Thank you

INVITING EVERYONE TO THE PARTY

JUNE 2017

Disclaimer:

CONTEXT

is really important for most of the assertions here. So please stay in context.

IRISH-ISMS AHEAD

Craic, eejit, etc are totally ok words

THIS IS BACON

DELPHI

VB.NET

C# (SOME JS WHEN JQUERY WAS NEW, SOME JAVA)

F# / C# <- STARTED LEARNING FP WOO!

SCALA

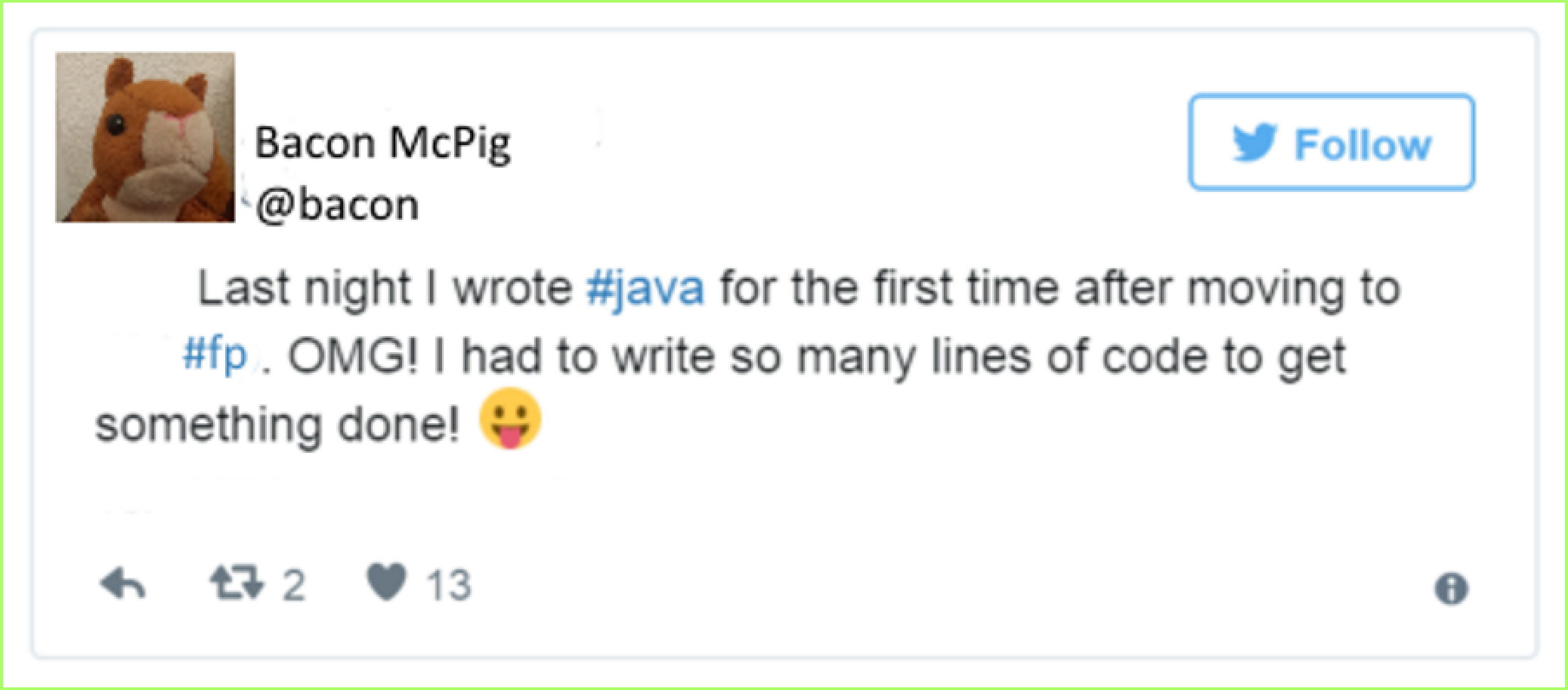
REFERENTIAL TRANSPARENCY

LESS MUTABLE STATE!!

NO EXCEPTIONS FOR FLOW CONTROL!!

.. AND MORE

BACON'S FRIENDS FELT AWKWARD



Last night I wrote #java for the first time after moving to #fp . OMG! I had to write so many lines of code to get something done! 😛

BACON'S FP

- **Typed** FP
- FP everywhere
- Aspiring to purity / Total functions

PROBLEMS WITH FP AS BACON UNDERSTANDS IT

- Dependency management
- Type tetris
- Complicated concepts

... is it worth it?... is it the best way?

BACON DREAMS OF WELL STRUCTURED PROGRAMS

Well-structured software is easy to write and to debug, and provides a collection of modules that can be reused to reduce future programming costs. [Why FP matters. John Hughes]

MEET OOOOO

- Works with Bacon
- Performance is everything
- Curious about functional approach

"FUNCTIONAL PROGRAMMING HAS EMERGED SINCE THE MID-2000S AS AN ATTRACTIVE BASIS FOR SOFTWARE CONSTRUCTION. ONE REASON IS THE INCREASING IMPORTANCE OF PARALLELISM AND DISTRIBUTION IN COMPUTING." ODERSKY, ROMPF APRIL 2014

"...ESPECIALLY ITS (SCALA) FOCUS ON PRAGMATIC CHOICES THAT UNIFY TRADITIONALLY DISPARATE PROGRAMMING-LANGUAGE PHILOSOPHIES (SUCH AS OBJECT-ORIENTED AND FUNCTIONAL PROGRAMMING)

ODERSKY, ROMPF APRIL 2014

SOLID LOOKS A LOT LIKE FP WHEN YOU SQUINT

ON SCALA AND F#/C#

(From it's creators)

Scala is very much about better component oriented programming for the Java platform. Although we do a good job of object oriented programming which is very nice in F#, we haven't thought to make fundamental improvements at the component level, in a sense. We are quite happy to say "You are making components? OK, make it a .NET component". *Don Syme - March 2009*

"...[Scala] focus on pragmatic choices that unify traditionally disparate programming-language philosophies (such as object-oriented and functional programming). The key lesson is these philosophies need not be contradictory in practice.

[Odersky, Rompf - April 2014]

Regarding functional and object-oriented programming, one fundamental choice is where to define pieces of functionality (...) ...and Scala gives programmers the choice.

Choice also involves responsibility, and in many cases novice Scala programmers need guidance to develop an intuitive sense of how to structure programs effectively.

[Odersky, Rompf, April 2014]

When Oooo and Bacon talk, they often disagree and call each other names

DOING + THINKING

WE BUILD SYSTEMS WITH:

LANGUAGE(S)

TOOLS: LIBRARIES, FRAMEWORKS

CONTEXT: USERS AND COMMUNITY

CONTEXT MATTERS

- Paradigms and how they interact
- Paradigms and how they shift

A PROGRAMMING PARADIGM

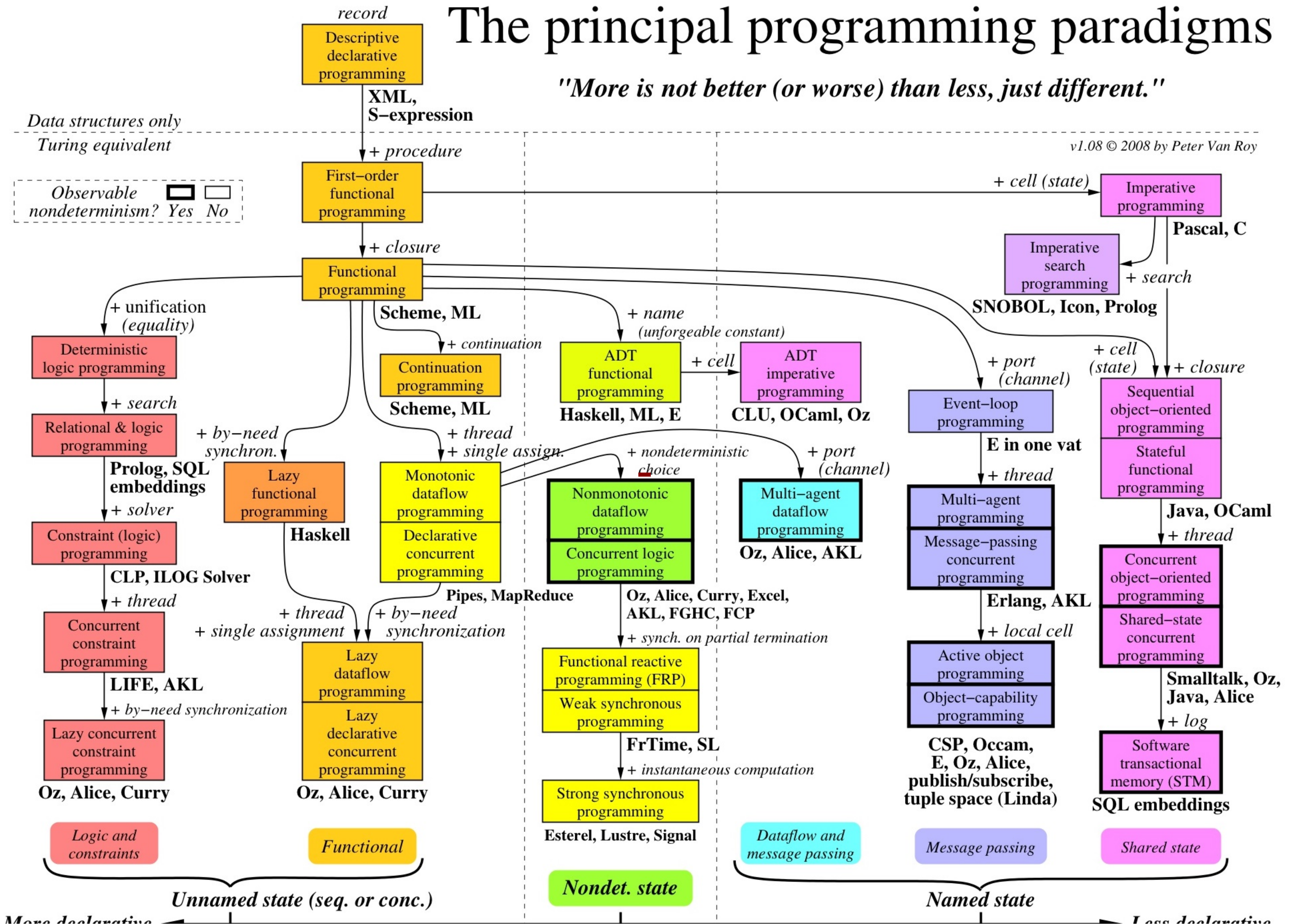
...is an approach to programming a computer based on a mathematical theory or a coherent set of principles.

[Programming Paradigms for Dummies: What Every Programmer Should Know - Peter Van Roy]

The principal programming paradigms

"More is not better (or worse) than less, just different."

v1.08 © 2008 by Peter Van Roy


record
Descriptive declarative programming
XML, S−expression
Data structures only
Turing equivalent
+ procedure
Observable nondeterminism? Yes No
First−order functional programming
+ cell (state)
Imperative programming
Pascal, C
Imperative search programming
+ search
SNOBOL, Icon, Prolog
+ closure
Functional programming
Scheme, ML
+ unification (equality)
Deterministic logic programming
+ search
Relational & logic programming
Prolog, SQL embeddings
+ solver
Constraint (logic) programming
CLP, ILOG Solver
+ thread
Concurrent constraint programming
LIFE, AKL
+ by−need synchronization
Lazy concurrent constraint programming
Oz, Alice, Curry
+ continuation
Continuation programming
Scheme, ML
+ by−need synchron.
Lazy functional programming
Haskell
+ thread
+ single assign.
Monotonic dataflow programming
Declarative concurrent programming
Pipes, MapReduce
+ thread
+ single assignment
+ by−need synchronization
Lazy dataflow programming
Lazy declarative concurrent programming
Oz, Alice, Curry
+ name (unforgeable constant)
ADT functional programming
Haskell, ML, E
+ cell
ADT imperative programming
CLU, OCaml, Oz
+ nondeterministic choice
Nonmonotonic dataflow programming
Concurrent logic programming
Oz, Alice, Curry, Excel, AKL, FGHC, FCP
+ synch. on partial termination
Functional reactive programming (FRP)
Weak synchronous programming
FrTime, SL
+ instantaneous computation
Strong synchronous programming
Esterel, Lustre, Signal
+ port (channel)
Multi−agent dataflow programming
Oz, Alice, AKL
+ port (channel)
Event−loop programming
E in one vat
+ thread
Multi−agent programming
Message−passing concurrent programming
Erlang, AKL
+ local cell
Active object programming
Object−capability programming
CSP, Occam, E, Oz, Alice, publish/subscribe, tuple space (Linda)
+ cell (state)
+ closure
Sequential object−oriented programming
Stateful functional programming
Java, OCaml
+ thread
Concurrent object−oriented programming
Shared−state concurrent programming
Smalltalk, Oz, Java, Alice
+ log
Software transactional memory (STM)
SQL embeddings
Logic and constraints
Functional
Dataflow and message passing
Message passing
Shared state
Unnamed state (seq. or conc.)
Nondet. state
Named state
More declarative
Less declarative

All but the smallest toy problems require different sets of concepts for different parts. This is why programming languages should support many paradigms.

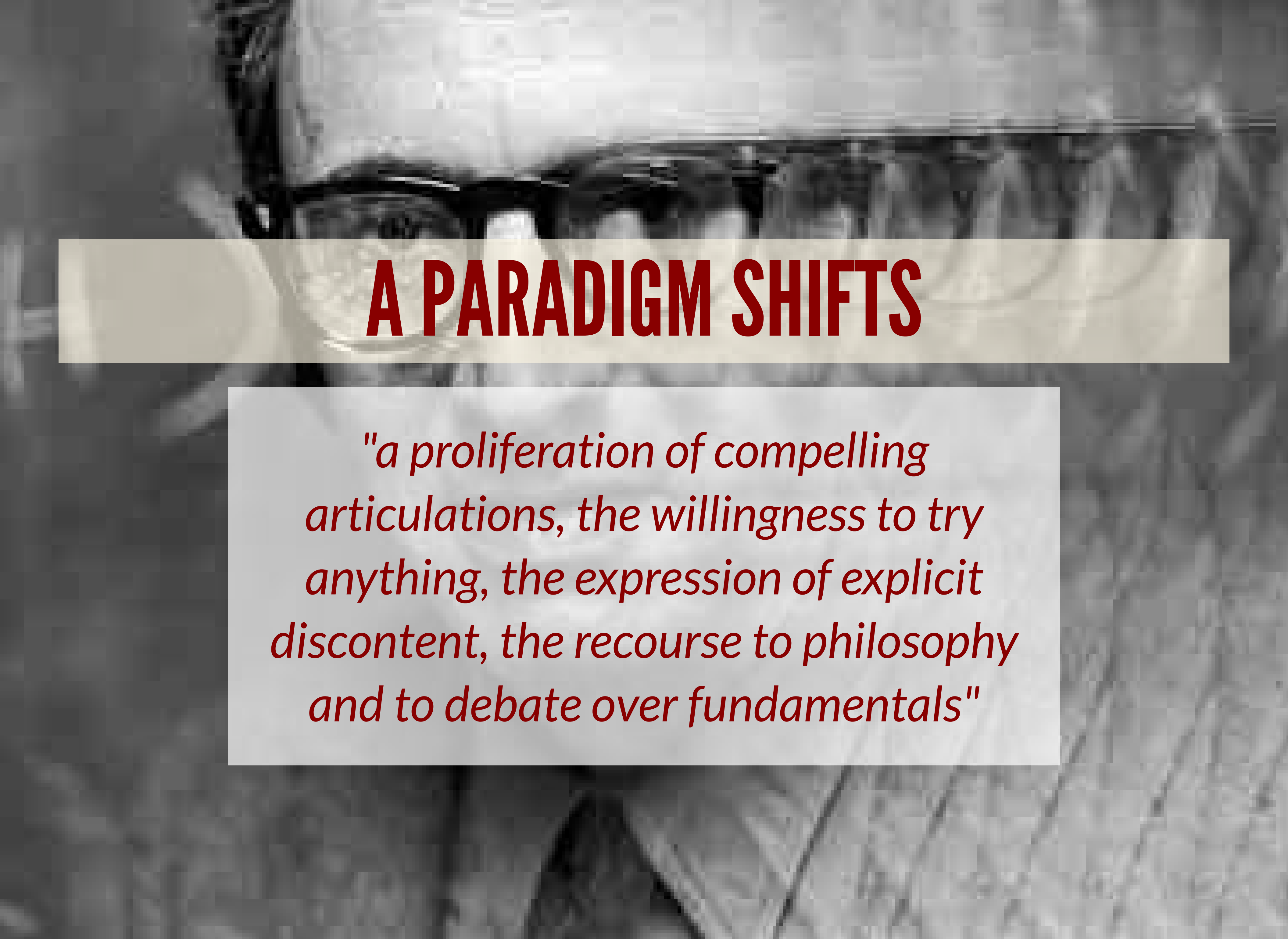
[Programming Paradigms for Dummies: What Every Programmer Should Know - Peter Van Roy]

A language should ideally support many concepts in a well-factored way, so that the programmer can choose the right concepts whenever they are needed without being encumbered by the others.

[Programming Paradigms for Dummies: What Every Programmer Should Know - Peter Van Roy]

...it is certainly not true that there is one "best" paradigm

[Programming Paradigms for Dummies: What Every Programmer Should Know - Peter Van Roy]



A PARADIGM SHIFTS

"a proliferation of compelling articulations, the willingness to try anything, the expression of explicit discontent, the recourse to philosophy and to debate over fundamentals"

Many langauges adding features generally associated with functional programming:

- lambdas
- functional data structures
- pattern matching, etc

C++, Java, C#

The decision to reject one paradigm is always simultaneously the decision to accept another, and the judgment leading to that decision involves the comparison of both paradigms with nature and with each other.

Kuhn, Thomas S.. The Structure of Scientific Revolutions: 50th Anniversary Edition (p. 78). University of Chicago Press. Kindle Edition.

It is, I think, particularly in periods of acknowledged crisis that scientists have turned to philosophical analysis as a device for unlocking the riddles of their field. Scientists have not generally needed or wanted to be philosophers.

Kuhn, Thomas S.. The Structure of Scientific Revolutions: 50th Anniversary Edition (p. 88). University of Chicago Press. Kindle Edition.

"... two scientific schools disagree about what is a problem and what a solution, they will inevitably talk through each other when debating the relative merits of their respective paradigms."

Kuhn, Thomas S.. The Structure of Scientific Revolutions: 50th Anniversary Edition (p. 109). University of Chicago Press. Kindle Edition.

"He argued that competing paradigms are "incommensurable": that is to say, there exists no objective way of assessing their relative merits."

Kuhn, Thomas S.. The Structure of Scientific Revolutions: 50th Anniversary Edition (p. 109). University of Chicago Press. Kindle Edition.

ALL THIS HAS HAPPENED BEFORE AND IT WILL HAPPEN AGAIN

AS PROGRAMMERS, WE

- cut corners
- have religious wars
- deal with terrible code
- deal with other people's terrible code (the wurst!)
- complain about the shortcommings of the current language we are using

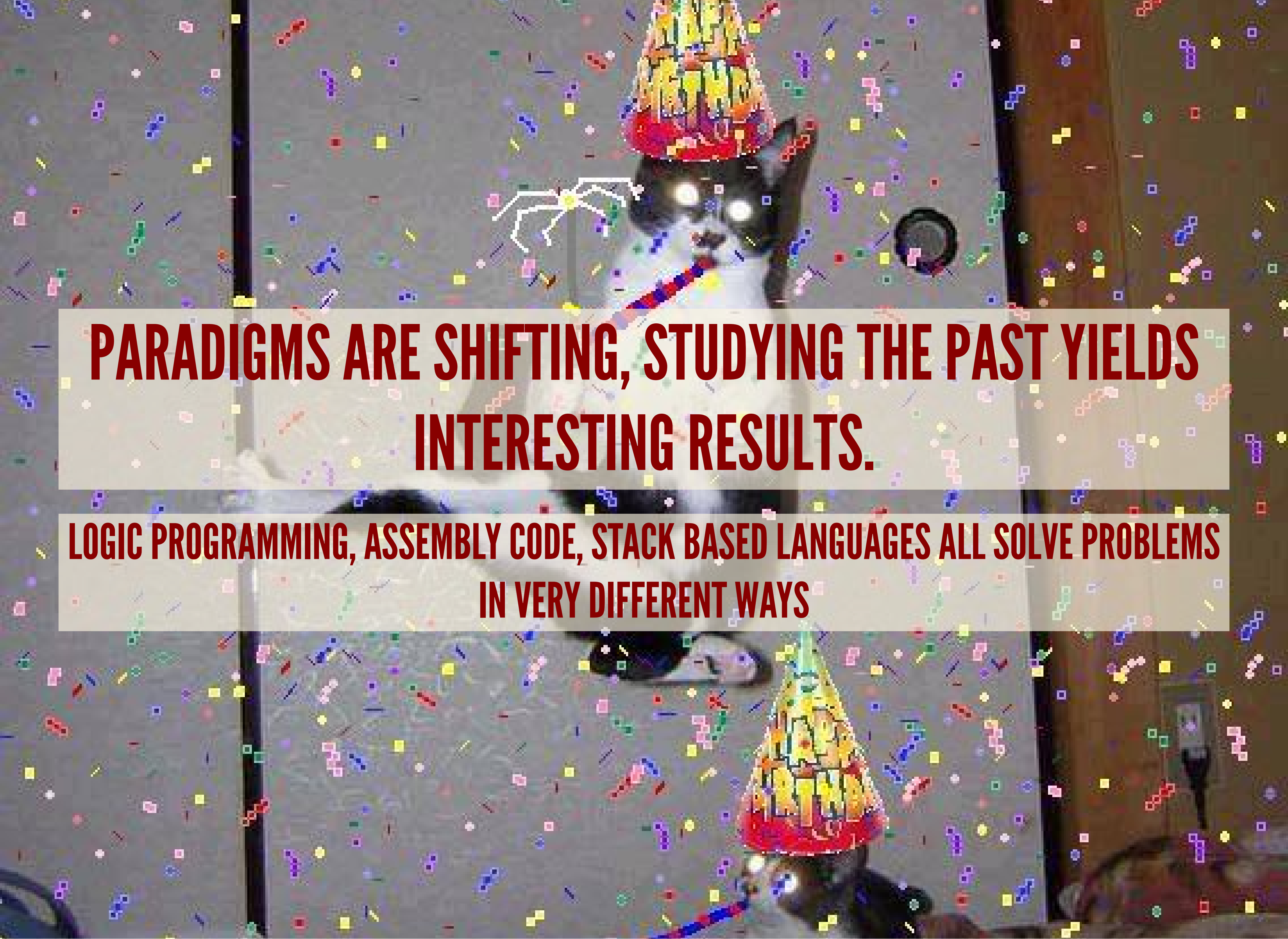
1978 Turing Award lecture by Floyd

To the designer of programming languages, I say: unless you can support the paradigms I use when I program, or at least support my extending your language into one that does support my programming methods, I don't need your shiny new languages; like an old car or house, the old language has limitations that I have learned to live with

To the teacher of programming, even more, I say: identify the paradigms you use, as fully as you can, then teach them explicitly. They will serve your students when Fortran has replaced Latin and Sanskrit as the archetypal dead language.

to the serious programmer: spend a part of your working day examining and refining your own methods. Even though programmers are always struggling to meet some future or past dead-line, methodological abstraction is a wise long term investment.

PEOPLE ARE PART OF THE CONTEXT, MAKE THEM PART OF YOUR CONTEXT



PARADIGMS ARE SHIFTING, STUDYING THE PAST YIELDS INTERESTING RESULTS.

LOGIC PROGRAMMING, ASSEMBLY CODE, STACK BASED LANGUAGES ALL SOLVE PROBLEMS IN VERY DIFFERENT WAYS

THE PARTY IS PROGRAMMING ....EVERYONE IS INVITED.

THANK YOU

ANDREA MAGNORSKY

@SILVERSPOON

SOURCES | REFERENCES

PAPERS

- Programming Paradigms for Dummies: What Every Programmer Should Know - Peter Van Roy
- The paradigms of programming
- The next 700 programming languages by peter landin
- Why Functional Programming Matters by John Hughes
- Joe Armstrong Thesis

ARTICLES, POSTS, VIDEOS

- A punchcard ate my programme by Walt Mankowski
- Clojure spec
- Lenses in F#
- F# Don Syme
- Programming paradigm
- The expression problem

IMAGES

- Animal party link
- Tea ceremony japan link
- Cats with hats link