Bibliography

N. I. Adams, IV, D. H. Bartley, G. Brooks, R. K. Dybvig, D. P. Friedman, R. Halstead, C. Hanson, C. T. Haynes, E. Kohlbecker, D. Oxley, K. M. Pitman, G. J. Rozas, G. L. Steele, Jr., G. J. Sussman, M. Wand, and H. Abelson. Revised^5 report on the algorithmic language Scheme. *SIGPLAN Notices*, 33(9):26–76, 1998. ISSN 0362-1340. 1.1

Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: principles, techniques, and tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1986. ISBN 0-201-10088-6. 2.1.2

Alexander Aiken and Manuel Fähndrich. Dynamic typing and subtype inference. In *FPCA '95: Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture*, pages 182–191, New York, NY, 1995. ACM. 3.3.1

Randall D. Beer. Preliminary report on a practical type inference system for Common Lisp. *SIGPLAN Lisp Pointers*, 1(2):5–11, 1987. ISSN 1045-3563. 3.3.2

Jan Benda, Tomas Matousek, and Ladislav Prosek. Phalanger: Compiling and running PHP applications on the Microsoft .NET platform. In *Proceedings of the .NET Technologies Conference*, pages 31–38, 2006. 2.3

Gregory Blajian, Roger Eggen, Maurice Eggen, and Gerald Pitts. Mono versus .NET: A comparative study of performance for distributed processing. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications*, pages 45–51, 2006. 2

Shri Borde. Tail call performance on x86, 2005. Available at `http://blogs.msdn.com/shrib/archive/2005/01/25/360370.aspx`. 4.2

Brent A. Fulgham and Isaac Gouy. The computer language benchmarks game, 2009. Available at http://shootout.alioth.debian.org. 4.1

Yannis Bres, Bernard Serpette, and Manuel Serrano. Bigloo.NET: compiling Scheme to .NET CLR. *Journal of Object Technology*, 9(3):71–94, October 2004. 2.3

PUC-Rio - Certificação Digital Nº 0510949/CA

David Broman. Tail call JIT conditions, 2007. Available at `http://blogs.msdn.com/davbr/pages/tail-call-jit-conditions.aspx`. 4.2

Luca Cardelli and Peter Wegner. On understanding types, data abstraction, and polymorphism. *ACM Computing Surveys*, 17(4):471–523, 1985. ISSN 0360-0300. 3.1, 3.1.3

Robert Cartwright and Mike Fagan. Soft typing. In *PLDI '91: Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, pages 278–292, New York, NY, 1991. ACM. 3.3.1

Bill Chiles and Alex Turner. Dynamic language runtime, 2009. Available at `http://dlr.codeplex.com/`. 1, 2.3

William Clinger. Common Larceny. In *Proceedings of the 2005 International Lisp Conference*, pages 101–107, 2005. 2.3

William D. Clinger and Lars Thomas Hansen. Lambda, the ultimate label or a simple optimizing compiler for Scheme. In *LFP '94: Proceedings of the 1994 ACM Conference on LISP and Functional Programming*, pages 128–139, New York, NY, 1994. ACM. 2.3

Antonio Cuni, Davide Ancona, and Armin Rigo. Faster than c#: efficient implementation of dynamic languages on .net. In *ICOOOLPS '09: Proceedings of the 4th workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages and Programming Systems*, pages 26–33, New York, NY, 2009. ACM. 2.3

Luis Damas and Robin Milner. Principal type-schemes for functional programs. In *POPL '82: Proceedings of the 9th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 207–212, New York, NY, 1982. ACM. 3.1, 3.3.1, 3.3.2

Ana Lúcia de Moura, Noemi Rodriguez, and Roberto Ierusalimschy. Coroutines in Lua. *Journal of Universal Computer Science*, 10(7):910–925, 2004. 1

L. Peter Deutsch and Allan M. Schiffman. Efficient implementation of the Smalltalk-80 system. In *POPL '84: Proceedings of the 11th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*, pages 297–302, New York, NY, 1984. ACM. 1, 2.3

Jitendra Dhamija. Introducing managed JScript, 2007. Available at `http://blogs.msdn.com/jscript/archive/2007/05/07/introducing-managed-jscript.aspx`. 2.3

PUC-Rio - Certificação Digital Nº 0510949/CA

ECMA. ECMAScript language specification, 1999. Available at http://www.ecma-international.org/publications/standards/Ecma-262.htm. 1

Daniel P. Friedman, Christopher T. Haynes, and Mitchell Wand. *Essentials of programming languages (2nd ed.).* MIT, Cambridge, MA, 2001. ISBN 0-262-06217-8. 2.1.2

Richard P. Gabriel. *Performance and evaluation of LISP systems.* MIT, Cambridge, MA, 1985. ISBN 0-262-07093-6. 4.1

Richard P. Gabriel and Kent M. Pitman. Endpaper: Technical issues of separation in function cells and value cells. *Lisp and Symbolic Computation*, 1(1):81–101, 1988. 3.3.2

Andy Georges, Dries Buytaert, and Lieven Eeckhout. Statistically rigorous Java performance evaluation. In *OOPSLA '07: Proceedings of the 22nd Annual ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications*, pages 57–76, New York, NY, 2007. ACM. 2

Carsten K. Gomard. Partial type inference for untyped functional programs. In *LFP '90: Proceedings of the 1990 ACM Conference on LISP and Functional Programming*, pages 282–287, New York, NY, 1990. ACM. 3.3.1

James Gosling, Bill Joy, Guy Steele, and Gilad Bracha. *The Java Language Specification.* Addison Wesley, 2005. ISBN 978-0321246783. 2.1.2

Dayong Gu, Clark Verbrugge, and Etienne M. Gagnon. Relative factors in performance analysis of Java virtual machines. In *VEE '06: Proceedings of the 2nd International Conference on Virtual Execution Environments*, pages 111–121, New York, NY, 2006. ACM. 2

Jennifer Hamilton. Language integration in the Common Language Runtime. *SIGPLAN Notices*, 38(2):19–28, 2003. ISSN 0362-1340. 2.3

Mark Hammond. Python for .NET: lessons learned, 2000. Available at `http://word-to-html.com/convertions/Python_for_NET.html`. 2.3

Fritz Henglein. Dynamic typing. In *ESOP'92: Proceedings of the 4th European Symposium on Programming*, pages 233–253, London, UK, 1992a. Springer-Verlag. 3.3.1, 5

Fritz Henglein. Global tagging optimization by type inference. In *LFP '92: Proceedings of the 1992 ACM Conference on LISP and Functional Programming*, pages 205–215, New York, NY, 1992b. ACM. 3.3.1

PUC-Rio - Certificação Digital Nº 0510949/CA

Fritz Henglein and Jakob Rehof. Safe polymorphic type inference for a dynamically typed language: translating Scheme to ML. In *FPCA '95: Proceedings of the Seventh International Conference on Functional Programming Languages and Computer Architecture*, pages 192–203, New York, NY, 1995. ACM. 3.3.1

David Herman, Aaron Tomb, and Cormac Flanagan. Space-efficient gradual typing. In *Proceedings of the Symposium on Trends in Functional Programming*, pages 1–16, April 2007. 5

Urs Hölzle and David Ungar. Optimizing dynamically-dispatched calls with run-time type feedback. In *PLDI '94: Proceedings of the ACM SIGPLAN 1994 Conference on Programming Language Design and Implementation*, pages 326–336, New York, NY, 1994. ACM Press. 1, 2.3

Urs Hölzle, Craig Chambers, and David Ungar. Optimizing dynamically-typed object-oriented languages with polymorphic inline caches. In *ECOOP '91: Proceedings of the European Conference on Object-Oriented Programming*, pages 21–38, London, UK, 1991. Springer-Verlag. 2.3

Jim Hugunin. IronPython: A fast Python implementation for .NET and Mono. In *Proceedings of PyCon DC 2004*, 2004. Available at `http://www.python.org/pycon/dc2004/papers/9`. 2.3

Jim Hugunin. Ironpython 2.0, 2008. Available at `http://ironpython.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=8365`. 2.3

Jim Hugunin. Ironpython 2.6 alpha 1, 2009. Available at `http://ironpython.codeplex.com/Release/ProjectReleases.aspx?ReleaseId=22982`. 2.3

Roberto Ierusalimschy. *Programming in Lua, Second Edition*. Lua.Org, 2006. ISBN 8590379825. 1, 1.1

Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. The implementation of Lua 5.0. *Journal of Universal Computer Science*, 11(7): 1159–1176, 7 2005. 2.1.1

Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. The evolution of Lua. In *HOPL III: Proceedings of the third ACM SIGPLAN Conference on History of Programming Languages*, pages 2–1–2–26, New York, NY, 2007. ACM Press. 1

John Lam. IronRuby, 2009. Available at `http://ironruby.net`. 2.3

PUC-Rio - Certificação Digital Nº 0510949/CA

Xavier Leroy and Pierre Weis. Polymorphic type inference and assignment. In *POPL '91: Proceedings of the 18th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 291–302, New York, NY, 1991. ACM. 3.1.3

Tim Lindholm and Frank Yellin. *The Java Virtual Machine Specification*. Prentice Hall, 1999. ISBN 978-0201432947. 1

Robert A. MacLachlan. The Python compiler for CMU Common Lisp. In *LFP '92: Proceedings of the 1992 ACM Conference on LISP and Functional Programming*, pages 235–246, New York, NY, 1992. ACM. 3.3.2

Dragos Manolescu, Brian Beckman, and Benjamin Livshits. Volta: Developing distributed applications by recompiling. *IEEE Software*, 25(5):53–59, 2008. ISSN 0740-7459. 1

Fabio Mascarenhas and Roberto Ierusalimschy. Running Lua scripts on the CLR through bytecode translation. *Journal of Universal Computer Science*, 11(7):1275–1290, 2005. 2.1.1, 2.3, 5

Fabio Mascarenhas and Roberto Ierusalimschy. Efficient compilation of Lua for the CLR. In *SAC '08: Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 217–221, New York, NY, 2008. ACM. 1, 4.2

Fabio Mascarenhas and Roberto Ierusalimschy. LuaInterface: Scripting the .NET CLR with Lua. *Journal of Universal Computer Science*, 10(7):892–909, 2004. 2.1.1

David Mertz. Charming Python: Functional programming in Python, 2001. Avaliable at `http://www.ibm.com/developerworks/library/l-prog2.html`. 2.1.2

Microsoft. ECMA C# and Common Language Infrastructure standards, 2005. Available at `http://msdn.microsoft.com/net/ecma/`. 1, 1.2, 2

Vance Morrison. Measure early and often for performance. *MSDN Magazine*, 9(4), 2008a. Available at `http://msdn.microsoft.com/en-us/magazine/cc500596.aspx`. 2

Vance Morrison. What's coming in .NET runtime performance in version v3.5 SP1, 2008b. Available at `http://blogs.msdn.com/vancem/archive/2008/05/12/what-s-coming-in-net-runtime-performance-in-version-v3-5-sp1.aspx`. 4.2, 4.2

PUC-Rio - Certificação Digital Nº 0510949/CA

Llewellyn Pritchard. IronScheme, 2009. Available at `http://www.codeplex.com/IronScheme`. 2.3

John C. Reynolds. Definitional interpreters for higher-order programming languages. *Higher Order and Symbolic Computation*, 11(4):363–397, 1998. ISSN 1388-3690. 5

Martin Richards. Richards Benchmark, 1999. Available at `http://www.cl.cam.ac.uk/~mr10/Bench.html`. 4.1

Grant Richins. Tail call improvements in .NET framework 4, 2009. Available at `http://blogs.msdn.com/clrcodegeneration/archive/2009/05/11/tail-call-improvements-in-net-framework-4.aspx`. 4.2

Bernard Paul Serpette and Manuel Serrano. Compiling Scheme to JVM bytecode: a performance study. *SIGPLAN Notices*, 37(9):259–270, 2002. ISSN 0362-1340. 2.3

Manuel Serrano and Marc Feeley. Storage use analysis and its applications. In *ICFP '96: Proceedings of the First ACM SIGPLAN International Conference on Functional Programming*, pages 50–61, New York, NY, 1996. ACM. 2.3, 3.3.2

Manuel Serrano and Pierre Weis. Bigloo: A portable and optimizing compiler for strict functional languages. In *Proceedings of the Static Analysis Symposium*, pages 366–381, 1995. 2.3

Olin Shivers. Control flow analysis in Scheme. In *PLDI '88: Proceedings of the ACM SIGPLAN 1988 Conference on Programming Language Design and Implementation*, pages 164–174, New York, NY, 1988. ACM. 3.3.2

Jeremy Siek and Walid Taha. Gradual typing for objects. In *Proceedings of ECOOP 2007: European Conference on Object-Oriented Programming*, pages 2–27, 2007. 5

Jeremy G. Siek and Walid Taha. Gradual typing for functional languages. In *Proceedings of the Scheme and Functional Programming Workshop*, pages 81–92, September 2006. 5

Jeremy G. Siek and Manish Vachharajani. Gradual typing with unification-based inference. In *DLS '08: Proceedings of the 2008 Symposium on Dynamic languages*, pages 1–12, New York, NY, 2008. ACM. 5

PUC-Rio - Certificação Digital Nº 0510949/CA

Michael Sperber, R. Kent Dybvig, Matthew Flatt, Anton Van Straaten, Richar Kelsey, Willian Clinger, Jonathan Rees, Robert Bruce Findler, and Jacob Matthews. Revised^6 report on the algorithmic language Scheme, 2007. Available at `http://www.r6rs.org`. 2.3

Guy L. Steele, Jr. Rabbit: A compiler for Scheme. Technical report, MIT, Cambridge, MA, 1978. 1.1

Sun Microsystems. Supporting dynamically typed languages on the Java platform, 2008. JSR 292, available at `http://jcp.org/en/jsr/detail?id=292`. 1

Sam Tobin-Hochstadt and Matthias Felleisen. The design and implementation of Typed Scheme. *SIGPLAN Notices*, 43(1):395–406, 2008. ISSN 0362-1340. 5

Alex Turner and Bill Chiles. Sites, binders, and dynamic object interop spec, 2009. Available at `http://dlr.codeplex.com/Project/Download/FileDownload.aspx?DownloadId=68830`. 2.3

James Vastbinder. A .NET Triumvirate: IronScheme, IronLisp, and Xacc, 2008. Available at `http://www.infoq.com/news/2008/01/leppie-ironscheme`. 2.3

Andrew K. Wright and Robert Cartwright. A practical soft type system for scheme. *ACM Transasctions on Programming Languagens and Systems (TOPLAS)*, 19(1):87–152, 1997. ISSN 0164-0925. 3.1, 3.3.1

Andrew K. Wright and Suresh Jagannathan. Polymorphic splitting: an effective polyvariant flow analysis. *ACM Transactions on Programming Languages and Systems*, 20(1):166–207, 1998. ISSN 0164-0925. 3.3.2

Dachuan Yu, Andrew Kennedy, and Don Syme. Formalization of generics for the .NET Common Language Runtime. *SIGPLAN Notices*, 39(1):39–51, 2004. ISSN 0362-1340. 1.2

PUC-Rio - Certificação Digital Nº 0510949/CA

PUC-Rio - Certificação Digital Nº 0510949/CA

A
Operational Semantics

This appendix presents a big-step operational semantics for the simplified Lua language in Figure 3.3 as a series of inference rules for relations $\xrightarrow{s}$ (statements and expression lists), $\xrightarrow{l}$ (lvalues), and $\xrightarrow{e}$ (expressions). Each relation is from a memory $\mathcal{M}$, an environment $\mathcal{E}$ and a term a to another memory $\mathcal{M}'$ and a value v that depends on the relation.

A memory $\mathcal{M}$ is a function $\mathbf{Loc} \to \mathbf{Value} \cup \{\bot\}$. **Loc** is the set of locations. **Value** is the set of Lua values and also the set of output values of the relation $\xrightarrow{e}$. Lua values are numbers, strings, **nil**, **true**, **false**, closures, which are a pair of environment $\mathcal{E}$ and function term f, and tables, which are locations pointing to functions $\mathbf{Value} \to \mathbf{Loc} \cup \{\bot\}$.

The set of output values for relation $\xrightarrow{l}$ is **Loc**, and for relation $\xrightarrow{s}$ it is the set of value tuples $\mathbf{Value}^*$.

An environment $\mathcal{E}$ is a function $\mathbf{Var} \to \mathbf{Loc}$ that maps variables to locations in memory.

The functions index, app, arith and less are primitives that model part of Lua's extensible semantics. The functions index, arith and less take a memory and two values and return a memory and a value, and app takes a memory, a value and a tuple and returns a memory and another tuple. The output memory of these functions is the same as the input memory for all locations that are $\bot$ in the input memory and are not part of any of the input values.

A.1
Semantic Rules

Rule SKIP:

$$\mathcal{M}, \mathcal{E}, \mathbf{skip} \xrightarrow{s} \mathcal{M}, \bot$$

Rule SEQ-RETURN:

$$\frac{\mathcal{M}, \mathcal{E}, s_1 \xrightarrow{s} \mathcal{M}', v \quad v \neq \bot}{\mathcal{M}, \mathcal{E}, s_1; s_2 \xrightarrow{s} \mathcal{M}', v}$$

Rule SEQ:

$$\frac{\mathcal{M}, \mathcal{E}, s_1 \xrightarrow{s} \mathcal{M}', \bot \quad \mathcal{M}', \mathcal{E}, s_2 \xrightarrow{s} \mathcal{M}'', v}{\mathcal{M}', \mathcal{E}, s_1; s_2 \xrightarrow{s} \mathcal{M}'', v}$$

Rule RETURN:

$$\frac{\mathcal{M}, \mathcal{E}, el \xrightarrow{s} \mathcal{M}', v}{\mathcal{M}, \mathcal{E}, \textbf{return}\ el \xrightarrow{s} \mathcal{M}', v}$$

Rule IF-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \textbf{false} \quad \mathcal{M}', \mathcal{E}, s_2 \xrightarrow{s} \mathcal{M}'', v}{\mathcal{M}, \mathcal{E}, \textbf{if}\ e\ \textbf{then}\ s_1\ \textbf{else}\ s_2 \xrightarrow{s} \mathcal{M}'', v}$$

Rule IF-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \textbf{nil} \quad \mathcal{M}', \mathcal{E}, s_2 \xrightarrow{s} \mathcal{M}'', v}{\mathcal{M}, \mathcal{E}, \textbf{if}\ e\ \textbf{then}\ s_1\ \textbf{else}\ s_2 \xrightarrow{s} \mathcal{M}'', v}$$

Rule IF-TRUE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', v \quad \mathcal{M}', \mathcal{E}, s_1 \xrightarrow{s} \mathcal{M}'', u \quad v \neq \textbf{false} \quad v \neq \textbf{nil}}{\mathcal{M}, \mathcal{E}, \textbf{if}\ e\ \textbf{then}\ s_1\ \textbf{else}\ s_2 \xrightarrow{s} \mathcal{M}'', u}$$

Rule WHILE-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \textbf{false}}{\mathcal{M}, \mathcal{E}, \textbf{while}\ e\ \textbf{do}\ s \xrightarrow{s} \mathcal{M}', \bot}$$

Rule WHILE-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \textbf{nil}}{\mathcal{M}, \mathcal{E}, \textbf{while}\ e\ \textbf{do}\ s \xrightarrow{s} \mathcal{M}', \bot}$$

Rule WHILE-RETURN:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', v \quad \mathcal{M}', \mathcal{E}, s \xrightarrow{s} \mathcal{M}'', u \quad v \neq \textbf{false} \quad v \neq \textbf{nil} \quad u \neq \bot}{\mathcal{M}, \mathcal{E}, \textbf{while}\ e\ \textbf{do}\ s \xrightarrow{s} \mathcal{M}'', u}$$

Rule WHILE-TRUE:

$$\frac{\begin{array}{c}\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', v \quad \mathcal{M}', \mathcal{E}, s \xrightarrow{s} \mathcal{M}'', \bot \quad \mathcal{M}'', \mathcal{E}, \textbf{while}\ e\ \textbf{do}\ s \xrightarrow{s} \mathcal{M}''', u \\ v \neq \textbf{false} \quad v \neq \textbf{nil}\end{array}}{\mathcal{M}, \mathcal{E}, \textbf{while}\ e\ \textbf{do}\ s \xrightarrow{s} \mathcal{M}''', u}$$

Rule LOCAL-1:

$$\frac{\mathcal{M}, \mathcal{E}, el \xrightarrow{s} \mathcal{M}', \bot \quad \mathcal{M}'[\vec{m} \mapsto \vec{\textbf{nil}}], \mathcal{E}[\vec{x} \mapsto \vec{m}], s \xrightarrow{s} \mathcal{M}'', v \quad \mathcal{M}'(m_k) = \bot}{\mathcal{M}, \mathcal{E}, \textbf{local}\ \vec{x} = el\ \textbf{in}\ s \xrightarrow{s} \mathcal{M}'', v}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule LOCAL-2:

$$\frac{\mathcal{M}, \mathcal{E}, el \xrightarrow{s} \mathcal{M}', \vec{v} \quad \mathcal{M}'[\vec{m} \mapsto \vec{v_a}], \mathcal{E}[\vec{x} \mapsto \vec{m}], s \xrightarrow{s} \mathcal{M}'', u \quad \mathcal{M}'(m_k) = \bot}{\mathcal{M}, \mathcal{E}, \textbf{local}\ \vec{x} = el\ \textbf{in}\ s \xrightarrow{s} \mathcal{M}'', v}$$

where $v_{a_k} = v_k$ if $k \leq |\vec{v}|$ and $v_{a_k} = \textbf{nil}$ otherwise and $|\vec{m}| = |\vec{x}|$.

Rule ASSIGN-1:

$$\frac{\mathcal{M}, \mathcal{E}, l_1 \xrightarrow{l} \mathcal{M}_1, m_1 \ldots \mathcal{M}_{k-1}, \mathcal{E}, l_k \xrightarrow{l} \mathcal{M}', m_k \quad \mathcal{M}', \mathcal{E}, el \xrightarrow{s} \mathcal{M}'', \bot}{\mathcal{M}, \mathcal{E}, \vec{l} = el \xrightarrow{s} \mathcal{M}''[\vec{m} \mapsto \vec{\textbf{nil}}], \bot}$$

Rule ASSIGN-2:

$$\frac{\mathcal{M}, \mathcal{E}, l_1 \xrightarrow{l} \mathcal{M}_1, m_1 \ldots \mathcal{M}_{k-1}, \mathcal{E}, l_k \xrightarrow{l} \mathcal{M}', m_k \quad \mathcal{M}', \mathcal{E}, el \xrightarrow{s} \mathcal{M}'', \vec{v}}{\mathcal{M}, \mathcal{E}, \vec{l} = el \xrightarrow{s} \mathcal{M}''[\vec{m} \mapsto \vec{v_a}], \bot}$$

where $v_{a_k} = v_k$ if $k \leq |\vec{v}|$ and $v_{a_k} = \textbf{nil}$ otherwise and $|\vec{m}| = |\vec{l}|$.

Rule APP-STAT:

$$\frac{\mathcal{M}, \mathcal{E}, e(el)_1 \xrightarrow{e} \mathcal{M}', v}{\mathcal{M}, \mathcal{E}, e(el)_0 \xrightarrow{s} \mathcal{M}', \bot}$$

Rule VAR-LVAL:

$$\mathcal{M}, \mathcal{E}, x \xrightarrow{s} \mathcal{E}(x)$$

Rule VAR-RVAL:

$$\mathcal{M}, \mathcal{E}, x \xrightarrow{e} \mathcal{M}(\mathcal{E}(x))$$

Rule CONS:

$$\mathcal{M}, \mathcal{E}, \{\} \xrightarrow{e} \mathcal{M}[t \mapsto \lambda x.\bot], t \quad \mathcal{M}(t) = \bot$$

Rule TAB-LVAL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', t \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v \quad t \in \textbf{Table} \quad \mathcal{M}''(t)(v) \neq \bot}{\mathcal{M}, \mathcal{E}, e_1[e_2] \xrightarrow{s} \mathcal{M}'', \mathcal{M}''(t)(v)}$$

Rule TAB-LVAL-NEW:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', t \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v \quad t \in \textbf{Table} \quad \mathcal{M}''(t)(v) = \bot}{\mathcal{M}, \mathcal{E}, e_1[e_2] \xrightarrow{s} \mathcal{M}''[t \mapsto \mathcal{M}''(t)[v \mapsto l], l \quad \mathcal{M}''(l) = \bot}$$

Rule TAB-RVAL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', t \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v \quad t \in \textbf{Table} \quad \mathcal{M}''(t)(v) \neq \bot}{\mathcal{M}, \mathcal{E}, e_1[e_2] \xrightarrow{s} \mathcal{M}'', \mathcal{M}''(\mathcal{M}''(t)(v))}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule TAB-RVAL-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', t \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v \quad t \in \textbf{Table} \quad \mathcal{M}''(t)(v) = \bot}{\mathcal{M}, \mathcal{E}, e_1[e_2] \xrightarrow{s} \mathcal{M}'', \textbf{nil}}$$

Rule TAB-META:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \notin \textbf{Table}}{\mathcal{M}, \mathcal{E}, e_1[e_2] \xrightarrow{e} \text{index}(\mathcal{M}'', v_1, v_2)}$$

Rule EL-EMPTY:

$$\mathcal{M}, \mathcal{E}, \textbf{nothing} \xrightarrow{s} \mathcal{M}, \bot$$

Rule EL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}_1, v_1 \ldots \mathcal{M}_{k-1}, \mathcal{E}, e_k \xrightarrow{e} \mathcal{M}', v_k \quad k = |\vec{v}|}{\mathcal{M}, \mathcal{E}, \vec{e} \xrightarrow{s} \mathcal{M}', \vec{v}}$$

Rule EL-MEXP:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}_1, v_1 \ldots \mathcal{M}_{k-1}, \mathcal{E}, e_k \xrightarrow{e} \mathcal{M}', v_k \quad \mathcal{M}', \mathcal{E}, me \xrightarrow{s} \mathcal{M}'', u \quad k = |\vec{v}|}{\mathcal{M}, \mathcal{E}, \vec{e}, me \xrightarrow{s} \mathcal{M}'', \vec{v}u}$$

Rule APP-CLOSURE:

$$\frac{\begin{array}{c}\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \langle \mathcal{E}', \textbf{fun}(\vec{x})\ b \rangle \quad \mathcal{M}', \mathcal{E}, el \xrightarrow{s} \mathcal{M}'', \vec{v} \\ \mathcal{M}''[\vec{m} \mapsto \vec{v_a}], \mathcal{E}'[\vec{x} \mapsto \vec{m}], b \xrightarrow{s} \mathcal{M}''', u \quad \mathcal{M}''(m_k) = \bot\end{array}}{\mathcal{M}, \mathcal{E}, e(el)_n \xrightarrow{s} \mathcal{M}''', u}$$

where $v_{a_k} = v_k$ if $k \leq |\vec{v}|$ and $v_{a_k} = \textbf{nil}$ otherwise and $|\vec{m}| = |\vec{x}|$.

Rule APP-META:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, el \xrightarrow{s} \mathcal{M}'', v_2 \quad v_1 \notin \textbf{Closure}}{\mathcal{M}, \mathcal{E}, e(el)_n \xrightarrow{s} \text{app}(\mathcal{M}'', v_1, v_2)}$$

Rule APP-FIRST-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e(el)_n \xrightarrow{s} \mathcal{M}', \bot}{\mathcal{M}, \mathcal{E}, e(el)_1 \xrightarrow{e} \mathcal{M}', \textbf{nil}}$$

Rule APP-FIRST:

$$\frac{\mathcal{M}, \mathcal{E}, e(el)_n \xrightarrow{s} \mathcal{M}', \vec{v}}{\mathcal{M}, \mathcal{E}, e(el)_1 \xrightarrow{e} \mathcal{M}', v_1}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule ARITH:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \in \mathbf{Number} \quad v_2 \in \mathbf{Number}}{\mathcal{M}, \mathcal{E}, e_1 \oplus e_2 \xrightarrow{e} \mathcal{M}'', \mathcal{E}, v1 \oplus v_2}$$

Rule ARITH-META:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \notin \mathbf{Number} \vee v_2 \notin \mathbf{Number}}{\mathcal{M}, \mathcal{E}, e_1 \oplus e_2 \xrightarrow{e} \mathrm{arith}(\mathcal{M}'', v1, v_2)}$$

Rule EQ-TRUE:

$$\frac{\mathcal{M}, env, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} v_2 \quad v_1 = v_2}{\mathcal{M}, \mathcal{E}, e_1 == e_2 \xrightarrow{e} \mathcal{M}'', \mathbf{true}}$$

Rule EQ-FALSE:

$$\frac{\mathcal{M}, env, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} v_2 \quad v_1 \neq v_2}{\mathcal{M}, \mathcal{E}, e_1 == e_2 \xrightarrow{e} \mathcal{M}'', \mathbf{false}}$$

Rule LESS-TRUE:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \in \mathbf{Number} \quad v_2 \in \mathbf{Number} \quad v_1 < v_2}{\mathcal{M}, \mathcal{E}, e_1 \oplus e_2 \xrightarrow{e} \mathcal{M}'', \mathcal{E}, \mathbf{true}}$$

Rule LESS-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \in \mathbf{Number} \quad v_2 \in \mathbf{Number} \quad v_1 \not< v_2}{\mathcal{M}, \mathcal{E}, e_1 \oplus e_2 \xrightarrow{e} \mathcal{M}'', \mathcal{E}, \mathbf{false}}$$

Rule LESS-META:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \notin \mathbf{Number} \vee v_2 \notin \mathbf{Number}}{\mathcal{M}, \mathcal{E}, e_1 \oplus e_2 \xrightarrow{e} \mathrm{less}(\mathcal{M}'', \mathcal{E}, v1 \oplus v_2)}$$

Rule AND-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad v_1 = \mathbf{nil}}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{and}\ e_2 \xrightarrow{e} \mathcal{M}', \mathbf{nil}}$$

Rule AND-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad v_1 = \mathbf{false}}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{and}\ e_2 \xrightarrow{e} \mathcal{M}', \mathbf{false}}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule AND:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2 \quad v_1 \neq \mathbf{false} \quad v_1 \neq \mathbf{nil}}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{and} \ e_2 \xrightarrow{e} \mathcal{M}'', v2}$$

Rule OR-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', \mathbf{nil} \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{or} \ e_2 \xrightarrow{e} \mathcal{M}'', v2}$$

Rule OR-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', \mathbf{false} \quad \mathcal{M}', \mathcal{E}, e_2 \xrightarrow{e} \mathcal{M}'', v_2}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{or} \ e_2 \xrightarrow{e} \mathcal{M}'', v2}$$

Rule OR:

$$\frac{\mathcal{M}, \mathcal{E}, e_1 \xrightarrow{e} \mathcal{M}', v_1 \quad v_1 \neq \mathbf{false} \quad v_1 \neq \mathbf{nil}}{\mathcal{M}, \mathcal{E}, e_1 \ \mathbf{or} \ e_2 \xrightarrow{e} \mathcal{M}', v1}$$

Rule NOT-NIL:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \mathbf{nil}}{\mathcal{M}, \mathcal{E}, \mathbf{not} \ e \xrightarrow{e} \mathcal{M}', \mathbf{true}}$$

Rule NOT-FALSE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', \mathbf{false}}{\mathcal{M}, \mathcal{E}, \mathbf{not} \ e \xrightarrow{e} \mathcal{M}', \mathbf{true}}$$

Rule NOT-TRUE:

$$\frac{\mathcal{M}, \mathcal{E}, e \xrightarrow{e} \mathcal{M}', v \quad v \neq \mathbf{false} \quad v \neq \mathbf{nil}}{\mathcal{M}, \mathcal{E}, \mathbf{not} \ e \xrightarrow{e} \mathcal{M}', \mathbf{false}}$$

Rule FUN:

$$\mathcal{M}, \mathcal{E}, \mathbf{fun}(\vec{x}) \ b \xrightarrow{e} \mathcal{M}, \langle \mathcal{E}, \mathbf{fun}(\vec{x}) \ b \rangle$$

PUC-Rio - Certificação Digital Nº 0510949/CA

B
Typing Rules

This appendix presents the full set of typing rules, including those already mentioned in Section 3.1.4.

Rule SKIP:

$$\Gamma \vdash \textbf{skip} : \textbf{void}$$

Rule SEQ-VOID:

$$\frac{\Gamma \vdash s_1 : \textbf{void} \quad \Gamma \vdash s_2 : \textbf{void}}{\Gamma \vdash s_1; s_2 : \textbf{void}}$$

Rule SEQ-1:

$$\frac{\Gamma \vdash s_1 : \tau \quad \Gamma \vdash s_2 : \textbf{void}}{\Gamma \vdash s_1; s_2 : \tau}$$

Rule SEQ-2:

$$\frac{\Gamma \vdash s_1 : \textbf{void} \quad \Gamma \vdash s_2 : \tau}{\Gamma \vdash s_1; s_2 : \tau}$$

Rule SEQ-BOTH:

$$\frac{\Gamma \vdash s_1 : \tau_1 \quad \Gamma \vdash s_2 : \tau_2 \quad \tau_1 \rightsquigarrow \upsilon \quad \tau_2 \rightsquigarrow \upsilon}{\Gamma \vdash s_1; s_2 : \upsilon}$$

Rule IF-VOID:

$$\frac{\Gamma \vdash e : \tau_e \quad \Gamma \vdash s_1 : \textbf{void} \quad \Gamma \vdash s_2 : \textbf{void}}{\Gamma \vdash \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2 : \textbf{void}}$$

Rule IF-1:

$$\frac{\Gamma \vdash e : \tau_e \quad \Gamma \vdash s_1 : \tau \quad \Gamma \vdash s_2 : \textbf{void}}{\Gamma \vdash \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2 : \tau}$$

Rule IF-2:

$$\frac{\Gamma \vdash e : \tau_e \quad \Gamma \vdash s_1 : \textbf{void} \quad \Gamma \vdash s_2 : \tau}{\Gamma \vdash \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2 : \tau}$$

Rule IF-BOTH:

$$\frac{\Gamma \vdash e : \tau_e \quad \Gamma \vdash s_1 : \tau_1 \quad \Gamma \vdash s_2 : \tau_2 \quad \tau_1 \rightsquigarrow \upsilon \quad \tau_2 \rightsquigarrow \upsilon}{\Gamma \vdash \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2 : \upsilon}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule WHILE-VOID:

$$\frac{\Gamma \vdash e\colon \tau \quad \Gamma \vdash s\colon \mathbf{void}}{\Gamma \vdash \mathbf{while}\ e\ \mathbf{do}\ s\colon \mathbf{void}}$$

Rule WHILE:

$$\frac{\Gamma \vdash e\colon \tau \quad \Gamma \vdash s\colon \upsilon}{\Gamma \vdash \mathbf{while}\ e\ \mathbf{do}\ s\colon \upsilon}$$

Rule RETURN:

$$\frac{\Gamma \vdash el\colon \tau \quad \tau \rightsquigarrow \upsilon}{\Gamma \vdash \mathbf{return}\ el\colon \upsilon}$$

Rule LOCAL-DROP-VOID:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \mathbf{void} \quad m \geq |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \mathbf{void}}$$

Rule LOCAL-DROP:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \omega \quad m \geq |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \omega}$$

Rule LOCAL-FILL-VOID:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \mathbf{void} \quad m < |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k \quad \mathbf{nil} \rightsquigarrow \tau_l \quad l > m}{\Gamma \vdash \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \mathbf{void}}$$

Rule LOCAL-FILL:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \omega \quad m < |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k \quad \mathbf{nil} \rightsquigarrow \tau_l \quad l > m}{\Gamma \vdash \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \omega}$$

Rule LOCAL-VAR-DROP-VOID:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \mathbf{void} \quad m \geq |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \mathbf{void}}$$

Rule LOCAL-VAR-DROP:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \omega \quad m \geq |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \omega}$$

Rule LOCAL-VAR-FILL-VOID:

$$\frac{\begin{array}{c}\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \mathbf{void} \quad m < |\vec{x}| \\ \upsilon_k \rightsquigarrow \tau_k \quad \tau_l = \mathcal{D} \quad l > m\end{array}}{\Gamma \vdash \mathbf{local}\ \vec{x} = el\ \mathbf{in}\ s\colon \mathbf{void}}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule LOCAL-VAR-FILL:

$$\frac{\Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad \Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s : \omega \quad m < |\vec{x}| \quad \upsilon_k \rightsquigarrow \tau_k \quad \tau_l = \mathcal{D} \quad l > m}{\Gamma \vdash \textbf{local}\ \vec{x} = el\ \textbf{in}\ s : \omega}$$

Rule ASSIGN-DROP:

$$\frac{\Gamma \vdash l_k : \tau_k \quad \Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad m \geq |\vec{l}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash \vec{l} = el : \textbf{void}}$$

Rule ASSIGN-FILL:

$$\frac{\Gamma \vdash l_k : \tau_k \quad \Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \quad m < |\vec{l}| \quad \upsilon_k \rightsquigarrow \tau_k \quad \textbf{nil} \rightsquigarrow \tau_l \quad l > m}{\Gamma \vdash \vec{l} = el : \textbf{void}}$$

Rule ASSIGN-VAR-DROP:

$$\frac{\Gamma \vdash l_k : \tau_k \quad \Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad m \geq |\vec{l}| \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash \vec{l} = el : \textbf{void}}$$

Rule ASSIGN-VAR-FILL:

$$\frac{\Gamma \vdash l_k : \tau_k \quad \Gamma \vdash el : \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad m < |\vec{l}| \quad \upsilon_k \rightsquigarrow \tau_k \quad \tau_l = \mathcal{D} \quad l > m}{\Gamma \vdash \vec{l} = el : \textbf{void}}$$

Rule EL-EMPTY:

$$\frac{}{\Gamma \vdash \textbf{nothing} : \textbf{empty}}$$

Rule EL:

$$\frac{\Gamma \vdash e_k : \tau_k \quad n = |\vec{e}|}{\Gamma \vdash \vec{e} : \tau_1 \times \ldots \times \tau_n}$$

Rule EL-MEXP-EMPTY:

$$\frac{\Gamma \vdash e_k : \tau_k \quad \Gamma \vdash me : \textbf{empty} \quad n = |\vec{e}|}{\Gamma \vdash \vec{e}, me : \tau_1 \times \ldots \times \tau_n}$$

Rule EL-MEXP:

$$\frac{\Gamma \vdash e_k : \tau_k \quad \Gamma \vdash me : \upsilon_1 \times \ldots \times \upsilon_m \quad n = |\vec{e}|}{\Gamma \vdash \vec{e}, me : \tau_1 \times \ldots \times \tau_n \times \upsilon_1 \times \ldots \times \upsilon_m}$$

Rule EL-VAR-1:

$$\frac{\Gamma \vdash e_k : \tau_k \quad \Gamma \vdash me : \mathcal{D}^* \quad n = |\vec{e}|}{\Gamma \vdash \vec{e}, me : \tau_1 \times \ldots \times \tau_n \times \mathcal{D}^*}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule EL-VAR-2:

$$\frac{\Gamma \vdash e_k : \tau_k \quad \Gamma \vdash me\colon \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad n = |\vec{e}|}{\Gamma \vdash \vec{e}, me : \tau_1 \times \ldots \times \tau_n \times \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^*}$$

Rule APP-DROP:

$$\frac{\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \rightarrow \sigma \quad \Gamma \vdash el\colon \upsilon_1 \times \ldots \times \upsilon_m \quad m \geq n \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash f(el)_n : \sigma}$$

Rule APP-FILL:

$$\frac{\begin{array}{c}\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \rightarrow \sigma \\ \Gamma \vdash el\colon \upsilon_1 \times \ldots \times \upsilon_m \quad m < n \quad \upsilon_k \rightsquigarrow \tau_k \quad \mathbf{nil} \rightsquigarrow \tau_l \quad l > m\end{array}}{\Gamma \vdash f(el)_n : \sigma}$$

Rule APP-VAR-DROP:

$$\frac{\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \rightarrow \sigma \quad \Gamma \vdash el\colon \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad m \geq n \quad \upsilon_k \rightsquigarrow \tau_k}{\Gamma \vdash f(el)_n : \sigma}$$

Rule APP-VAR-FILL:

$$\frac{\begin{array}{c}\Gamma \vdash f : \tau_1 \times \ldots \times \tau_n \rightarrow \sigma \\ \Gamma \vdash el\colon \upsilon_1 \times \ldots \times \upsilon_m \times \mathcal{D}^* \quad m < n \quad \upsilon_k \rightsquigarrow \tau_k \quad \tau_l = \mathcal{D} \quad l > m\end{array}}{\Gamma \vdash f(el)_n : \sigma}$$

Rule APP-DYN:

$$\frac{\Gamma \vdash f : \tau \quad \Gamma \vdash el\colon \upsilon \quad \tau \rightsquigarrow \mathcal{D} \quad \upsilon \rightsquigarrow \mathcal{D}^*}{\Gamma \vdash f(el)_n : \mathcal{D}^*}$$

Rule APP-STAT:

$$\frac{\Gamma \vdash e(el)_n\colon \tau}{\Gamma \vdash e(el)_0\colon \mathbf{void}}$$

Rule APP-FIRST:

$$\frac{\Gamma \vdash e(el)_n\colon \tau_1 \times \ldots \times \tau_n}{\Gamma \vdash e(el)_1\colon \tau_1}$$

Rule APP-FIRST-NIL:

$$\frac{\Gamma \vdash e(el)_n\colon \mathbf{empty}}{\Gamma \vdash e(el)_1\colon \mathbf{nil}}$$

Rule APP-FIRST-DYN:

$$\frac{\Gamma \vdash e(el)_n\colon \mathcal{D}^*}{\Gamma \vdash e(el)_1\colon \mathcal{D}}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule CONS:

$$\frac{\forall i,j,\sigma.(i \neq j \wedge \sigma \leadsto \tau_i) \rightarrow \sigma \not\leadsto \tau_j \quad \mathbf{nil} \leadsto \upsilon_k}{\Gamma \vdash \{\} : \tau_1 \mapsto \upsilon_1 \wedge \ldots \wedge \tau_n \mapsto \upsilon_n}$$

Rule CONS-DYN:

$$\frac{\mathbf{nil} \leadsto \upsilon}{\Gamma \vdash \{\} : \mathcal{D} \mapsto \upsilon}$$

Rule INDEX:

$$\frac{\Gamma \vdash e_1 : \tau_1 \mapsto \upsilon_1 \wedge \ldots \wedge \tau_n \mapsto \upsilon_n \quad \Gamma \vdash e_2 : \sigma \quad \sigma \leadsto \tau_k}{\Gamma \vdash e_1[e_2] : \upsilon_k}$$

Rule INDEX-DYN:

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \upsilon \quad \tau \leadsto \mathcal{D} \quad \upsilon \leadsto \mathcal{D}}{\Gamma \vdash e_1[e_2] : \mathcal{D}}$$

Rule ARITH-NUM:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \leadsto \mathbf{Number} \quad \tau_2 \leadsto \mathbf{Number}}{\Gamma \vdash e_1 \oplus e_2 : \mathbf{Number}}$$

Rule ARITH-DYN:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \not\leadsto \mathbf{Number} \vee \tau_2 \not\leadsto \mathbf{Number} \quad \tau_1 \leadsto \mathcal{D} \quad \tau_2 \leadsto \mathcal{D}}{\Gamma \vdash e_1 \oplus e_2 : \mathcal{D}}$$

Rule EQ:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \leadsto \tau_2 \vee \tau_2 \leadsto \tau_1}{\Gamma \vdash e_1 == e_2 : \mathbf{Bool}}$$

Rule LESS-NUM:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \leadsto \mathbf{Number} \quad \tau_2 \leadsto \mathbf{Number}}{\Gamma \vdash e_1 < e_2 : \mathbf{Bool}}$$

Rule LESS-DYN:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \not\leadsto \mathbf{Number} \vee \tau_2 \not\leadsto \mathbf{Number} \quad \tau_1 \leadsto \mathcal{D} \quad \tau_2 \leadsto \mathcal{D}}{\Gamma \vdash e_1 < e_2 : \mathcal{D}}$$

Rule AND-NIL:

$$\frac{\Gamma \vdash e_1 : \mathbf{nil}}{\Gamma \vdash e_1 \ \mathbf{and} \ e_2 : \mathbf{nil}}$$

Rule AND-FALSE:

$$\frac{\Gamma \vdash e_1 : \mathbf{false}}{\Gamma \vdash e_1 \ \mathbf{and} \ e_2 : \mathbf{false}}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

Rule AND-TRUE:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_1 : \tau_2 \quad \tau_1 \neq \mathbf{false} \quad \mathbf{nil} \not\leadsto \tau_1}{\Gamma \vdash e_1 \ \mathbf{and}\ e_2 : \tau_2}$$

Rule AND:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_1 : \tau_2 \quad \tau_1 \neq \mathbf{nil} \quad \mathbf{nil} \leadsto \tau_1 \quad \tau_1 \leadsto \upsilon \quad \tau_2 \leadsto \upsilon}{\Gamma \vdash e_1 \ \mathbf{and}\ e_2 : \upsilon}$$

Rule OR-NIL:

$$\frac{\Gamma \vdash e_1 : \mathbf{nil} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \ \mathbf{or}\ e_2 : \tau}$$

Rule OR-FALSE:

$$\frac{\Gamma \vdash e_1 : \mathbf{false} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \ \mathbf{or}\ e_2 : \tau}$$

Rule OR-TRUE:

$$\frac{\Gamma \vdash e_1 : \tau \quad \tau \neq \mathbf{false} \quad \mathbf{nil} \not\leadsto \tau}{\Gamma \vdash e_1 \ \mathbf{or}\ e_2 : \tau}$$

Rule OR:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \tau_1 \neq \mathbf{nil} \quad \mathbf{nil} \leadsto \tau_1 \quad \tau_1 \leadsto \upsilon \quad \tau_2 \leadsto \upsilon}{\Gamma \vdash e_1 \ \mathbf{or}\ e_2 : \upsilon}$$

Rule NOT-NIL:

$$\frac{\Gamma \vdash e : \mathbf{nil}}{\Gamma \vdash \mathbf{not}\ e : \mathbf{true}}$$

Rule NOT-FALSE:

$$\frac{\Gamma \vdash e : \mathbf{false}}{\Gamma \vdash \mathbf{not}\ e : \mathbf{true}}$$

Rule NOT-TRUE:

$$\frac{\Gamma \vdash e : t \quad \tau \neq \mathbf{false} \quad \mathbf{nil} \not\leadsto \tau}{\Gamma \vdash \mathbf{not}\ e : \mathbf{false}}$$

Rule NOT:

$$\frac{\Gamma \vdash e : t \quad \tau \neq \mathbf{nil} \quad \mathbf{nil} \leadsto \tau}{\Gamma \vdash \mathbf{not}\ e : \mathbf{Bool}}$$

Rule FUN-EMPTY:

$$\frac{\Gamma \vdash s; \mathbf{return}\ el : \upsilon}{\Gamma \vdash \mathbf{fun}()\ s; \mathbf{return}\ el : \tau_1 \times \ldots \times \tau_n \rightarrow \upsilon}$$

Rule FUN:

$$\frac{\Gamma[\vec{x} \mapsto \vec{\tau}] \vdash s; \mathbf{return}\ el : \upsilon}{\Gamma \vdash \mathbf{fun}(\vec{x})\ s; \mathbf{return}\ el : \tau_1 \times \ldots \times \tau_n \rightarrow \upsilon \quad n \geq |\vec{x}|}$$

PUC-Rio - Certificação Digital Nº 0510949/CA

C
Collected Benchmark Results

	base	single	box	intern	prop	infer
binary-trees	2.140	1.987	2.256	1.143	0.857	0.193
fannkuch	1.825	1.835	3.315	3.397	1.684	0.829
fib-iter	0.567	0.504	1.780	1.824	0.130	0.062
fib-memo	0.468	0.230	0.318	0.340	0.351	0.147
fib-rec	1.806	0.540	1.311	1.297	1.304	0.773
mandelbrot	1.632	1.588	5.608	5.551	0.146	0.145
n-body	2.656	2.516	4.169	3.123	3.084	0.853
n-sieve	1.901	1.887	3.115	3.138	1.352	0.296
nsieve-bits	0.883	0.567	1.377	1.403	1.028	0.574
partial-sum	0.647	0.493	1.152	1.184	0.680	0.407
recursive	1.639	0.450	1.418	1.425	1.488	0.110
spectral-norm	1.894	1.186	3.212	3.245	3.175	0.210
richards	0.343	0.308	0.329	0.217	0.202	0.047
richards-tail	N/A	N/A	N/A	N/A	N/A	N/A
richards-oo	0.383	0.304	0.327	0.227	0.197	0.052
richards-oo-tail	N/A	N/A	N/A	N/A	N/A	N/A
richards-oo-meta	0.691	0.634	0.463	0.315	0.274	0.242
richards-oo-cache	0.493	0.352	0.370	0.241	0.216	0.186

Table C.1: Benchmark running times for Mono 2.4, in seconds

PUC-Rio - Certificação Digital Nº 0510949/CA

	base	single	box	intern	prop	infer
binary-trees	0.686	0.671	0.686	0.269	0.218	0.031
fannkuch	1.373	1.392	1.267	1.312	0.895	0.696
fib-iter	0.189	0.189	0.236	0.232	0.037	0.029
fib-memo	0.183	0.156	0.154	0.176	0.166	0.023
fib-rec	0.920	0.644	0.287	0.265	0.265	0.131
mandelbrot	1.102	1.115	0.858	0.985	0.037	0.041
n-body	1.952	1.798	1.817	1.277	1.197	0.191
n-sieve	1.336	1.326	1.244	1.258	0.996	0.205
n-sieve-bits	0.365	0.343	0.359	0.363	0.296	0.199
partial-sum	0.314	0.291	0.283	0.269	0.168	0.158
recursive	0.712	0.511	0.253	0.242	0.222	0.047
spectral-norm	1.178	1.125	0.840	0.850	0.825	0.162
richards	0.240	0.230	0.211	0.156	0.148	0.055
richards-tail	0.255	0.234	0.215	0.160	0.150	0.064
richards-oo	0.255	0.218	0.201	0.158	0.127	0.055
richards-oo-tail	N/A	N/A	1.094	0.827	0.628	0.060
richards-oo-meta	0.425	0.398	0.298	0.211	0.201	0.179
richards-oo-cache	0.281	0.252	0.240	0.162	0.156	0.152

Table C.2: Benchmark running times for .NET 3.5 SP1, in seconds

	base	single	box	intern	prop	infer
binary-trees	0.691	0.690	0.597	0.252	0.203	0.035
fannkuch	1.182	1.406	1.196	1.301	0.895	0.671
fib-iter	0.204	0.218	0.243	0.268	0.044	0.034
fib-memo	0.178	0.165	0.150	0.172	0.173	0.029
fib-rec	0.968	0.665	0.318	0.297	0.296	0.138
mandelbrot	1.093	1.120	1.007	0.998	0.042	0.045
n-body	1.747	1.986	1.658	1.303	1.190	0.183
n-sieve	1.437	1.450	1.292	1.259	0.983	0.206
nsieve-bits	0.346	0.356	0.354	0.381	0.318	0.193
partial-sum	0.331	0.304	0.281	0.283	0.180	0.161
recursive	0.735	0.525	0.277	0.266	0.258	0.052
spectral-norm	1.152	1.131	0.854	0.873	0.848	0.107
richards	0.280	0.259	0.217	0.159	0.152	0.050
richards-tail	0.278	0.261	0.214	0.162	0.156	0.063
richards-oo	0.279	0.263	0.201	0.158	0.140	0.054
richards-oo-tail	N/A	N/A	1.212	0.690	0.394	0.055
richards-oo-meta	0.470	0.460	0.291	0.207	0.202	0.190
richards-oo-cache	0.308	0.283	0.228	0.169	0.162	0.152

Table C.3: Benchmark running times for .NET 4.0 Beta 1, in seconds

PUC-Rio - Certificação Digital Nº 0510949/CA

PUC-Rio - Certificação Digital Nº 0510949/CA

	lua	luajit
binary-trees	0.259	0.184
fannkuch	1.228	0.707
fib-iter	0.499	0.083
fib-memo	0.264	0.109
fib-rec	0.835	0.170
mandelbrot	0.503	0.108
n-body	0.680	0.347
n-sieve	0.655	0.532
n-sieve-bits	0.318	0.134
partial-sum	0.282	0.136
recursive	0.704	0.128
spectra-lnorm	0.705	0.269
richards	0.133	0.054
richards-tail	0.137	0.059
richards-oo	0.116	0.052
richards-oo-tail	0.128	0.056
richards-oo-meta	0.140	0.068
richards-oo-cache	0.138	0.062

Table C.4: Benchmark running times for Lua 5.1.4 and LuaJIT 1.1.5, in seconds

binary-trees	0.137
fannkuch	1.983
fib-iter	0.553
fib-memo	0.102
fib-rec	0.706
mandelbrot	1.061
n-body	1.581
n-sieve	0.753
n-sieve-bits	0.626
partial-sum	0.402
recursive	0.476
spectral-norm	1.838
richards	0.676

Table C.5: Benchmark running times for IronPython 2.0, in seconds