Bibliography

[AC96] Martin Abadi and Luca Cardelli. *A Theory of Objects*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1996. 4.2

[ACF+13] Esteban Allende, Oscar Callaú, Johan Fabry, Éric Tanter, and Marcus Denker. Gradual typing for Smalltalk. *Science of Computer Programming*, August 2013. 2.3, 6.2

[ACPP89] Martin Abadi, Luca Cardelli, Benjamin Pierce, and Gordon Plotkin. Dynamic typing in a statically-typed language. In *Proceedings of the 16th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '89, pages 213–227, New York, NY, USA, 1989. ACM. 2.1, 2.3

[AD03] Christopher Anderson and Sophia Drossopoulou. BabyJ: From object based to class based programming via types. *Electronic Notes in Theoretical Computer Science*, 82(8):53–81, 2003. 2.1, 2.4

[AFSW11] Amal Ahmed, Robert Bruce Findler, Jeremy G. Siek, and Philip Wadler. Blame for all. In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, pages 201–214, New York, NY, USA, 2011. ACM. 2.3

[AFT13] Esteban Allende, Johan Fabry, and Éric Tanter. Cast insertion strategies for gradually-typed objects. In *Proceedings of the 9th Symposium on Dynamic Languages*, DLS '13, pages 27–36, New York, NY, USA, 2013. ACM. 2.3, 3, 6.2

[BAT14] Gavin Bierman, Martín Abadi, and Mads Torgersen. Understanding TypeScript. In *ECOOP 2014–Object-Oriented Programming*, pages 257–281. Springer, 2014. 6.2

[BBH+13] Andrew P. Black, Kim B. Bruce, Michael Homer, James Noble, Amy Ruskin, and Richard Yannow. Seeking Grace: A new object-oriented language for novices. In *Proceedings of the 44th ACM*

Technical Symposium on Computer Science Education, SIGCSE '13, pages 129–134, New York, NY, USA, 2013. ACM. 2.3, 6.2

[BBHN12] Andrew P. Black, Kim B. Bruce, Michael Homer, and James Noble. Grace: The absence of (inessential) difficulty. In *Proceedings of the ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*, Onward! '12, pages 85–98, New York, NY, USA, 2012. ACM. 2.3

[BG93] Gilad Bracha and David Griswold. Strongtalk: Typechecking Smalltalk in a production environment. In *Proceedings of the Eighth Annual Conference on Object-Oriented Programming Systems, Languages, and Applications*, OOPSLA '93, pages 215–230, New York, NY, USA, 1993. ACM. 2.2

[Bra96] Gilad Bracha. The Strongtalk type system for Smalltalk. In *Proceedings of the Workshop on Extending the Smalltalk Language*, OOPSLA, 1996. 2.2

[Bra04] Gilad Bracha. Pluggable type systems. In *Proceedings of the Workshop on Revival of Dynamic Languages*, OOPSLA, October 2004. 1, 2.2

[BS12] Ambrose Bonnaire-Sergeant. A practical optional type system for Clojure. Master's thesis, University of Western Australia, 2012. 2.4, 6.2

[Car84] Luca Cardelli. A semantics of multiple inheritance. In *Proceedings of the International Symposium on Semantics of Data Types*, pages 51–67, New York, NY, USA, 1984. Springer-Verlag New York, Inc. 4.2

[Car86] Luca Cardelli. Amber. In *Proceedings of the Thirteenth Spring School of the LITP on Combinators and Functional Programming Languages*, pages 21–47, London, UK, UK, 1986. Springer-Verlag. 4.2

[CF91] Robert Cartwright and Mike Fagan. Soft typing. In *Proceedings of the ACM SIGPLAN 1991 Conference on Programming Language Design and Implementation*, PLDI '91, pages 278–292, New York, NY, USA, 1991. ACM. 2.1

[Cha14] Pierre Chapuis. Lua HTTP Digest. `https://github.com/catwell/lua-http-digest`, 2014. [Accessed February 2015]. 5

[Cla14] Kevin Clancy. Lua Analyzer. `https://bitbucket.org/kevinclancy/lua-analyzer/`, 2014. [Accessed February 2015]. 6.1

[Cor11] Leaf Corcoran. MoonScript: A programmer friendly language that compiles to Lua. `http://moonscript.org/`, 2011. [Accessed February 2015]. 6.1

[dF14] Luiz Henrique de Figueiredo. lmd5 – A message digest library for Lua based on OpenSSL. `http://www.tecgraf.puc-rio.br/~lhf/ftp/lua/`, 2014. [Accessed Februrary 2015]. 3.8, 5

[Ern13] Emil Ernerfeldt. Sol. `https://github.com/emilk/sol`, 2013. [Accessed Februrary 2015]. 6.1

[Fac14] Facebook. Hack. `http://hacklang.org/`, 2014. [Accessed Februrary 2015]. 6.2

[FF02] Robert Bruce Findler and Matthias Felleisen. Contracts for higher-order functions. In *Proceedings of the 7th ACM SIGPLAN International Conference on Functional Programming*, ICFP '02, pages 48–59, New York, NY, USA, 2002. ACM. 2.1, 2.3

[Fla06] Cormac Flanagan. Hybrid type checking. In *Conference Record of the 33rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '06, pages 245–256, New York, NY, USA, 2006. ACM. 2.4, 5.3

[Fle07] Fabien Fleutot. Metalua: Static meta-programming for Lua. `https://github.com/fab13n/metalua`, 2007. [Accessed February 2015]. 6.1

[Fle13] Fabien Fleutot. Tidal Lock: Gradual static typing for Lua. `https://github.com/fab13n/metalua/tree/tilo/src/tilo`, 2013. [Accessed Februrary 2015]. 6.1

[FT07] Fabien Fleutot and Laurence Tratt. Contrasting compile-time meta-programming in Metalua and Converge. In *Proceedings of the Workshop on Dynamic Languages and Applications*, 2007. 6.1

[GKT+06] Jessica Gronski, Kenneth Knowles, Aaron Tomb, Stephen N. Freund, and Cormac Flanagan. Sage: Hybrid checking for flexible specifications. In *Scheme and Functional Programming Workshop*, pages 93–104, September 2006. 2.4

[Goo11] Google. Dart. `https://www.dartlang.org/`, 2011. [Accessed February 2015]. 2.2, 3, 6.2

[GSK11] Arjun Guha, Claudiu Saftoiu, and Shriram Krishnamurthi. Typing local control and state using flow analysis. In *Proceedings of the 20th European Conference on Programming Languages and Systems: Part of the Joint European Conferences on Theory and Practice of Software*, ESOP'11/ETAPS'11, pages 256–275, Berlin, Heidelberg, 2011. Springer-Verlag. 3.3, 6.2

[HBNB13] Michael Homer, Kim B. Bruce, James Noble, and Andrew P. Black. Modules as gradually-typed objects. In *Proceedings of the 7th Workshop on Dynamic Languages and Applications*, DYLA '13, pages 1:1–1:8, New York, NY, USA, 2013. ACM. 2.3, 6.2

[Hen94] Fritz Henglein. Dynamic typing: Syntax and proof theory. *Science of Computer Programming*, 22(3):197–230, June 1994. 2.1, 2.3

[Hoe12] Rob Hoelz. Typical. `https://github.com/hoelzro/lua-typical`, 2012. [Accessed February 2015]. 5

[HTF07] David Herman, Aaron Tomb, and Cormac Flanagan. Space-efficient gradual typing. In *Trends in Functional Programming*, pages 1–18, 2007. 2.3

[IdFC11] Roberto Ierusalimschy, Luiz Henrique de Figueiredo, and Waldemar Celes. Lua 5.2 Reference Manual. `http://www.lua.org/manual/5.2/manual.html`, 2011. [Accessed February 2015]. 5, 5.1

[Ier08] Roberto Ierusalimschy. LPeg - parsing expression grammars for Lua. `http://www.inf.puc-rio.br/~roberto/lpeg/lpeg.html`, 2008. [Accessed February 2015]. 5.8

[Ier09] Roberto Ierusalimschy. A text pattern-matching tool based on parsing expression grammars. *Software: Practice & Experience*, 39(3):221–258, March 2009. 5.8

[JS11] R. James and A. Sabry. Yield: Mainstream delimited continuations. In *First International Workshop on the Theory and Practice of Delimited Continuations (TPDC 2011)*, page 20, 2011. 5.1

[Kep04] Kepler Project. LuaFileSystem. `http://keplerproject.github.io/luafilesystem/`, 2004. [Accessed February 2015]. 5

[Leh14a] Jukka Lehtosalo. mypy: Optional static typing for Python. `http://www.mypy-lang.org/`, 2014. [Accessed February 2015]. 6.2

[Leh14b] Jukka Lehtosalo. typing: Annotate your Python code. `https://github.com/JukkaL/typing`, 2014. [Accessed Februrary 2015]. 6.2

[LG11] Jukka Lehtosalo and David J. Greaves. Language with a pluggable type system and optional runtime monitoring of type errors. In *International Workshop on Scripts to Programs*, STOP '11, January 2011. 2.4

[LPGK13] Benjamin S. Lerner, Joe Gibbs Politz, Arjun Guha, and Shriram Krishnamurthi. TeJaS: Retrofitting type systems for JavaScript. In *Proceedings of the 9th Symposium on Dynamic Languages*, DLS '13, pages 1–16, New York, NY, USA, 2013. ACM. 6.2

[Maj15] Dibyendu Majumdar. Ravi. `https://github.com/dibyendumajumdar/ravi`, 2015. [Accessed Februrary 2015]. 6.1

[Man11] David Manura. LuaInspect. `https://github.com/davidm/lua-inspect`, 2011. [Accessed February 2015]. 6.1

[Mel14] Peter Melnichenko. Luacheck. `https://github.com/mpeterv/luacheck`, 2014. [Accessed February 2015]. 6.1

[MI09] Ana Lúcia De Moura and Roberto Ierusalimschy. Revisiting coroutines. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 31(2):6:1–6:31, February 2009. 5.1

[Mic12] Microsoft. TypeScript. `http://www.typescriptlang.org/`, 2012. [Accessed Februrary 2015]. 2.2, 3, 6.2

[MMI13] Hisham Muhammad, Fabio Mascarenhas, and Roberto Ierusalimschy. LuaRocks: a declarative and extensible package management system for Lua. In *Brazilian Symposium on Programming Languages*, pages 16–30. Springer, 2013. 2.5

[Moo07] Colin Moock. *Essential ActionScript 3.0*. O'Reilly, 2007. 2.3

[Neh07] Diego Nehab. LuaSocket. `http://w3.impa.br/~diego/software/luasocket/`, 2007. [Accessed February 2015]. 5

[Neh08] Diego Nehab. Filters, sources, sinks, and pumps or functional programming for the rest of us. In Luiz Henrique de Figueiredo, Waldemar Celes, and Roberto Ierusalimschy, editors, *Lua Programming Gems*. Lua.org, Rio de Janeiro, 2008. 5.3

[NN99] Flemming Nielson and Hanne Riis Nielson. Type and effect systems. In *Correct System Design*, pages 114–136. Springer, 1999. 5.1

[OTMW04] Xinming Ou, Gang Tan, Yitzhak Mandelbaum, and David Walker. Dynamic typing with dependent types. *Exploring new frontiers of theoretical informatics*, pages 437–450, 2004. 2.1

[Pea13] David J. Pearce. A calculus for constraint-based flow typing. In *Proceedings of the 15th Workshop on Formal Techniques for Java-like Programs*, FTfJP '13, pages 7:1–7:7, New York, NY, USA, 2013. ACM. 6.2

[PGK12] Joe Gibbs Politz, Arjun Guha, and Shriram Krishnamurthi. Semantics and types for objects with first-class member names. In *FOOL 2012: 19th International Workshop on Foundations of Object-Oriented Languages*, page 37, 2012. 6.2

[Pie02] Benjamin C. Pierce. *Types and Programming Languages*. MIT Press, Cambridge, MA, USA, 2002. 1

[PT00] Benjamin C. Pierce and David N. Turner. Local type inference. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 22(1):1–44, January 2000. 1, 3

[RTSF13] Brianna M. Ren, John Toman, T. Stephen Strickland, and Jeffrey S. Foster. The Ruby Type Checker. In *Object-Oriented Program Languages and Systems (OOPS) Track at ACM Symposium on Applied Computing*, pages 1565–1572, Coimbra, Portugal, March 2013. 6.2

[Sch14] Thijs Schreijer. Lua modulo11 number generator and verificator. `https://github.com/Tieske/mod11`, 2014. [Accessed February 2015]. 5

[SGT09] Jeremy G. Siek, Ronald Garcia, and Walid Taha. Exploring the design space of higher-order casts. In *Proceedings of the 18th European Symposium on Programming Languages and Systems:*

Held As Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009, ESOP '09, pages 17–31, Berlin, Heidelberg, 2009. Springer-Verlag. 2.3

[ST06] Jeremy G. Siek and Walid Taha. Gradual typing for functional languages. In *Scheme and Functional Programming Workshop*, pages 81–92, September 2006. 1, 2.2, 2.3, 2.3, 2.3, 2.3

[ST07] Jeremy G. Siek and Walid Taha. Gradual typing for objects. In *Proceedings of the 21st European Conference on Object-Oriented Programming*, ECOOP'07, pages 2–27, Berlin, Heidelberg, 2007. Springer-Verlag. 2.2, 2.3, 2.3, 2.4, 3, 3.1, 4.2

[Ste82] Guy L. Steele, Jr. An overview of Common LISP. In *Proceedings of the 1982 ACM Symposium on LISP and Functional Programming*, LFP '82, pages 98–107, New York, NY, USA, 1982. ACM. 2.1

[SVB13] Jeremy G. Siek, Michael M. Vitousek, and Shashank Bharadwaj. Gradual typing for mutable objects. Technical report, University of Colorado, 2013. [to be published]. 3, 3.1, 4.2

[SVCB15] Jeremy G. Siek, Michael M. Vitousek, Matteo Cimini, and John Tang Boyland. Refined criteria for gradual typing. In Thomas Ball, Rastislav Bodik, Shriram Krishnamurthi, Benjamin S. Lerner, and Greg Morrisett, editors, *1st Summit on Advances in Programming Languages (SNAPL 2015)*, volume 32 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 274–293, Dagstuhl, Germany, 2015. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. 2.4

[SW10] Jeremy G. Siek and Philip Wadler. Threesomes, with and without blame. In *Proceedings of the 37th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '10, pages 365–376, New York, NY, USA, 2010. ACM. 2.3

[Tan07] Audrey Tang. Perl 6: Reconciling the irreconcilable. In *Proceedings of the 34th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '07, page 1, New York, NY, USA, 2007. ACM. 2.3

[Tha90] Satish Thatte. Quasi-static typing. In *Proceedings of the 17th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '90, pages 367–381, New York, NY, USA, 1990. ACM. 2.1, 2.3

[THF06] Sam Tobin-Hochstadt and Matthias Felleisen. Interlanguage migration: From scripts to programs. In *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, OOPSLA '06, pages 964–974, New York, NY, USA, 2006. ACM. 1, 2.4

[THF08] Sam Tobin-Hochstadt and Matthias Felleisen. The design and implementation of Typed Scheme. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '08, pages 395–406, New York, NY, USA, 2008. ACM. 2.4, 6.2

[THF10] Sam Tobin-Hochstadt and Matthias Felleisen. Logical types for untyped languages. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, pages 117–128, New York, NY, USA, 2010. ACM. 3.3, 6.2

[Vit13] Michael Vitousek. Reticulated Python. `https://github.com/mvitousek/reticulated`, 2013. [Accessed Februrary 2015]. 2.3

[VKSB14] Michael M. Vitousek, Andrew M. Kent, Jeremy G. Siek, and Jim Baker. Design and evaluation of gradual typing for Python. In *Proceedings of the 10th ACM Symposium on Dynamic Languages*, DLS '14, pages 45–56, New York, NY, USA, 2014. ACM. 2.3, 6.2

[vR14] Guido van Rossum. PEP483 - The Theory of Type Hints. `https://www.python.org/dev/peps/pep-0483/`, 2014. [Accessed Februrary 2015]. 6.2

[vRLL14] Guido van Rossum, Jukka Lehtosalo, and Lukasz Langa. PEP484 - Type Hints. `https://www.python.org/dev/peps/pep-0484/`, 2014. [Accessed Februrary 2015]. 6.2

[WF09] Philip Wadler and Robert Bruce Findler. Well-typed programs can't be blamed. In *Proceedings of the 18th European Symposium on Programming Languages and Systems: Held As Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2009*, ESOP '09, pages 1–16, Berlin, Heidelberg, 2009. Springer-Verlag. 2.3

[Win11] Johnni Winther. Guarded type promotion: Eliminating redundant casts in Java. In *Proceedings of the 13th Workshop on Formal*

Techniques for Java-Like Programs, FTfJP '11, pages 6:1–6:8, New York, NY, USA, 2011. ACM. 6.2

A
Glossary

bottom type A type that is subtype of all types.

closed table type A table type that does not provide any guarantees about keys with types not listed in the table type. See complete definition in page 63.

coercion A relation that allows converting values from one type to values of another type without error.

consistency A relation used by gradual typing to check the interaction between the dynamic type and other types. See complete definition in page 18.

consistent-subtyping A relation that combines consistency and subtyping. See complete definition in page 20.

contravariant A part of a type constructor is contravariant when it reverses the subtyping order, that is, the part T_1 of a type being a subtype of the corresponding part T_2 of another type implies that $T_2 <: T_1$.

covariant A part of a type constructor is covariant when it preserves the subtyping order, that is, the part T_1 of a type being a subtype of the corresponding part T_2 of another type implies that $T_1 <: T_2$.

depth subtyping A subtyping relation over records that allows variance in the type of record fields.

dynamic type A type used by gradual typing to denote unknown values. See complete definition in page 18.

filter type A type used by Typed Lua to discriminate the type of local variables inside control flow statements. See complete definition in page 65.

fixed table type A table type which guarantees that there are no keys with a type that is not one of its key types, and that can have any number of *fixed* or *closed* references. See complete definition in page 63.

flow typing An approach that combines static typing and flow analysis to allow variables to have different types at different parts of the program.

free assigned variable A free variable that appears in an assignment.

gradual type system A type system that uses either the consistency relation or the consistent-subtyping relation instead of type equality to perform static type checking. See complete definition in page 18.

gradual typing An approach that uses a gradual type system to allow static and dynamic typing in the same code, but inserting run-time checks between statically typed and dynamically typed code. See complete definition in page 18.

invariant A part of a type constructor is invariant when it forbids variance, that is, the part T_1 of a type being a subtype of the corresponding part T_2 of another type implies that $T_1 <: T_2$ and $T_2 <: T_1$. It is also a way to define type equality through subtyping.

metatable A Lua table that allows changing the behavior of other tables it is attached to.

nominal type system A type system that uses the type names to check the compatibility among them.

open table type A table type which guarantees that there are no keys with a type that is not one of its key types, and that only have *closed* references. See complete definition in page 63.

optional type system A type system that allows combining static and dynamic typing in the same language, but without affecting the run-time semantics. See complete definition in page 17.

projection environment An environment used by Typed Lua to handle unions of second-level types that are bound to projection types.

projection type A type used by Typed Lua to discriminate the type of local variables that have a dependency relation. See complete definition in page 65.

prototype object An object that works like a class, that is, it is an object from which other objects inherit its attributes.

self-like delegation A technique to implement inheritance in dynamically typed languages through prototype objects.

sound type system A type system that does not type check all programs that contain a type error.

structural type system A type system that uses type structures to check the compatibility among them.

table refinement An operation from Typed Lua that allows programmers to change a table type to include new fields or to specialize existing fields. See complete definition in page 80.

top type A type that is supertype of all types.

type environment An environment that binds variable names to types.

type tag A tag that describes the type of a value during run-time in dynamically typed languages.

unique table type A table type which guarantees that there are no keys with a type that is not one of its key types, and that has only one reference. See complete definition in page 63.

unsound type system A type system that type checks certain programs that contain type errors.

userdata A Lua data type that allows Lua to hold values from applications or libraries that are written in C.

vararg expression A Lua expression that can result in an arbitrary number of values.

variadic function A function that can use an arbitrary number of arguments.

variance A property that defines the subtyping order between the components of a type constructor.

width subtyping A subtyping relation over records that allows the subtype to include fields that do not exist in the supertype.

B
The syntax of Typed Lua

This appendix presents the complete syntax of Typed Lua.

$$
\begin{aligned}
\textit{chunk} ::= {} & \textit{block} \\
\textit{block} ::= {} & \{\textit{stat}\}\ [\textit{retstat}] \\
\textit{stat} ::= {} & \text{`;'} \\
| {} & \textit{varlist}\ \text{`='}\ \textit{explist} \\
| {} & \textit{functioncall} \\
| {} & \textit{label} \\
| {} & \textbf{break} \\
| {} & \textbf{goto}\ \textit{Name} \\
| {} & \textbf{do}\ \textit{block}\ \textbf{end} \\
| {} & \textbf{while}\ \textit{exp}\ \textbf{do}\ \textit{block}\ \textbf{end} \\
| {} & \textbf{repeat}\ \textit{block}\ \textbf{until}\ \textit{exp} \\
| {} & \textbf{if}\ \textit{exp}\ \textbf{then}\ \textit{block}\ \{\textbf{elseif}\ \textit{exp}\ \textbf{then}\ \textit{block}\}\ [\textbf{else}\ \textit{block}]\ \textbf{end} \\
| {} & \textbf{for}\ \textit{Name}\ \text{`='}\ \textit{exp}\ \text{`,'}\ \textit{exp}\ [\text{`,'}\ \textit{exp}]\ \textbf{do}\ \textit{block}\ \textbf{end} \\
| {} & \textbf{for}\ \textit{namelist}\ \textbf{in}\ \textit{explist}\ \textbf{do}\ \textit{block}\ \textbf{end} \\
| {} & [\textbf{const}]\ \textbf{function}\ \textit{funcname}\ \textit{funcbody} \\
| {} & \textbf{local function}\ \textit{Name}\ \textit{funcbody} \\
| {} & \textbf{local}\ \textit{namelist}\ [\text{`='}\ \textit{explist}] \\
| {} & [\textbf{local}]\ \textbf{typealias}\ \textit{Name}\ \text{`='}\ \textit{type} \\
| {} & [\textbf{local}]\ \textbf{interface}\ \textit{typedec} \\
\textit{retstat} ::= {} & \textbf{return}\ [\textit{explist}]\ [\text{`;'}] \\
\textit{label} ::= {} & \text{`::'}\ \textit{Name}\ \text{`::'} \\
\textit{funcname} ::= {} & \textit{Name}\ \{\text{`.'}\ \textit{Name}\}\ [\text{`:'}\ \textit{Name}] \\
\textit{varlist} ::= {} & [\textbf{const}]\ \textit{var}\ \{\text{`,'}\ [\textbf{const}]\ \textit{var}\} \\
\textit{var} ::= {} & \textit{Name} \mid \textit{prefixexp}\ \text{`['}\ \textit{exp}\ \text{`]'} \mid \textit{prefixexp}\ \text{`.'}\ \textit{Name} \\
\textit{namelist} ::= {} & \textit{Name}\ [\text{`:'}\ \textit{type}]\ \{\text{`,'}\ \textit{Name}\ [\text{`:'}\ \textit{type}]\}
\end{aligned}
$$

$$
\begin{aligned}
\mathit{explist} &::= \mathit{exp}\ \{\text{`,'}\ \mathit{exp}\}\\
\mathit{exp} &::= \textbf{nil} \mid \textbf{false} \mid \textbf{true} \mid \mathit{Number} \mid \mathit{String} \mid \text{`...'} \mid \mathit{functiondef}\\
&\quad \mid \mathit{prefixexp} \mid \mathit{tableconstructor} \mid \mathit{exp}\ \mathit{binop}\ \mathit{exp} \mid \mathit{unop}\ \mathit{exp}\\
\mathit{prefixexp} &::= \mathit{var} \mid \mathit{functioncall} \mid \text{`('}\ \mathit{exp}\ \text{`)'}\\
\mathit{functioncall} &::= \mathit{prefixexp}\ \mathit{args} \mid \mathit{prefixexp}\ \text{`:'}\ \mathit{Name}\ \mathit{args}\\
\mathit{args} &::= \text{`('}\ [\mathit{explist}]\ \text{`)'} \mid \mathit{tableconstructor} \mid \mathit{String}\\
\mathit{functiondef} &::= \textbf{function}\ \mathit{funcbody}\\
\mathit{funcbody} &::= \text{`('}\ [\mathit{parlist}]\ \text{`)'}\ [\text{`:'}\ \mathit{rettype}]\ \mathit{block}\ \textbf{end}\\
\mathit{parlist} &::= \mathit{namelist}\ [\text{`,'}\ \text{`...'}\ [\text{`:'}\ \mathit{type}]] \mid \text{`...'}\ [\text{`:'}\ \mathit{type}]\\
\mathit{tableconstructor} &::= \text{`\{'}\ [\mathit{fieldlist}]\ \text{`\}'}\\
\mathit{fieldlist} &::= [\textbf{const}]\ \mathit{field}\ \{\mathit{fieldsep}\ [\textbf{const}]\ \mathit{field}\}\ [\mathit{fieldsep}]\\
\mathit{field} &::= \text{`[']}\ \mathit{exp}\ \text{`]'}\ \text{`='}\ \mathit{exp} \mid \mathit{Name}\ \text{`='}\ \mathit{exp} \mid \mathit{exp}\\
\mathit{fieldsep} &::= \text{`,'} \mid \text{`;'}\\
\mathit{binop} &::= \text{`+'} \mid \text{`-'} \mid \text{`*'} \mid \text{`/'} \mid \text{`//'} \mid \text{`\^{}'} \mid \text{`\%'}\\
&\quad \mid \text{`\&'} \mid \text{`\~{}'} \mid \text{`|'} \mid \text{`>>'} \mid \text{`<<'} \mid \text{`..'}\\
&\quad \mid \text{`<'} \mid \text{`<='} \mid \text{`>'} \mid \text{`>='} \mid \text{`=='} \mid \text{`\~{}='}\\
&\quad \mid \textbf{and} \mid \textbf{or}\\
\mathit{unop} &::= \text{`-'} \mid \textbf{not} \mid \text{`\#'} \mid \text{`\~{}'}\\
\mathit{typedec} &::= \mathit{Name}\ \{\mathit{decitem}\}\ \textbf{end}\\
\mathit{decitem} &::= \mathit{idlist}\ \text{`:'}\ \mathit{idtype}\\
\mathit{idtype} &::= \mathit{type} \mid \mathit{methodtype}\\
\mathit{idlist} &::= \mathit{id}\ \{\text{`,'}\ \mathit{id}\}\\
\mathit{id} &::= [\textbf{const}]\ \mathit{Name}\\
\mathit{type} &::= \mathit{primarytype}\ [\text{`?'}]\\
\mathit{primarytype} &::= \mathit{literaltype} \mid \mathit{basetype} \mid \textbf{nil} \mid \textbf{value} \mid \textbf{any} \mid \textbf{self} \mid \mathit{Name}\\
&\quad \mid \mathit{functiontype} \mid \mathit{tabletype} \mid \mathit{primarytype}\ \text{`|'}\ \mathit{primarytype}\\
\mathit{literaltype} &::= \textbf{false} \mid \textbf{true} \mid \mathit{Int} \mid \mathit{Float} \mid \mathit{String}\\
\mathit{basetype} &::= \textbf{boolean} \mid \textbf{integer} \mid \textbf{number} \mid \textbf{string}\\
\mathit{functiontype} &::= \mathit{tupletype}\ \text{`->'}\ \mathit{rettype}\\
\mathit{tupletype} &::= \text{`('}\ [\mathit{typelist}]\ \text{`)'}\\
\mathit{typelist} &::= \mathit{type}\ \{\text{`,'}\ \mathit{type}\}\ [\text{`*'}]\\
\mathit{rettype} &::= \mathit{type} \mid \mathit{uniontuple}\ [\text{`?'}]\\
\mathit{uniontuple} &::= \mathit{tupletype} \mid \mathit{uniontuple}\ \text{`|'}\ \mathit{uniontuple}
\end{aligned}
$$

$$
\begin{aligned}
\textit{tabletype} &::= \text{`\{'}\ [\textit{tabletypebody}]\ \text{`\}'} \\
\textit{tabletypebody} &::= \textit{maptype} \mid \textit{recordtype} \\
\textit{maptype} &::= [\textit{keytype}\ \text{`:'}]\ \textit{type} \\
\textit{keytype} &::= \textit{basetype} \mid \textbf{value} \\
\textit{recordtype} &::= \textit{recordfield}\ \{\text{`,'}\ \textit{recordfield}\}\ [\text{`,'}\ \textit{type}] \\
\textit{recordfield} &::= [\textbf{const}]\ \textit{literaltype}\ \text{`:'}\ \textit{type} \\
\textit{methodtype} &::= \textit{tupletype}\ \text{`=>'}\ \textit{rettype}
\end{aligned}
$$

C
The type system of Typed Lua

This appendix presents the complete type system of Typed Lua.

C.1 Subtyping rules

(S-LITERAL)

$$\Sigma \vdash L <: L$$

(S-FALSE)

$$\Sigma \vdash \textbf{false} <: \textbf{boolean}$$

(S-TRUE)

$$\Sigma \vdash \textbf{true} <: \textbf{boolean}$$

(S-STRING)

$$\Sigma \vdash \mathit{string} <: \textbf{string}$$

(S-INT1)

$$\Sigma \vdash \mathit{int} <: \textbf{integer}$$

(S-INT2)

$$\Sigma \vdash \mathit{int} <: \textbf{number}$$

(S-FLOAT)

$$\Sigma \vdash \mathit{float} <: \textbf{number}$$

(S-BASE)

$$\Sigma \vdash B <: B$$

(S-INTEGER)

$$\Sigma \vdash \textbf{integer} <: \textbf{number}$$

(S-NIL)

$$\Sigma \vdash \textbf{nil} <: \textbf{nil}$$

(S-VALUE)

$$\Sigma \vdash F <: \mathbf{value}$$

(S-ANY)

$$\Sigma \vdash \mathbf{any} <: \mathbf{any}$$

(S-SELF)

$$\Sigma \vdash \mathbf{self} <: \mathbf{self}$$

(S-UNION1)

$$\frac{\Sigma \vdash F_1 <: F \quad \Sigma \vdash F_2 <: F}{\Sigma \vdash F_1 \cup F_2 <: F}$$

(S-UNION2)

$$\frac{\Sigma \vdash F <: F_1}{\Sigma \vdash F <: F_1 \cup F_2}$$

(S-UNION3)

$$\frac{\Sigma \vdash F <: F_2}{\Sigma \vdash F <: F_1 \cup F_2}$$

(S-FUNCTION)

$$\frac{\Sigma \vdash S_3 <: S_1 \quad \Sigma \vdash S_2 <: S_4}{\Sigma \vdash S_1 \to S_2 <: S_3 \to S_4}$$

(S-PAIR)

$$\frac{\Sigma \vdash F_1 <: F_2 \quad \Sigma \vdash P_1 <: P_2}{\Sigma \vdash F_1 \times P_1 <: F_2 \times P_2}$$

(S-VARARG1)

$$\frac{\Sigma \vdash F_1 \cup \mathbf{nil} <: F_2 \cup \mathbf{nil}}{\Sigma \vdash F_1* <: F_2*}$$

(S-VARARG2)

$$\frac{\Sigma \vdash F_1 \cup \mathbf{nil} <: F_2 \quad \Sigma \vdash F_1* <: P_2}{\Sigma \vdash F_1* <: F_2 \times P_2}$$

(S-VARARG3)

$$\frac{\Sigma \vdash F_1 <: F_2 \cup \mathbf{nil} \quad \Sigma \vdash P_1 <: F_2*}{\Sigma \vdash F_1 \times P_1 <: F_2*}$$

(S-UNION4)

$$\frac{\Sigma \vdash S_1 <: S \quad \Sigma \vdash S_2 <: S}{\Sigma \vdash S_1 \sqcup S_2 <: S}$$

(S-UNION5)

$$\frac{\Sigma \vdash S <: S_1}{\Sigma \vdash S <: S_1 \sqcup S_2}$$

(S-UNION6)

$$\frac{\Sigma \vdash S <: S_2}{\Sigma \vdash S <: S_1 \sqcup S_2}$$

(S-TABLE1)

$$\frac{\forall i\ \exists j \quad \Sigma \vdash F_j <: F'_i \quad \Sigma \vdash F'_i <: F_j \quad \Sigma \vdash V_j <:_c V'_i}{\Sigma \vdash \{\overline{F{:}V}\}_{fixed|closed} <: \{\overline{F'{:}V'}\}_{closed}}$$

(S-TABLE2)

$$\frac{\begin{array}{c}\forall i\ \forall j \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash V_i <:_u V'_j \\ \forall j\ \nexists i \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash \mathbf{nil} <:_o V'_j\end{array}}{\Sigma \vdash \{\overline{F{:}V}\}_{unique} <: \{\overline{F'{:}V'}\}_{closed}}$$

(S-TABLE3)

$$\frac{\begin{array}{c}\forall i\ \exists j \quad \Sigma \vdash F_i <: F'_j \wedge \Sigma \vdash V_i <:_u V'_j \\ \forall j\ \nexists i \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash \mathbf{nil} <:_o V'_j\end{array}}{\Sigma \vdash \{\overline{F{:}V}\}_{unique} <: \{\overline{F'{:}V'}\}_{unique|open|fixed}}$$

(S-TABLE4)

$$\frac{\begin{array}{c}\forall i\ \forall j \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash V_i <:_c V'_j \\ \forall j\ \nexists i \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash \mathbf{nil} <:_o V'_j\end{array}}{\Sigma \vdash \{\overline{F{:}V}\}_{open} <: \{\overline{F'{:}V'}\}_{closed}}$$

(S-TABLE5)

$$\frac{\begin{array}{c}\forall i\ \exists j \quad \Sigma \vdash F_i <: F'_j \wedge \Sigma \vdash V_i <:_c V'_j \\ \forall j\ \nexists i \quad \Sigma \vdash F_i <: F'_j \rightarrow \Sigma \vdash \mathbf{nil} <:_o V'_j\end{array}}{\Sigma \vdash \{\overline{F{:}V}\}_{open} <: \{\overline{F'{:}V'}\}_{open|fixed}}$$

(S-TABLE6)

$$\frac{\begin{array}{c}\forall i\ \exists j \quad \Sigma \vdash F_i <: F'_j \quad \Sigma \vdash F'_j <: F_i \quad \Sigma \vdash V_i <:_c V'_j \\ \forall j\ \exists i \quad \Sigma \vdash F_i <: F'_j \quad \Sigma \vdash F'_j <: F_i \quad \Sigma \vdash V_i <:_c V'_j\end{array}}{\Sigma \vdash \{\overline{F{:}V}\}_{fixed} <: \{\overline{F'{:}V'}\}_{fixed}}$$

(S-FIELD1)
$$\frac{\Sigma \vdash F_1 <: F_2 \quad \Sigma \vdash F_2 <: F_1}{\Sigma \vdash F_1 <:_c F_2}$$

(S-FIELD2)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash \textbf{const } F_1 <:_c \textbf{const } F_2}$$

(S-FIELD3)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash F_1 <:_c \textbf{const } F_2}$$

(S-FIELD4)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash F_1 <:_u F_2}$$

(S-FIELD5)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash \textbf{const } F_1 <:_u \textbf{const } F_2}$$

(S-FIELD6)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash \textbf{const } F_1 <:_u F_2}$$

(S-FIELD7)
$$\frac{\Sigma \vdash F_1 <: F_2}{\Sigma \vdash F_1 <:_u \textbf{const } F_2}$$

(S-FIELD8)
$$\frac{\Sigma \vdash \textbf{nil} <: F}{\Sigma \vdash \textbf{nil} <:_o F}$$

(S-FIELD9)
$$\frac{\Sigma \vdash \textbf{nil} <: F}{\Sigma \vdash \textbf{nil} <:_o \textbf{const } F}$$

(S-AMBER)
$$\frac{\Sigma[x_1 <: x_2] \vdash F_1 <: F_2}{\Sigma \vdash \mu x_1.F_1 <: \mu x_2.F_2}$$

(S-ASSUMPTION)
$$\frac{x_1 <: x_2 \in \Sigma}{\Sigma \vdash x_1 <: x_2}$$

(S-UNFOLDR)
$$\frac{\Sigma \vdash F_1 <: [x \mapsto \mu x.F_2]F_2}{\Sigma \vdash F_1 <: \mu x.F_2}$$

(S-UNFOLDL)
$$\frac{\Sigma \vdash [x \mapsto \mu x.F_1]F_1 <: F_2}{\Sigma \vdash \mu x.F_1 <: F_2}$$

(S-EXPRESSION)

$$\Sigma \vdash T <: T$$

(S-PAIR2)

$$\frac{\Sigma \vdash T_1 <: T_2 \quad \Sigma \vdash E_1 <: E_2}{\Sigma \vdash T_1 \times E_1 <: T_2 \times E_2}$$

(S-VARARG4)

$$\frac{\Sigma \vdash T_1 \cup \mathbf{nil} <: T_2 \cup \mathbf{nil}}{\Sigma \vdash T_1* <: T_2*}$$

(S-VARARG5)

$$\frac{\Sigma \vdash T_1 \cup \mathbf{nil} <: T_2 \quad \Sigma \vdash T_1* <: E_2}{\Sigma \vdash T_1* <: T_2 \times E_2}$$

(S-VARARG6)

$$\frac{\Sigma \vdash T_1 <: T_2 \cup \mathbf{nil} \quad \Sigma \vdash E_1 <: T_2*}{\Sigma \vdash T_1 \times E_1 <: T_2*}$$

(C-ANY1)

$$\Sigma \vdash F \lesssim \mathbf{any}$$

(C-ANY2)

$$\Sigma \vdash \mathbf{any} \lesssim F$$

C.2 Typing rules

(T-SKIP)

$$\Gamma_1, \Pi \vdash \mathbf{skip}, \Gamma_1$$

(T-SEQ)

$$\frac{\Gamma_1, \Pi \vdash s_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash s_2, \Gamma_3}{\Gamma_1, \Pi \vdash s_1 \ ; \ s_2, \Gamma_3}$$

(T-ASSIGNMENT)

$$\frac{\Gamma_1, \Pi \vdash el : S_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash \overline{l} : S_2, \Gamma_3 \quad S_1 \lesssim S_2}{\Gamma_1, \Pi \vdash \overline{l} = el, \Gamma_3}$$

(T-METHOD1)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{unique} \\ \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad \nexists i \in 1..n \; L \lesssim F_i \quad n = |\overline{F{:}V}| \\ closeall(\Gamma_1)[self \mapsto \mathbf{self}, \overline{id \mapsto F}, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ F_o = \{\overline{F{:}V}, L{:}\mathbf{const\ self} \times F_1 \times ... \times F_n \times \mathbf{nil}* \to S\}_{unique} \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_o], frv(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s))\end{array}}{\Gamma_1, \Pi \vdash \mathbf{fun}\ id_1{:}id_2\ (\overline{id{:}F}){:}S\ s, \Gamma_5}$$

(T-METHOD2)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{open} \\ \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad \nexists i \in 1..n \; L \lesssim F_i \quad n = |\overline{F{:}V}| \\ closeall(\Gamma_1)[self \mapsto \mathbf{self}, \overline{id \mapsto F}, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ F_o = \{\overline{F{:}V}, L{:}\mathbf{const\ self} \times F_1 \times ... \times F_n \times \mathbf{nil}* \to S\}_{open} \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_o], frv(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s))\end{array}}{\Gamma_1, \Pi \vdash \mathbf{fun}\ id_1{:}id_2\ (\overline{id{:}F}){:}S\ s, \Gamma_5}$$

(T-METHOD3)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{unique} \\ \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad \nexists i \in 1..n \; L \lesssim F_i \quad n = |\overline{F{:}V}| \\ closeall(\Gamma_1)[self \mapsto \mathbf{self}, \overline{id \mapsto F}, ... \mapsto F, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ F_o = \{\overline{F{:}V}, L{:}\mathbf{const\ self} \times F_1 \times ... \times F_n \times F* \to S\}_{unique} \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_o], frv(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s))\end{array}}{\Gamma_1, \Pi \vdash \mathbf{fun}\ id_1{:}id_2\ (\overline{id{:}F}, ...{:}F){:}S\ s, \Gamma_5}$$

(T-METHOD4)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{open} \\ \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad \nexists i \in 1..n \; L \lesssim F_i \quad n = |\overline{F{:}V}| \\ closeall(\Gamma_1)[self \mapsto \mathbf{self}, \overline{id \mapsto F}, ... \mapsto F, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ F_o = \{\overline{F{:}V}, L{:}\mathbf{const\ self} \times F_1 \times ... \times F_n \times F* \to S\}_{open} \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_o], frv(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\mathbf{fun}\ (\overline{id{:}T}){:}S\ s))\end{array}}{\Gamma_1, \Pi \vdash \mathbf{fun}\ id_1{:}id_2\ (\overline{id{:}F}, ...{:}F){:}S\ s, \Gamma_5}$$

(T-METHOD5)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{unique} \quad \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad n = |\overline{F{:}V}| \\ \exists i \in 1..n \; L <: F_i \wedge F_i <: L \wedge \textbf{const self} \times F_1 \times ... \times F_n \times \textbf{nil}* \rightarrow S <: V_i \\ closeall(\Gamma_1)[self \mapsto \textbf{self}, \overline{id \mapsto F}, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ V_i \mapsto \textbf{const self} \times F_1 \times ... \times F_n \times \textbf{nil}* \rightarrow S \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_s], frv(\textbf{fun } (\overline{id{:}T}){:}S \; s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\textbf{fun } (\overline{id{:}T}){:}S \; s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } id_1{:}id_2 \; (\overline{id{:}F}){:}S \; s, \Gamma_5}$$

(T-METHOD6)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{open} \quad \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad n = |\overline{F{:}V}| \\ \exists i \in 1..n \; L <: F_i \wedge F_i <: L \wedge \textbf{const self} \times F_1 \times ... \times F_n \times \textbf{nil}* \rightarrow S <: V_i \\ closeall(\Gamma_1)[self \mapsto \textbf{self}, \overline{id \mapsto F}, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ V_i \mapsto \textbf{const self} \times F_1 \times ... \times F_n \times \textbf{nil}* \rightarrow S \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_s], frv(\textbf{fun } (\overline{id{:}T}){:}S \; s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\textbf{fun } (\overline{id{:}T}){:}S \; s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } id_1{:}id_2 \; (\overline{id{:}F}){:}S \; s, \Gamma_5}$$

(T-METHOD7)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{unique} \quad \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad n = |\overline{F{:}V}| \\ \exists i \in 1..n \; L <: F_i \wedge F_i <: L \wedge \textbf{const self} \times F_1 \times ... \times F_n \times F* \rightarrow S <: V_i \\ closeall(\Gamma_1)[self \mapsto \textbf{self}, \overline{id \mapsto F}, ... \mapsto F, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ V_i \mapsto \textbf{const self} \times F_1 \times ... \times F_n \times F* \rightarrow S \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_s], frv(\textbf{fun } (\overline{id{:}T}){:}S \; s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\textbf{fun } (\overline{id{:}T}){:}S \; s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } id_1{:}id_2 \; (\overline{id{:}F}, ...{:}F){:}S \; s, \Gamma_5}$$

(T-METHOD8)

$$\frac{\begin{array}{c}\Gamma_1(id_1) = F_s \quad F_s = \{\overline{F{:}V}\}_{open} \quad \Gamma_1, \Pi \vdash id_2 : L, \Gamma_2 \quad n = |\overline{F{:}V}| \\ \exists i \in 1..n \; L <: F_i \wedge F_i <: L \wedge \textbf{const self} \times F_1 \times ... \times F_n \times F* \rightarrow S <: V_i \\ closeall(\Gamma_1)[self \mapsto \textbf{self}, \overline{id \mapsto F}, ... \mapsto F, \sigma \mapsto F_s], \Pi[\rho \mapsto S] \vdash s, \Gamma_3 \\ V_i \mapsto \textbf{const self} \times F_1 \times ... \times F_n \times F* \rightarrow S \\ \Gamma_4 = openset(\Gamma_1[id_1 \mapsto F_s], frv(\textbf{fun } (\overline{id{:}T}){:}S \; s)) \\ \Gamma_5 = closeset(\Gamma_4, fav(\textbf{fun } (\overline{id{:}T}){:}S \; s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } id_1{:}id_2 \; (\overline{id{:}F}, ...{:}F){:}S \; s, \Gamma_5}$$

(T-WHILE1)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash e : F, \Gamma_2 \quad closeall(\Gamma_2), \Pi \vdash s, \Gamma_3 \\ \Gamma_4 = closeset(\Gamma_2, fav(s)) \\ \Gamma_5 = openset(\Gamma_4, frv(s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{while } e \textbf{ do } s, \Gamma_5}$$

(T-WHILE2)

$$\frac{\begin{array}{c}\Gamma_1(id) = F \\ closeall(\Gamma_1[id \mapsto \phi(F, filter(F, \textbf{nil}))]), \Pi \vdash s, \Gamma_2 \\ \Gamma_3 = openset(\Gamma_1[id \mapsto F], frv(s)) \\ \Gamma_4 = closeset(\Gamma_3, fav(s))\end{array}}{\Gamma_1, \Pi \vdash \textbf{while } id \textbf{ do } s, \Gamma_4}$$

(T-IF1)

$$\frac{\Gamma_1, \Pi \vdash e : T, \Gamma_2 \quad \Gamma_2, \Pi \vdash s_1 : \Gamma_3 \quad \Gamma_2, \Pi \vdash s_2 : \Gamma_4 \quad \Gamma_5 = join(\Gamma_3, \Gamma_4)}{\Gamma_1 \vdash \textbf{if } e \textbf{ then } s_1 \textbf{ else } s_2, \Gamma_5}$$

(T-IF2)

$$\frac{\begin{array}{c}\Gamma_1(id) = F \quad F_t = fot(F, \textbf{nil}) \quad F_e = fit(F, \textbf{nil}) \\ \Gamma_1[id \mapsto \phi(F, F_t)], \Pi \vdash s_1, \Gamma_2 \\ \Gamma_1[id \mapsto \phi(F, F_e)], \Pi \vdash s_2, \Gamma_3 \\ \Gamma_4 = join(\Gamma_2, \Gamma_3)\end{array}}{\Gamma_1, \Pi \vdash \textbf{if } id \textbf{ then } s_1 \textbf{ else } s_2, \Gamma_4[id \mapsto F]}$$

(T-IF3)

$$\frac{\begin{array}{c}\Gamma_1(id) = F \quad F_t = fot(F, \textbf{nil}) \quad F_e = fit(F, \textbf{nil}) \\ F_e = \textbf{void} \quad \Gamma_1[id \mapsto \phi(F, F_t)], \Pi \vdash s_1, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \textbf{if } id \textbf{ then } s_1 \textbf{ else } s_2, \Gamma_2[id \mapsto F]}$$

(T-IF4)

$$\frac{\begin{array}{c}\Gamma_1(id) = F \quad F_t = fot(F, \textbf{nil}) \quad F_e = fit(F, \textbf{nil}) \\ F_t = \textbf{void} \quad \Gamma_1[id \mapsto \phi(F, F_e)], \Pi \vdash s_2, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \textbf{if } id \textbf{ then } s_1 \textbf{ else } s_2, \Gamma_2[id \mapsto F]}$$

(T-IF5)

$$\frac{\begin{array}{c}\Gamma_1(id) = \pi_i^x \\ S_t = fopt(\Pi(x), \mathbf{nil}, i) \quad S_e = fipt(\Pi(x), \mathbf{nil}, i) \\ \Gamma_1, \Pi[x \mapsto S_t] \vdash s_1, \Gamma_2 \\ \Gamma_1, \Pi[x \mapsto S_e] \vdash s_2, \Gamma_3 \\ \Gamma_4 = join(\Gamma_2, \Gamma_3)\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ id\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_4}$$

(T-IF6)

$$\frac{\begin{array}{c}\Gamma_1(id) = \pi_i^x \\ S_t = fopt(\Pi(x), \mathbf{nil}, i) \\ fit(proj(\Pi(x), i), \mathbf{nil}) = \mathbf{void} \\ \Gamma_1, \Pi[x \mapsto S_t] \vdash s_1, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ id\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_2}$$

(T-IF7)

$$\frac{\begin{array}{c}\Gamma_1(id) = \pi_i^x \\ S_e = fipt(\Pi(x), \mathbf{nil}, i) \\ fot(proj(\Pi(x), i), \mathbf{nil}) = \mathbf{void} \\ \Gamma_1, \Pi[x \mapsto S_e] \vdash s_2, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ id\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_2}$$

(T-IF8)

$$\frac{\begin{array}{c}\Gamma_1(id) = \phi(F_1, F_2) \quad F_t = fit(F_2, \mathbf{string}) \quad F_e = fot(F_2, \mathbf{string}) \\ \Gamma_1[id \mapsto \phi(F_1, F_t)], \Pi \vdash s_1, \Gamma_2 \\ \Gamma_1[id \mapsto \phi(F_1, F_e)], \Pi \vdash s_2, \Gamma_3 \\ \Gamma_4 = join(\Gamma_2, \Gamma_3)\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ type(id) == \text{“}string\text{”}\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_4[id \mapsto F_1]}$$

(T-IF9)

$$\frac{\begin{array}{c}\Gamma_1(id) = \phi(F_1, F_2) \quad F_t = fit(F_2, \mathbf{string}) \quad F_e = fot(F_2, \mathbf{string}) \\ F_t = \mathbf{void} \quad \Gamma_1[id \mapsto \phi(F_1, F_e)], \Pi \vdash s_2, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ type(id) == \text{“}string\text{”}\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_2[id \mapsto F_1]}$$

(T-IF10)

$$\frac{\begin{array}{c}\Gamma_1(id) = \phi(F_1, F_2) \quad F_t = fit(F_2, \mathbf{string}) \quad F_e = fot(F_2, \mathbf{string}) \\ F_e = \mathbf{void} \quad \Gamma_1[id \mapsto \phi(F_1, F_t)], \Pi \vdash s_1, \Gamma_2\end{array}}{\Gamma_1, \Pi \vdash \mathbf{if}\ type(id) == \text{“}string\text{”}\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2, \Gamma_2[id \mapsto F_1]}$$

(T-LOCAL1)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash el : S, \Gamma_2 \\ S \lesssim F_1 \times ... \times F_n \times \textbf{value}* \quad n = \mid \overline{id{:}F} \mid \\ \Gamma_2[\overline{id \mapsto F}], \Pi \vdash s, \Gamma_3\end{array}}{\Gamma_1, \Pi \vdash \textbf{local } \overline{id{:}F} = el \textbf{ in } s, (\Gamma_3 - \{\overline{id}\})[\overline{id \mapsto \Gamma_2(id)}]}$$

(T-LOCAL2)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash el : E, \Gamma_2, (x, S) \\ \Gamma_3 = \Gamma_2[id_1 \mapsto infer(E, 1), ..., id_n \mapsto infer(E, n)] \\ \Gamma_3, \Pi[x \mapsto S] \vdash s, \Gamma_4 \quad n = \mid \overline{id} \mid\end{array}}{\Gamma_1, \Pi \vdash \textbf{local } \overline{id} = el \textbf{ in } s, (\Gamma_4 - \{\overline{id}\})[\overline{id \mapsto \Gamma_2(id)}]}$$

(T-LOCALREC)

$$\frac{\Gamma_1[id \mapsto F], \Pi \vdash e : F_1, \Gamma_2 \quad F_1 \lesssim F \quad \Gamma_2, \Pi \vdash s, \Gamma_3}{\Gamma_1, \Pi \vdash \textbf{rec } id{:}F = e \textbf{ in } s, (\Gamma_3 - \{id\})[\overline{id \mapsto \Gamma_2(id)}]}$$

(T-RETURN)

$$\frac{\Gamma_1 \vdash el : S_1, \Gamma_2 \quad \Pi(\rho) = S_2 \quad S_1 \lesssim S_2}{\Gamma_1 \vdash \textbf{return } el, \Gamma_2}$$

(T-STMAPPLY1)

$$\frac{\Gamma_1, \Pi \vdash e(el) : S, \Gamma_2}{\Gamma_1, \Pi \vdash \lfloor e(el) \rfloor_0, \Gamma_2}$$

(T-STMINVOKE1)

$$\frac{\Gamma_1, \Pi \vdash e{:}n(el) : S, \Gamma_2}{\Gamma_1, \Pi \vdash \lfloor e{:}n(el) \rfloor_0, \Gamma_2}$$

(T-NIL)

$$\Gamma_1, \Pi \vdash \textbf{nil} : \textbf{nil}, \Gamma_1$$

(T-FALSE)

$$\Gamma_1, \Pi \vdash \textbf{false} : \textbf{false}, \Gamma_1$$

(T-TRUE)

$$\Gamma_1, \Pi \vdash \textbf{true} : \textbf{true}, \Gamma_1$$

(T-INT)

$$\Gamma_1, \Pi \vdash \mathit{int} : \mathit{int}, \Gamma_1$$

(T-FLOAT)

$$\Gamma_1, \Pi \vdash \mathit{float} : \mathit{float}, \Gamma_1$$

(T-STR)

$$\Gamma_1, \Pi \vdash \mathit{string} : \mathit{string}, \Gamma_1$$

(T-IDREAD1)

$$\frac{\Gamma_1(id) = F}{\Gamma_1, \Pi \vdash id : close(F), \Gamma_1[id \mapsto open(F)]}$$

(T-IDREAD2)

$$\frac{\Gamma_1(id) = F}{\Gamma_1, \Pi \vdash id : fix(F), \Gamma_1[id \mapsto fix(F)]}$$

(T-IDREAD3)

$$\frac{\Gamma_1(id) = \phi(F_1, F_2)}{\Gamma_1, \Pi \vdash id : F_2, \Gamma_1}$$

(T-IDREAD4)

$$\frac{\Gamma_1(id) = \pi_i^x}{\Gamma_1, \Pi \vdash id : proj(\Pi(x), i), \Gamma_1}$$

(T-INDEXREAD1)

$$\frac{\Gamma_1(id) = \{\overline{F{:}V}\} \quad \Gamma_1, \Pi \vdash e_2 : F, \Gamma_2 \quad \exists i \in 1..n \; F \lesssim F_i \quad n = |\overline{F{:}V}|}{\Gamma_1, \Pi \vdash id[e_2] : rconst(V_i), \Gamma_2}$$

(T-INDEXREAD2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \{\overline{F{:}V}\}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3 \quad \exists i \in 1..n \; F \lesssim F_i \quad n = |\overline{F{:}V}|}{\Gamma_1, \Pi \vdash e_1[e_2] : rconst(V_i), \Gamma_3}$$

(T-INDEXREAD3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1[e_2] : \mathbf{any}, \Gamma_3}$$

(T-COERCE1)

$$\frac{\Gamma_1(id) <: F \quad tag(F, closed)}{\Gamma_1, \Pi \vdash <F> id : F, \Gamma_1[id \mapsto reopen(F)]}$$

(T-COERCE2)

$$\frac{\Gamma_1(id) <: F \quad tag(F, fixed)}{\Gamma_1, \Pi \vdash <F> id : F, \Gamma_1[id \mapsto F]}$$

(T-FUNCTION1)

$$\frac{\begin{array}{c} closeall(\Gamma_1)[\overline{id \mapsto F}], \Pi[\rho \mapsto S] \vdash s, \Gamma_2 \\ \Gamma_3 = openset(\Gamma_1, frv(\textbf{fun } (\overline{id{:}F}){:}S\ s)) \\ \Gamma_4 = closeset(\Gamma_3, fav(\textbf{fun } (\overline{id{:}F}){:}S\ s)) \end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } (\overline{id{:}F}){:}S\ s : F_1 \times ... \times F_n \times \textbf{nil}* \to S, \Gamma_4}$$

(T-FUNCTION2)

$$\frac{\begin{array}{c} closeall(\Gamma_1)[\overline{id \mapsto F}, ... \mapsto F], \Pi[\rho \mapsto S] \vdash s, \Gamma_2 \\ \Gamma_3 = openset(\Gamma_1, frv(\textbf{fun } (\overline{id{:}F}){:}S\ s)) \\ \Gamma_4 = closeset(\Gamma_3, fav(\textbf{fun } (\overline{id{:}F}){:}S\ s)) \end{array}}{\Gamma_1, \Pi \vdash \textbf{fun } (\overline{id{:}F}, ...{:}F){:}S\ s : F_1 \times ... \times F_n \times F* \to S, \Gamma_4}$$

(T-CONSTRUCTOR1)

$$\frac{\begin{array}{c} \Gamma_1, \Pi \vdash ([e_1] = e_2)_i : (F_i, V_i), \Gamma_{i+1} \quad T = \{F_1{:}V_1, ..., F_n{:}V_n\}_{unique} \\ wf(T) \quad n = \mid \overline{[e_1] = e_2} \mid \quad \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+1}) \end{array}}{\Gamma_1, \Pi \vdash \{\ \overline{[e_1] = e_2}\ \} : T, \Gamma_f}$$

(T-CONSTRUCTOR2)

$$\frac{\begin{array}{c} \Gamma_1, \Pi \vdash ([e_1] = e_2)_i : (F_i, V_i), \Gamma_{i+1} \\ \Gamma_1, \Pi \vdash me : F_{n+1} \times ... \times F_{n+m} \times F_{n+m+1}*, \Gamma_{n+2} \\ T = \{F_1{:}V_1, ..., F_n{:}V_n, 1{:}F_{n+1}, ..., m{:}F_{n+m}, \textbf{integer}{:}F_{n+m+1} \cup \textbf{nil}\}_{unique} \\ wf(T) \quad n = \mid \overline{[e_1] = e_2} \mid \quad \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+2}) \end{array}}{\Gamma_1, \Pi \vdash \{\ \overline{[e_1] = e_2}\ \} : T, \Gamma_f}$$

(T-FIELD)

$$\frac{\Gamma_1, \Pi \vdash e_2 : V, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_1 : F, \Gamma_3}{\Gamma_1, \Pi \vdash [e_1] = e_2 : (F, vt(F, V)), \Gamma_3}$$

(T-ARITH1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \textbf{integer} \quad F_2 <: \textbf{integer}}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{integer}, \Gamma_3}$$

(T-ARITH2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \textbf{integer} \quad F_2 <: \textbf{number}}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{number}, \Gamma_3}$$

(T-ARITH3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \textbf{number} \quad F_2 <: \textbf{integer}}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{number}, \Gamma_3}$$

(T-ARITH4)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \textbf{number} \quad F_2 <: \textbf{number}}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{number}, \Gamma_3}$$

(T-ARITH5)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \textbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{any}, \Gamma_3}$$

(T-ARITH6)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : \textbf{any}, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 + e_2 : \textbf{any}, \Gamma_3}$$

(T-CONCAT1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \textbf{string} \quad F_2 <: \textbf{string}}{\Gamma_1, \Pi \vdash e_1 \mathbin{..} e_2 : \textbf{string}, \Gamma_3}$$

(T-CONCAT2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \textbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \mathbin{..} e_2 : \textbf{any}, \Gamma_3}$$

(T-CONCAT3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : \textbf{any}, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \mathbin{..} e_2 : \textbf{any}, \Gamma_3}$$

(T-EQUAL)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 == e_2 : \mathbf{boolean}, \Gamma_3}$$

(T-ORDER1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \mathbf{number} \quad F_2 <: \mathbf{number}}{\Gamma, \Pi \vdash e_1 < e_2 : \mathbf{boolean}, \Gamma_3}$$

(T-ORDER2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \mathbf{string} \quad F_2 <: \mathbf{string}}{\Gamma_1, \Pi \vdash e_1 < e_2 : \mathbf{boolean}}$$

(T-ORDER3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 < e_2 : \mathbf{any}, \Gamma_3}$$

(T-ORDER4)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : \mathbf{any}, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 < e_2 : \mathbf{any}, \Gamma_3}$$

(T-BITWISE1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad F_1 <: \mathbf{integer} \quad F_2 <: \mathbf{integer}}{\Gamma_1, \Pi \vdash e_1 \,\&\, e_2 : \mathbf{integer}, \Gamma_3}$$

(T-BITWISE2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \,\&\, e_2 : \mathbf{any}, \Gamma_3}$$

(T-BITWISE3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : \mathbf{any}, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \,\&\, e_2 : \mathbf{any}, \Gamma_3}$$

(T-AND1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{nil}, \Gamma_2}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{and}\ e_2 : \mathbf{nil}, \Gamma_2}$$

(T-AND2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{false}, \Gamma_2}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{and}\ e_2 : \mathbf{false}, \Gamma_2}$$

(T-AND3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{nil} \cup \mathbf{false}, \Gamma_2}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{and}\ e_2 : \mathbf{nil} \cup \mathbf{false}, \Gamma_2}$$

(T-AND4)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad \mathbf{nil} \not\lesssim F_1 \quad \mathbf{false} \not\lesssim F_1}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{and}\ e_2 : F_2, \Gamma_3}$$

(T-AND5)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{and}\ e_2 : F_1 \cup F_2, \Gamma_3}$$

(T-OR1)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F, \Gamma_2 \quad \mathbf{nil} \not\lesssim F \quad \mathbf{false} \not\lesssim F}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{or}\ e_2 : F, \Gamma_2}$$

(T-OR2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{nil}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{or}\ e_2 : F, \Gamma_3}$$

(T-OR3)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{false}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{or}\ e_2 : F, \Gamma_3}$$

(T-OR4)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{nil} \cup \mathbf{false}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{or}\ e_2 : F, \Gamma_3}$$

(T-OR5)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3}{\Gamma_1, \Pi \vdash e_1 \ \mathbf{or}\ e_2 : filter(filter(F_1, \mathbf{nil}), \mathbf{false}) \cup F_2, \Gamma_3}$$

(T-NOT1)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{nil}, \Gamma_2}{\Gamma_1, \Pi \vdash \mathbf{not}\ e : \mathbf{true}, \Gamma_2}$$

(T-NOT2)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{false}, \Gamma_2}{\Gamma_1, \Pi \vdash \mathbf{not}\ e : \mathbf{true}, \Gamma_2}$$

(T-NOT3)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{nil} \cup \mathbf{false}, \Gamma_2}{\Gamma_1, \Pi \vdash \mathbf{not}\ e : \mathbf{true}, \Gamma_2}$$

(T-NOT4)

$$\frac{\Gamma_1, \Pi \vdash e : F \quad \textbf{nil} \not\lesssim F \quad \textbf{false} \not\lesssim F}{\Gamma_1, \Pi \vdash \textbf{not}\ e : \textbf{false}, \Gamma_2}$$

(T-NOT5)

$$\frac{\Gamma_1, \Pi \vdash e : F, \Gamma_2}{\Gamma_1, \Pi \vdash \textbf{not}\ e : \textbf{boolean}, \Gamma_2}$$

(T-LEN1)

$$\frac{\Gamma_1, \Pi \vdash e : F, \Gamma_2 \quad F <: \textbf{string}}{\Gamma_1, \Pi \vdash \#\ e : \textbf{integer}, \Gamma_2}$$

(T-LEN2)

$$\frac{\Gamma_1, \Pi \vdash e : F, \Gamma_2 \quad F <: \{\}_{closed}}{\Gamma_1, \Pi \vdash \#\ e : \textbf{integer}, \Gamma_2}$$

(T-LEN3)

$$\frac{\Gamma_1, \Pi \vdash e : \textbf{any}, \Gamma_2}{\Gamma_1, \Pi \vdash \#\ e : \textbf{any}, \Gamma_2}$$

(T-EXPAPPLY1)

$$\frac{\Gamma_1, \Pi \vdash e(el) : S, \Gamma_2}{\Gamma_1, \Pi \vdash \lfloor e(el) \rfloor_1 : proj(S, 1), \Gamma_2}$$

(T-EXPINVOKE1)

$$\frac{\Gamma_1, \Pi \vdash e{:}n(el) : S, \Gamma_2}{\Gamma_1, \Pi \vdash \lfloor e{:}n(el) \rfloor_1 : proj(S, 1), \Gamma_2}$$

(T-EXPDOTS)

$$\frac{\Gamma_1, \Pi \vdash ... : F*, \Gamma_2}{\Gamma_1, \Pi \vdash \lfloor ... \rfloor_1 : F \cup \textbf{nil}, \Gamma_2}$$

(T-IDWRITE1)

$$\frac{\Gamma_1(id) = F}{\Gamma_1, \Pi \vdash id_l : F, \Gamma_1}$$

(T-IDWRITE2)

$$\frac{\Gamma_1(id) = \phi(F_1, F_2)}{\Gamma_1, \Pi \vdash id_l : F_1, \Gamma_1[id \mapsto F_1]}$$

(T-INDEXWRITE1)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash e_1 : \{\overline{F{:}V}\}, \Gamma_2 \\ \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3 \quad \exists i \in 1..n \; F \lesssim F_i \wedge \neg const(V_i) \quad n = |\overline{F{:}V}|\end{array}}{\Gamma_1, \Pi \vdash e_1[e_2]_l : V_i, \Gamma_3}$$

(T-INDEXWRITE2)

$$\frac{\Gamma_1, \Pi \vdash e_1 : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F, \Gamma_3}{\Gamma_1, \Pi \vdash e_1[e_2]_l : \mathbf{any}, \Gamma_3}$$

(T-REFINE1)

$$\frac{\begin{array}{c}\Gamma_1(id) = \{\overline{F{:}V}\}_{unique} \\ \Gamma_1, \Pi \vdash e : F_{new}, \Gamma_2 \quad \nexists i \in 1..n \; F_{new} \lesssim F_i \quad V_{new} = vt(F_{new}, V) \quad n = |\overline{F{:}V}|\end{array}}{\Gamma_1, \Pi \vdash id[e]{<}V{>} : V_{new}, \Gamma_2[id \mapsto \{\overline{F{:}V}, F_{new}{:}V_{new}\}_{unique}]}$$

(T-REFINE2)

$$\frac{\begin{array}{c}\Gamma_1(id) = \{\overline{F{:}V}\}_{open} \\ \Gamma_1, \Pi \vdash e : F_{new}, \Gamma_2 \quad \nexists i \in 1..n \; F_{new} \lesssim F_i \quad V_{new} = vt(F_{new}, V) \quad n = |\overline{F{:}V}|\end{array}}{\Gamma_1, \Pi \vdash id[e]{<}V{>} : V_{new}, \Gamma_2[id \mapsto \{\overline{F{:}V}, F_{new}{:}V_{new}\}_{open}]}$$

(T-LHSLIST)

$$\frac{\Gamma_1, \Pi \vdash l_i : F_i, \Gamma_{i+1} \quad \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+1}) \quad n = |\; \bar{l} \;|}{\Gamma_1, \Pi \vdash \bar{l} : F_1 \times ... \times F_n \times \mathbf{value}*, \Gamma_f}$$

(T-EXPLIST1)

$$\frac{\Gamma_1, \Pi \vdash e_i : F_i, \Gamma_{i+1} \quad \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+1}) \quad n = |\; \overline{e} \;|}{\Gamma_1, \Pi \vdash \overline{e} : F_1 \times ... \times F_n \times \mathbf{nil}*, \Gamma_f}$$

(T-EXPLIST2)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash e_i : F_i, \Gamma_{i+1} \quad \Gamma_1, \Pi \vdash me : F_{n+1} \times ... \times F_{n+m} \times F_{n+m+1}*, \Gamma_{n+2} \\ \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+2}) \quad n = |\; \overline{e} \;|\end{array}}{\Gamma_1, \Pi \vdash \overline{e}, me : F_1 \times ... \times F_{n+m} \times F_{n+m+1}*, \Gamma_f}$$

(T-EXPLIST3)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash e_i : F_i, \Gamma_{i+1} \quad \Gamma_1, \Pi \vdash me : S, \Gamma_{n+2} \\ S = F_{n+1} \times ... \times F_{n+m} \times \mathbf{nil}* \sqcup F'_{n+1} \times ... \times F'_{n+m} \times \mathbf{nil}* \\ \Gamma_f = merge(\Gamma_1, ..., \Gamma_{n+2}) \quad n = \mid \overline{e} \mid\end{array}}{\Gamma_1, \Pi \vdash \overline{e}, me : F_1 \times ... \times F_n \times \pi_1^x \times ... \times \pi_m^x \times \mathbf{nil}*, \Gamma_f, (x, S)}$$

(T-APPLY1)

$$\frac{\Gamma_1, \Pi \vdash e : S_1 \to S_2, \Gamma_2 \quad \Gamma_2, \Pi \vdash el : S_3, \Gamma_3 \quad S_3 \lesssim S_1}{\Gamma_1, \Pi \vdash e(el) : S_2, \Gamma_3}$$

(T-APPLY2)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash el : S, \Gamma_3}{\Gamma_1, \Pi \vdash e_1(el) : \mathbf{any}*, \Gamma_3}$$

(T-INVOKE1)

$$\frac{\begin{array}{c}\Gamma_1, \Pi \vdash e : F, \Gamma_2 \\ \Gamma_2[\sigma \mapsto F], \Pi \vdash e[id] : \mathbf{const}\ S_1 \to S_2, \Gamma_3 \\ \Gamma_3[\sigma \mapsto F], \Pi \vdash el : S_3, \Gamma_4 \quad F \times S_3 \lesssim [\mathbf{self} \mapsto F]S_1\end{array}}{\Gamma_1, \Pi \vdash e{:}id(el) : [\mathbf{self} \mapsto F]S_2, \Gamma_4}$$

(T-INVOKE2)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{any}, \Gamma_2 \quad \Gamma_2, \Pi \vdash el : S, \Gamma_3}{\Gamma_1, \Pi \vdash e{:}id(el) : \mathbf{any}*, \Gamma_3}$$

(T-DOTS)

$$\frac{\Gamma_1(...) = F}{\Gamma_1, \Pi \vdash ... : F*, \Gamma_1}$$

(T-SELF)

$$\frac{\Gamma_1, \Pi \vdash e : \mathbf{self}, \Gamma_2 \quad \Gamma_2(\sigma) = F}{\Gamma_1, \Pi \vdash e : F, \Gamma_2}$$

(T-SETMETATABLE1)

$$\frac{\Gamma_1(id) = \mathbf{self}}{\Gamma_1, \Pi \vdash setmetatable(\{\}, \{[\text{“}\_\_index\text{”}] = id\}) : \mathbf{self}, \Gamma_1}$$

(T-SETMETATABLE2)

$$\frac{\Gamma_1(id) = \{F_1{:}V_1, ..., F_n{:}V_n\}_{fixed}}{\Gamma_1, \Pi \vdash setmetatable(\{\}, \{[\text{“}\_\_index\text{”}] = id\}) : \{F_1{:}V_1, ..., F_n{:}V_n\}_{open}, \Gamma_1}$$

(T-SETMETATABLE3)

$$\frac{\Gamma_1, \Pi \vdash e : T, \Gamma_2 \quad T = \{F_1{:}V_1, ..., F_n{:}V_n\}_{closed} \quad \Gamma_1(id) = \textbf{self} \quad \Gamma_1(\sigma) <: T}{\Gamma_1, \Pi \vdash setmetatable(e, \{[\text{“}\_\_index\text{”}] = id\}) : \textbf{self}, \Gamma_2[\sigma \mapsto T]}$$

(T-UNFOLD)

$$\frac{\Gamma_1, \Pi \vdash e : \mu x.F, \Gamma_2}{\Gamma_1, \Pi \vdash e : [x \mapsto \mu x.F]F, \Gamma_2}$$

(T-FOLD)

$$\frac{\Gamma_1, \Pi \vdash e : [x \mapsto \mu x.F]F, \Gamma_2}{\Gamma_1, \Pi \vdash e : \mu x.F, \Gamma_2}$$

(T-TERNARY)

$$\frac{\Gamma_1, \Pi \vdash e_1 : F_1, \Gamma_2 \quad \Gamma_2, \Pi \vdash e_2 : F_2, \Gamma_3 \quad \Gamma_3, \Pi \vdash e_3 : F_2, \Gamma_4}{\Gamma_1, \Pi \vdash e_1 \textbf{ and } e_2 \textbf{ or } e_3 : F_2, \Gamma_4}$$

C.3 Auxiliary functions

$$\begin{aligned}
wf(\{\overline{F : V}\}_{unique|open|fixed|closed}) = \forall i \; ((\nexists j \; i \neq j \; \wedge \; F_i \lesssim F_j) \; \wedge \; wf(V_i) \wedge \\
\neg tag(V_i, unique) \; \wedge \; \neg tag(V_i, open)) \\
wf(\textbf{const } F) = wf(F) \\
wf(F_1 \cup F_2) = wf(F_1) \; \wedge \; wf(F_2) \\
wf(\mu x.F) = wf(F) \\
wf(S_1 \to S_2) = wf(S_1) \; \wedge \; wf(S_2) \\
wf(S_1 \sqcup S_2) = wf(S_1) \; \wedge \; wf(S_2) \\
wf(F*) = wf(F) \\
wf(F \times P) = wf(F) \; \wedge \; wf(P) \\
wf(F) = \top \quad \text{for all other cases}
\end{aligned}$$

$$\begin{aligned}
tag(F_1 \cup F_2, t) = tag(F_1, t) \; \vee \; tag(F_2, t) \\
tag(\{\overline{F{:}V}\}_t, t) = \top \\
tag(\{\overline{F{:}V}\}_{t_1}, t_2) = \bot \\
tag(F, t) = \bot
\end{aligned}$$

$$
\begin{aligned}
vt(L, V) &= fix(V) \\
vt(F_1, F_2) &= nil(fix(F_2)) \\
vt(F_1, \textbf{const } F_2) &= \textbf{const } nil(fix(F_2))
\end{aligned}
$$

$$
nil(T) = \begin{cases} T & \text{if } \textbf{nil} \lesssim T \\ T \cup \textbf{nil} & \text{otherwise} \end{cases}
$$

$$
\begin{aligned}
fix(F_1 \cup F_2) &= fix(F_1) \cup fix(F_2) \\
fix(\{\overline{F{:}V}\}_{unique|open}) &= \{\overline{F{:}V}\}_{fixed} \\
fix(F) &= F
\end{aligned}
$$

$$
\begin{aligned}
close(F_1 \cup F_2) &= close(F_1) \cup close(F_2) \\
close(\{\overline{F{:}V}\}_{unique|open}) &= \{\overline{F{:}V}\}_{closed} \\
close(F) &= F
\end{aligned}
$$

$$
\begin{aligned}
open(F_1 \cup F_2) &= open(F_1) \cup open(F_2) \\
open(\{\overline{F{:}V}\}_{unique}) &= \{\overline{F{:}V}\}_{open} \\
open(F) &= F
\end{aligned}
$$

$$
\begin{aligned}
reopen(\{\overline{F : V}\}_{closed}) &= \{\overline{F : V}\}_{open} \\
reopen(F) &= F
\end{aligned}
$$

$$
\begin{aligned}
rconst(\textbf{const } F) &= F \\
rconst(F) &= F
\end{aligned}
$$

$$const(\textbf{const}\ F) = \top$$
$$const(F) = \bot$$

$$proj(S_1 \sqcup S_2, i) = proj(S_1, i) \cup proj(S_2, i)$$
$$proj(F*, i) = nil(F)$$
$$proj(F \times P, 1) = F$$
$$proj(F \times P, i) = proj(P, i-1)$$
$$proj(E*, i) = nil(E)$$
$$proj(T \times E, 1) = T$$
$$proj(T \times E, i) = proj(E, i-1)$$

$$infer(T_1 \times ... \times T_n*, i) = \begin{cases} general(T_i) & \text{if } i < n \\ general(nil(T_n)) & \text{if } i >= n \end{cases}$$

$$general(\textbf{false}) = \textbf{boolean}$$
$$general(\textbf{true}) = \textbf{boolean}$$
$$general(int) = \textbf{integer}$$
$$general(float) = \textbf{number}$$
$$general(string) = \textbf{string}$$
$$general(F_1 \cup F_2) = general(F_1) \cup general(F_2)$$
$$general(S_1 \to S_2) = general2(S_1) \to general2(S_2)$$
$$general(\{F_1{:}V_1, ..., F_n{:}V_n\}_{tag}) = \{F_1{:}general(V_1), ..., F_n{:}general(V_n)\}_{tag}$$
$$general(\mu x.F) = \mu x.general(F)$$
$$general(T) = T$$

$$general2(F*) = general(F)*$$
$$general2(F \times P) = general(F) \times general2(P)$$
$$general2(S_1 \sqcup S_2) = general2(S_1) \sqcup general2(S_2)$$

$$closeall(\Gamma[id_1 \mapsto T_1, ..., id_n \mapsto T_n]) = \Gamma[id_1 \mapsto close(T_1), ..., id_n \mapsto close(T_n)]$$

$$closeset(\Gamma, \{id_1, ..., id_n\}) = \Gamma[id_1 \mapsto close(\Gamma(id_1)), ..., id_n \mapsto close(\Gamma(id_n))]$$

$$openset(\Gamma, \{id_1, ..., id_n\}) = \Gamma[id_1 \mapsto open(\Gamma(id_1)), ..., id_n \mapsto open(\Gamma(id_n))]$$

$$\begin{aligned}
merge(\overline{\Gamma}) &= reduce(\overline{\Gamma}, merge2) \\
merge2(\Gamma_1, \Gamma_2) &= \{\overline{(id, merget(\Gamma_1(id), \Gamma_2(id))}\} \\
merget(T_1, T_2) &= T_1 \quad \text{if } T_2 \lesssim T_1 \\
merget(T_1, T_2) &= T_2 \quad \text{if } T_1 \lesssim T_2 \\
&\quad \text{the next case applies if} \\
&\quad \overline{V^l \lesssim_u V_r \vee V^r \lesssim_u V_l} \\
&\quad \text{and the right side is } wf \\
merget(\{\overline{F : V^l}, \overline{F' : V'}\}_{unique}, & \\
\{\overline{F : V^r}, \overline{F'' : V''}\}_{unique}) &= \{\overline{F : sup_u(V^l, V^r)}, \\
&\quad \overline{F' : V'}, \\
&\quad \overline{F'' : V''}\}_{unique} \\
&\quad \text{the next case applies if} \\
&\quad \overline{V^l \lesssim_c V_r \vee V^r \lesssim_c V_l} \\
&\quad \text{and the right side is } wf \\
merget(\{\overline{F : V^l}, \overline{F' : V'}\}_{unique|open}, & \\
\{\overline{F : V^r}, \overline{F'' : V''}\}_{unique|open}) &= \{\overline{F : sup_c(V^l, V^r)}, \\
&\quad \overline{F' : V'}, \\
&\quad \overline{F'' : V''}\}_{open} \\
merget(T_1, T_2) &= \bot \quad \text{otherwise}
\end{aligned}$$

$$\begin{aligned}
sup_u(V_1, V_2) &= V_2 \quad \text{if } V_1 \lesssim_u V_2 \\
sup_u(V_1, V_2) &= V_1 \quad \text{if } V_2 \lesssim_u V_1
\end{aligned}$$

$$sup_c(V_1, V_2) = V_2 \quad \text{if } V_1 \lesssim_c V_2$$
$$sup_c(V_1, V_2) = V_1 \quad \text{if } V_2 \lesssim_c V_1$$

$$join(\Gamma_1, \Gamma_2) = \{\overline{(id, joint(\Gamma_1(id), \Gamma_2(id))}\}$$
$$joint(T_1, T_2) = T_1 \quad \text{if } T_2 \lesssim T_1$$
$$joint(T_1, T_2) = T_2 \quad \text{if } T_1 \lesssim T_2$$
$$\text{the next case applies if}$$
$$\overline{V^l \lesssim_u V_r} \vee \overline{V^r \lesssim_u V_l}$$
$$\text{and the right side is } wf$$
$$joint(\{\overline{F : V^l}, \overline{F' : V'}\}_{unique}, \{\overline{F : V^r}, \overline{F'' : V''}\}_{unique}) = \{\overline{F : sup_u(V^l, V^r)},$$
$$\overline{F' : nil(V')},$$
$$\overline{F'' : nil(V'')}\}_{unique}$$
$$joint(T_1, T_2) = \bot \quad \text{otherwise}$$

$$filter(F_1 \cup F_2, F_1) = filter(F_2, F_1)$$
$$filter(F_1 \cup F_2, F_2) = filter(F_1, F_2)$$
$$filter(F_1 \cup F_2, F_3) = filter(F_1, F_3) \cup filter(F_2, F_3)$$
$$filter(F_1, F_2) = F_1$$

$$fopt(P_1 \sqcup P_2, F, i) = P_2 \quad \text{if } fot(proj(P_1, i), F) = \textbf{void}$$
$$fopt(P_1 \sqcup P_2, F, i) = P_1 \quad \text{if } fot(proj(P_2, i), F) = \textbf{void}$$
$$fopt(P_1 \sqcup P_2, F, i) = P_1 \sqcup P_2 \quad \text{otherwise}$$
$$fopt(P \sqcup S, F, i) = fopt(S, F, i) \quad \text{if } fot(proj(P, i), F) = \textbf{void}$$
$$fopt(P \sqcup S, F, i) = P \sqcup fopt(S, F, i) \quad \text{otherwise}$$
$$fopt(S \sqcup P_2, F, i) = fopt(S, F, i) \quad \text{if } fot(proj(P, i), F) = \textbf{void}$$
$$fopt(S \sqcup P, F, i) = fopt(S, F, i) \sqcup P \quad \text{otherwise}$$
$$fopt(S_1 \sqcup S_2, F, i) = fopt(S_1, F, i) \sqcup fopt(S_2, F, i)$$

$$
\begin{aligned}
fot(F_1 \cup F_2, F_3) &= fot(F_1, F_3) && \text{if } fot(F_2, F_3) = \textbf{void} \\
fot(F_1 \cup F_2, F_3) &= fot(F_2, F_3) && \text{if } fot(F_1, F_3) = \textbf{void} \\
fot(F_1 \cup F_2, F_3) &= fot(F_1, F_3) \cup fot(F_2, F_3) && \text{otherwise} \\
fot(F_1, F_2) &= \textbf{void} && \text{if } F_1 <: F_2 \text{ and } F_2 <: F_1 \\
fot(F_1, F_2) &= F_1 && \text{otherwise}
\end{aligned}
$$

$$
\begin{aligned}
fipt(P_1 \sqcup P_2, F, i) &= P_2 && \text{if } fit(proj(P_1, i), F) = \textbf{void} \\
fipt(P_1 \sqcup P_2, F, i) &= P_1 && \text{if } fit(proj(P_2, i), F) = \textbf{void} \\
fipt(P_1 \sqcup P_2, F, i) &= P_1 \sqcup P_2 && \text{otherwise} \\
fipt(P \sqcup S, F, i) &= fipt(S, F, i) && \text{if } fit(proj(P, i), F) = \textbf{void} \\
fipt(P \sqcup S, F, i) &= P \sqcup fipt(S, F, i) && \text{otherwise} \\
fipt(S \sqcup P_2, F, i) &= fipt(S, F, i) && \text{if } fit(proj(P, i), F) = \textbf{void} \\
fipt(S \sqcup P, F, i) &= fipt(S, F, i) \sqcup P && \text{otherwise} \\
fipt(S_1 \sqcup S_2, F, i) &= fipt(S_1, F, i) \sqcup fipt(S_2, F, i)
\end{aligned}
$$

$$
\begin{aligned}
fit(F_1 \cup F_2, F_3) &= fit(F_1, F_3) && \text{if } fit(F_2, F_3) = \textbf{void} \\
fit(F_1 \cup F_2, F_3) &= fit(F_2, F_3) && \text{if } fit(F_1, F_3) = \textbf{void} \\
fit(F_1 \cup F_2, F_3) &= fit(F_1, F_3) \cup fit(F_2, F_3) && \text{otherwise} \\
fit(F_1, F_2) &= F_1 && \text{if } F_1 <: F_2 \text{ and } F_2 <: F_1 \\
fit(F_1, F_2) &= \textbf{void} && \text{otherwise}
\end{aligned}
$$