

30 Years of Bumping into Philip

Roberto Ierusalimschy



Pontifícia
Universidade
Católica do
Rio de Janeiro

Date: Thu, 26 Jan 2017 18:25:23 +0000
From wadler@inf.ed.ac.uk Thu Jan 26 16:26:01
2017
From: Philip Wadler <wadler@inf.ed.ac.uk>
To: Roberto Ierusalimschy <roberto@inf.puc-rio.br>
Subject: Possible visit in April 2017?

Roberto,

We met at WG 2.8 in Aussois in October 2013.
[...]

Of course I knew whom he was (one of my Idols)!

<http://perevodik.net/en/posts/39/>

“If programming languages were cars”



“Haskell: How do I drive this thing? Despite being older than many languages on the list, Haskell is more modern in most ways.”

IFIP WG 2.8 Aussois, October 2013



“Lua: cute, efficient, and becoming very trendy.”

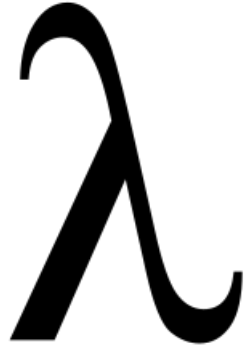
What is Lua?

- Yet another scripting language
 - a “dysfunctional language”?
- Not totally unlike Python, Ruby, Perl, etc.

For me, Haskell was an elegant, theoretical language with some interesting real-life applications.

For me, Haskell was an elegant, theoretical language with some interesting real-life applications.

For them, Haskell was an industrial language possibly tainted by too many users.

A large, bold, black lambda symbol (λ) is centered in the upper half of the slide. It is enclosed within a large, blue, stylized right square bracket that spans most of the width of the slide.

Nonetheless (?), Scheme has been continually a strong influence over the evolution of Lua.

And we had met before that meeting...



**I SIMPÓSIO
BRASILEIRO DE
LINGUAGENS
DE
PROGRAMAÇÃO**
4 a 6 de setembro
de 1996



SOCIEDADE
BRASILEIRA DE
COMPUTAÇÃO

10:30 CONFERÊNCIA II

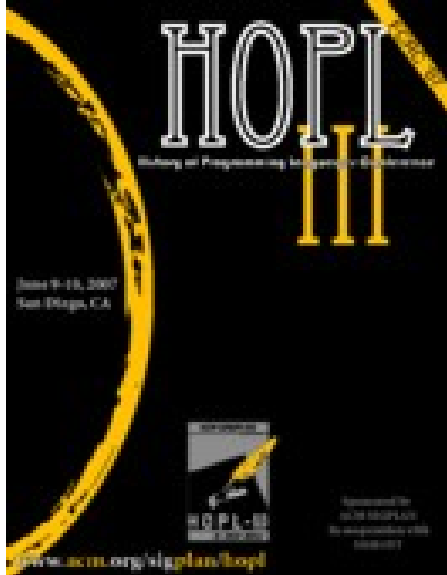
[Lazy vs. Strict](#)

[Philip Wadler \(University of Glasgow, Scotland\)](#)

(With the Lambda performance, but yet without the full costume.)



2007: III ACM History of Programming Languages



The evolution of Lua

Roberto Ierusalimschy, Luiz Henrique de Figueiredo, Waldemar Celes

A history of Haskell: being lazy with class

Paul Hudak, John Hughes, Simon Peyton Jones, Philip Wadler



Roberto Ierusalimschy
Luiz Henrique de Figueiredo
Waldemar Celes

Roberto Ierusalimsky (PUC-Rio),
Luiz Henrique de Figueiredo (IMPA),
and Waldemar Celes (PUC-Rio)

The evolution of Lua

June 9, 2007

Session Chair: Julia Lawall (DIKU)



© ACM COPYRIGHT NOTICE. Copyright © 2007
by the Association for Computing Machinery
<http://www.acm.org/pubs/copyright_policy/#Notice>

History of Programming Languages Conference III

June 9 - 10, 2007
San Diego, California



Influences of Embedding in the Language

- Coroutines
- Integers
- Exception handling
- “Lambda: The Ultimate API”

Roberto Ierusalimschy, Luiz H. de Figueiredo, Waldemar Celes.
Passing a Language through the Eye of a Needle. [Communications of the ACM](#), 54(7):38-43, July 2011.



After that message, Philip gave a talk at PUC-Rio.

We met again in Lisbon, Jan 2019.
(PC for HOPL IV)

PUC-Rio: Course notes

Staff

- **Instructor** [Philip Wadler](#)
- **Host** [Roberto Ierusalimsky](#)

Schedule

Lectures and tutorials take place Fridays and some Thursdays in 548L.

- **13.00–14.30pm** Lecture
- **14.30–16.00pm** Tutorial

Fri 29 Mar	Naturals
------------	----------

Informatics-Huawei Distinguished Visitor Scheme		
Eligibility	• The aim of this scheme is to enable internationally renowned academics to visit the School of Informatics. • Applications are invited in all areas of research covered by the School, with the exception of cybersecurity. • Visitors will typically be professors or of equivalent standing (consideration will be given to their academic record). • The scheme covers permanent employees of overseas universities or equivalent research institutions. • Visitors must remain employed by their home institution during the visit and draw their usual salary.	
Criteria	• Every visitor must have an sponsor within the School of Informatics. Sponsors should be members of academic staff. • The sponsor submits the application, after having obtained the agreement in principle of the visitor and their home institution. • Visitors must stay in Edinburgh (or nearby) for the duration of the visit and contribute to the academic life of the host institute and the School. • The assumption is that visitors will travel to Edinburgh for their visit; should this be impossible due to travel restrictions or other public health measures, virtual visits can be allowed on a case-by-case basis.	
Value	The distinguished visitor scheme will cover: • The cost of traveling to Edinburgh for the visitor and accompanying family (partner and dependent children). One return trip per person per visit will be funded. • The additional cost of living in Edinburgh (e.g., accommodation, childcare). The indicative amount for this is up to £3,200 per month. • Research expenses, which can include the cost for running experiments, small items of equipment, and part-funding for PhD students or RAs working on the project the visitor undertakes. • Students and RAs funded this way would normally be already based in Edinburgh and would work temporarily with the visitor. • All expenses must be budgeted for and justified as part of the application. • The minimum duration of a visit is 2 months, the maximum 12 months. The scheme can also fund multiple visits by the same visitors, possibly over more than one academic year. • It is anticipated that up to 10 visitors per year will be hosted.	
Deadlines and Decisions		
Application Deadline	Decision Date	Visit Start Window
13 November 2020	End of November	March to July

Three months (Dec 2021-Feb 2022) at the University of Edinburgh.

Work in a model for the type system of Pallene, a companion language for Lua.

Visit somewhat impacted by the Omicron variant (first detected one week before the trip).

Shallow Types

Pallene is a statically-typed language that operates over (untyped) Lua data structures. Types for primitives are always correct (allowing unboxing); types for data structures work as “views”, telling the compiler what to expect/check when accessing them.

Shallow Types

An identity function

```
-- in Pallene
function id (a : {{int}}) : {{int}}
  return a
end
```

```
-- in Lua
id(4)          error
id({3, 4})     ok
id({{3, 4}, {5, 6}})  ok
```

Shallow Types

Example of two views into the same array: B is an alias for A, but with a different type.

```
let A:{{int}} = {} : {{int}} in
let B:int = A as int in
  A[1] := B;      -- A[1] becomes a self-reference to A
  B[2] := 10;     -- A[2] also becomes 10
  A[1][2]         -- returns 10
```

Final Remarks

- It is an honor to be a friend of Philip Wadler
- It is one of the main entries in my CV



Parabéns!