Programming in the Next Dimension

Adrian King
6 April 2017

What Is the Future?

- A lot of the future will look like the past.
  - Sumerian persisted as a ceremonial language for almost 2000 years after it stopped being spoken for everyday purposes.
  - How long will Fortran last?
- So we're mainly interested in the parts of the future that look different from the past.

What Is Programming?

- Some things are clearly not programming.

What Isn't Programming?

- Some things might be programming.

- Is any kind of communication with a machine programming?
  - Setting a thermostat?
  - Shoveling coal into a steam locomotive?
- Is natural language programming?
  - Asking Siri where to get lunch?
  - Asking your coworker to join you for lunch?

Programming Is Formal

- To me, programming differs from natural communication in that it is precise and detailed.
  - Communication in natural languages is often sloppy and elliptical.
- To be a proper program, the validity and behavior of the program must be specified formally.

The Constraints of Formality

- If we accept that programming is a formally defined activity, that greatly constrains what we can consider programming.
- Natural language (or other informal) communication with machines is also very interesting, but not programming in this sense.

Formalisms

- A well-known formalism is the lambda calculus.
- It represents a program as a one-dimensional sequence of symbols.

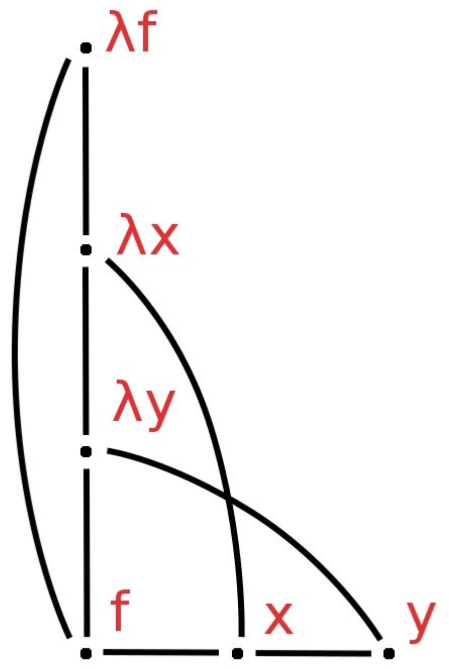
Haskell:

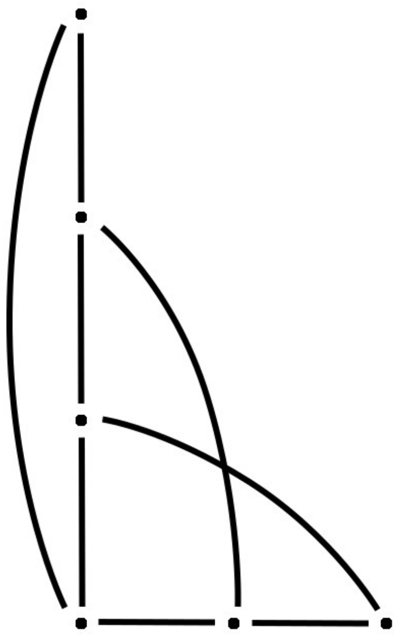
```
flip :: (a -> b -> c) -> b -> a -> c
flip f x y = f y x
```

Untyped lambda calculus:

$\lambda f.\ \lambda x.\ \lambda y.\ f\ y\ x$

Wait!
Count Those Dimensions Again





- We can adopt 2-dimensional conventions for depicting a lambda calculus term. Maybe a one-dimensional formalism isn't so one-dimensional after all.

Writing vs. Geometry

- Geometric shapes can be an alternative to text for representing relationships in programs.
- But written language has been around for 5000 years, and has survived many predictions of its demise.

Fortress: Another Foray into the 2nd Dimension

- Sun's experimental language Fortress was an attempt to support conventional (2-dimensional) mathematical notation, such as:

$$\sum_{k=1}^{n} a_k x^k$$

- Fortress also supported a more conventional, linear textual format:

```
SUM[k←1:n] a[k] x^k
```

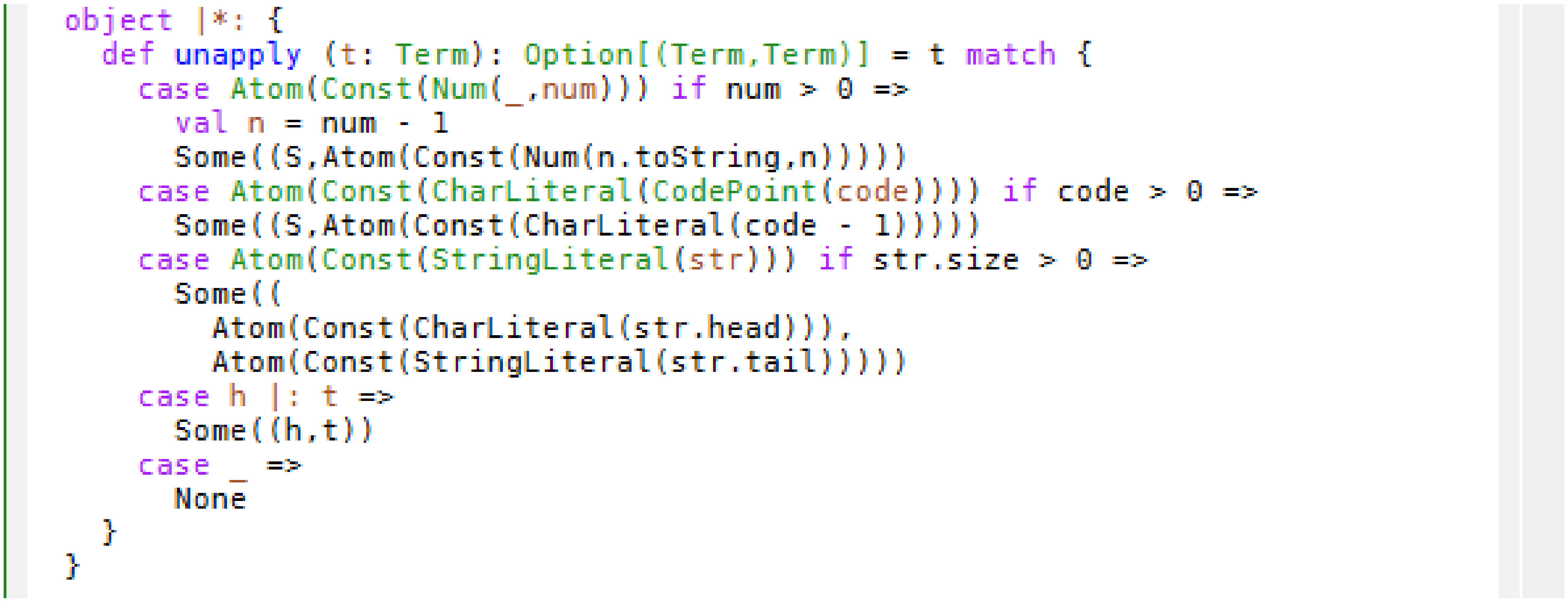
- Some users thought this format was more convenient. Existing text manipulation tools impart a lot of inertia.

But We Already Live in Flatland

- Most of us already think our 1-dimensional programs are 2-dimensional. After all, we can see them on our 2-dimensional screens!

Phatter Than Flat

- In fact, most text editors display code in *three* dimensions.
- Ooh, colors!



```
object |*: {
  def unapply (t: Term): Option[(Term,Term)] = t match {
    case Atom(Const(Num(_,num))) if num > 0 =>
      val n = num - 1
      Some((S,Atom(Const(Num(n.toString,n)))))
    case Atom(Const(CharLiteral(CodePoint(code)))) if code > 0 =>
      Some((S,Atom(Const(CharLiteral(code - 1)))))
    case Atom(Const(StringLiteral(str))) if str.size > 0 =>
      Some((
        Atom(Const(CharLiteral(str.head))),
        Atom(Const(StringLiteral(str.tail)))))
    case h |: t =>
      Some((h,t))
    case _ =>
      None
  }
}
```

Prediction Time

- My guess: programming languages won't change much, at least internally.
- But our tools may offer us views with more dimensions when we work with code.
- Volumes, sounds, more? (Please don't say smells.)