# HasView

Brandon Wang and Kaushik Iyer
University of California at Berkeley

## Introduction

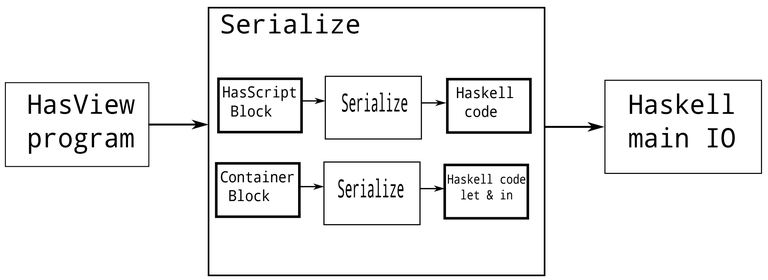
- A system that allows visual Haskell programming
- Use ideas from dataflow programming:
  - functions ↦ blocks
  - recursion ↦ nesting
  - control flow ↦ arrows
- Requirements:

**Extensible** Need to be able to easily write new structures — use Python

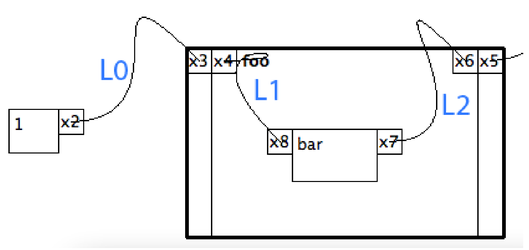
**Modular** Class based structure to allow for abstraction of GUI elements

**Compilation** Need to be able to compile to GHC-compatible Haskell

## Serialization




## Serialization and Resolution Example




## Serialization and Resolution Explanation

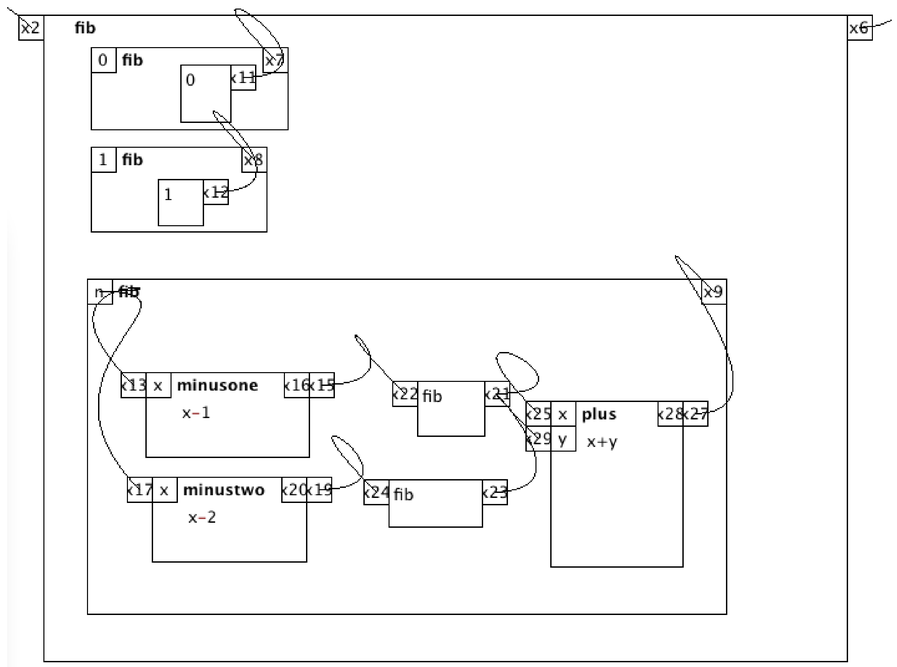
Serialization

```
foo x4 = let l1 = x4
         in l2
```

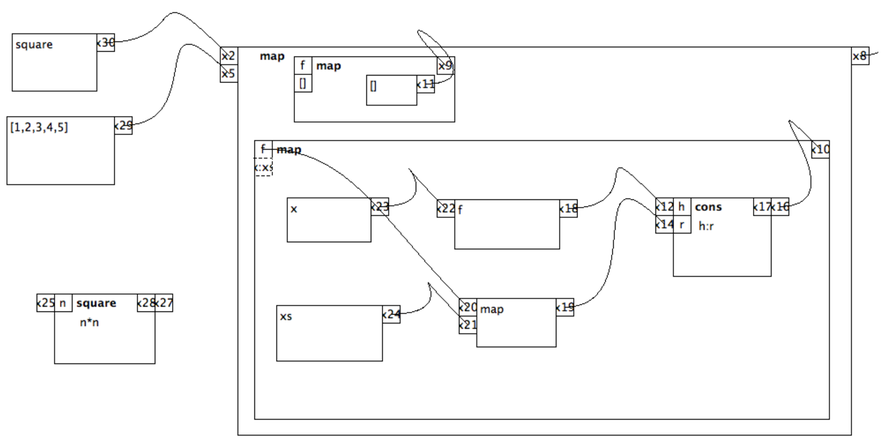
Resolution

```
l2 = (bar l1)
```

## Calculating `fib`




## Calculating `map`




## Compiled `fib` code

```
main = let fib 0 = let l1 = 0
                   in l1
           fib 1 = let l2 = 1
                   in l2
           fib n = let l6 = n
                       l7 = n
                       minustwo x = x - 2
                       l12 = plus l10 l11
                       minusone x = x - 1
                       plus x y = x + y
                       l10 = (fib 18)
                       l11 = (fib 19)
                       l8 = minusone l6
                       l9 = minustwo l7
                   in l12
           l0 = fib l13
           l13 = 25
       in print l0
```

## Compiled `map` code

```
main = let l10 = map  19 18
           map  f [] = let l0 = []
                       in l0
           map  f (x:xs) = let l6 = (map  l5 l4)
                               l7 = cons  l3 l6
                               l4 = xs
                               l5 = f
                               l2 = x
                               l3 = (f  l2)
                               cons  h r = h:r
                           in l7
           square n = n*n
           l8 = [1,2,3,4,5]
           l9 = square
       in print l10
```


Mail: {kiyer,brandonwang}@berkeley.edu