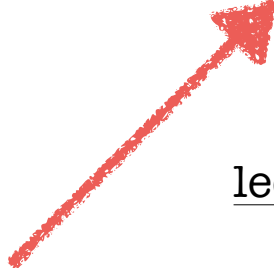


(Embedded) Domain-Specific Languages

and Free Life Advice

Lee Pike
Galois, Inc.
leepike@galois.com
[@pike7464](#)



Worth
every penny



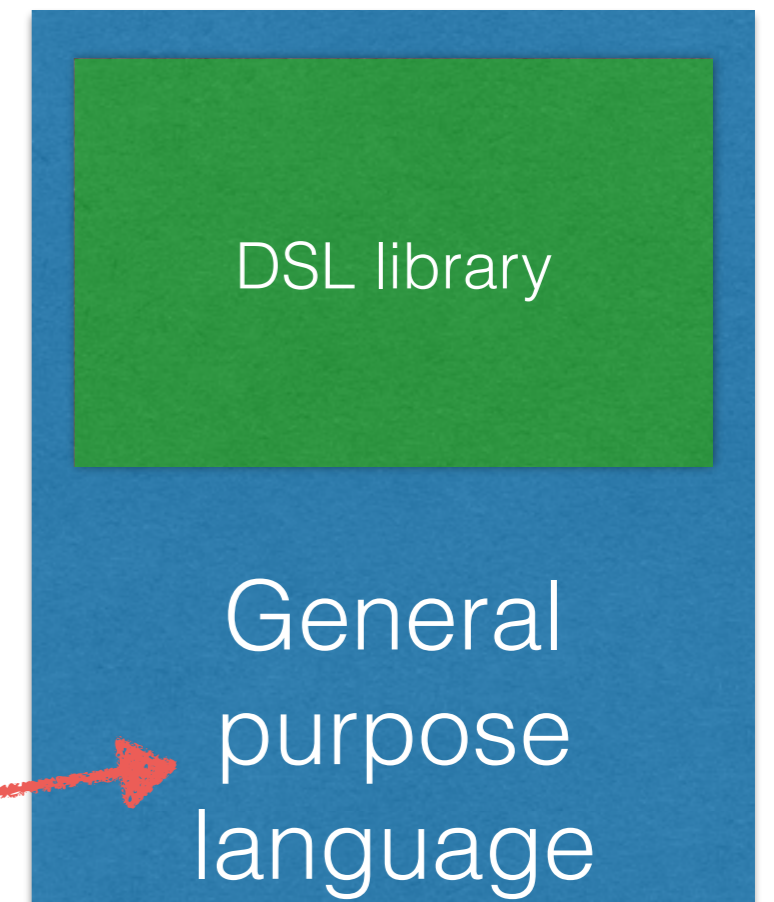
| galois |

halfway home
for recovering academics

DSLs

- DSLs: Excel, MATLAB, awk, Make, LaTeX, SQL
- Non-DSLs: Java, Haskell, C
- Embedded DSLs:
 - A DSL as a library
 - Fast to build, easy to manipulate

“Host language”



DSLs Are Fun

Language expert



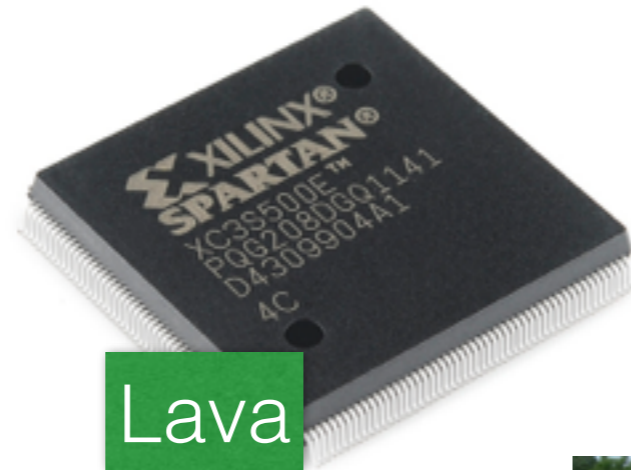
Domain expert

Change the World!

one EDSL at a time



FeldSpar



Lava



Ivory



Atom



Copilot

A jet!

Domain-Specific Languages and Code Synthesis Using Haskell

By Andy Gill

Communications of the ACM, Vol. 57 No. 6, Pages 42-49

10.1145/2605205

[Comments](#)



There are many ways to give instructions to a computer: an electrical engineer might write a MATLAB program; a database administrator might write an SQL script; a hardware engineer might write in Verilog; and an accountant might write a spreadsheet with embedded formulas. Aside from the difference in language used in each of these examples, there is an important difference in *form* and *idiom*. Each uses a language customized to the job at hand, and each builds computational requests in a form both familiar and productive for programmers (although accountants may not think of themselves as programmers). In short, each of these examples uses a Domain-Specific Language (DSL).

[Programming Languages](#)

May 15, 2014

[Volume 12, issue 4](#)

Design Exploration through Code-generating DSLs

High-level DSLs for low-level programming

Bo Joel Svensson, Indiana University

Mary Sheeran, Chalmers University of Technology

Ryan Newton, Indiana University

<https://queue.acm.org/detail.cfm?id=2617811>

<https://queue.acm.org/detail.cfm?id=2626374>

A Calculator

```
data Expr =
  Lit Integer
  | Var String
  | Add Expr Expr
  | Sub Expr Expr
  deriving (Show, Read, Eq)

lit :: Integer -> Expr
lit = Lit

var :: String -> Expr
var = Var

(."+") :: Expr -> Expr -> Expr
a .+ b = Add a b

(.-) :: Expr -> Expr -> Expr
a .- b = Sub a b

expr :: Expr
expr = lit 3 .+ var "x" .- var "y"

-- > expr
-- Sub (Add (Lit 3) (Var "x")) (Var "y")
```



Calculating

```
type Env = Map String Integer
```

```
eval :: Env -> Expr -> Integer
```

```
eval env e = case e of
```

```
  Lit x    -> x
```

```
  Var s    -> lookup s env
```

```
  Add a b  -> eval env a + eval env b
```

```
  Sub a b  -> eval env a - eval env b
```

```
env :: Env
```

```
env = insert "x" 4 (insert "y" 5 empty)
```

```
expr :: Expr
```

```
expr = lit 3 .+ var "x" .- var "y"
```

```
-- > eval env expr
```

```
-- 2
```


Don't-Miss DSL Talks

9:00-10:00 Keynote: Ras Bodik (University of Washington)
Program Synthesis: Opportunities for the next Decade

9:00-10:00 Keynote: Mary Sheeran, Chalmers University of Technology
Functional Programming and Hardware

Wednesday, 2 September

9:00-10:00 Keynote: John Hughes (University of Cambridge)
Tribute to John Hughes

10:00-11:00 Break

14:50 Break

Code Generation

15:10 Guilt Free Ivory
Trevor Elliott, Lee Pike, Simon Winwood, Pat Hickey, James Bielman, Jamey Sharp, Eric Seidel and John Launchbury

Management of Financial Multi-Party Contracts
past, change the future, FRPNow!

Chalmers University of Technology (Sweden); Koen Claessen, Chalmers University of Technology (Sweden)
Bahr, University of Copenhagen (Denmark); Jost Berthold, Commonwealth Bank of Australia (Australia); Martin Elsman, University of Copenhagen (Denmark)

A Fast Compiler for NetKAT

Steffen Smolka, Cornell University (USA); Spiridon Eliopoulos, Inhabited Type LLC (USA); Nate Foster, Cornell University (USA); Arjun Guha, University of Massachusetts Amherst (USA)

Shamless Plug!

FRP (Elm)

```
Mouse.clicks : Signal ()
```

```
Signal.map    : (a -> b)          -> Signal a -> Signal b
```

```
Signal.foldp : (a -> b -> b) -> b -> Signal a -> Signal b
```

```
countClick : Signal Int
```

```
countClick =
```

```
  Signal.foldp (\clk count -> count + 1) 0 Mouse.clicks
```

```
main : Signal Element
```

```
main =
```

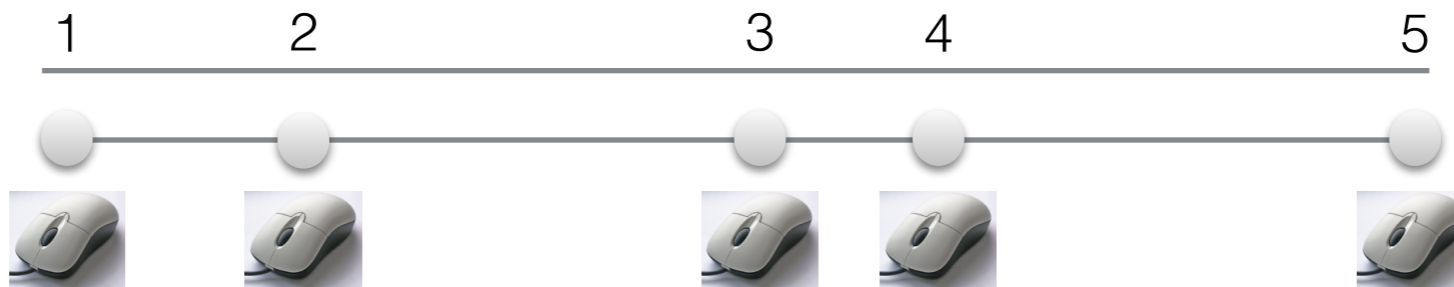
```
  let go count =
```

```
    if count > 10
```

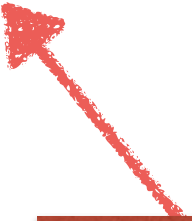
```
      then show count
```

```
      else show "waiting"
```

```
  in Signal.map go countClick
```



Some Challenges



Feel free
to solve

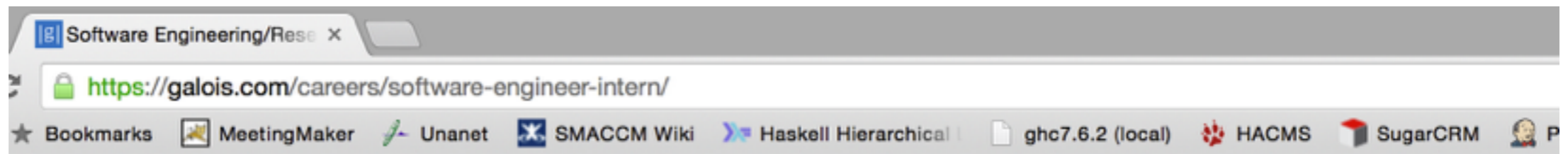
- Sharing and recursion
- Syntax
- Types

Free Advice!

1. Why are you getting a Ph.D?
2. Do you recall the Leslie Lamport's dissertation?
3. If you don't write it down, it never happened.

Shamless Plug, the Sequel

A Serious Lack of Shame



Current Opening

Software Engineering/Research Intern

Galois is currently seeking software engineering and research interns for **Winter/Spring of 2016** at all educational levels. We are committed to matching interns with exciting and engaging engineering work that fits their particular interests, creating lasting value for interns, Galois, and our community. A Galois internship is a chance to tackle cutting-edge, meaningful problems in a uniquely collaborative environment with world-leading researchers.

Important Dates

- Applications due: **October 1st, 2015**
- Internship period (flexible): 12 weeks during **January – April, 2016**

About Galois

Our mission is to create trustworthiness in critical systems. We're in the business of taking blue-sky ideas and turning them into real-world technology solutions. We've been developing real-world systems for over ten years using functional programming, language design, and formal methods.