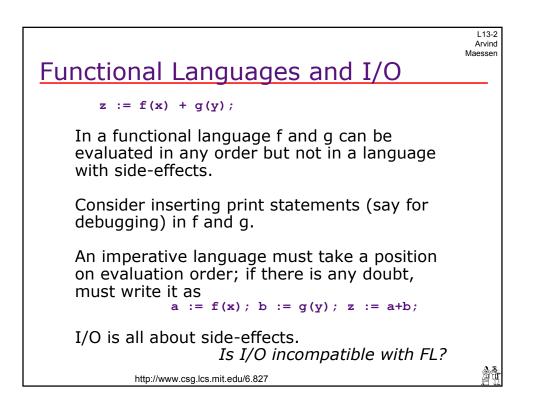
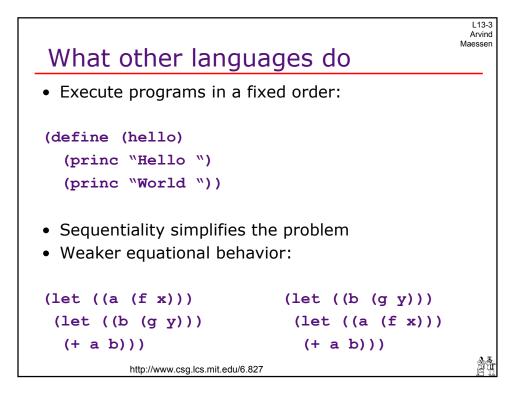
Using Monads for Input and Output Arvind Jan-Willem Maessen Laboratory for Computer Science M.I.T. Lecture 15 http://www.csg.lcs.mit.edu/6.827





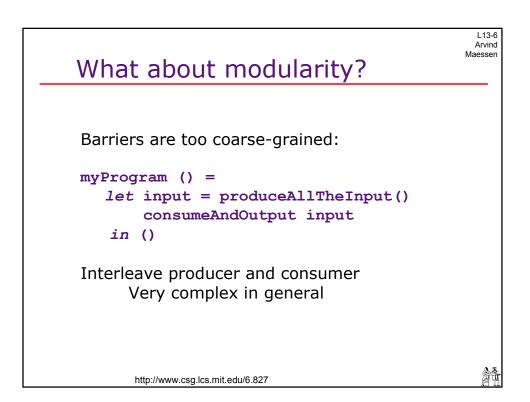
	L13-4
	Arvind
Print string	aessen
i inic sering	
<pre>printString :: String -&gt; ()</pre>	-
printString "Hello World!"	
but what about	
let	
printString "Hello "	
printString "World!"	
in ()	
The string may be printed all jumbled up.	
5, 1, 5, 1	
alternatives:	
Output convention	
Forced sequencing (Usually not available	
in pure FL's)	
	<b>A</b> .Ā.
http://www.csg.lcs.mit.edu/6.827	EU

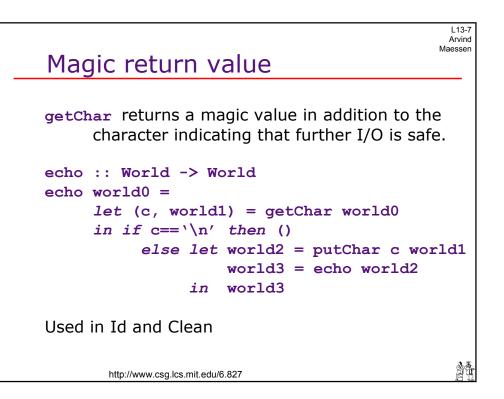
L13-5 Arvind Maessen

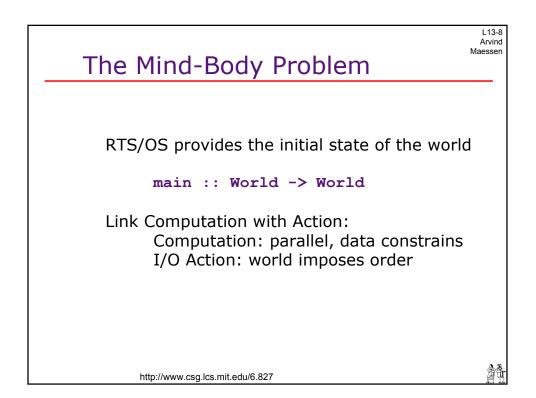
會告

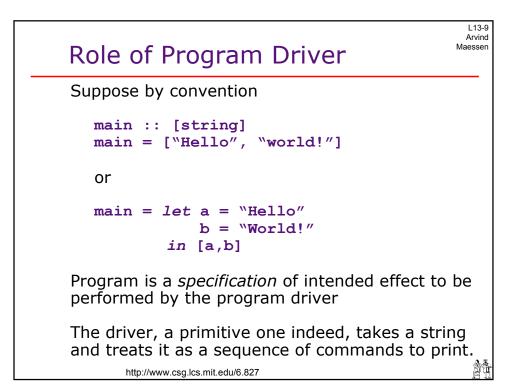
## Need for Sequencing

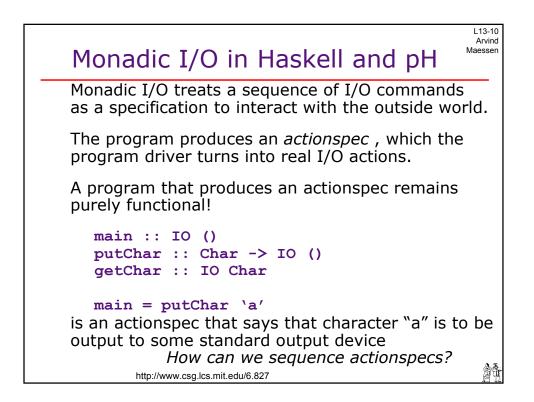
http://www.csg.lcs.mit.edu/6.827

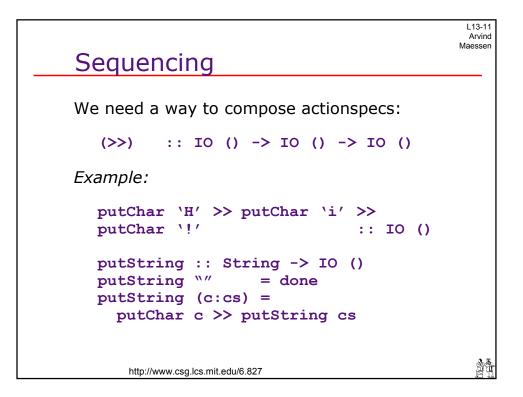


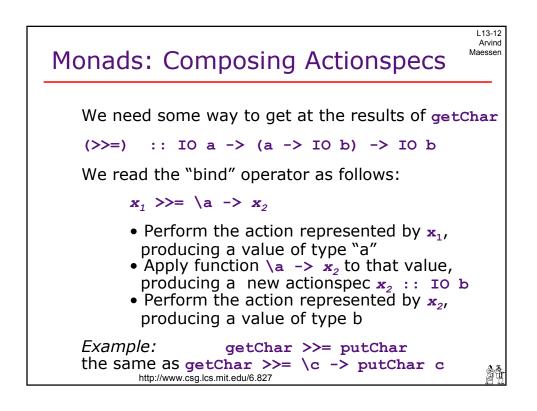


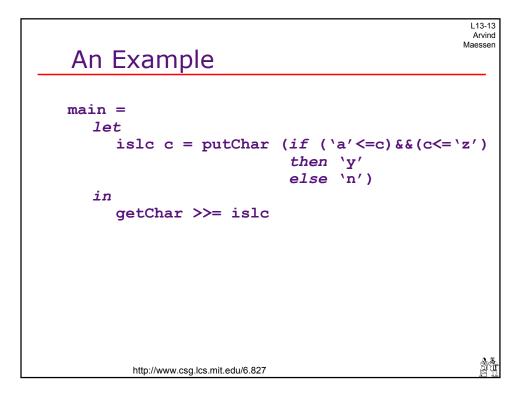


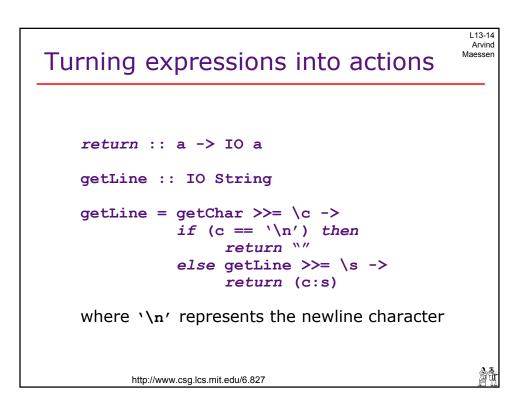


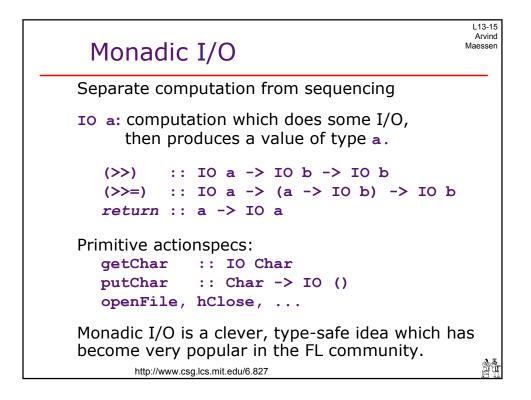


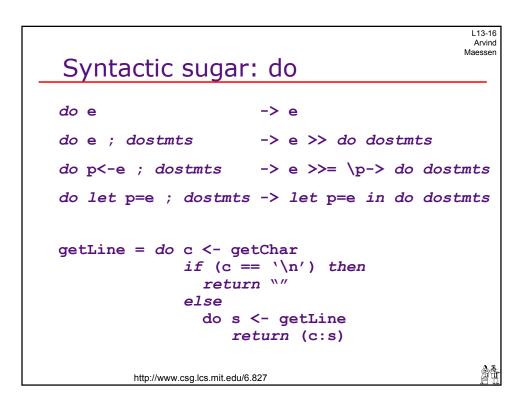








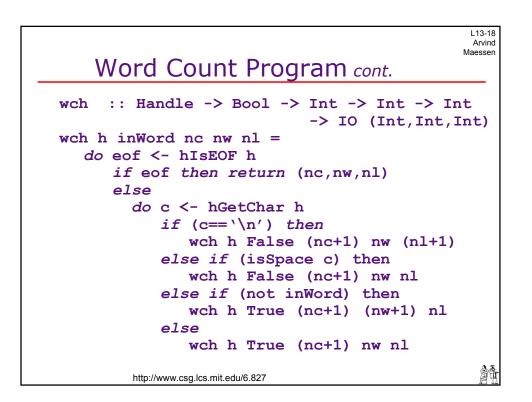




L13-17 Arvind Maessen

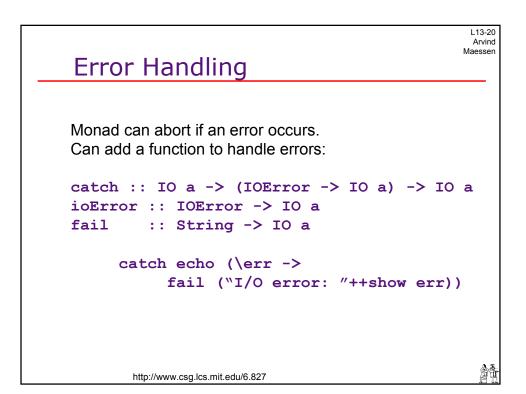
## Example: Word Count Program

```
type Filepath = String
data IOMode = ReadMode | WriteMode | ...
data Handle = ... implemented as built-in type
openFile :: FilePath -> IOMode -> IO Handle
hClose
          :: Handle \rightarrow IO ()
hIsEOF
          :: Handle -> IO Bool
hGetChar :: Handle -> IO Char
          :: String -> IO (Int, Int, Int)
WC
wc filename =
     do h <- openFile filename ReadMode
         (nc,nw,nl) <- wch h False 0 0 0
         hClose h
         return (nc,nw,nl)
                                                 論
        http://www.csg.lcs.mit.edu/6.827
```



L13-19 Arvind Maessen

## Maessen Calling WC main :: IO () main = do [filename] <- getArgs</pre> (nc,nw,nl) <- wc filename</pre> putStr " " putStr (show nc) putStr " \_\_\_\_ putStr (show nw) putStr " \_\_\_\_\_// putStr (show nl) putStr " " putStr filename putStr "\n" 2A http://www.csg.lcs.mit.edu/6.827



L13-21 Arvind Maessen

## An Example

