Pure Functional Programming and it's Benefits (case study Haskell)

Massyl Nait Mouloud, Architect/Developer at Sfeir

12 November 2013

FP Origins

- First high-level FP language Lisp (McCarthy), developed in the late 1950s
- Lisp the seed for functional languages family
- Haskell originally designed in the late 1980s (Hudak, Wadler, 1988), inspired by languages developed by David Turner in the early 1980s.

Named after the logician Haskell B. Curry, with Alonzo Church, established the theoretical foundations of functional programming (Lambda Calculus)

What is Haskell

- Pure functional programming language
- Statically typed, with inference type (Hindley/Milner)
- Non-strict (GHC lazy)
- Fast

Pure FP (case Haskell)

Pros

- Reasoning on your code (rely on the powerful tool that is MATHS/induction)
- Less error prone, almost no bugs
- Relying on property based frameworks to generate tests based on invariants
- Fast prototyping and development
- Easy maintenance beacause concisness and clarity of the code

Cons

- Lazyness is double-edged sword
- It may appear scary at first (another approach/philosophy)

Sample code Java versus Haskell

Even numbers from Aaron Contorer

```
final int LIMIT=50;
int[] a = new int[LIMIT];
int[] b = new int[LIMIT - 5];
for (int i=0;i < LIMIT;i++) {
  a[i] = (i+1)*2;
  if (i >=5) b[i - 5] = a[i];
}
```

```
a = [2,4..100]
b = drop 5 a
```

Conway sequence from me

```
conWay r l = iterate next' [r] !! (l-1)
next' = concatMap (\xs -> [length xs, head xs]) . group
```

Advice to embrace FP paradigm

- Unlearn what you know in OO, Imperative paradigms
- Train yourself and be tenacious

Conclusion

You have to learn another paradigm (mainly FP)