Goals of the Luau Type System, Two Years On

LILY BROWN, ANDY FRIESEN, and ALAN JEFFREY, Roblox, USA

In HATRA 2021, we presented *The Goals Of The Luau Type System*, describing the human factors issues with designing a type system for a language with a heterogeneous developer community. In this extended abstract we provide a progress report on how the design is being delivered.

1 RECAP

Luau [7] is the scripting language used by the Roblox [8] platform for shared 3D experiences. Luau extends the Lua [5] language, notably by providing type-driven tooling such as autocomplete and API documentation (as well as traditional type error reporting). Roblox has hundreds of millions of users, and millions of creators, ranging from children learning to program for the first time to professional development studios.

In HATRA 2021, we presented *The Goals Of The Luau Type System* [1], describing the human factors issues with designing a type system for a language with a heterogeneous developer community. The design flows from the needs of the different communities: beginners are focused on immediate goals ("the stairs should light up when a player walks on them") and less on the code quality concerns of more experienced developers; for all users type-driven tooling is important for productivity. These needs result in a design with two modes:

- *non-strict mode*, aimed at non-professionals, focused on minimizing false positives, and
- *strict mode*, aimed at professionals, focused on minimizing false negatives (i.e. type soundness).

For both communities, type-driven tooling is important, so we provide *infallible type inference*, which infers types no matter the inputs.

2 NEW STUFF

The main improvement is that Luau now has support for *semantic subtyping* [4]. Semantic subtyping interprets types as sets of values, and subtyping as set inclusion [3]. This is aligned with the *minimize false positives* goal of Luau non-strict mode, since semantic subtyping only reports a failure of subtyping when there is a value which inhabits the candidate subtype, but not the candidate supertype. This has the added benefit of improving error reporting in the prototype implementation: since the prototype is constructive, the witness for the failure of subtyping can help drive error reports.

The other big improvement is how we view gradual typing, Rather than the approach of [9], which uses *consistent subtyping* where any $\lesssim T \lesssim$ any for any type T, we adopt an approach based on *error suppression*, where any is an error-suppressing type, and any failures of subtyping involving error-suppressing types are not reported. Users can explicitly suppress type errors by declaring variables with type any, and since an expression with a type error has an error-suppressing type we avoid cascading errors. Error-suppression is in production Luau, and is mechanically verified [2].

3 FURTHER WORK

Currently the type inference system uses greedy inference, which is very sensitive to order of declarations, and can easily result in false positives. This needs replaced by some form of local type inference [6].

Currently, non-strict mode operates in the style of gradual type systems by inferring any as the type for local variables. This does not play well with type-directed tooling, for example any cannot provide autocomplete suggestions. We plan to improve non-strict mode's type inference and interaction with error-suppression.

REFERENCES

[1] L. Brown, A. Friesen, and A. S. A. Jeffrey. 2021. Goals of the Luau Type System. In *Proc. Human Aspects of Types and Reasoning Assistants*. https://asaj.org/papers/hatra21.pdf

[2] L. Brown and A. S. A. Jeffrey. 2023. Luau Prototype Typechecker. https://github.com/luau-lang/agda-typeck

[3] G. Castagna and A. Frisch. 2005. A Gentle Introduction to Semantic Subtyping. In *Proc. Principles and Practice of Declarative Programming*.

[4] A. S. A. Jeffrey. 2022. Semantic Subtyping in Luau. Roblox Technical Blog. https://blog.roblox.com/2022/11/semantic-subtyping-luau/

[5] Lua.org and PUC-Rio. 2023. The Lua Programming Language. https://lua.org

[6] B. C. Pierce and D. N. Turner. 2000. Local Type Inference. *ACM Trans. Program. Lang. Syst.* 22, 1 (2000), 1–44.

[7] Roblox. 2023. The Luau Programming Language. https://luau-lang.org

[8] Roblox. 2023. Reimagining the way people come together. https://corp.roblox.com

[9] J. G. Siek and W. Taha. 2007. Gradual Typing for Objects. In *Proc. European Conf Object-Oriented Programming*. 2–27.