# The Parrot VM

(40’)

Its goals and challenges for dynamic languages

François Perrad

francois.perrad@gadz.org

## About Me

- Parrot-porter since Oct 2005
- In the TOP 10 of committers
- Releaser Manager of Parrot 1.1.0
- Packager for Windows (with mingw)
  - http://sourceforge.net/projects/parrotwin32/
  - ~1 daily build on Smolder
- Mostly a HLL implementor
  - Lua
  - WMLScript (bytecode translator)
  - Markdown
  - ...

## About Parrot VM

- a Virtual Machine started in 2001 ...
- targets dynamic languages
  - multi paradigm : functional, OO, ...
  - Register based (not stack based)
  - Continuation Passing Style (CPS)
- allows interoperability between HLL
  (High Level Language)
  - reuse components
- a great environment for HLL implementor

- Visible side effects
  - CLR/DLR
  - JSR 292

## About commodity

- *commodity* is some good for which there is demand, but which is supplied without qualitative differentiation across a market

- An example
  - generic pharmaceuticals

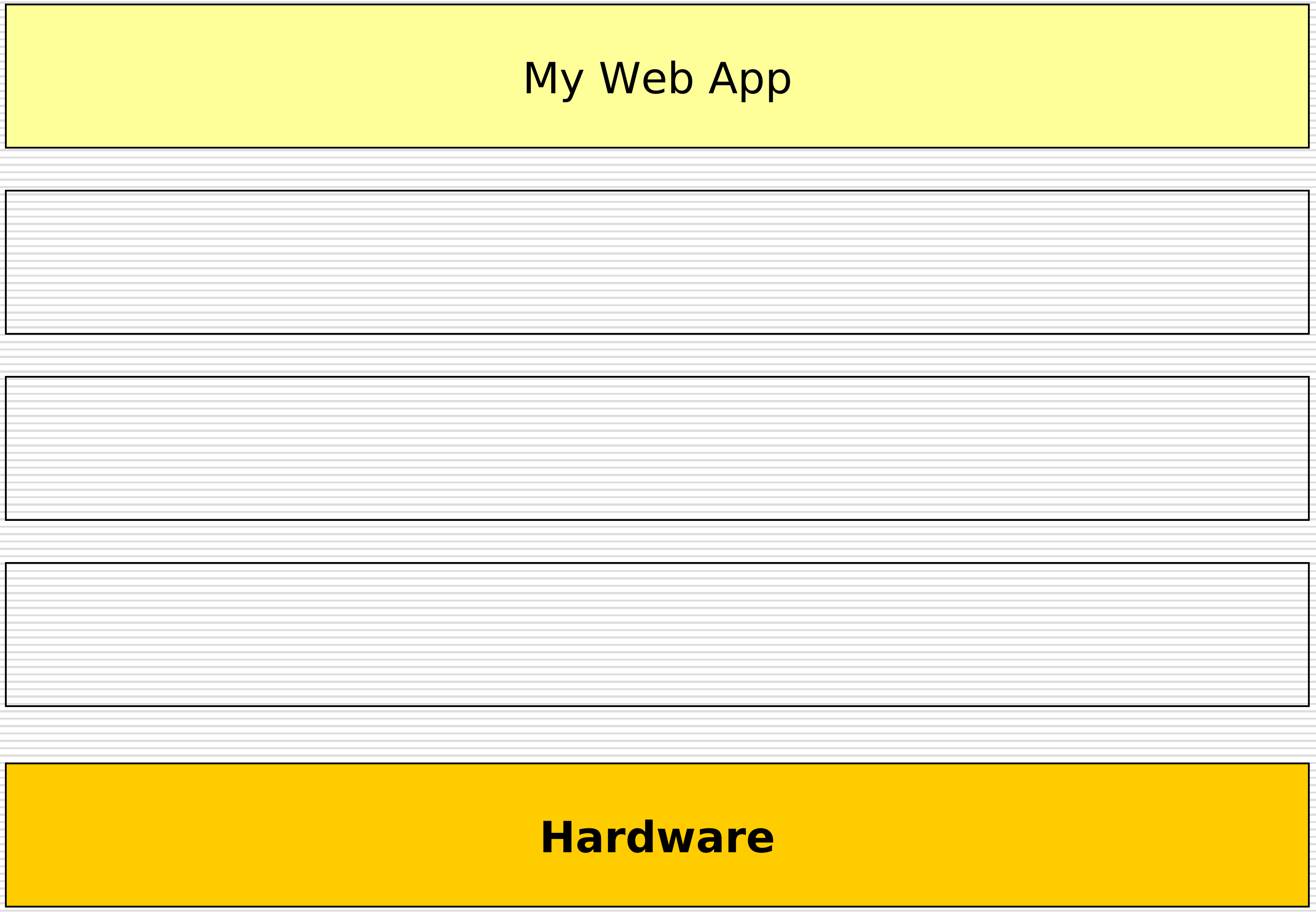
- the interest for customers is lower prices

# About commoditization

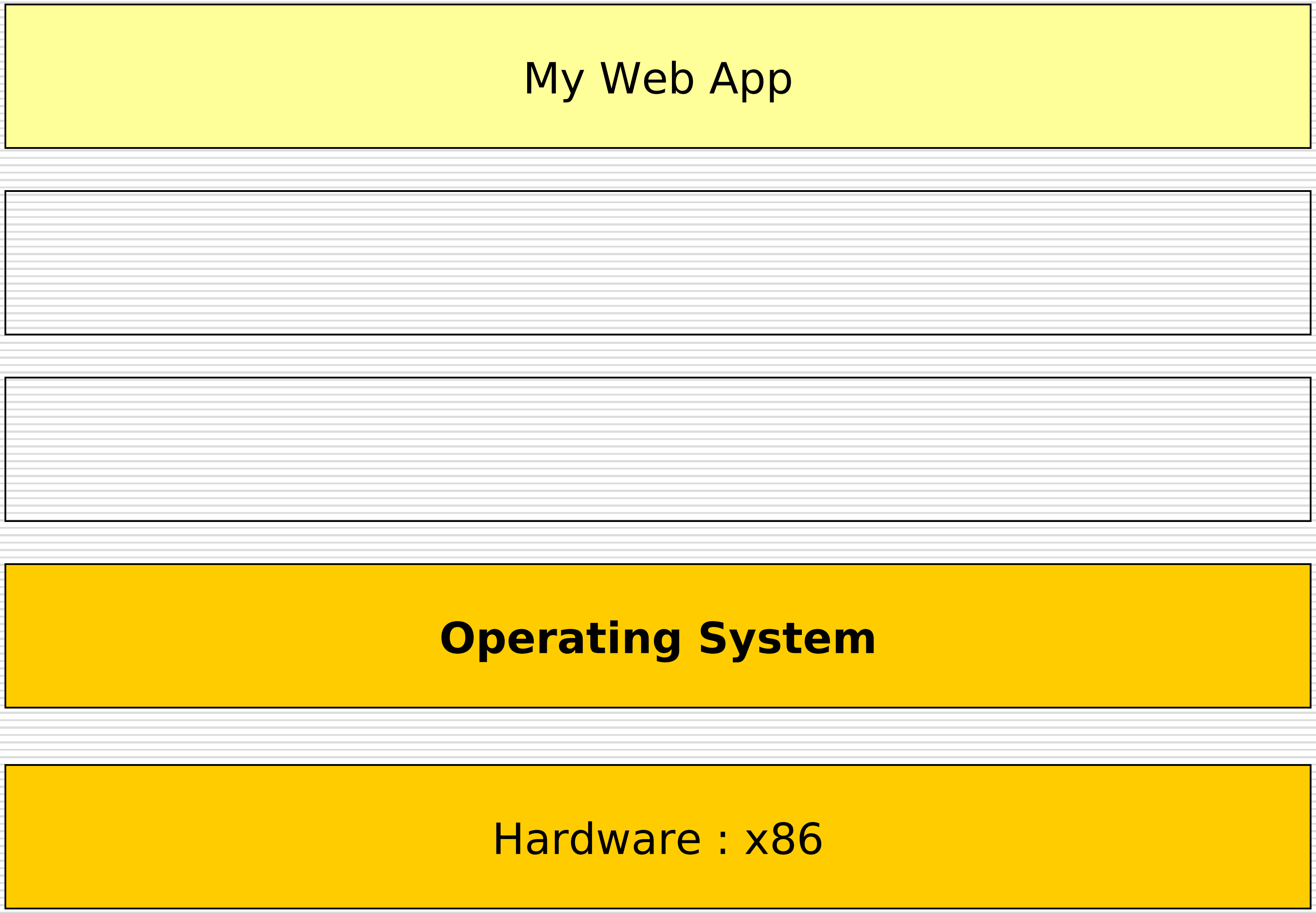
- *commoditization* is the process by which goods that have economic value and are distinguishable in terms of attributes (uniqueness or brand) end up becoming simple commodities in the eyes of the market or consumers.
- An example : Eclipse IDE
  - IBM *gives/opens* its codebase
  - the IDE market disappears
  - competitors die or survive in niche segment
  - a new market : plugins for Eclipse
  - Who is best placed in this market ?
  - Who controls the Eclipse roadmap ?

# A stack of commodities

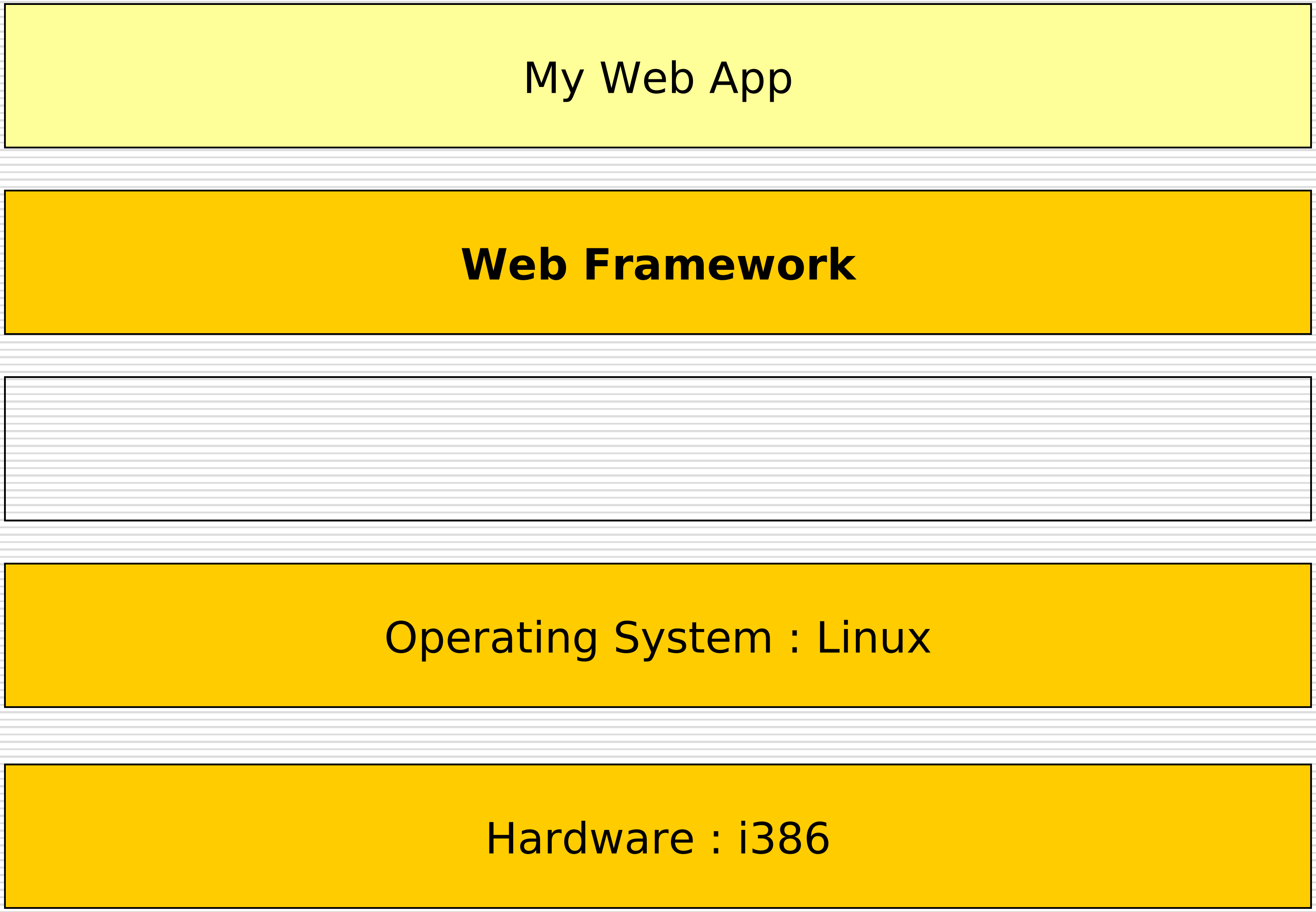
# A stack of commodities



## A stack of commodities



# A stack of commodities



## A stack of commodities

My Web App

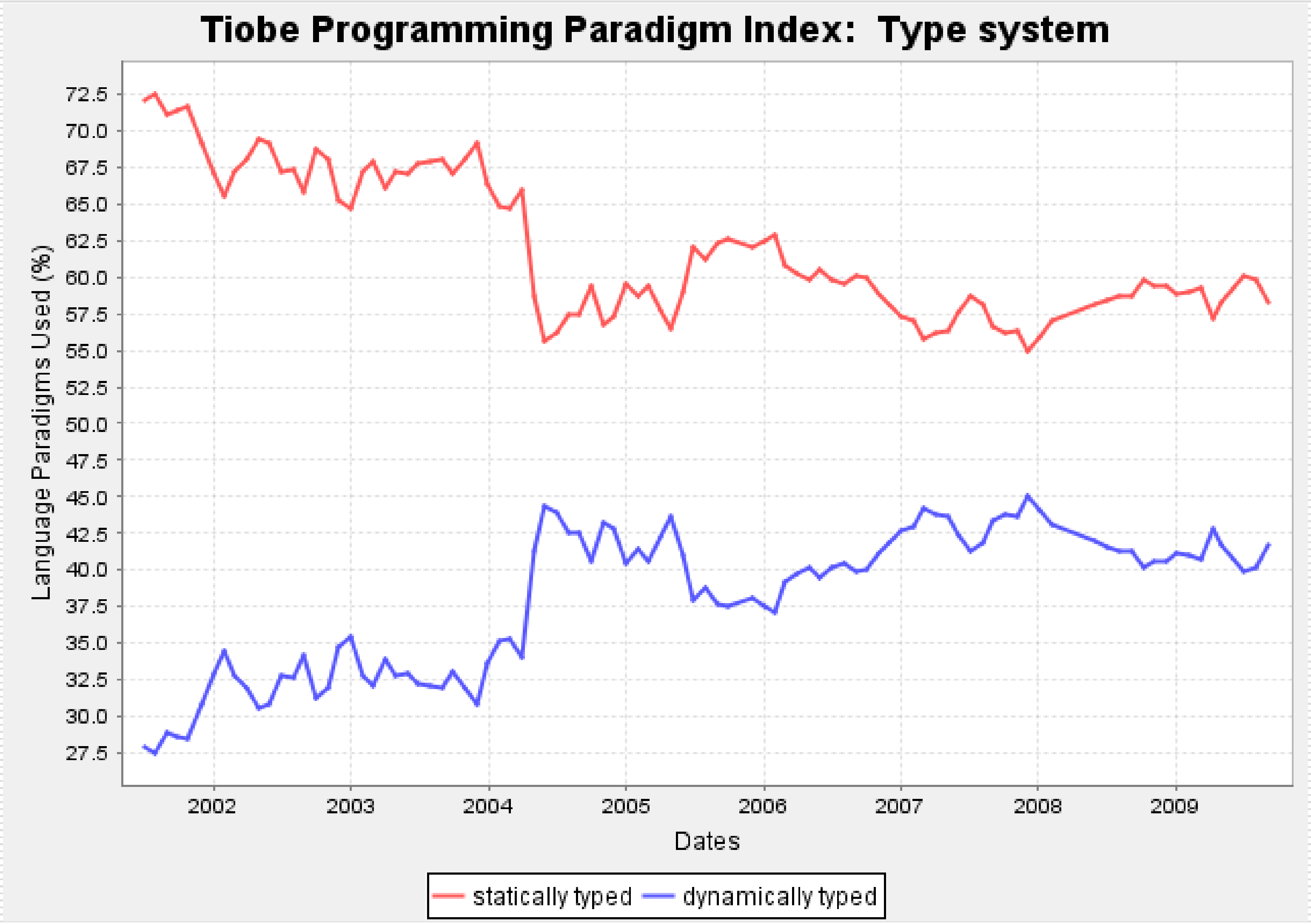
Web Framework

**Virtual Machine (Interpreter)**

Operating System : Linux

Hardware : i386 – **multi core**

# Dynamic languages grow



## C/C+ implementation

| | |
|---|---|
| PHP | PHP 4.x, PHP 5.x |
| VisualBasic | - |
| Perl | Perl 5.8, Perl 5.10 |
| Python | CPython 2.6, CPython 3.1, Stackless, Psyco V2 (JIT), Unladen Swallow 2.6 (llvm) |
| JavaScript | SpiderMonkey, V8, ... |
| Ruby | Ruby 1.8, Ruby 1.9 (YARV), Rubinius |
| Lua | Lua 5.1, LuaJIT 5.1 |

## JVM implementation

| | |
|---|---|
| PHP | - |
| Perl | - |
| Python | JPython 2.5 |
| JavaScript | Rhino |
| Ruby | JRuby 1.8 |
| Lua | Kahlua (J2ME) |
| Scala | Scala 2.7 |
| Groovy | Groovy 1.6 |

## .NET/mono implementation

| | |
|---|---|
| PHP | Phalanger |
| VisualBasic | *VB.NET* |
| Perl | - |
| Python | IronPython |
| JavaScript | *JScript.NET* |
| Ruby | IronRuby |
| Lua | - |
| Scala | ? |

## Challenges

- Languages must move on VM
  - Initial C implementation is just a prototyp or a prove of concept
- Open Source resources are limited
- Efforts are duplicated
  - in language
  - in libraries binding
  - in module & framework
- core language or full modules ?

## Conditions for success

- Official Specifications (vs reference implementation)
  - rubyspec.org/ (Rubinius)
  - perlcabal.org/syn/ (Perl6)
- Official & huge Test Suite
  - svn.pugscode.org/pugs/t/spec/ (Perl6)
- Mostly implements the language with the language itself
  - codespeak.net/pypy/ (Python)
- User friendly policies for support, version, maintenance, …

## Features for HLL implementor

- Multiple runcore (opcode interpreter)
    - switch, C goto, profiler, ...
- GC - Garbage Collection
- Exception mechanism
- Unicode support (ICU)
- PBC - a bytecode file format
- Dynamic PMC (PolyMorphic Container)
    - build the HLL types as object
    - as shared library
- Dynamic Opcode
    - as shared library
- NCI - Native Call Interface
    - a FFI mechanism

## Features for HLL implementor

- PIR – Parrot Intermediate Representation
    - the interface of Parrot
    - an OO assembler, with rich calling convention
    - 4 types : int, num, string, pmc
- PCT - Parrot Compiler Toolkit
    - PGE – Parrot Grammar Engine
        - parse the source
        - produce an AST (Abtract Syntax Tree)
    - NQP – Not Quite Perl
        - describes action on AST
    - HLLCompiler : base class for a compiler
    - PIR generation from AST

## Roadmap : some milestones

- 2005 : PGE
- Jan 2008 : NQP availability
  - Start of Rakudo dev.
- Mar 2009 : Parrot 1.0
  - API stable (with deprecation rules)
  - Installable
  - Languages leave the nest
- Jul 2009 : Parrot 1.4
  - refactoring

# Roadmap : next milestones

- ??? : Plumage – Module ecosystem
  - trac.parrot.org/parrot/wiki/ModuleEcosystem
  - gitorious.org/parrot-plumage/parrot-plumage
- Jan 2010 : Parrot 2.0
- 2010 Q1 : Rakudo Star
  - a large subset of Perl6 usable in *real* world
- Updated roadmap
  trac.parrot.org/parrot/report/14

## Bibliography / Webography

- www.parrot.org
- trac.parrot.org
- F. Elie, Économie du logiciel libre, Eyrolles
- Virtual Machine Showdown: Stack Versus Registers
- The Structure and Performance of *Efficient Interpreters*
- How to *not* *write Virtual Machines for Dynamic Languages*

"*one bytecode to rule them all*"