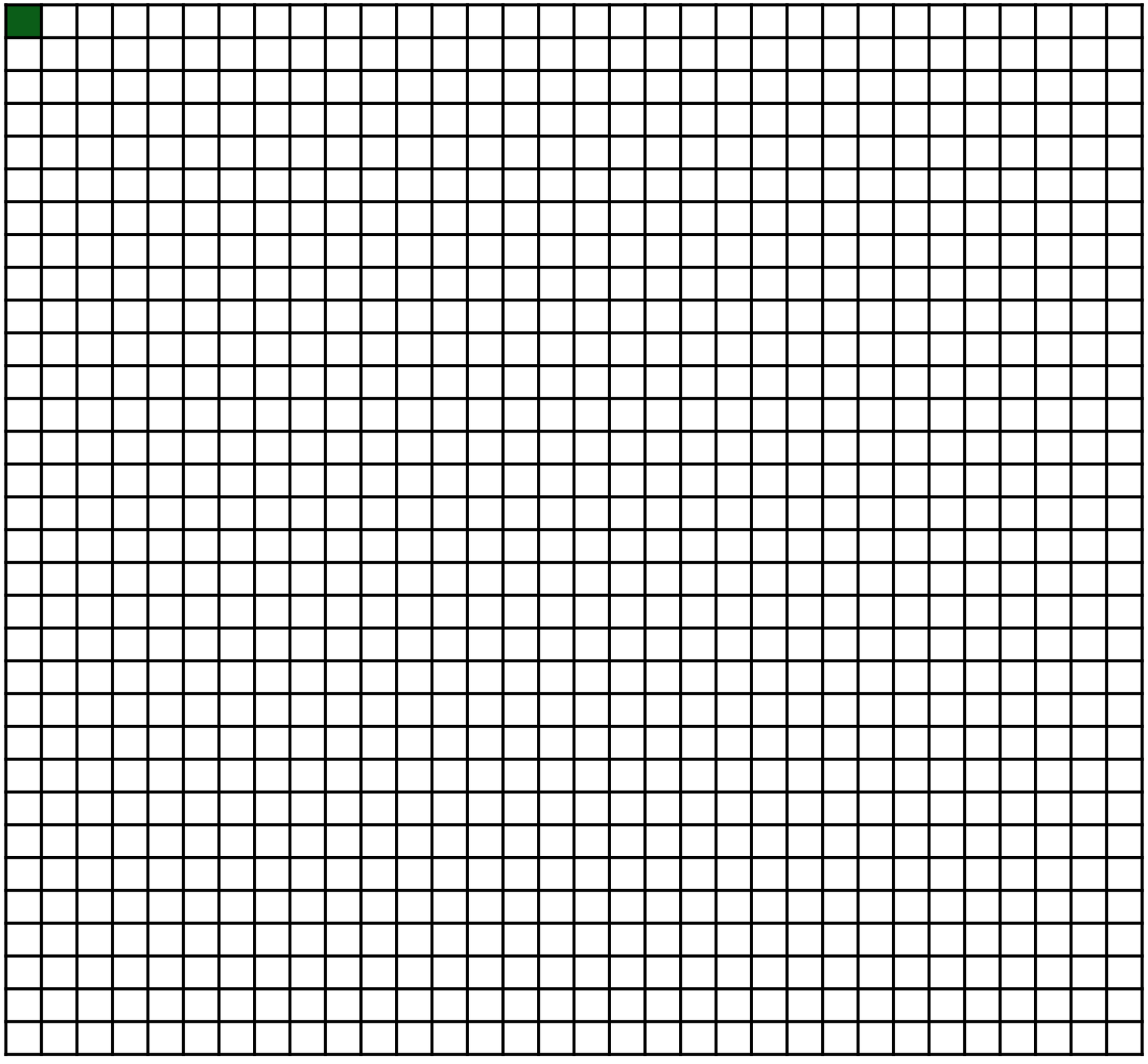
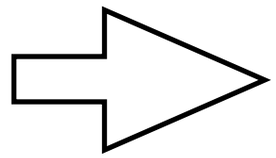
Stream Fusion in Continuation Passing Style

Ben Lippmeier
SAWDAP 9/12/2014



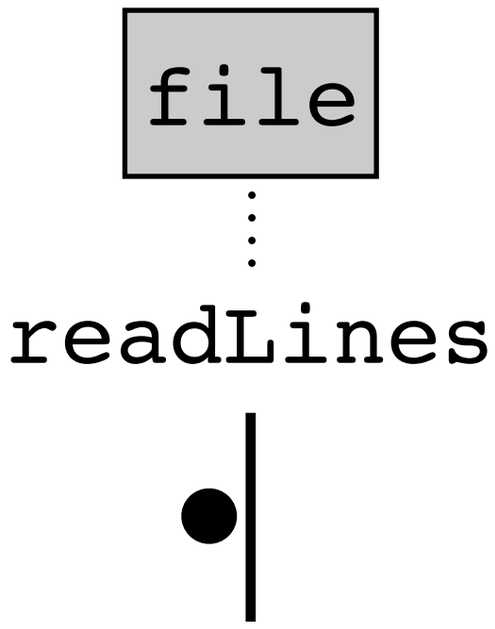
```
BM-1     1990-12-31    1150.16  0  1
BM-1     1991-01-31    5169.95  0  0
BM-1     1991-02-28    1447.14  1  1
BM-1     1991-03-31    2193.71  0  0
BM-1     1991-04-30    1412.16  0  1
BM-1     1991-05-31    3224.93  1  1
BM-1     1991-06-30    3343.45  0  0
BM-103   1996-04-30    3421.69  0  0
BM-103   1996-05-31    3333.66  1  1
BM-103   1996-06-30    6412.49  0  0
BM-103   1996-07-31    1417.03  0  0
BM-103   1996-08-31    1451.25  1  1
BM-103   1996-09-30    5452.64  0  0
BM-103   1996-10-31    1435.04  0  0
BM-103   1996-11-30    4475.29  1  1
BM-106   2007-05-31    1651.47  0  0
BM-106   2007-06-30    6472.68  0  0
BM-106   2007-07-31    7357.13  1  1
BM-106   2007-08-31    1481.75  0  0
BM-106   2007-09-30    1405.94  0  0
BM-106   2007-10-31    1350.67  0  1
BM-106   2007-11-30    1295.94  1  0
BM-106   2007-12-31    1240.67  0  0
BM-106   2008-01-31    1216.87  1  1
BM-106   2008-02-29    1556.26  0  0
BM-106   2008-03-31    1220.99  1  0
.. on and on for few hundred GB ..
```

```
BM-1     1990-12-31    1150.16  0  1
BM-1     1991-01-31    5169.95  0  0
BM-1     1991-02-28    1447.14  1  1
BM-1     1991-03-31    2193.71  0  0
BM-1     1991-04-30    1412.16  0  1
BM-1     1991-05-31    3224.93  1  1
BM-1     1991-06-30    3343.45  0  0
BM-103   1996-04-30    3421.69  0  0
BM-103   1996-05-31    3333.66  1  1
BM-103   1996-06-30    6412.49  0  0
BM-103   1996-07-31    1417.03  0  0
BM-103   1996-08-31    1451.25  1  1
BM-103   1996-09-30    5452.64  0  0
BM-103   1996-10-31    1435.04  0  0
BM-103   1996-11-30    4475.29  1  1
BM-106   2007-05-31    1651.47  0  0
BM-106   2007-06-30    6472.68  0  0
BM-106   2007-07-31    7357.13  1  1
BM-106   2007-08-31    1481.75  0  0
BM-106   2007-09-30    1405.94  0  0
BM-106   2007-10-31    1350.67  0  1
BM-106   2007-11-30    1295.94  1  0
BM-106   2007-12-31    1240.67  0  0
BM-106   2008-01-31    1216.87  1  1
BM-106   2008-02-29    1556.26  0  0
BM-106   2008-03-31    1220.99  1  0
.. on and on for few hundred GB ..
```

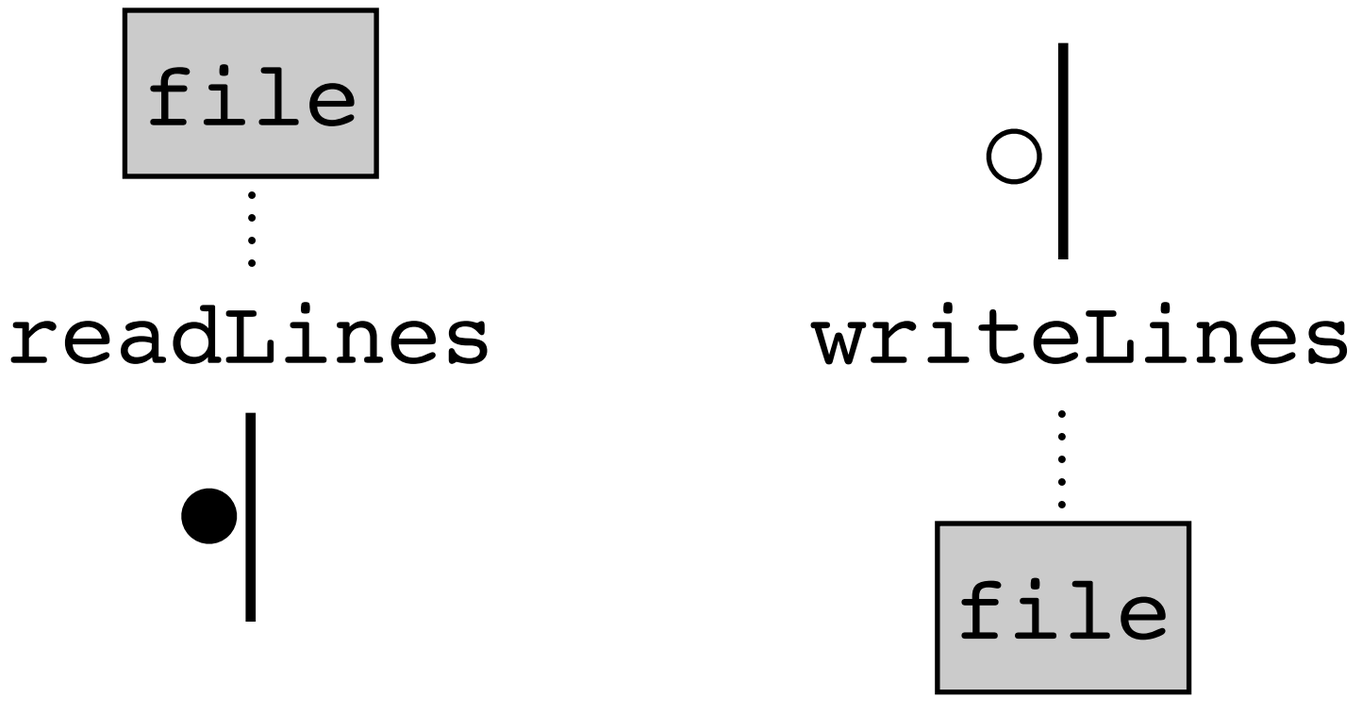


```
BM-1     15450.16    1
BM-103   345121.69   1
BM-106   165451.47   0
…
```

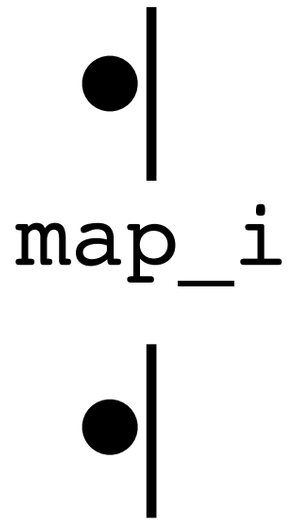
```
map :: (a -> b) -> A a -> A b
```



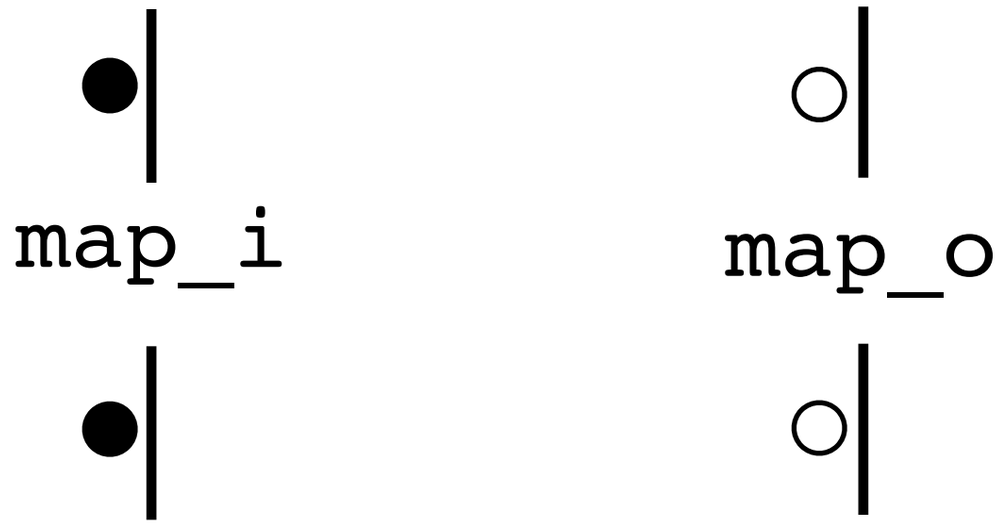

```
readLines :: FilePath -> I String
```




```
readLines :: FilePath -> I String

writeLines :: FilePath -> O String
```

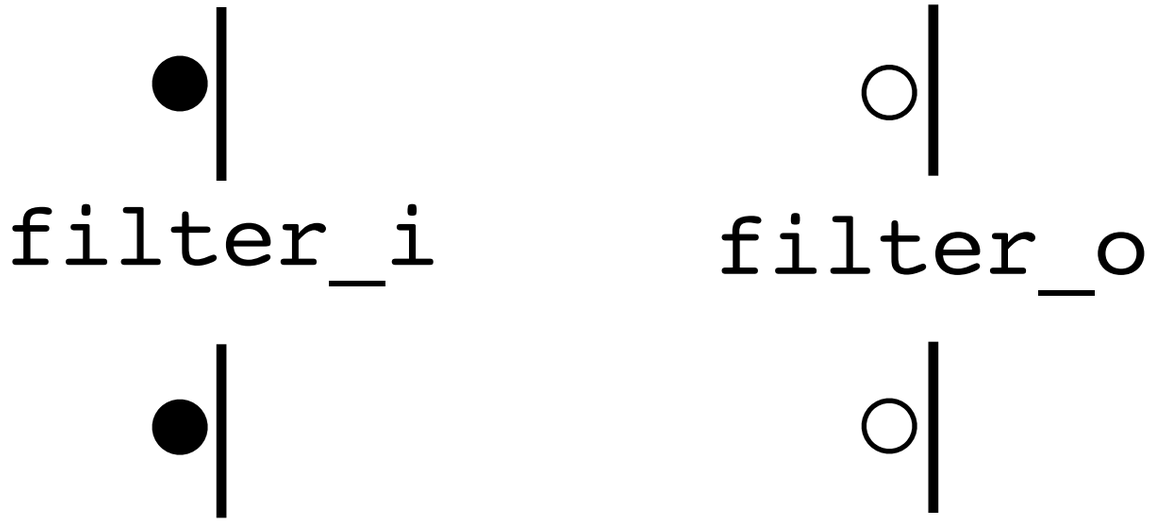



```
map_i :: (a -> b) -> I a -> I b
```

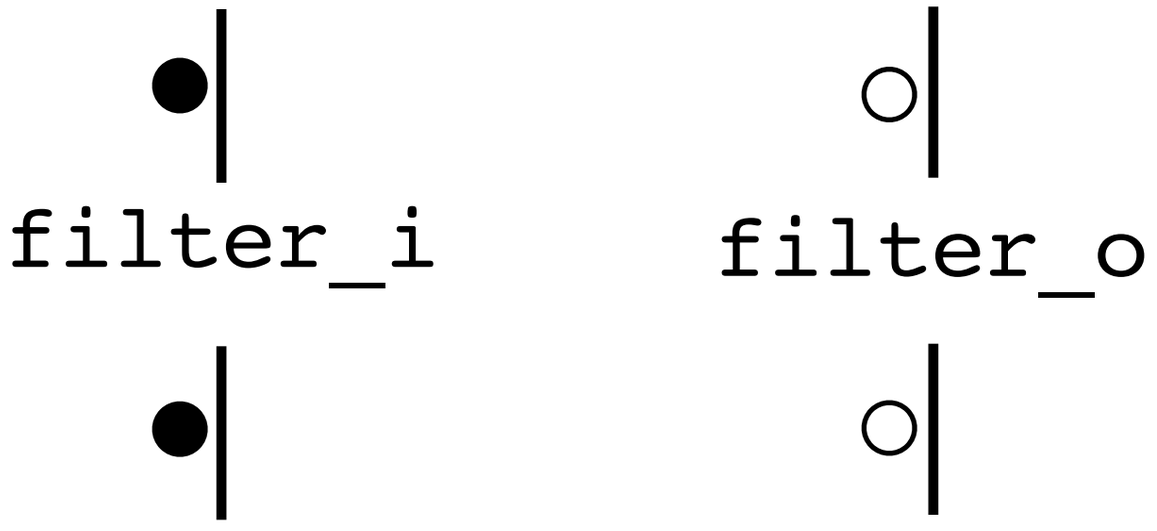



```
map_i :: (a -> b) -> I a -> I b

map_o :: (a -> b) -> O b -> O a
```

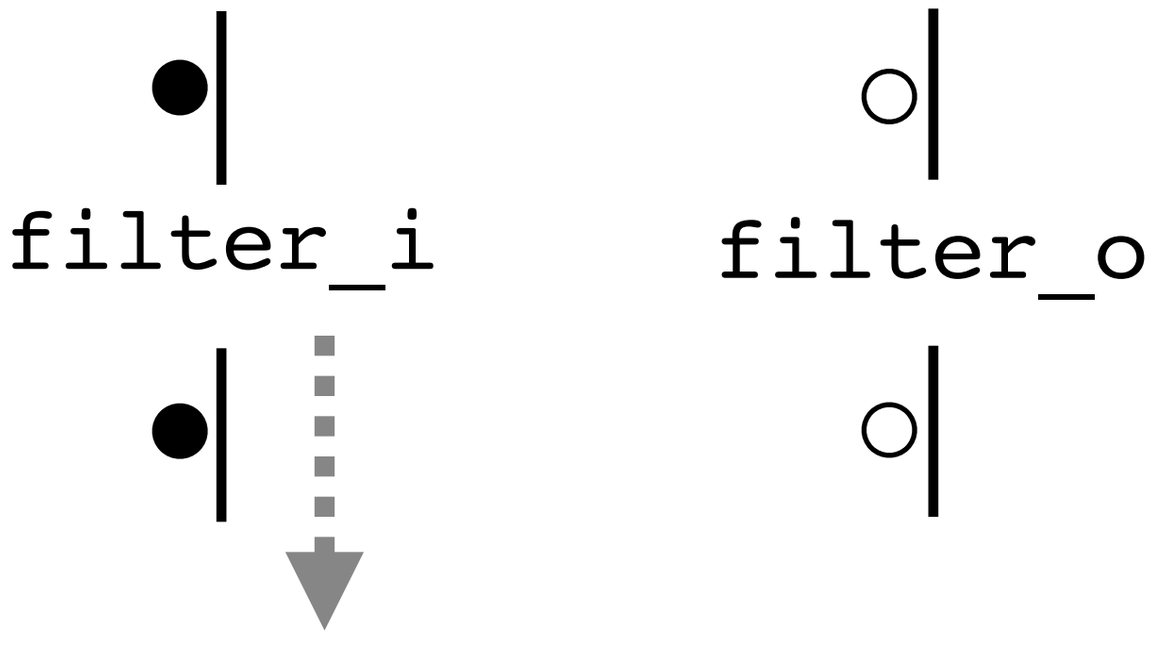
“contramap”



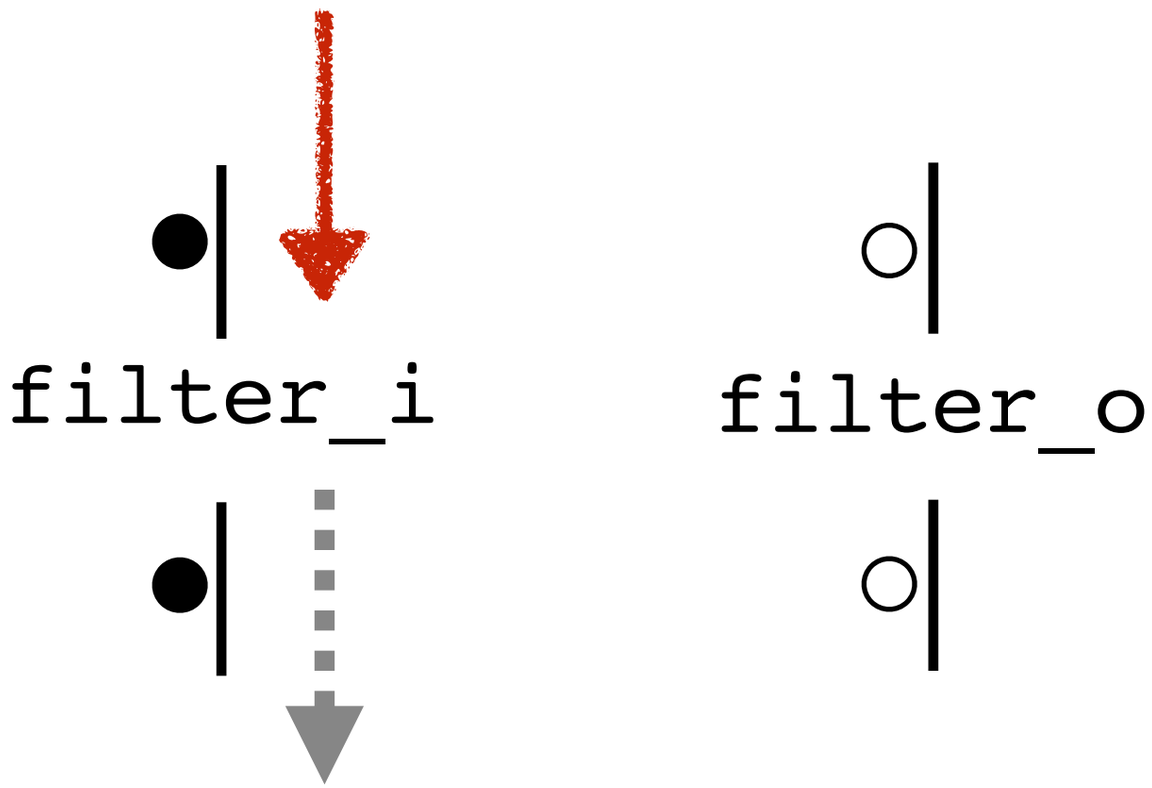

```
filter_i :: (a -> Bool) -> I a -> I a

filter_o :: (a -> Bool) -> O a -> O a
```



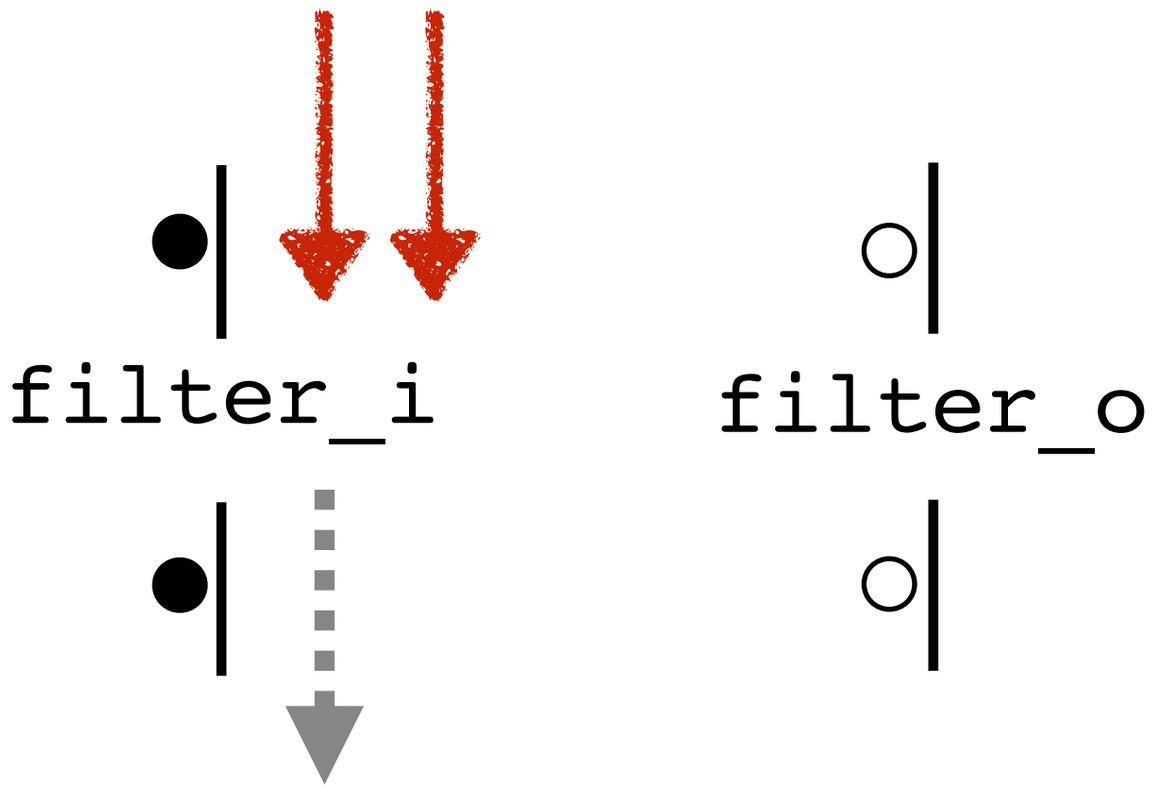

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



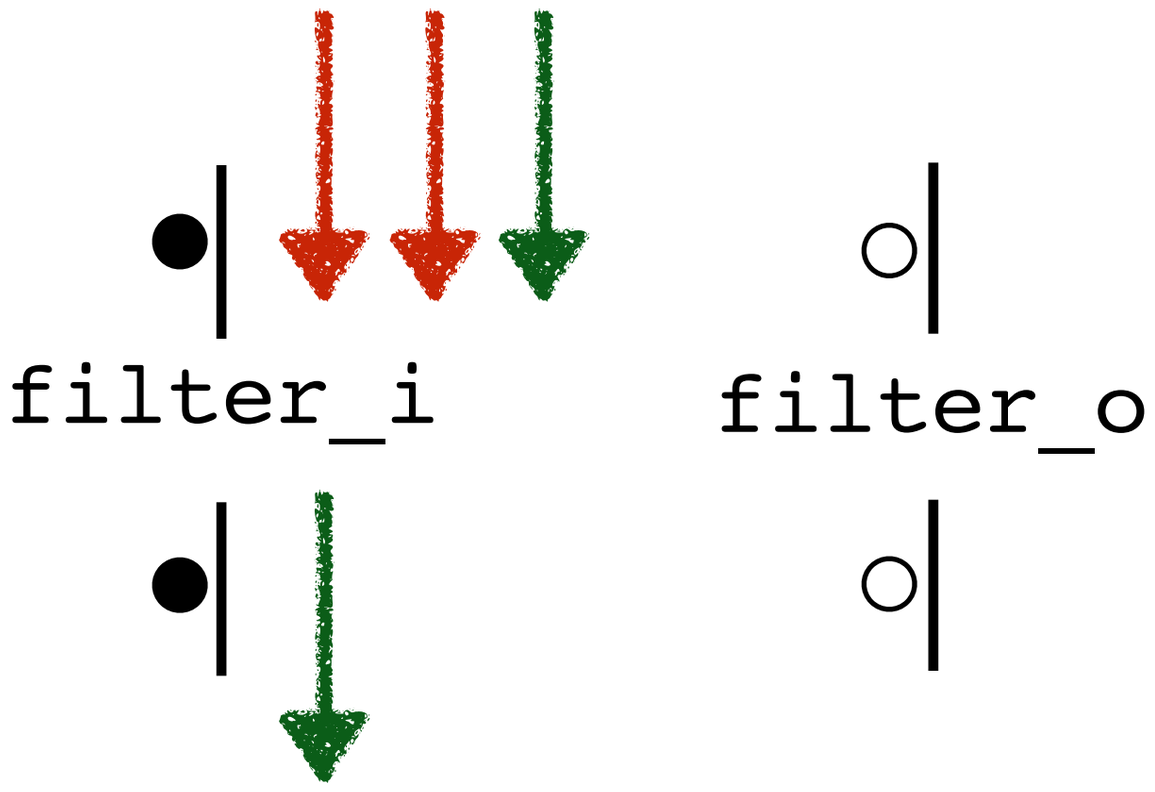

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



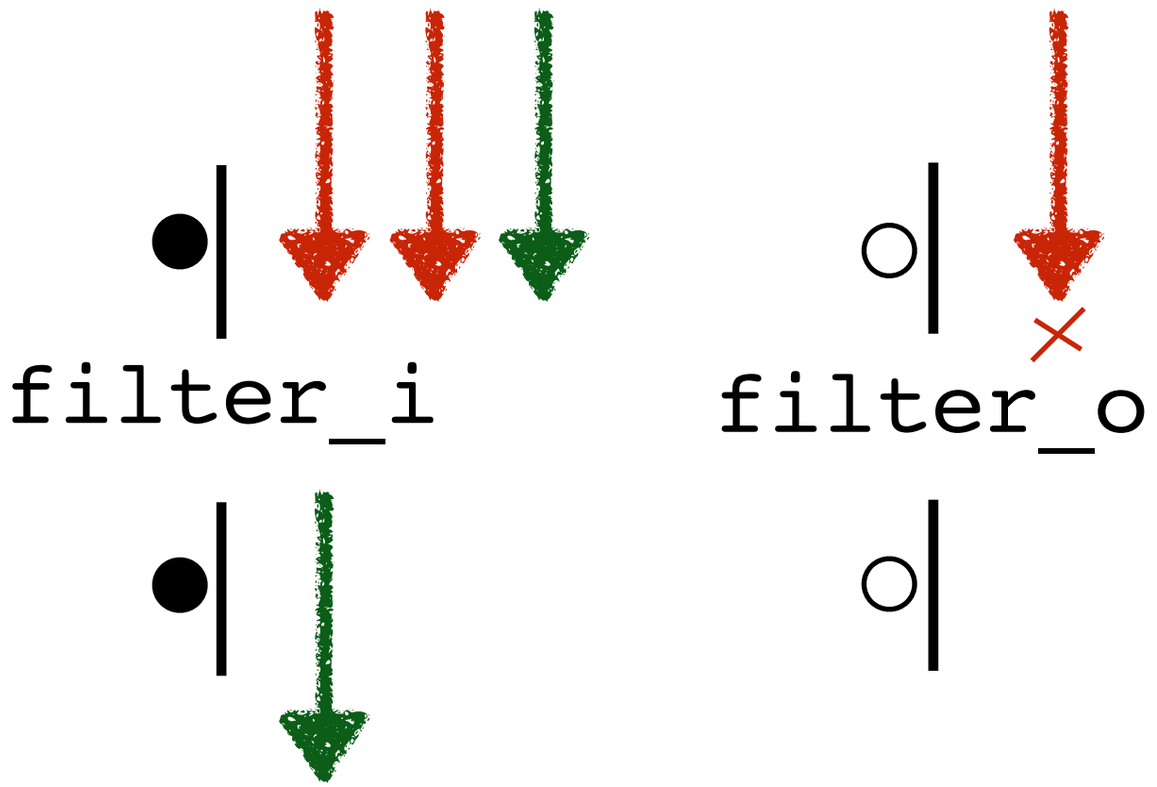

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



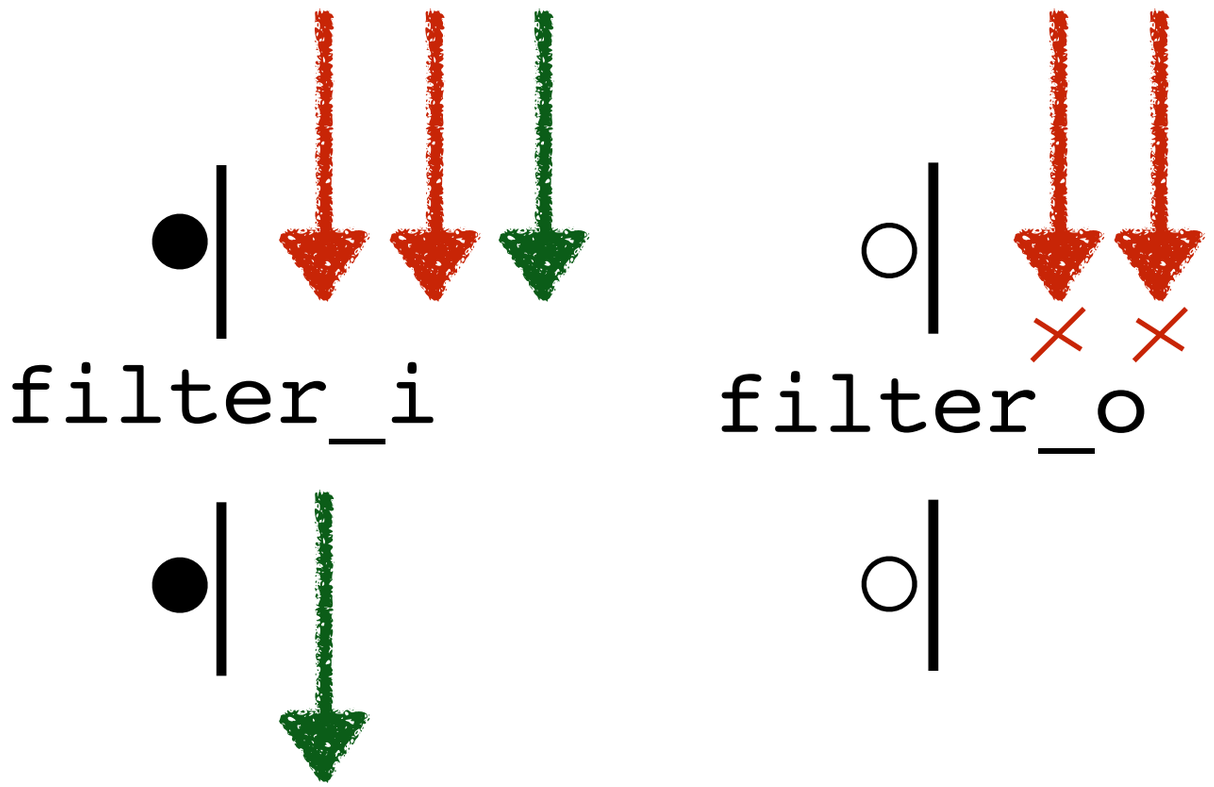

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



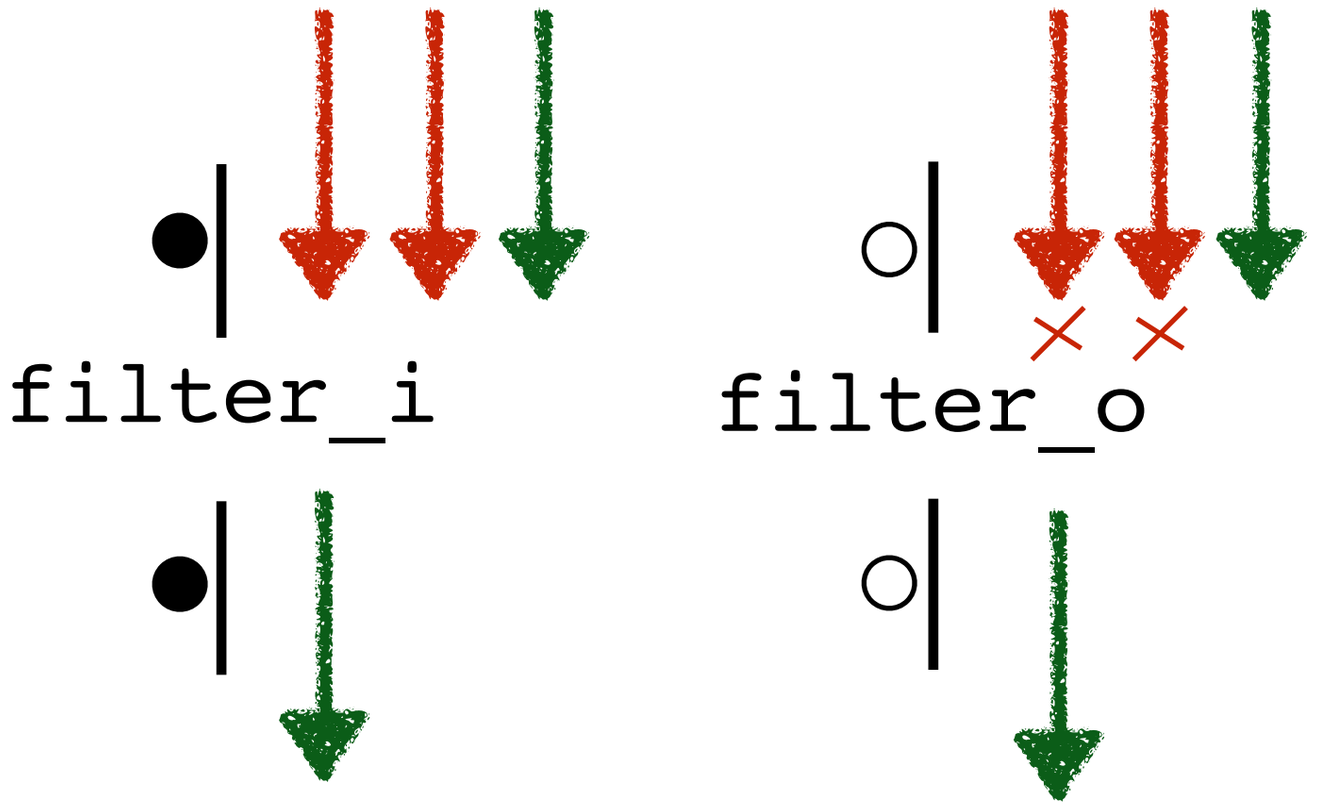

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



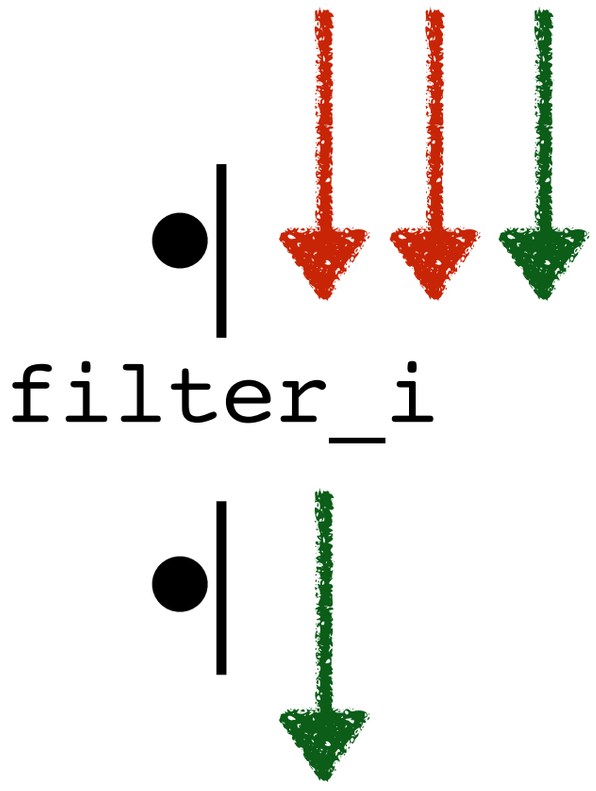

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```

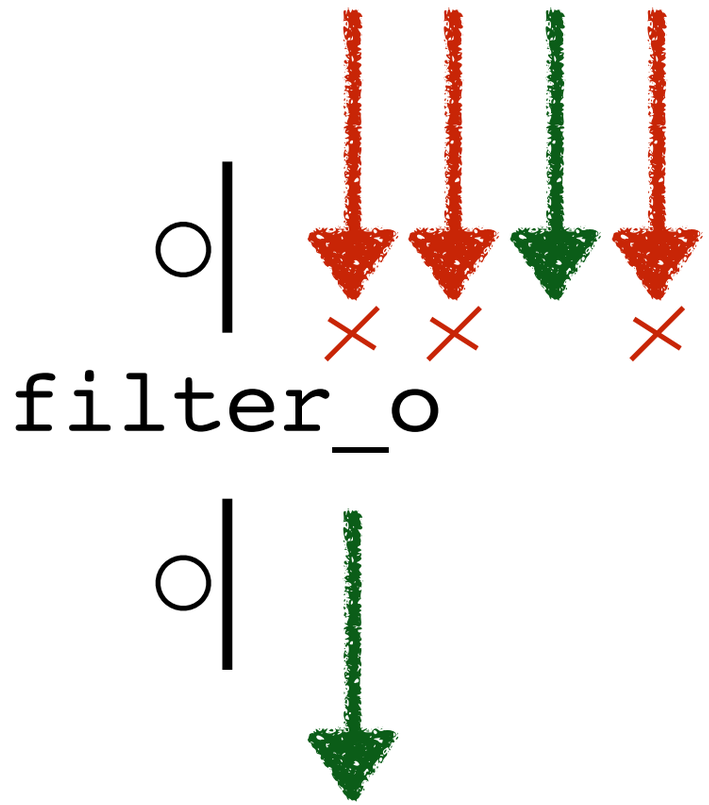



```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```

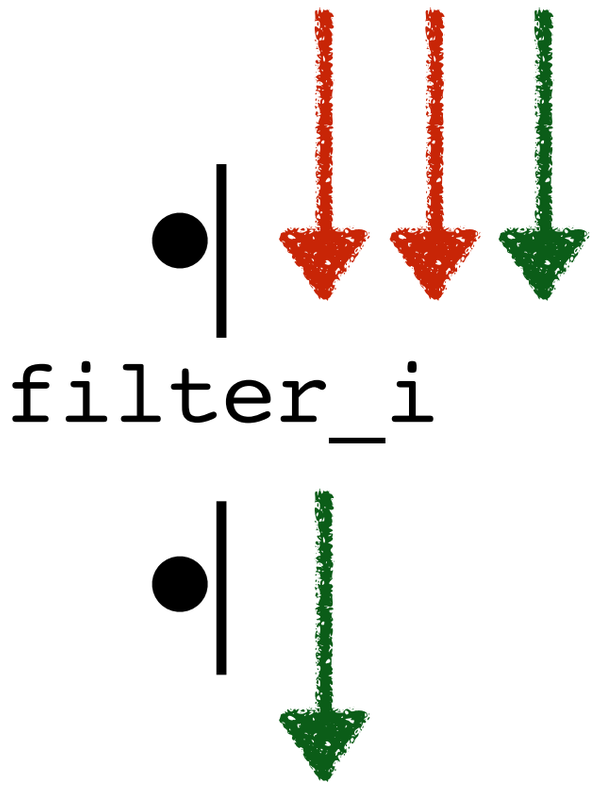


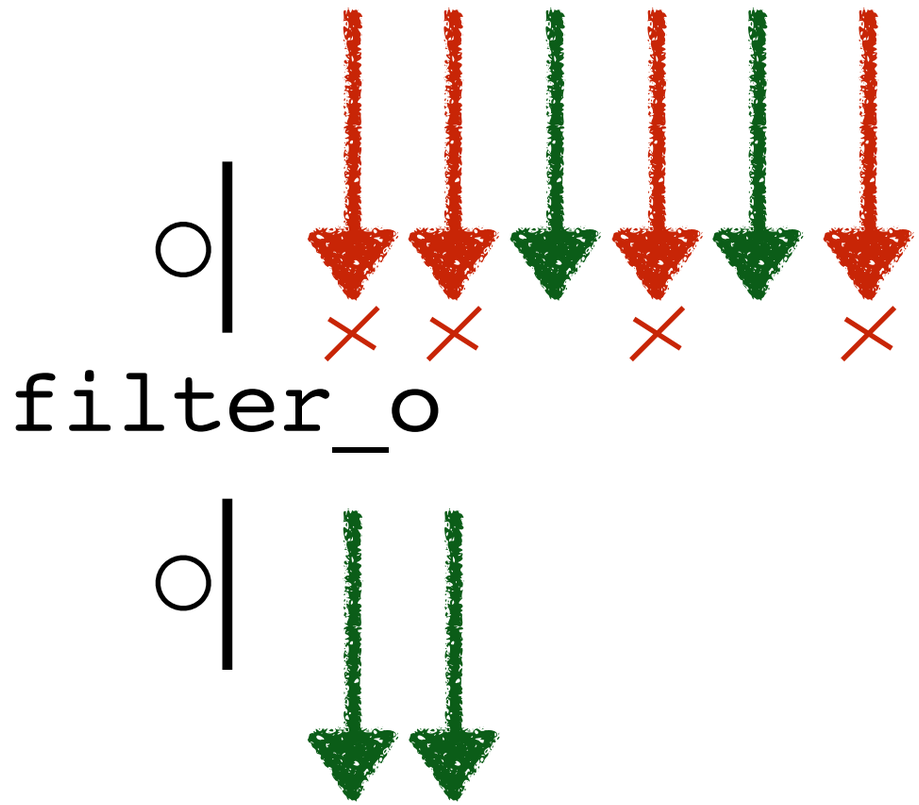

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```



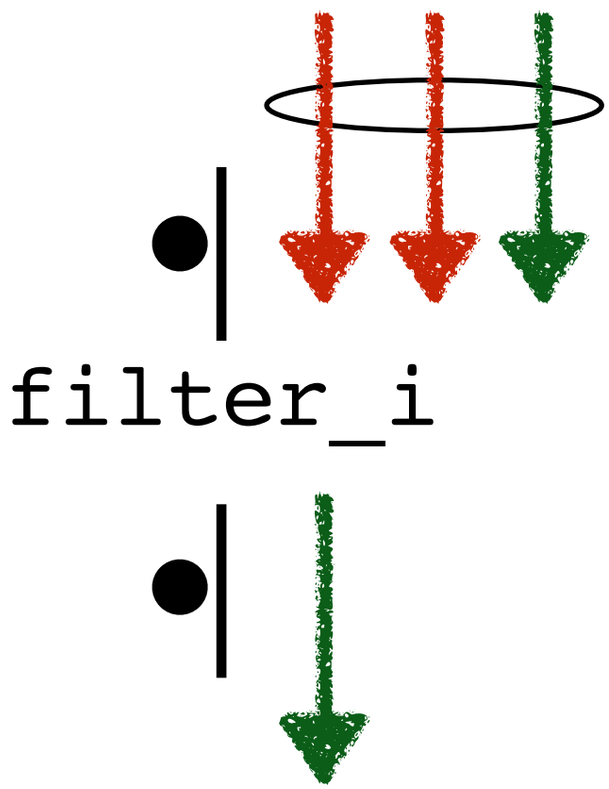



```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```

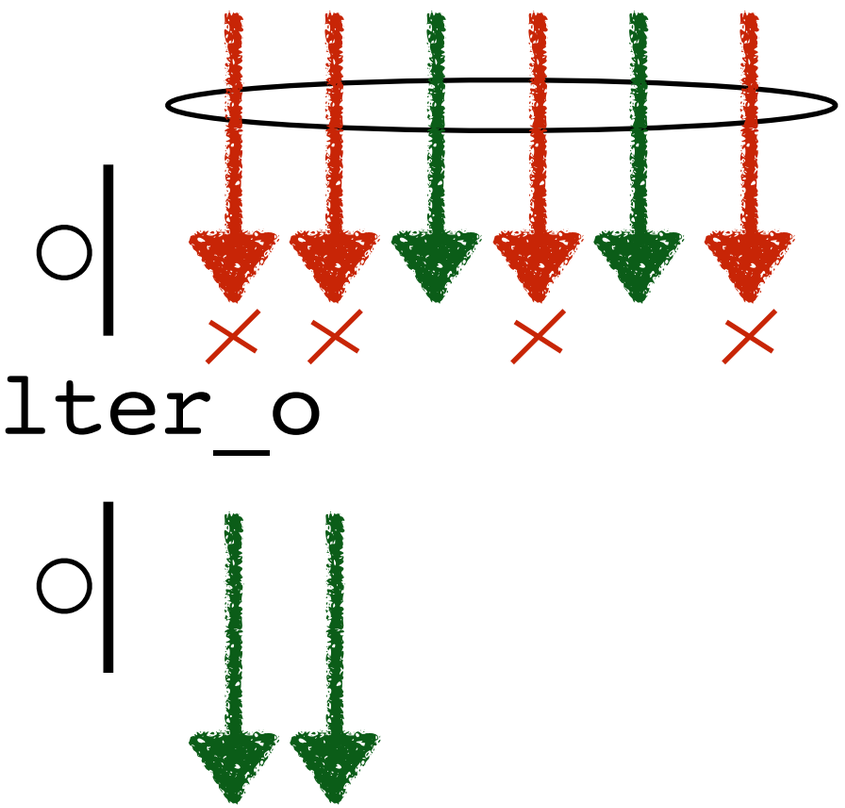





```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```

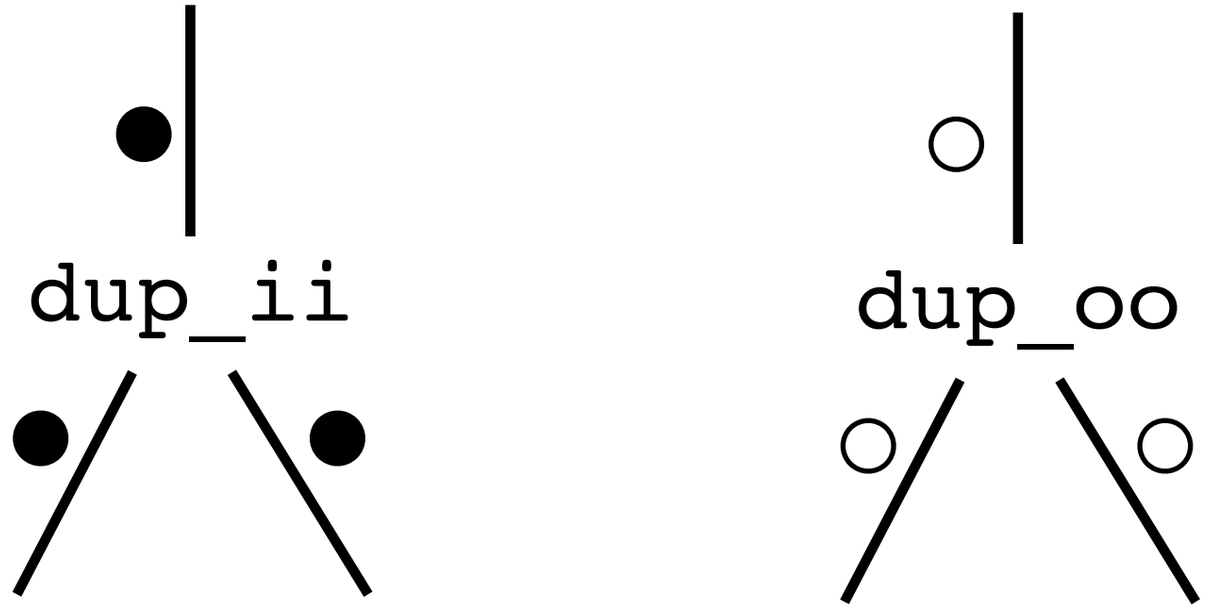


filter_i

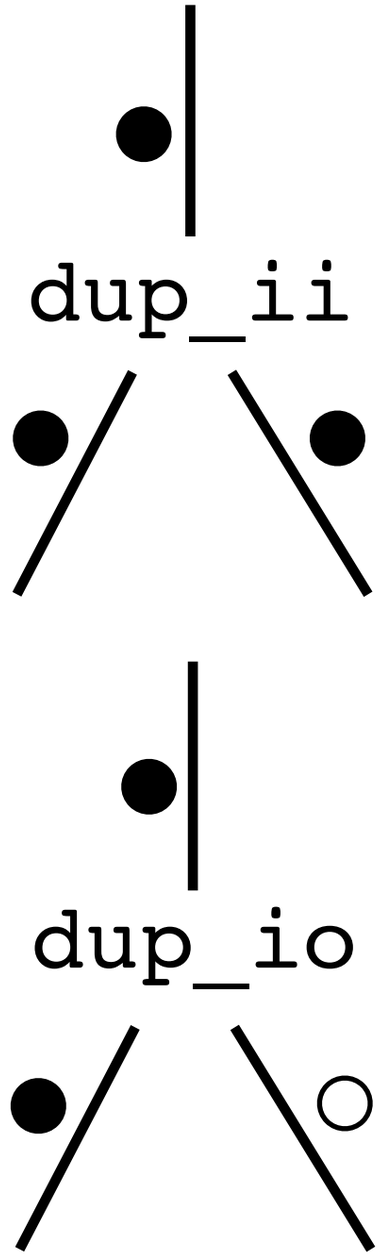


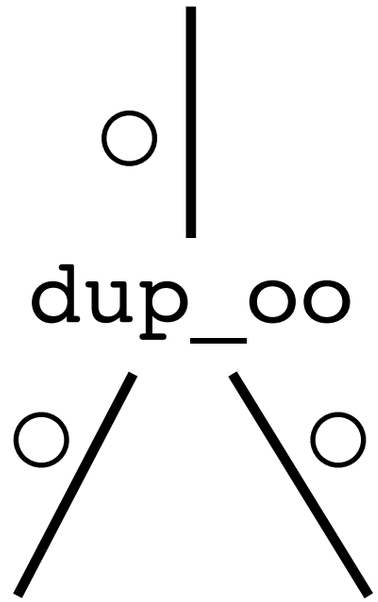
filter_o

```
filter_i :: (a -> Bool) -> I a -> I b

filter_o :: (a -> Bool) -> O a -> O a
```

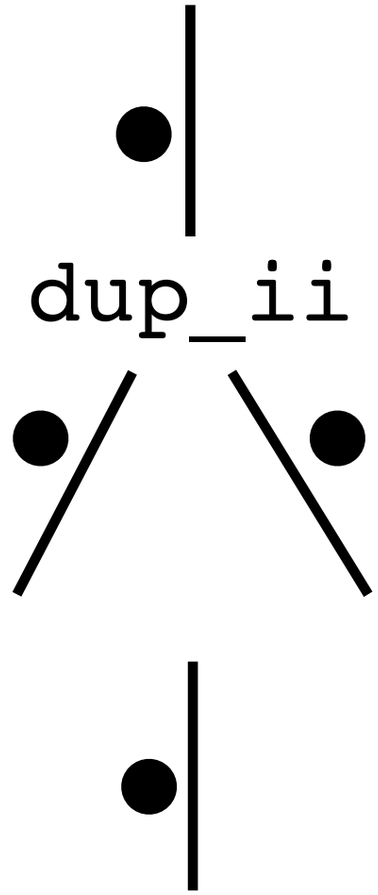


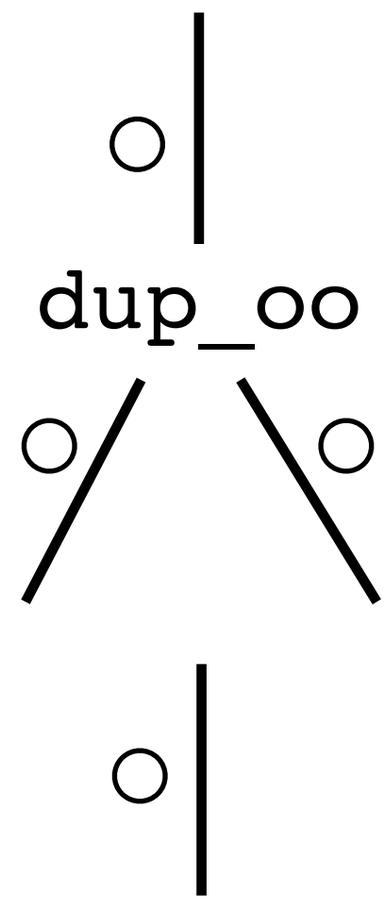

```
dup_ii :: I a —> (I a, I a)
dup_oo :: O a -> O a -> O a
```

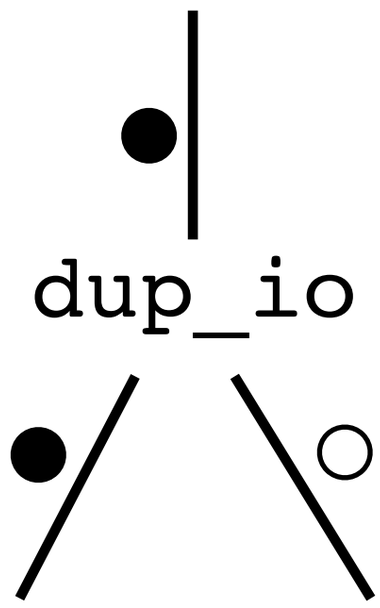


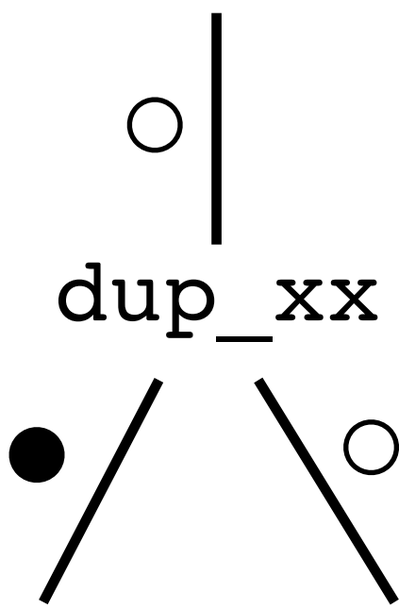



```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
```



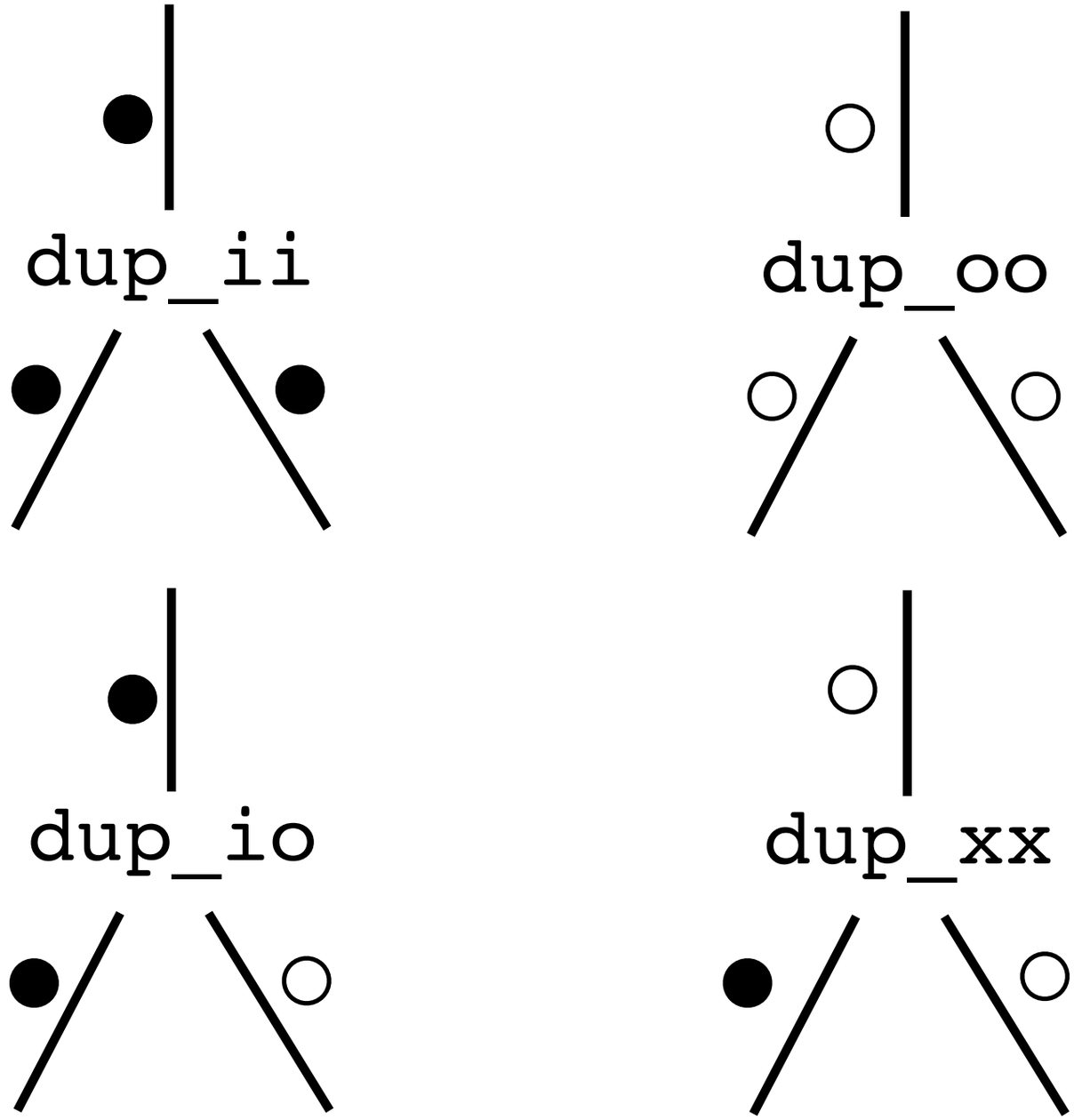






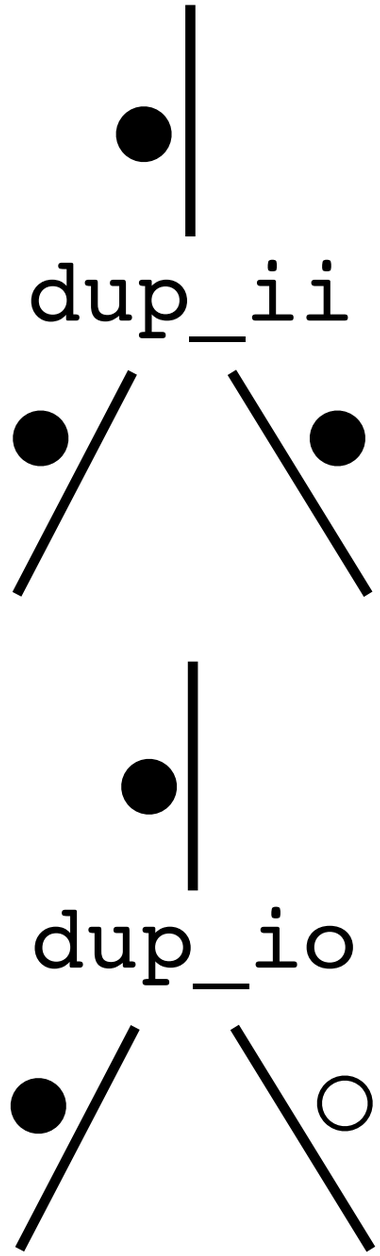

```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
dup_xx :: I a -> O a -> O a
```

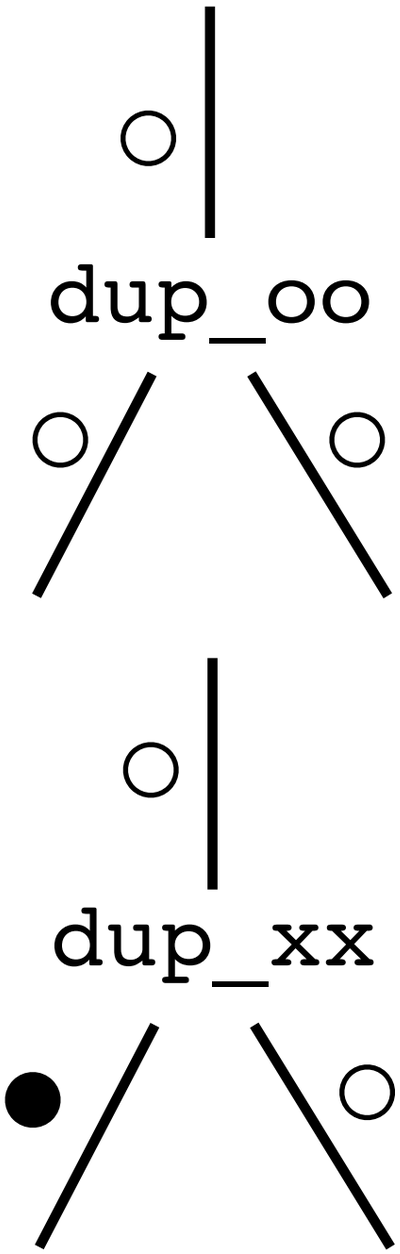
Buffer




```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
dup_xx :: I a -> O a -> O a
```

Buffer

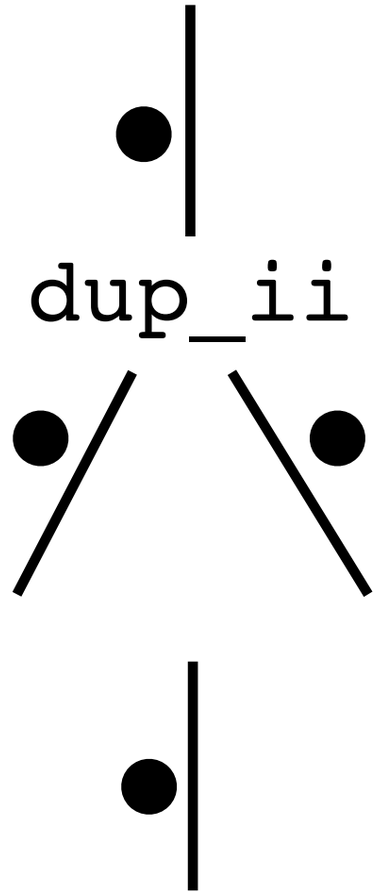


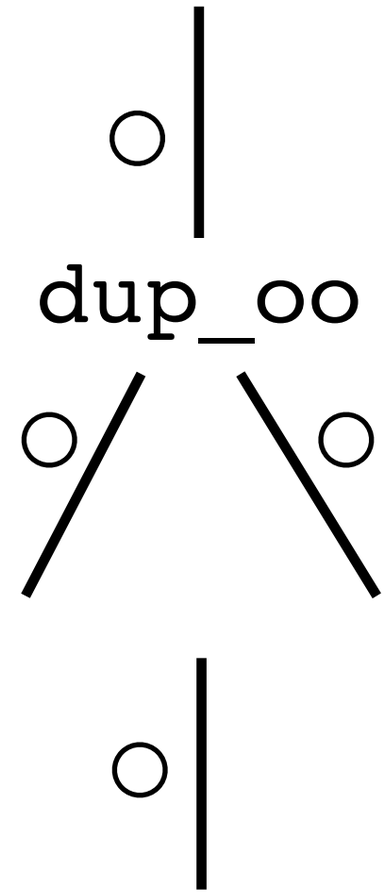


OK


```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
dup_xx :: I a -> O a -> O a
```

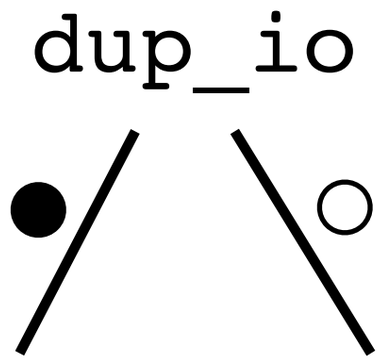
Buffer

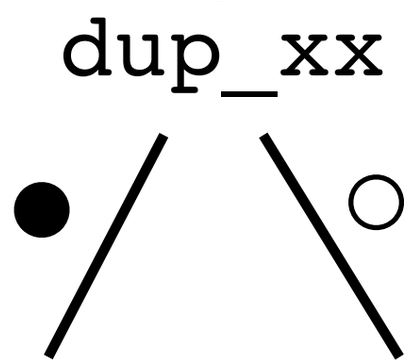




OK

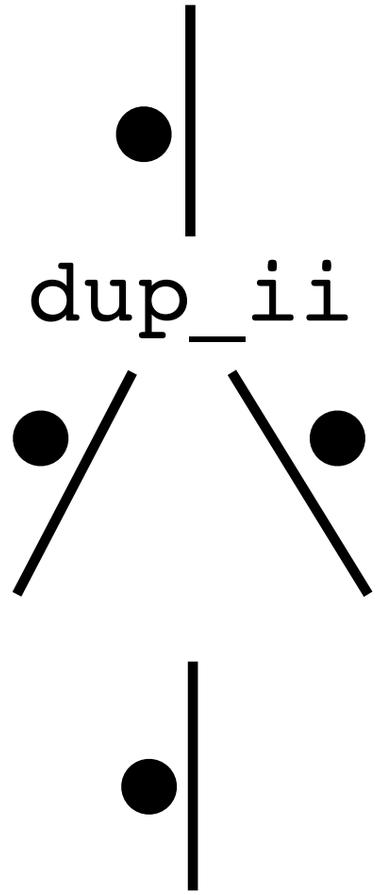
OK

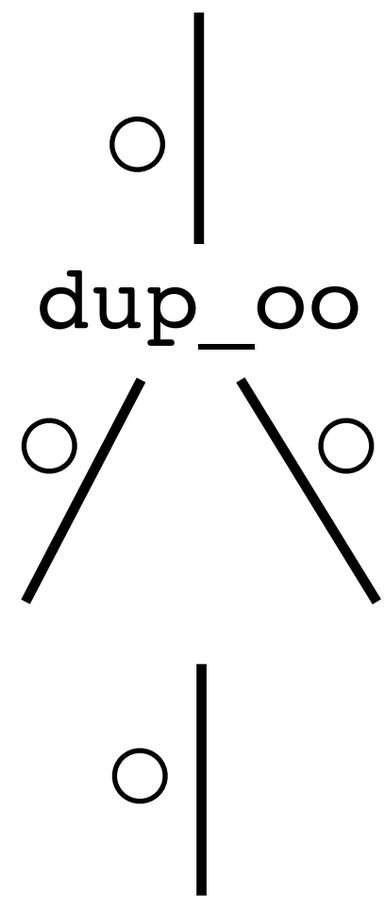





```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
dup_xx :: I a -> O a -> O a
```

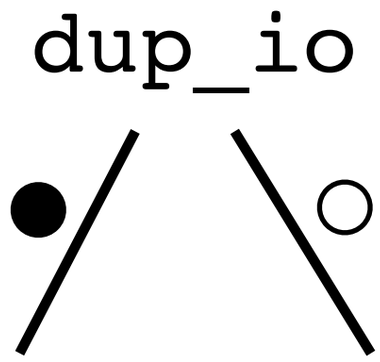
Buffer

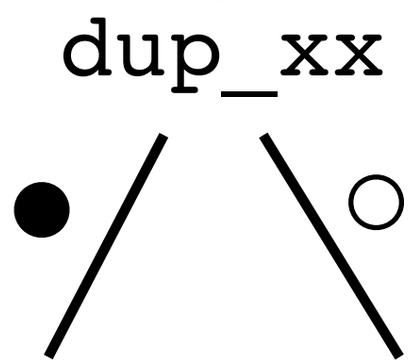




OK

OK



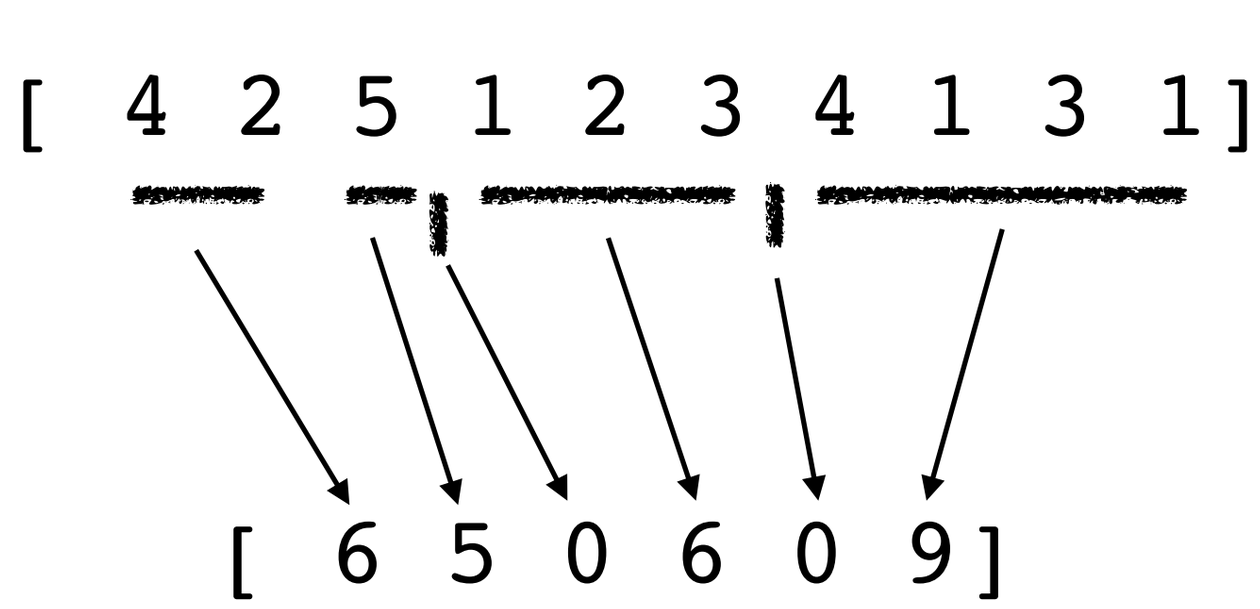


Invalid

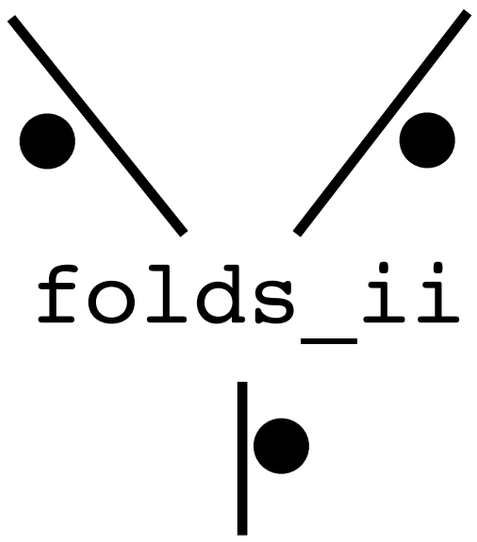
```
dup_ii :: I a —> (I a, I b)
dup_oo :: O a -> O a -> O a
dup_io :: I a -> O a -> I a
dup_xx :: I a -> O a -> O a
```

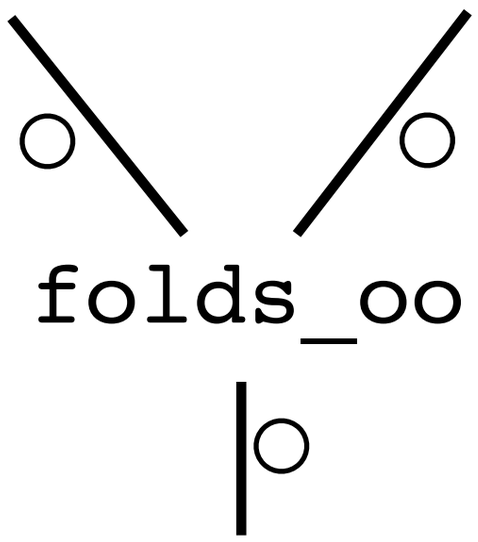
```
folds :: (a -> a -> a) -> a
      -> A Int -> A a -> A a
```

```
[ 2 1 0 3 0 4]
```



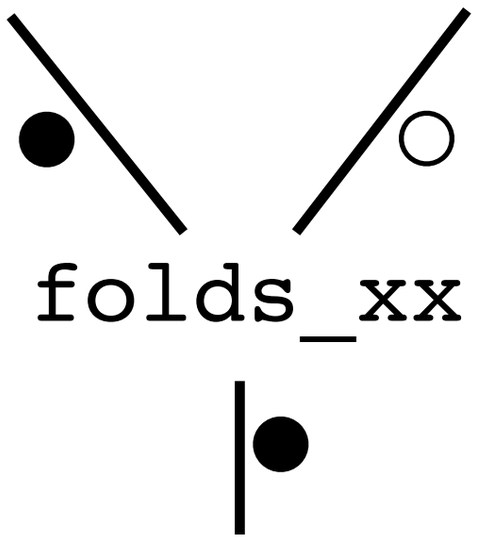
OK

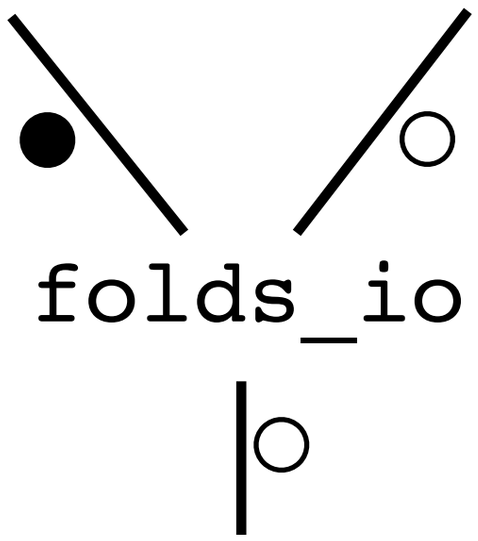




Buffer

Invalid





OK

```
folds_ii :: .. -> I Int -> I a -> I a
folds_oo :: .. -> O Int -> O a -> O a
folds_xx :: .. -> I Int -> O a -> I a
folds_io :: .. -> I Int -> O a -> O a
```

Continuations

```
type Sink a    = (a -> IO (), IO ())
```

push eject

```
type Sink a   = (a -> IO (), IO ())
```

push eject

```
type Source a = IO (Maybe a)
```

??

```haskell
type Sink a   = (a -> IO (), IO ())
```

push eject

```haskell
type Source a = IO (Maybe a)
```

??

```haskell
type Source a = (a -> IO ()) -> IO () -> IO ()
```

eat eject

```
folds_ii :: (a -> a -> a)    — worker function
         -> a                — neutral value
         -> Source Int       — segment lengths
         -> Source a         — segment data
         -> IO (Source a)

folds_ii f z (Source pullLen) (Source pullX)
 = return $ Source pull_folds
 where
        pull_folds eat eject
         = pullLen eat_len eject_len
         where
               eat_len (I# len) = loop_folds len z
               eject_len        = eject

               loop_folds !n !acc
                | tagToEnum# (n ==# 0#) = eat acc
                | otherwise
                = pullX eat_x eject_x
                where
                      eat_x x = loop_folds (n -# 1#) (f acc x)
                      eject_x = eject

        {-# INLINE [1] pull_folds #-}
{-# INLINE [2] folds_ii #-}
```

Drain

```
groupsum :: FilePath -> FilePath -> FilePath -> IO ()
groupsum fileInSegs fileInVals fileOutSums
 = do
        -- Group the input segment file to get segment lengths.
        isegs    <- sourceFileLines fileInSegs
        isegLens <- groups_i isegs

        -- Read floating point values from input file.
        istrings <- sourceFileLines fileInVals
        ivals    <- map_i readDouble istrings

        -- Sum up values according to the segment descriptor.
        isums    <- folds_ii (+) 0 isegLens ivals

        -- Create a sink that converts floats back to strings
        -- and writes them to the output file.
        ofile    <- sinkFileBytes fileOutSums
        ofloat   <- map_o (showDoubleFixed 2) ofile

        -- Drain sums into the output.
        drain isums ofloat
```

```
drain :: Source a -> Sink a -> IO ()
```

Questions?