The Evolutionary Ecology of Technology: The Case of Programming Languages

Silvia Crafa

Universita’ di Padova

…numerical computing

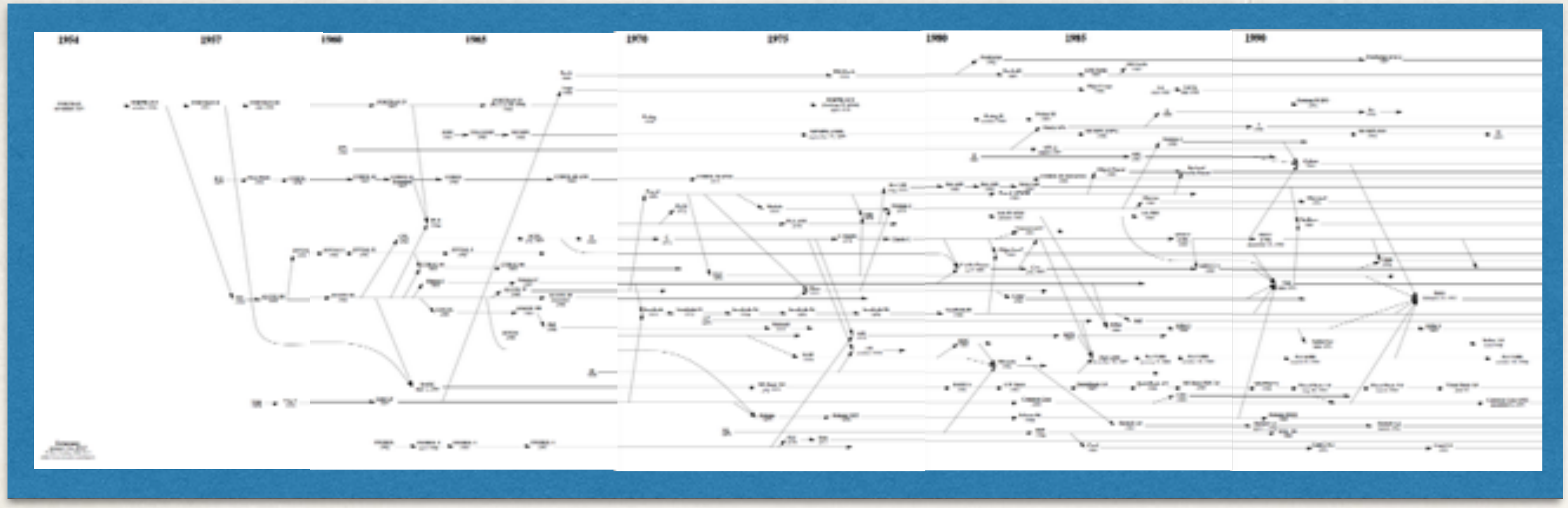
The programming languages timeline

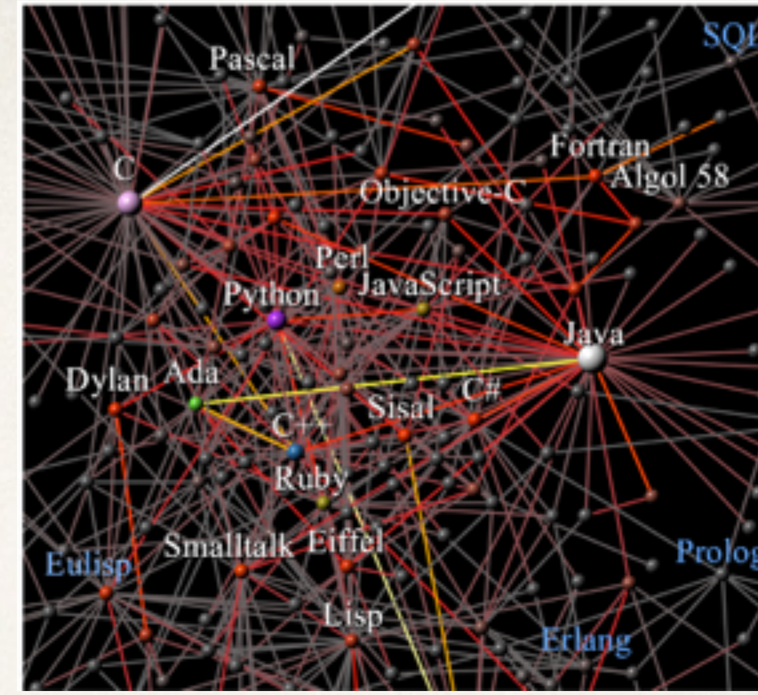
try to grasp the **evolutionary process** that
guided / unfolded behind
the fortune of mainstream PLs

the quest for "good"
programming abstractions

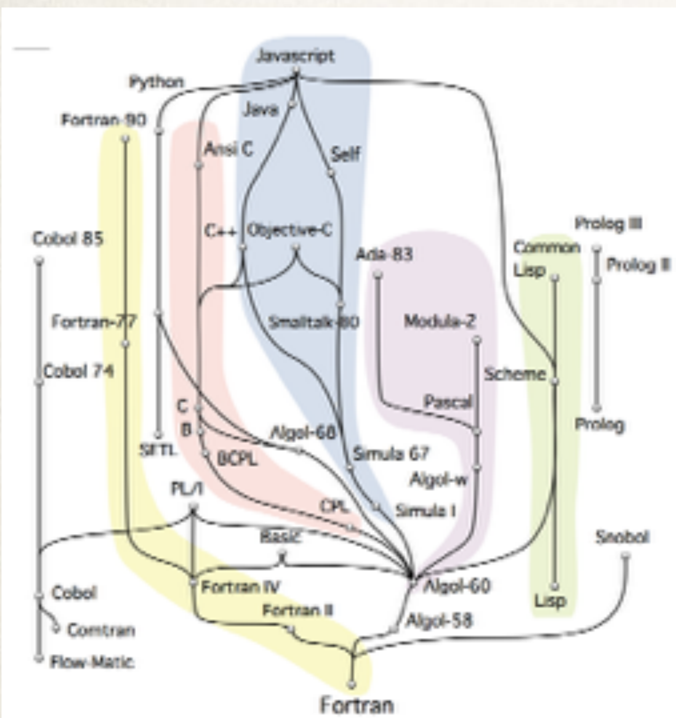
Time

- **When** a language has been **invented** VS when became **popular**?
- **Why** has been invented VS why became popular?



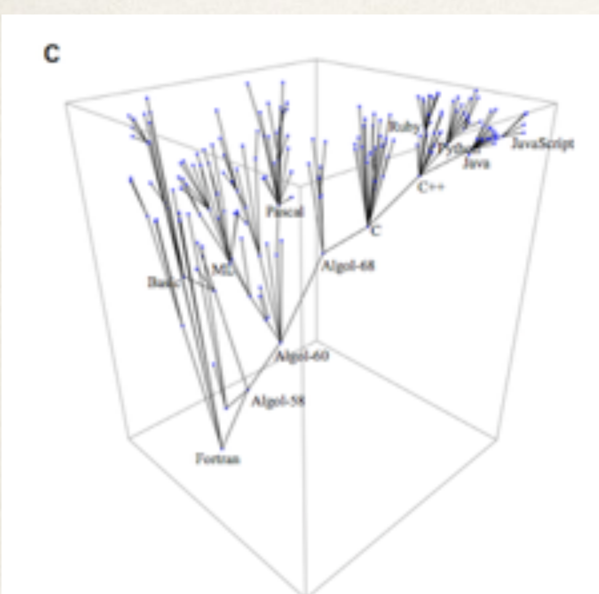


SQL
Pascal
C
Fortran
Algol 58
Objective-C
Perl
JavaScript
Python
Java
Dylan
Ada
C#
Sisal
C++
Ruby
Eulisp
Smalltalk
Eiffel
Prolog
Lisp
Erlang

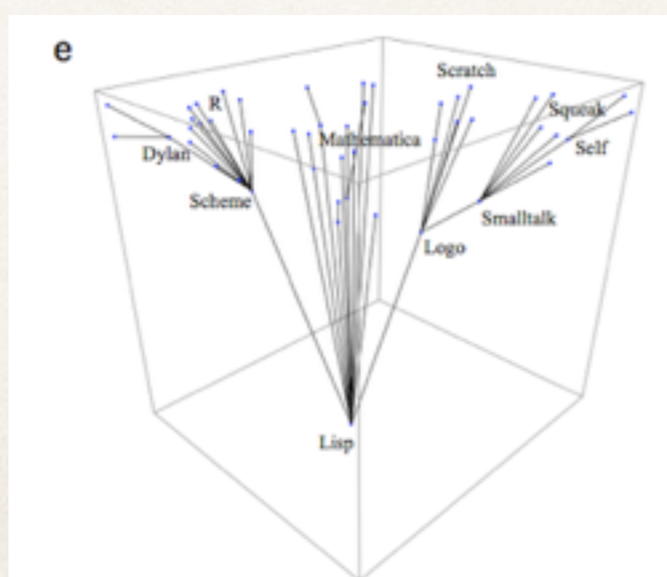


Javascript
Python
Fortran 90
Java
Ansi C
Self
Cobol 85
C++
Objective-C
Ada 83
Common Lisp
Prolog III
Prolog II
Fortran 77
Smalltalk-80
Modula-2
Cobol 74
Scheme
Pascal
C
B
SFTL
BCPL
Algol-68
Simula 67
Algol-w
Prolog
PL/I
CPL
Simula I
Basic
Snobol
Cobol
Fortran IV
Algol-60
Lisp
Comtran
Fortran II
Algol-58
Flow-Matic
Fortran

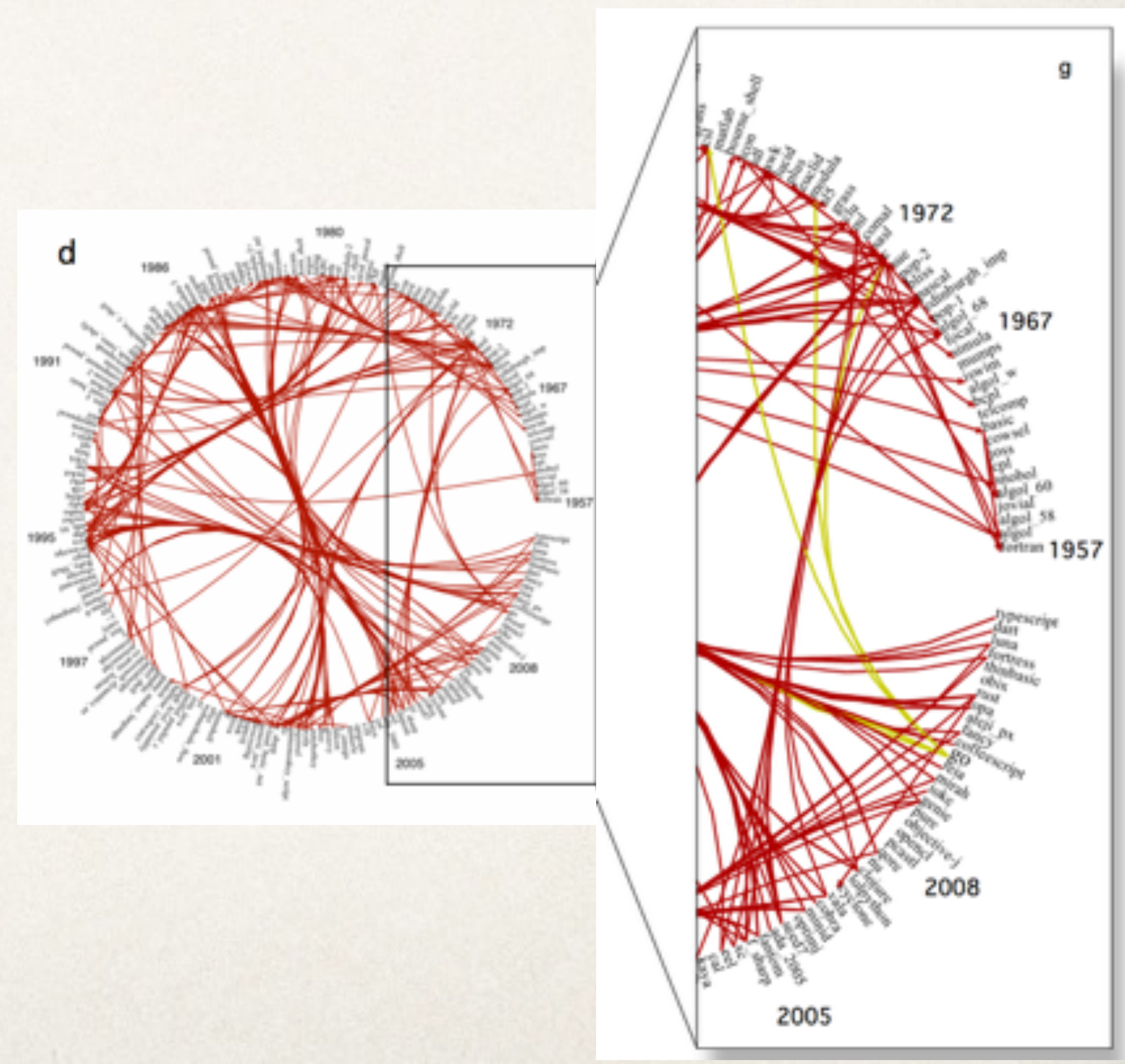
The Graph Of Programming Languages
C
Java
C-Sharp
Perl
Python
Haskell
Smalltalk
Lisp
Ruby



c
Ruby
Python
JavaScript
Java
C++
Pascal
C
Algol-68
ML
Basic
Algol-60
Algol-58
Fortran



e
Scratch
R
Squeak
Dylan
Mathematica
Self
Scheme
Smalltalk
Logo
Lisp



d
1972
1967
1957
2008
2005
g
1972
1967
1957
2008
2005

Human language as a culturally transmitted replicator

Mark Pagel

NATURE REVIEWS | GENETICS
VOLUME 10 | JUNE 2009 | 405

Punctuated equilibrium in the large scale evolution of programming languages

Sergi Valverde*[1,2] and Ricard Solé†[1,3,2]
[1] *ICREA-Complex Systems Lab, Universitat Pompeu Fabra, Dr Aiguader 88, 08003 Barcelona, Spain*
[2] *Institut de Biologia Evolutiva, UPF-CSIC, Psg Barceloneta 37, 08003 Barcelona, Spain*
[3] *Santa Fe Institute, 1399 Hyde Park Road, Santa Fe NM 87501, USA*

SANTA FE INSTITUTE

The evolutionary ecology of technological innovations

Ricard V. Solé*,[1,2,3] Sergi Valverde,[1,2] Marti Rosas Casals,[4,1] Stuart Kauffman,[3] Doyne Farmer,[3] and Niles Eldredge[5]

Complexity 2013

A phylogenetic approach to cultural evolution

Ruth Mace and Clare J. Holden

Department of Anthropology, University College London, Gower Street, London, UK, WC1E 6BT

TRENDS in Ecology and Evolution
Vol.20 No.3 March 2005

The evolutionary ecology of technological innovations

Ricard V. Solé*,[1,2,3] Sergi Valverde,[1,2] Marti Rosas Casals,[4,1] Stuart Kauffman,[3] Doyne Farmer,[3] and Niles Eldredge[5]

Motivations

Is it possible to formulate a theory of technological evolution?

Technological change displays numerous life-like features, suggesting a deep connection with biological evolution. But some differences are also noticeable.

- **descent with variation**
- **selection**
- **convergence**
- **extinction**
- **rapid change and diversification**
- **punctuated pattern**
- **coevolution**
- **macro-evolutionary trends**
- **niche construction**
- **exaptation**
- **…**

- tech innovations are examples of **planned design**: long-term goals, efficiency,
- together with a **clear notion of *progress*** (*measures*)
- **"The Lazarus effect"**
- **…**

The evolutionary ecology of technological innovations

Ricard V. Solé*,[1,2,3] Sergi Valverde,[1,2] Marti Rosas Casals,[4,1] Stuart Kauffman,[3] Doyne Farmer,[3] and Niles Eldredge[5]

Motivations

Is it possible to formulate a theory of technological evolution?

Technological change displays numerous life-like features, suggesting a deep connection with biological evolution. But some differences are also noticeable.

tinkering

a widespread ***reuse*** and ***combination*** of available elements
to build new structures

- **Technology is highly dependent on the combination of preexisting inventions**. Adding new simple elements can completely reset the path of future technologies
- In biology, once established, solutions to problems are seldom replaced.


The evolutionary ecology of technological innovations

Ricard V. Solé*,[1,2,3] Sergi Valverde,[1,2] Marti Rosas Casals,[4,1] Stuart Kauffman,[3] Doyne Farmer,[3] and Niles Eldredge[5]

Motivations

- consider the role played by **social and economic factors**:
    - issues of *compatibility*, but also *market dominance* or *trends*, often make it impossible for better solutions to enter, so that the dominant technology is **stuck to suboptimal solutions** (*JavaScript* and *Web Solutions*)
    - coevolution of **economy** and technology: novel technologies can deeply transform how economy is organised and how new economic regimes emerge (*Internet*, *Cloud*, *BigData*, *CS Education*)

The evolutionary ecology of technological innovations

Ricard V. Solé*,[1,2,3] Sergi Valverde,[1,2] Marti Rosas Casals,[4,1] Stuart Kauffman,[3] Doyne Farmer,[3] and Niles Eldredge[5]

Motivations

The **availability of data** is crucial:

for *information technology* we have the complete fossil record

while in *biology* we have rich data on the history of phyla

- The **phylogeny of technology** is not hierarchical, but rather is more similar to that of bacteria

  reticulate networks, instead of trees, appear to be more appropriate when dealing with cultural dynamics. The reticulated nature is largely due to the rapid and large information exchange, and the introduction of different types of innovations.

- We need **to identify the scales** at which technological hierarchies operate.

  In biology, such hierarchies can be described including different levels, from population dynamics to genotype-phenotypic maps. Information technology, with all its richness and multiplicity of scales, offers our best to achieve this goal.

	Biological Evo	Language Evo	PL Evo
<u>Discrete</u> <u>heritable</u> units:	nucleotides, aminoacids, genes	words, phonemes, syntax	primitives, phrases, modules, *styles*
Mode of inheritance:	parent off-spring, rare clonal	parents, groups, prestige bias (cultural traits)	teaching, companies, backward compatibility, prestige or trend bias
Mutation:	genetic alteration	new words, mistakes, sound changes, innovation	specification update, new version *e.g. Python 3.3.3, Python 3.4.0*
Selection:	natural selection	social selection and trends	market, social selection, trends (everything on web) *stuck to suboptimal solutions*

	Biological Evo	Language Evo	PL Evo
Hybridisation:	species mixes	language Creoles	??
Horizontal transfer:	horizontal gene tranfer	borrowing	??
Fossils:	fragmented fossil records	ancient texts	??
Extinction:	species (mass) extinction	language death	??

What is a species?

What is a Programming Language?

A formal constructed language:

formally defined **syntax**

Needed by the parser!

semantics explaining the meaning of language phrases

The PL boundaries are precisely (and finitely) defined by the Language Specification

No hybrids!

a code mixing Java and C++ constructs will not compile…unless we define a new language, i.e. a new species

We know what a species is!

differently form biology and human languages

Programming Paradigms

- A **programming paradigm** is a fundamental **style** of computer programming, it characterises the structure of programs

  imperative, functional, object-oriented, declarative, logic, …

- PLs are designed to support ***one*** or ***many*** paradigms; they are usually **classified in terms of paradigms**

if a **PL** is a ***species***, a **paradigm** is a group/family/***class***

new paradigms emerge (*speciation*),
compete (*selection*) and
often merge (***hybridise***)

multilevel evolution
and multilevel selection

	Biological Evo	Language Evo	PL Evo
Hybridisation:	species mixes	language Creoles	no hybridisation hybrid code does not run
Horizontal transfer:	horizontal gene tranfer	borrowing	lateral influence but no hybrid
Fossils:	fragmented fossil records	ancient texts	abandoned languages deprecated features PL for old hardware
Extinction:	species (mass) extinction	language death	language death for high level PL “no” mass extinction *Cobol survives, what about* *Objective-C after Swift?*

What is a species?

Coevolution:

PLs co-evolve with hardware
(e.g. multicores, GPUs, Cloud, IoT)
and with programmers (PL theory)

Macro-Evolutionary Trend:

PLs increase their abstraction level.

focus on "***what to do***" rather than on "***how to do it***"

This is due to more efficient hardware, which supports stratifications of virtual machines, and enhanced theory

Niche Construction:

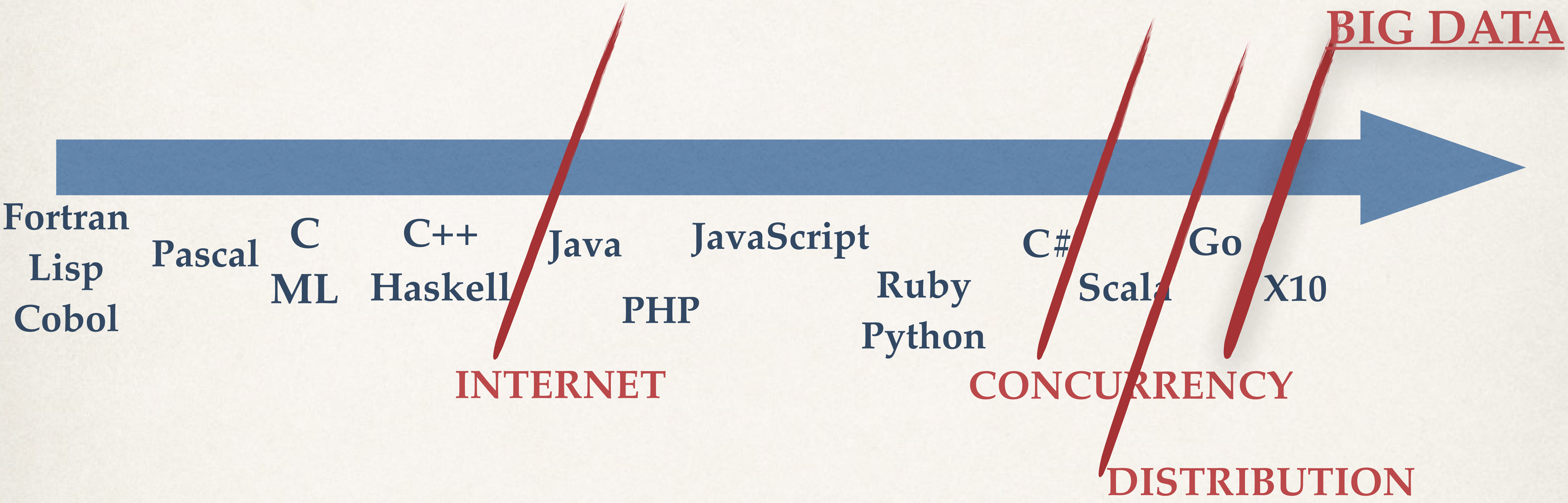
Web PL ecosystem

- page content: HTML5
- page apperance: CSS
- Client side: JavaScript
- Server side: Php, CGI
- data: XML

Exaptation:

after 50 years, functional abstractions appear to work well in concurrent programming

The programming languages timeline



Changes need a catalyst .....linearize evolutionary leaps!

- multicore —> concurrent programming
- cloud computing —> distributed programming
- big data applications —> High Performance Computing

The Quest for good Abstractions

Easy to think
Easy to reason about

Expressiveness
Performance

different abstraction levels!

- Big Data Application Framework
  - Map - Reduce Model
  - Bulk Synchronous Parallel Model
- Message Passing Model
- Shared Memory
- GPU Concurrency Model

which abstractions <u>interoperate</u> productively?

Moving towards conclusions

- **Modern Mainstream Programming Languages:**
  - become more declarative/high-level, moving stuff into the runtime
  - productively mix paradigms
  - heterogeneous concurrency models (Distribution)

- **What is the right level of abstraction?**
  - What are good abstractions? Expressive, flexible, easy to reason about, easy to implement in a scalable/resilient way

- **What about theory?**

The role of PL theory

- Formal languages are **well suited to test** new abstractions and **new mix** of abstractions in a concise and expressive model. i.e. they allow for ***experimentation in a controlled environment.***
  - Asynchrony, locality, scope extrusion, futures, mobility, security, timing, probability, ecc., have been studied both in isolation and in combination

- To develop **formal (and *mechanisable*) techniques to reason** about software systems

- When working in a formal framework it is easier to **distinguish the different abstraction levels involved**: study them separately and then integrating them

Conclusions

- cloud computing, reactive programming, BigData bring about **new shuffle** of **old issues** and **new problems** (scalability, heterogeneity, fault tolerance, security, privacy, efficiency)

- this scenario will act **as the environment operating a selection** over the features of actual PLs.

- hence **"language mutations"** will appear **to adapt** to these new requirements, and to **co-evolve with hardware** evolution.

PLs struggle for life in the language arena.

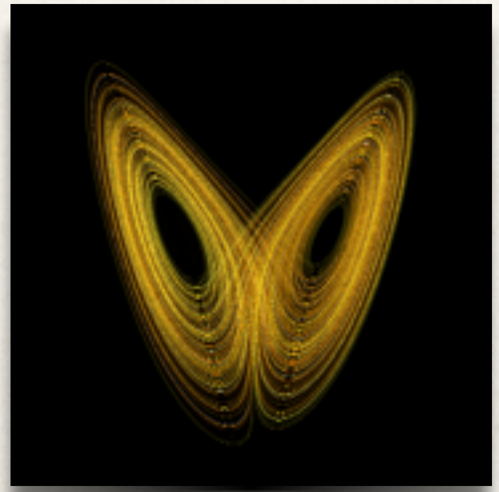
- Will only survive those equipped with **higher plasticity**, either in their design choices or in their marketing strategies?

…what's Java8 if not a form of ***adaptation***?

About Numerical Computing

Consider a mathematical problem to be solved numerically

- The Lorenz system: a system of 3 ordinary differential equations nonlinear, three-dimensional and deterministic



notable for having chaotic solutions for certain parameter values and initial conditions

simplified math model for atmospheric convection, also in models for lasers, electric circuits, chemical reactions,…

About Numerical Computing

Maths

1. Specify the mathematical problem

CS (Matlab, Mathematica DSL, Python,…)

2. Write a software capable of solving the numerical problem
3. Run the numerical software to find solution

CS

4. Plot results into a graphic

Maths

Specify the math problem…

Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$ be the convection intensity, the maximum temperature difference and the stratification change respectively. The system equations are:

$$\dot{y_1}(t) = py_2(t) - py_1(t)$$

$$\dot{y_2}(t) = ry_1(t) - y_2(t) - y_1(t)y_3(t)$$

$$\dot{y_3}(t) = y_1(t)y_2(t) - by_3(t)$$

The discretization in time is made with the `Implicit Euler}` method. Simulation duration is set being $t_0 = 0, T = 2$ with $dt = 0.005$s.
The system initial conditions are the following: $y_1(0) = 1$, $y_2(0) = 2$ and $y_3(0) = 3$.
Results are shown in Figure

…in the Maths language

Specify ODE coefficients, ODEs, the time interval, the discretisation method in time, initial conditions

CS

```
Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$
be the convection intensity, the maximum temperature difference and
the stratification change respectively. The system equations are:
$$\dot{y_1}(t) = p y_2(t) - p y_1(t)$$
$$\dot{y_2}(t) = r y_1(t) - y_2(t) - y_1(t) y_3(t)$$
$$\dot{y_3}(t) = y_1(t) y_2(t) - b y_3(t)$$
The discretization in time is made with the
                    Implicit Euler} method.
Simulation duration is set being $t_0=0$,$T = 2$ with $dt = 0.005$s.

The system initial conditions are the following:
$y_1(0)=1$, $y_2(0)=2$ and $y_3(0)=3$.

Results are shown in Figure
```

…in LaTeX !

IDEA:
translate it into another PL so to **use it as input** of the numerical **solution software**

Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$ be the convection intensity, the maximum temperature difference and the stratification change respectively. The system equations are:

$$\dot{y_1}(t) = py_2(t) - py_1(t)$$
$$\dot{y_2}(t) = ry_1(t) - y_2(t) - y_1(t)y_3(t)$$
$$\dot{y_3}(t) = y_1(t)y_2(t) - by_3(t)$$

The discretization in time is made with the \setTimeDiscrMethod{Implicit Euler} method. Simulation duration is set being $t_0 = 0, T = 2$ with $dt = 0.005$s.
The system initial conditions are the following: $y_1(0) = 1$, $y_2(0) = 2$ and $y_3(0) = 3$.

problem specification

```
\begin{computable_expression}
  Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$
  be the convection intensity, the maximum temperature difference and
  the stratification change respectively. The system equations are:
  $$\dot{y_1}(t) = p y_2(t) - p y_1(t)$$
  $$\dot{y_2}(t) = r y_1(t) - y_2(t) - y_1(t) y_3(t)$$
  $$\dot{y_3}(t) = y_1(t) y_2(t) - b y_3(t)$$
  The discretization in time is made with the
  \setTimeDiscrMethod{Implicit Euler}
  Simulation duration is set being $t_0=0$,$T = 2$ with $dt = 0.005$s.

  The system initial conditions are the following:
  $y_1(0)=1$, $y_2(0)=2$ and $y_3(0)=3$.

  Results are shown in Figure \createFigure{y_1(t)}{y_2(t);y_3(t):'r--'}.
\end{computable_expression}
```

CFL tool

- parse a LaTeX text
- recognize a math problem
- generate a Python script that computes the solution

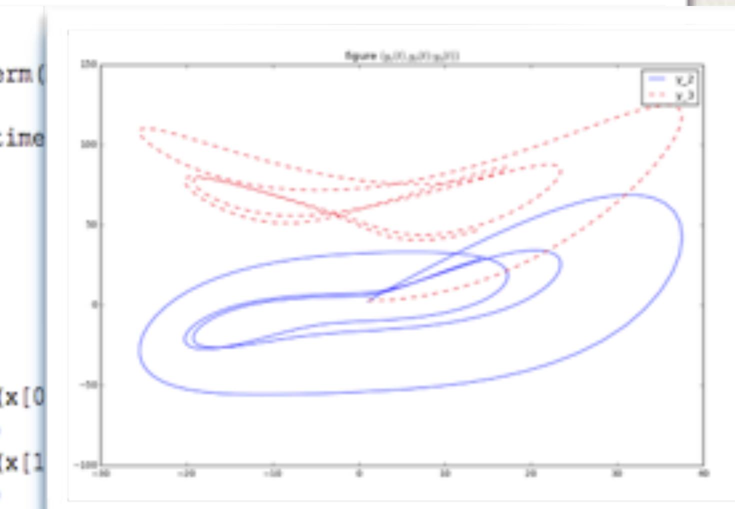
executable Python script

```
# declaration of known variables:
p = 10
b = 8/3
r = 64
# defintion of the time domain of the problem:
T = 2
dt = 0.005
t_0 = 0
# initialization of the problem unknown variables:
y_1 = OdeSolution(Function('1'))
y_2 = OdeSolution(Function('2'))
y_3 = OdeSolution(Function('3'))
y_1_hist1 = [0 for i in np.arange(t_0,T,dt)]
y_2_hist1 = [0 for i in np.arange(t_0,T,dt)]
y_3_hist1 = [0 for i in np.arange(t_0,T,dt)]
# setting of initial conditions:
x = [float(ic) for ic in OdeSystem.current.initcond]
x_cfl_past = np.copy(x)
dtmin = dt
it = 0
# solution of the initial-value problem:
for t in np.arange(t_0,T,dtmin):
    1*tder(y_1) == p*y_2 - nnl_term(p*nonLinear(y_1))
    1*tder(y_2) == r*y_1 - 1*y_2 - nnl_term(nonLinear(y_1)*nonLinear(y_3))
```

```
    1*tder(y_3) == y_1*y_2 - nnl_term(
    pr = Problem.current[0]
    pr.method.set_tDiscr('IEuler',time
    x = pr.method.solve(t)
    Problem.current = []

    y_1_hist1[it] = x[0]
    y_2_hist1[it] = x[1]
    y_3_hist1[it] = x[2]

    y_1 = OdeSolution(Function(str(x[0
    y_1.SetPastValue(x_cfl_past[0])
    y_2 = OdeSolution(Function(str(x[1
    y_2.SetPastValue(x_cfl_past[1])
    y_3 = OdeSolution(Function(str(x[2])))
    y_3.SetPastValue(x_cfl_past[2])
```



problem's resolution pattern

```
p.plot(y_1_hist1,y_3_hist1,'r--',label='y_3');
p.legend();
p.savefig('ce0test1m1fig1.png',bbox_inches=0,dpi=100);
```

Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$ be the convection intensity, the maximum temperature difference and the stratification change respectively. The system equations are:

$$\dot{y_1}(t) = p y_2(t) - p y_1(t)$$
$$\dot{y_2}(t) = r y_1(t) - y_2(t) - y_1(t) y_3(t)$$
$$\dot{y_3}(t) = y_1(t) y_2(t) - b y_3(t)$$

The discretization in time is made with the Implicit Euler} method. Simulation duration is set being $t_0 = 0, T = 2$ with $dt = 0.005$s.
The system initial conditions are the following: $y_1(0) = 1$, $y_2(0) = 2$ and $y_3(0) = 3$.
Results are shown in Figure

Maths

CFL: *computing from LaTeX*

a **numerical problem-solving environment** that converts the **specification** of a mathematical problem into an appropriate **resolution pattern** that can be directly executed

LaTeX

```
\begin{computable_expression}
  Let $p = 10$, $r = 64$, $b = 8/3$. Let $y_1(t)$, $y_2(t)$ and $y_3(t)$
  be the convection intensity, the maximum temperature difference and
  the stratification change respectively. The system equations are:
  $$\dot{y_1}(t) = p y_2(t) - p y_1(t)$$
  $$\dot{y_2}(t) = r y_1(t) - y_2(t) - y_1(t) y_3(t)$$
  $$\dot{y_3}(t) = y_1(t) y_2(t) - b y_3(t)$$
  The discretization in time is made with the
  \setTimeDiscrMethod{Implicit Euler} method.
  Simulation duration is set being $t_0=0$,$T = 2$ with $dt = 0.005$s.
```

Python

```
# declaration of known variables:
p = 10
b = 8/3
r = 64
# defintion of the time domain of the problem:
T = 2
dt = 0.005
t_0 = 0
# initialization of the problem unknown variables:
...
dtmin = dt
it = 0
# solution of the initial-value problem:
for t in np.arange(t_0,T,dtmin):
    1*tder(y_1) == p*y_2 - nnl_term(p*nonLinear(y_1))
```

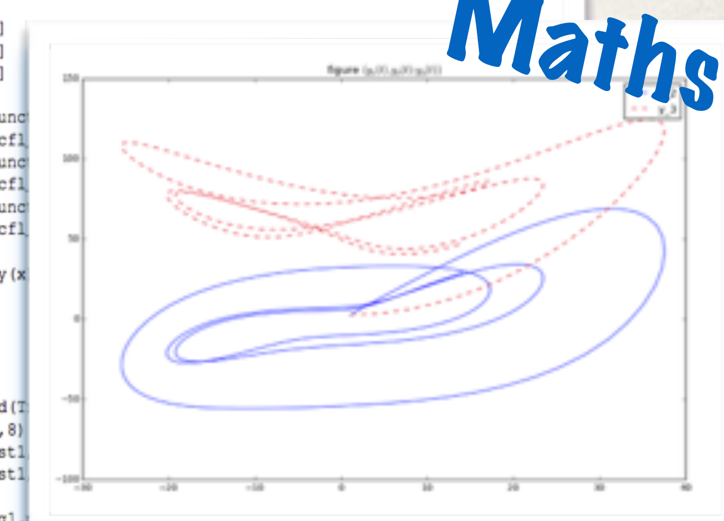
```
    1*tder(y_3) == y_1*y_2 - nnl_term(b*nonLinear(y_3))
    pr = Problem.current[0]
    pr.method.set_tDiscr('IEuler',timeInterval(0.0,dtmin),dtmin)
    x = pr.method.solve(t)
    Problem.current = []

    y_1_hist1[it] = x[0]
    y_2_hist1[it] = x[1]
    y_3_hist1[it] = x[2]

    y_1 = OdeSolution(Func
    y_1.SetPastValue(x_cfl
    y_2 = OdeSolution(Func
    y_2.SetPastValue(x_cfl
    y_3 = OdeSolution(Func
    y_3.SetPastValue(x_cfl

    x_cfl_past = np.copy(x
    it += 1
#endfor
# print of the results:
import pylab as p
p.clf()
pfig=p.figure(1); p.hold(T
pfig.set_size_inches(12,8)
p.plot(y_1_hist1,y_2_hist1
p.plot(y_1_hist1,y_3_hist1
p.legend();
p.savefig('ce0testlmlfig1.
```

Maths



...different abstraction levels...

The **gap** between the math def and the computation of its solution is **covered by relying on high-level mathematical abstractions**,

which

can be expressed both in LaTeX and in Python, helping both in problem recognition and in the generation of the code for the resolution pattern