Goals of the Luau Type System

Lily Brown Andy Friesen Alan Jeffrey

ROBLOX

Human Aspects of Types and Reasoning Assistants 2021

Creator Goals

Roblox

A platform for creating shared immersive 3D experiences:

- **Many**: 20 million experiences, 8 million creators.
- **At scale**: e.g. *Adopt Me!* has 10 billion plays.
- **Young**: 200+ kids' coding camps in 65+ countries.
- **Professional**: 345k creators monetizing experiences.

A very heterogenous community.

Roblox developer community

All developers are important:

- **Learners**: energetic creative community.
- **Professionals**: high-quality experiences.
- **Everyone inbetween**: some learners become professionals!

Satisfying everyone is sometimes challenging.

Roblox Studio

Demo time!

Learners have immediate goals

E.g. “when a player steps on the button, advance the slide”.

- Most goals can be met by the 3D environment editor.
- Some need programming, e.g. reacting to collisions or timers.
- Very different onboarding than “let’s learn to program”.
- “Google Stack Overflow” is a common workflow.
- Type-driven tools (e.g. autocomplete or API help) are useful.
- Some type errors (e.g. catching typos) are useful, but many are not.

Type systems should help or get out of the way.

Professionals have long-term goals

E.g. “decrease user churn” or “improve frame rate”.

- Code planning
- Code refactoring
- Defect detection

Type-driven development is a useful technique!

Luau Type System

Infallible types

Goal: support type-driven tools (e.g. autocomplete) for all programs.

- Traditional typing judgment: $\Gamma \vdash M : T$
- Infallible judgment: $\Gamma \vdash M \Rightarrow N : T$, where N may flag type errors
- For every M and Γ, there are N and T such that $\Gamma \vdash M \Rightarrow N : T$.

Related work:

- Type error reporting, program repair.
- Typed holes (e.g. in Hazel).

Strict types

Goal: no false negatives.

- *Strict mode* enabled by developers who want defect detection.
- *Business as usual* soundness via progress + preservation.
- *Gradual types* for programs with flagged type errors.

Related work:

- Lots and lots for type safety.
- Gradual typing, blame analysis, migratory types…

Nonstrict types

Goal: no false positives.

- *Nonstrict mode* enabled by developers who want type-drive tools.
- Not even obvious how to state the goals!
- A shot at it: a program is *incorrectly flagged* if it contains a flagged value (i.e. a flagged program has successfully terminated).
- Is progress + correct flagging what we want?

Related work:

- Success types (e.g. Erlang Dialyzer).
- Incorrectness Logic.

Mixing types

Goal: *support mixed strict/nonstrict development*.

- Strict/nonstrict mode is enabled per-module.
- What happens when a codebase is mixed?
- Combine progress + preservation with progress + correct flagging?

Related work:

- Not much?

Type inference

Goal: *provide benefits of type-directed tools to everyone*.

- Infer types for all variables, don't just give them type any.
- Luau includes System F, so everything is undecidable. Yay heuristics!
- Currently, strict and nonstrict mode infer different types. Boo!

Related work:

- Lots.

Thank you!

Roblox is hiring!