Decoding Lua: Formal Semantics for the Developer and the Semanticist

Mallku Soldevila[1], Beta Ziliani[1], Bruno Silvestre[2], Daniel Fridlender[3] and Fabio Mascarenhas[4]

[1]FAMAF/UNC and CONICET, [2]INF/UFG, [3]FAMAF/UNC, [4]DCC/UFRJ

Abstract

We provide formal semantics for a large subset of the Lua 5.2 programming language. We validate our model by mechanizing it and testing it against the test suite of the reference interpreter.

About Lua

- Lightweight imperative scripting language, featuring dynamic typing, automatic memory management, data description facilities, and metaprogramming mechanisms to adapt the language to specific domains [5].
- Used in diverse applications: game development [3], plugin development (the photo editing software Adobe Photoshop Lightroom, and the type-setting system LuaTex), web application firewalls, and embedded systems.

Modular semantics

$s ::= \textbf{if } e \textbf{ then } s \textbf{ else } s \textbf{ end} \mid \textbf{;} \mid \ldots$

$v ::= \textbf{nil} \mid \mathit{bool\_literal} \mid \ldots$

$e ::= v \mid e \ \mathit{binop}\ e \mid \ldots$

$\mathit{binop} ::= \textbf{and} \mid \textbf{or} \mid \ldots$

$$\frac{v \notin \{\textbf{nil}, \textbf{false}\}}{\textbf{if } v \textbf{ then } s_1 \textbf{ else } s_2 \textbf{ end} \ \rightarrow^{s} \ s_1}$$

$$\frac{v \in \{\textbf{nil}, \textbf{false}\}}{\textbf{if } v \textbf{ then } s_1 \textbf{ else } s_2 \textbf{ end} \ \rightarrow^{s} \ s_2}$$

$$\frac{op \in \{\textbf{and}, \textbf{or}\}}{v \ op \ e \ \rightarrow^{e} \ \delta(op, v, e)}$$

Figure 1: Syntax and semantics of (some) stateless statements and expressions

$s ::= \ldots \mid \textbf{local } x = e \textbf{ in } s \textbf{ end} \mid x = e$

$$\frac{\sigma' = (r, v), \sigma}{\sigma : \textbf{local } x = v \textbf{ in } s \textbf{ end} \ \rightarrow^{s\_\sigma} \ \sigma' : s[x \backslash r]}$$

$$\frac{\sigma' = \sigma[r := v]}{\sigma : r = v \ \rightarrow^{s\_\sigma} \ \sigma' : \textbf{;}} \qquad \sigma : r \rightarrow^{e\_\sigma} \ \sigma : \sigma(r)$$

Figure 2: Syntax and semantics of local variable definition and assignment.

$$E ::= [\,] \mid \textbf{if } E \textbf{ then } s \textbf{ else } s \textbf{ end} \mid \textbf{local } x = E \textbf{ in } s \textbf{ end} \mid x = E \mid E \ \mathit{binop}\ e \mid \mathit{unop}\ E$$

Figure 3: Evaluation contexts.

$$\frac{e \ \rightarrow^{e} \ e'}{\sigma : E[\![e]\!] \ \mapsto \ \sigma : E[\![e']\!]} \qquad \frac{s \ \rightarrow^{s} \ s'}{\sigma : E[\![s]\!] \ \mapsto \ \sigma : E[\![s']\!]}$$

$$\frac{\sigma : s \ \rightarrow^{s\_\sigma} \ \sigma' : s'}{\sigma : E[\![s]\!] \ \mapsto \ \sigma' : E[\![s']\!]}$$

Figure 4: Semantics of programs.

Lightweight mechanization with PLT Redex

File	Features tested	Coverage
calls.lua	functions and calls	77.83%
closure.lua	closures	48.5%
constructs.lua	syntax and short-circuit opts.	63.18%
events.lua	metatables	90.4%
locals.lua	local variables and environments	62.3%
math.lua	numbers and math lib	82.2%
nextvar.lua	tables, next, and for	53.24%
sort.lua	(parts of) table library	24.1%
vararg.lua	vararg	100%

Figure 5: Lua 5.2's test suite coverage.

Concepts modelled

The features modelled include:

- Every type of Lua value, except *coroutines* and *userdata* (see below);
- Metatables;
- Identity of closures;
- Dynamic execution of source code;
- Error handling;
- A large collection of the services of the standard library.

Conclusion and future work

The formal semantics given, together with its lightweight mechanization, make up a tool that both semanticists and Lua developers can use for understanding and extending the features of the language. Future work includes:

- Adding missing features (coroutines, new operators and metamethods of version 5.3, garbage collector).
- Tools for assisting in the translation of our PLT Redex model to a proof assistant (possibly Coq).
- Use the model to give formal guarantees of correctness of tools for code analysis and language extensions, such as Luacheck, Ravi and Typed Lua [4].

References

[1] M. Felleisen, R. B. Finlder, and M. Flatt. *Semantics Engineering with PLT Redex.* The MIT Press, 2009.

[2] A. Guha, C. Saftoiu, and S. Krishnamurthi. The essence of JavaScript. In *ECOOP '10*, 2010.

[3] R. Ierusalimschy, L. de Figueiredo, and W. Celes. The evolution of an extension language: a history of lua. In *Brazilian Symposium on Programming Languages*, 2001.

[4] A. M. Maidl, F. Mascarenhas, and R. Ierusalimschy. A formalization of Typed Lua. In *DLS '15*, 2015.

[5] L. H. d. F. R. Ierusalimschy and W. Celes. Lua – an extensible extension language. *Software: Practice and Experience*, 26(6):635–652, 1996.

Acknowledgements

We are very grateful to the anonymous reviewers for their insightful feedback.
This poster was done using the template originally created by the *Computational Physics and Biophysics Group* at Jacobs University, and modified by Nathaniel Johnston (`nathaniel@nathanieljohnston.com`).