in k v s'
inc “div” (“div”, v) = ((), (“div”, v + 1))

eval (Cte n)       = returnS n
eval (Div e1 e2) =
    eval e1        `bindS` \v1 ->
    eval e2        `bindS` \v2 ->
    inc “div”      `bindS` \_ ->
    returnS (v1 / v2)
```

Mónadas – Modificación 2

- Reescribimos la sintaxis por comodidad

```
returnS :: a -> StateT a
returnS x = \s -> (x, s)
bindS :: StateT a -> (a -> StateT b) -> StateT b
bindS m k = \s -> let (v, s') = m s in k v s'
inc “div” (“div”, v) = ((), (“div”, v + 1))

eval (Cte n)       = returnS n
eval (Div e1 e2) =
    eval e1        `bindS` \v1 ->
    eval e2        `bindS` \v2 ->
    inc “div”      `bindS` \_ ->
    returnS (v1 / v2)
```

Mónadas – Ejemplo básico

- Reescribimos el código y dibujamos los recuadros

```
eval (Cte n) = id n
eval (Div e1 e2) = let v1' = (eval e1)
                   in (\v1 -> let v2' = (eval e2)
                              in (\v2 -> id (v1 / v2)
                                 ) v2'
                      ) v1'
```

Mónadas – Ejemplo básico

- Rearmamos los recuadros

```
eval (Cte n) = id n
eval (Div e1 e2) = (\m k -> let v1' = m
                            in k v1')
                   (eval e1)
                   (\v1 -> (\m k -> let v2' = m
                                   in k v2')
                           (eval e2)
                           (\v2 -> id (v1 / v2)))
```

Mónadas – Ejemplo básico

- Damos nombre a los recuadros

```
returnId :: ??
returnId x  = x
bindId :: ??
bindId m k = let v = m in k v

eval (Cte n)       = returnId n
eval (Div e1 e2) = bindId (eval e1)
                          (\v1 -> bindId (eval e2)
                                         (\v2 -> returnId (v1 / v2)))
```

Mónadas – Ejemplo básico

- Damos nombre a los recuadros

```
returnId :: ??
returnId x  = x
bindId :: ??
bindId m k = let v = m in k v

eval (Cte n)       = returnId n
eval (Div e1 e2) = eval e1 `bindId` \v1 ->
                   eval e2 `bindId` \v2 ->
                   returnId (v1 / v2)
```

Mónadas – Ejemplo básico

- Damos nombre a los recuadros

```
returnId :: a -> Id a
returnId x  = x
bindId :: Id a -> (a -> Id b) -> Id b
bindId m k = let v = m in k v

eval (Cte n)       = returnId n
eval (Div e1 e2) = eval e1 `bindId` \v1 ->
                   eval e2 `bindId` \v2 ->
                   returnId (v1 / v2)
```

Retomar el camino principal…