Functions in Lua

Roberto Ierusalimschy

Pontifícia
Universidade
Católica do
Rio de Janeiro

What about Lua?

A simple, dynamic, imperative language, with tables, coroutines, and functions.


waze


CELESTIA


QSC


ADOBE® PHOTOSHOP® LIGHTROOM® 3


MEDIAWIKI


VLC
Media Player


SNORT

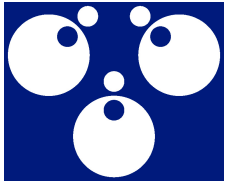

REAPER
Digital Audio Workstation


NMAP PROJECT


WIRESHARK


sas


torch
Memory




redis

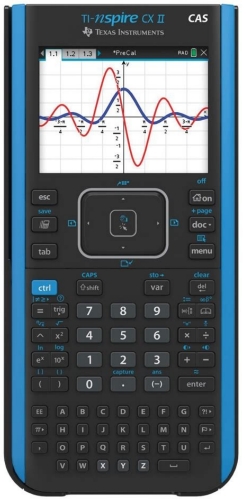

Fusion 8
The world's most advanced visual effects and motion graphics software, now available for Mac and Windows!
Public Beta Now Available


Q-SYS
Designer Software


neovim


iClone
Real-time 3D Animation


SAMSUNG


TI-nspire CX II
CAS
TEXAS INSTRUMENTS


cisco
RV325


PiCy


SIERRA
WIRELESS
AirLink® ES450


TV Interativa se faz com
ginga
Lua

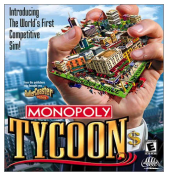

Games for Windows
COMPANY OF HEROES
THQ


Games for Windows
UNIVERSE AT WAR
SEGA


Games for Windows
DUNGEONS
kalypso


Tap Tap
Revenge


BLITZKRIEG
CDV


Introducing The World's First Competitive Sim!
MONOPOLY
TYCOON

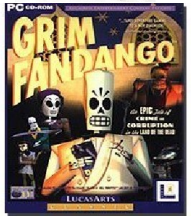

DRIVER
SAN FRANCISCO
UBISOFT


Games for Windows
PC DVD
The Lord of the Rings Online
Shadows of Angmar

ROBLOX


XBOX 360
LIVE
CRACKDOWN

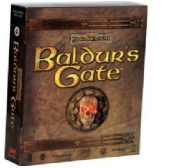

REQUIEM
MEMENTO MORI


PC CD-ROM
GRIM FANDANGO
LUCASARTS


XBOX
amped

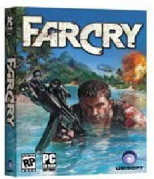

XBOX 360
HULK
MARVEL
SEGA


PlayStation.2
DESTROY ALL HUMANS!
THQ

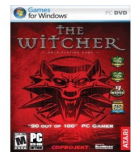

Baldur's Gate


RAIL SIMULATOR
EA


WORLD of WARCRAFT

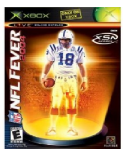

FARCRY
UBISOFT


IMPOSSIBLE CREATURES


Games for Windows
PC DVD
THE WITCHER

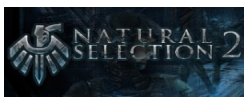

PC DVD
STALKER
THQ


Blossom


XBOX
NFL FEVER
18


soul ride
Snowboarding Simulator


Diner Dash 2
EA

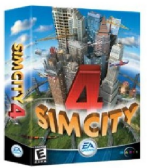

NATURAL SELECTION 2


AVALON
ONLINE


XBOX
Links


BRUTAL LEGEND
EA


XBOX 360
LIVE
FABLE II
Microsoft


SIMCITY
4
EA


ANGRY BIRDS
PLAY


PC DVD
SpellForce
THQ


NAPOLEON
TOTAL WAR
16
SEGA


Games for Windows
PC DVD
SUPREME COMMANDER
THQ


BATTLE ZONE
98 REDUX


XBOX 360
LIVE
TOP SPIN 2


XBOX 360
LIVE
SONIC
THERE BE HERO


PC DVD
YOU ARE EMPTY

Games for Windows
PC DVD
DUNGEONS

BUZZ!
Quiz TV

PlayStation.2
BRAVE

PlayStation
Home

MDK2

Games for Windows
PC DVD
CRYSIS

Games for Windows
PC DVD
SUPREME COMMANDER
Forged Alliance

HOMEWORLD2
SIERRA

PlayStation.2
PSYCHONAUTS

Bubble
Ball

THE INCREDIBLES
THQ

There

PC DVD
DAWN OF WAR
DARK CRUSADE
THQ

The Guild 2

LUCASARTS
MONKEY ISLAND

EMPIRE
TOTAL WAR
16
SEGA

PC DVD
CIVILIZATION
2K GAMES

plus+

XBOX 360
LIVE
CRACKDOWN

STAR WARS
EMPIRE AT WAR

MIGHT AND MAGIC
HEROES V

CRIMSON
STEAM PIRATES

What are the Goals?

- Portability
- Simplicity
- Small size
- Scripting

Portability

- Runs on most platforms we ever heard of: Posix (Linux, BSD, etc.), OS X, Windows, Android, iOS, Arduino, Raspberry Pi, Symbian, Nintendo DS, PSP, PS3, IBM z/OS, etc.
- Runs inside OS kernels: FreeBSD, Linux
- Runs directly on the bare metal, without an OS: NodeMCU ESP8266

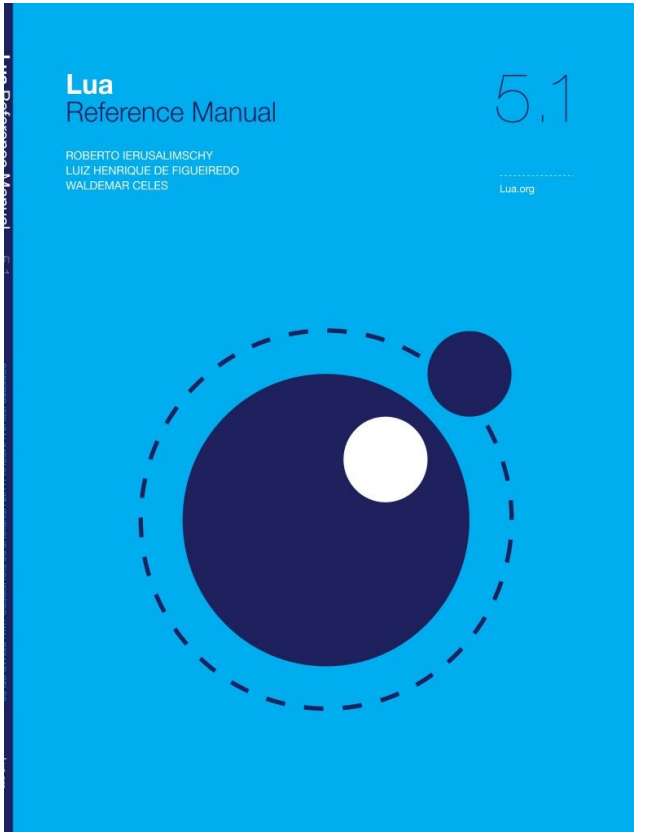
Simplicity

Reference manual with less than 100 pages (proxy for complexity).

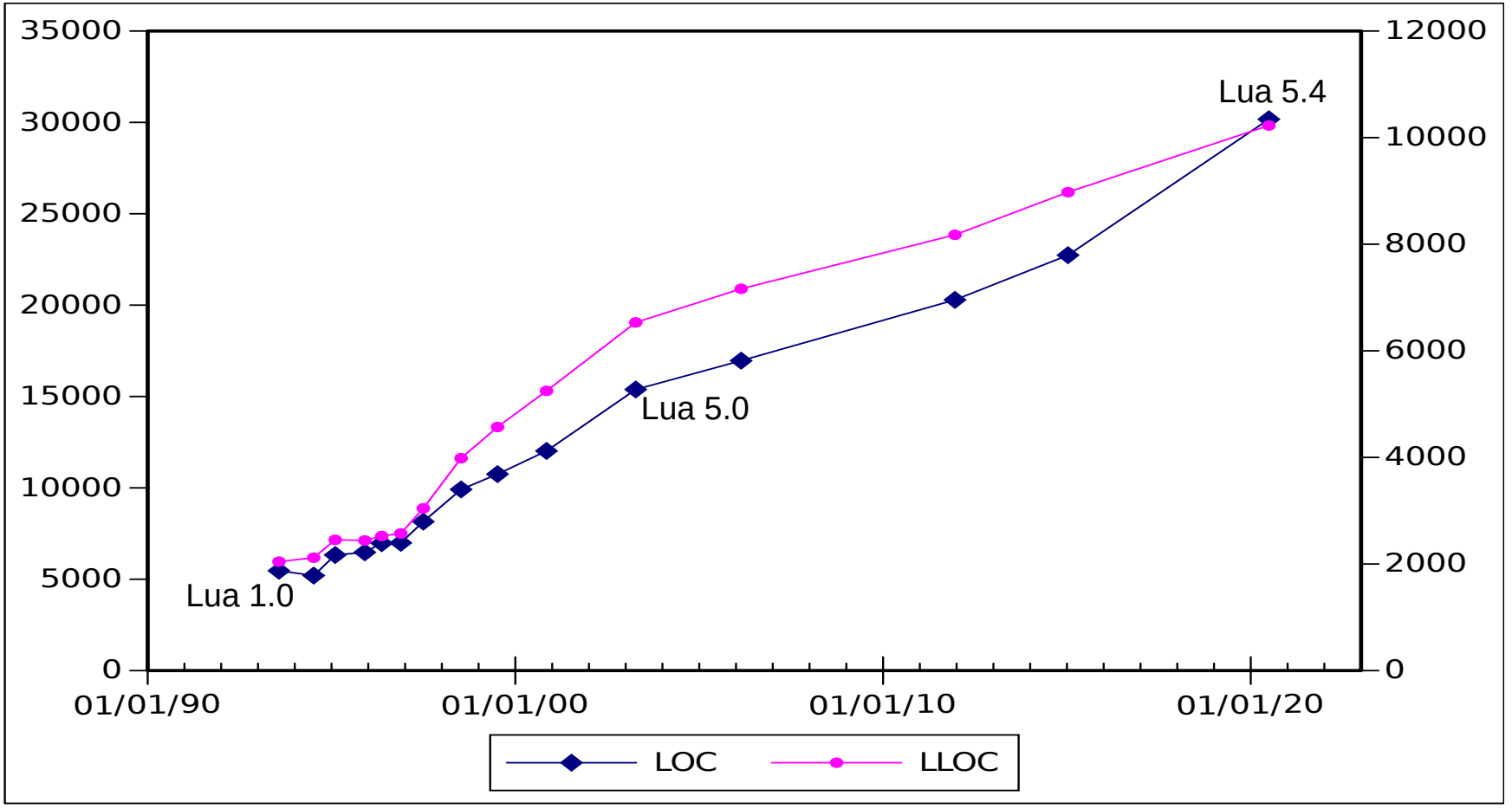
Documents the language, the libraries, and the C API.

(spine)

Lua Reference Manual 5.1 ROBERTO IERUSALIMSCHY / LUIZ HENRIQUE DE FIGUEIREDO / WALDEMAR CELES Lua.org



Size


35000
30000
25000
20000
15000
10000
5000
0
12000
10000
8000
6000
4000
2000
0
01/01/90
01/01/00
01/01/10
01/01/20
Lua 1.0
Lua 5.0
Lua 5.4
LOC
LLOC

Scripting

- Scripting language is not a synonym for dynamic language.
- Program written in two languages.
- System language implements the hard parts of the application.
- Scripting *glues* together the hard parts.

Lua and Scripting

- Lua is implemented as a library.
- Lua has been designed for scripting.
- Good for *embedding* and *extending.*
- Embedded in C/C++, Java, Fortran, C#, Perl, Ruby, Python, etc.

Tables

- Only data structure in Lua
- Associative arrays
  - maps keys of any type to values of any type
- Arrays: tables with integer keys
- Records: tables with field names (strings) as keys: `t.x` is sugar for `t["x"]`

Coroutines

- Collaborative threads
- Equivalent to one-shot continuations
  - we can write `call/1cc` with them

Functions

anonymous functions

function values

first-class functions

closures

lambdas

higher-order functions

What does “function” mean?

- Functions are first-class values.
- Functions can be nested.
- Nested functions have lexical scoping (with mutable variables).
- There are anonymous functions.

Properties Somewhat Independent

- C has functions as first-class values, but no nesting.
- Lisp (original) has functions as first-class values and anonymous functions, but no lexical scoping.
- Pascal has lexical scoping, but functions are not first-class values.

Properties Somewhat Independent

- *Blocks* in Ruby and Smalltalk are anonymous with lexical scoping, but they are not first-class values.
- Java has lexical scoping, but only for values.
- C++ needs manual “assigment conversion” for mutable external variables.

How Lua uses functions to achieve its goals

Simplicity/Small size

- All functions are anonymous.
- Syntax sugar for several typical constructs.

```
function foo (...) ... end
```

```
foo = function (...) ... end
```

```
local function foo (...) ... end
```

```
local foo;
foo = function (...) ... end
```

```
function foo (...) ... end
```

```
foo = function (...) ... end
```

```
local function foo (...) ... end
```

```
local foo;
foo = function (...) ... end
```

```
local foo = function (...) ... end
```

Eval

- `eval` is a hallmark of dynamic languages.
- Lua offers a `load` function instead, which returns a function.

```
local f = load(“print(10)”)
f()  --> 10
f()  --> 10
```

Load

- Clearly separates compilation from execution.
- `load` is a pure function.
- Any code always runs inside some function.
- It is easier to do `eval` from `load` than the reverse.

Modules

- Tables populated with functions

```
local math = require "math"
print(math.sqrt(10))
```

```
local math = require "math"
local sqrt = math.sqrt
print(sqrt(10))
```

Modules

- Syntactically, a module is a function that creates its table.
- Local variables are private to the module.
- the expression `require “math”` finds an adequate file, then loads and executes it; the returned value is the module table.

Exception Handling

- All done through two functions, `pcall` and `error`.
- `pcall` calls a function in *protected mode*, catching any error.
- `error` *raises* an error. (It calls the continuation of the inner enclosing `pcall`.)

```
try {
  <block/throw>
}
catch (err) {
  <exception code>
}
```

```
local ok, err = pcall(function ()
  <block/error>
end)
if not ok then
  <exception code>
end
```

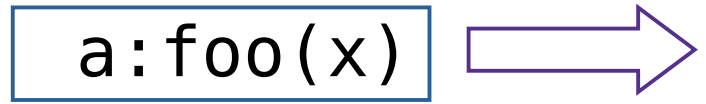
Exception Handling

- simple semantics
- no extra syntax
- simple to interface with other languages

Objects

- first-class functions + tables ≈ objects
- syntactic sugar for methods (colon syntax)

```
a:foo(x)
```



```
a.foo(a,x)
```

```
function a:foo (x)
  ...
end
```

```
a.foo = function (self,x)
  ...
end
```

Objects

- Flexible
- easy to interface with other languages
- clear semantics
- Few new concepts: a *method* is just a regular function

The Lua-C API

The Lua-C API

- Functions are constructs found in most languages, wich compatible basic semantics.
- Constructions based on functions are easier to translate between different languages.

The Lua-C API

- Modules and OO programming need no extra features in the Lua-C API.
  - all done with standard mechanisms for tables and functions.
- Exception handling and `load` go the opposite way: primitives in the API, exported to Lua.

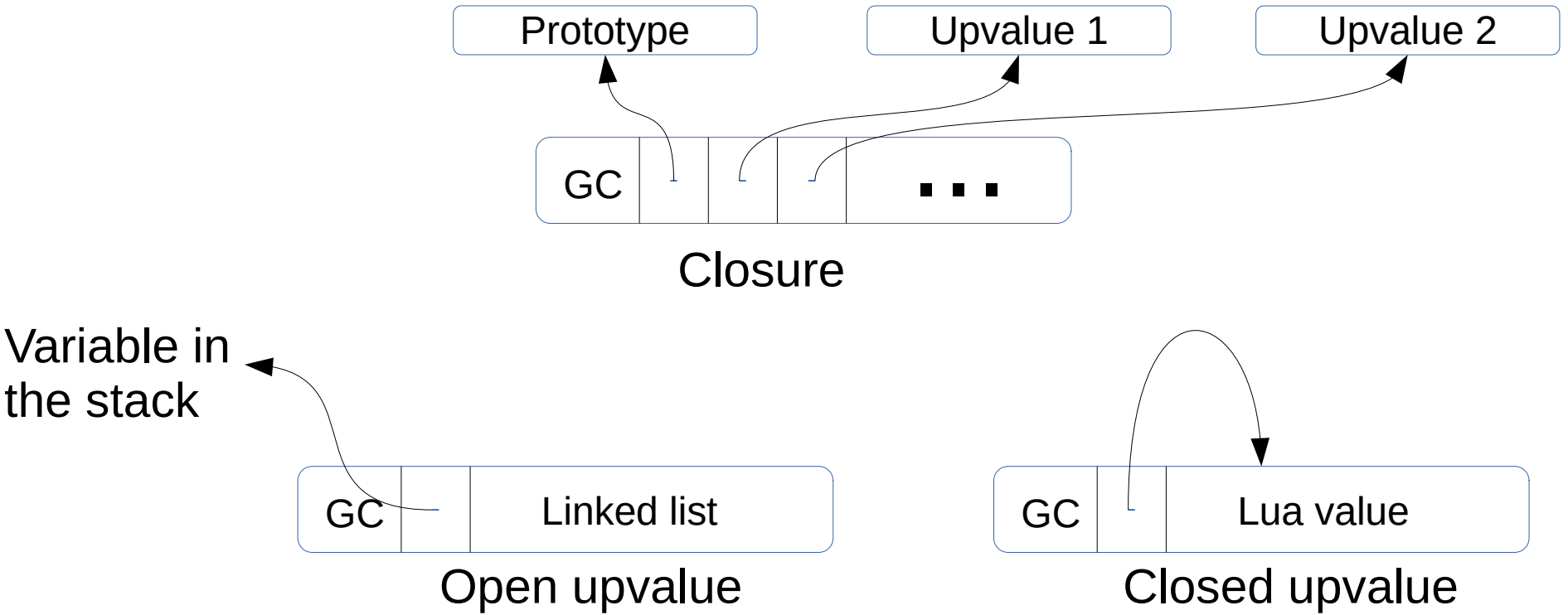
Implementation Requirements

- One-pass compiler.
- Safe for space.
- No assigment conversion.
- A function may use variables from several different stacks (coroutines).

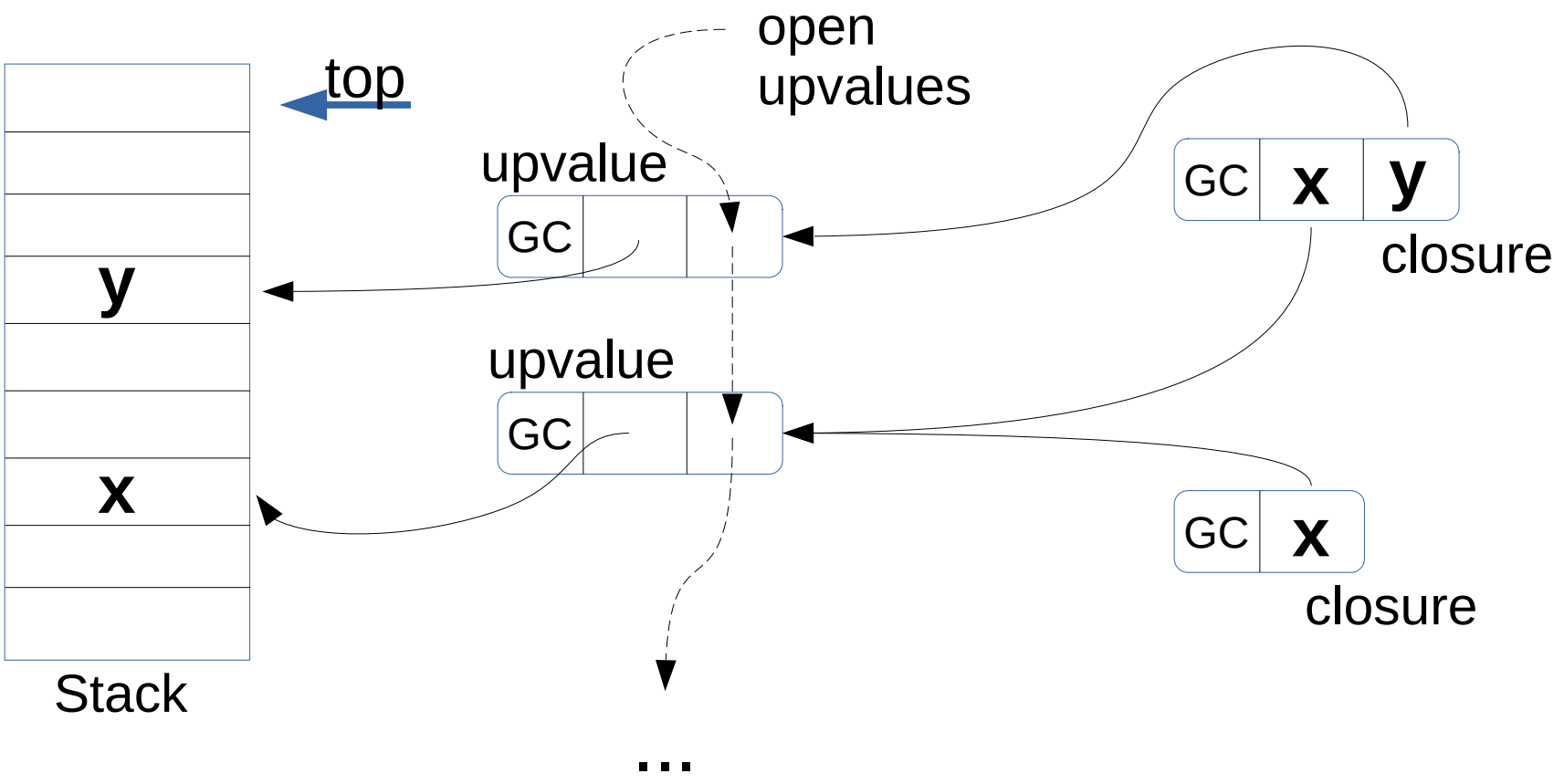
Implementation

- Geared towards imperative languages.
- Zero cost when not used: All local variables live on the stack.
- Lua uses *upvalues* to intermeditate the access to external variables.

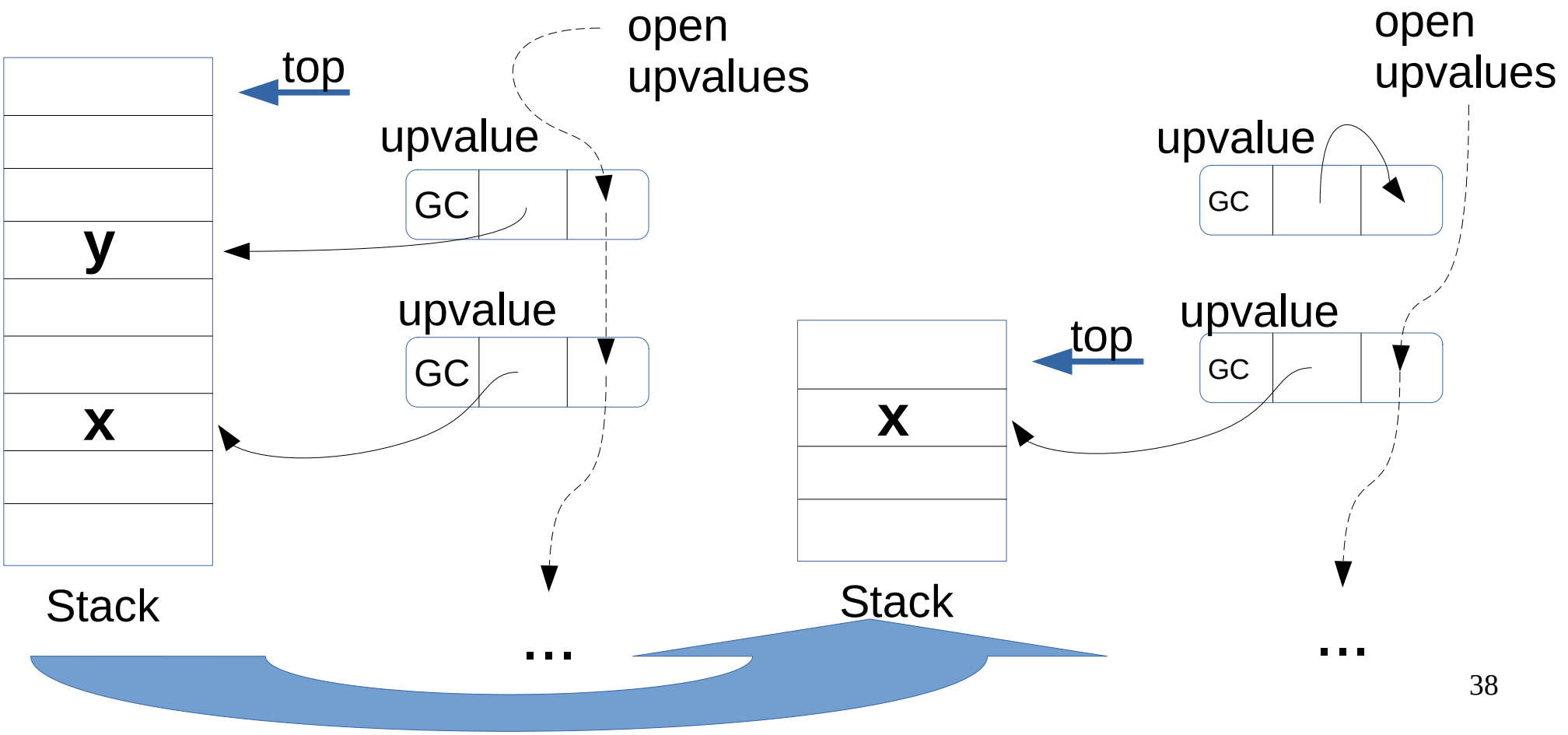
Basic data structures



List of open upvalues (for unicity)



Closing an upvalue - Closing a scope

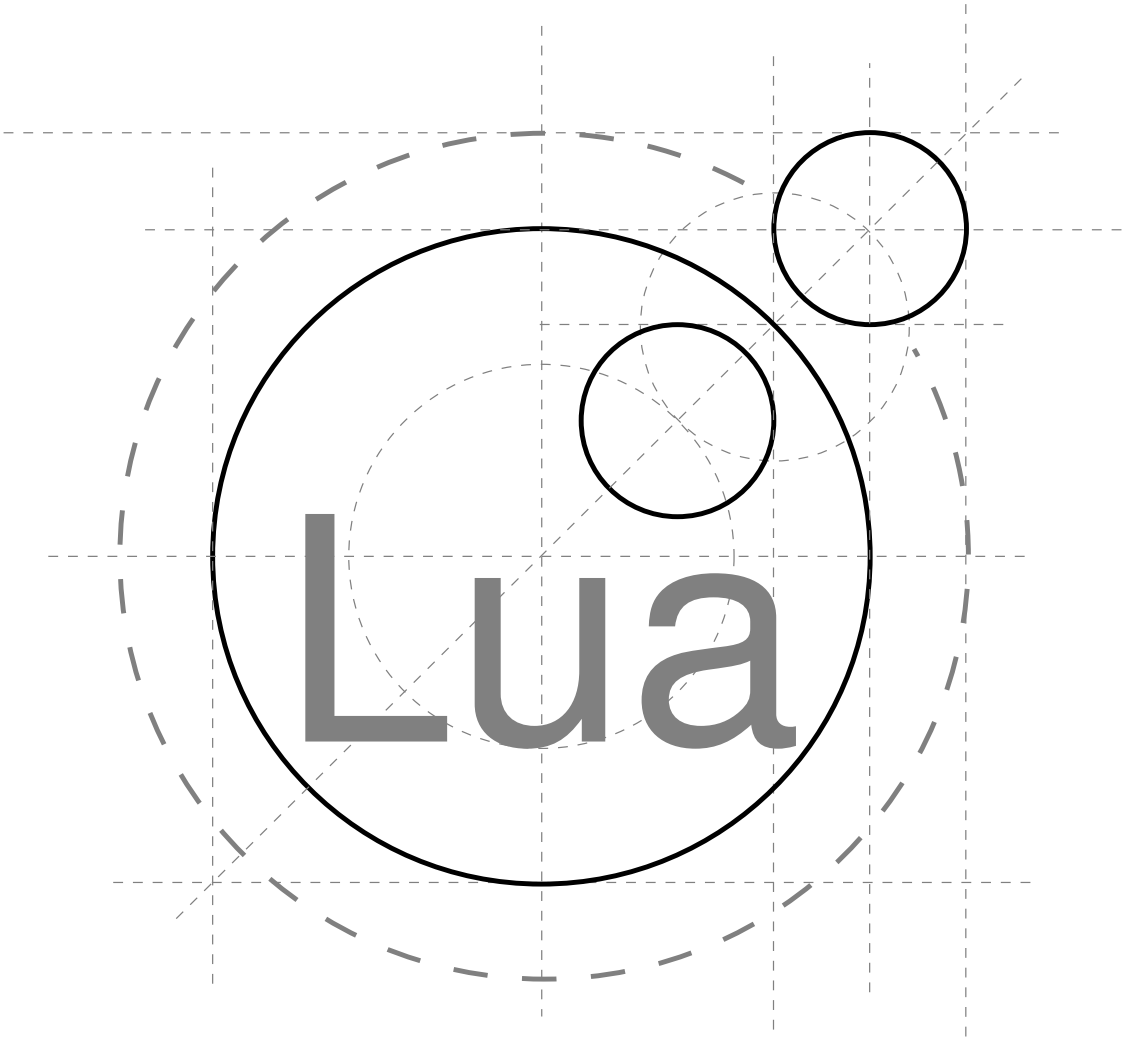


Several Details...

- Uses flattening for nesting.
- List of open upvalues is limited by program syntax.
- Unicity needed for mutability.
- A closure may point to upvalues in different stacks.

Final Remarks

- First-class functions are a key ingredient for programming in Lua.
- Lua itself uses functions for several basic constructs in the language.
- In Lua, the use of constructors based on first-class functions greatly helps to make the C API general.


Lua