minimalistic typed Lua is here

Hisham Muhammad hisham@konghq.com>

minimalism versus types

Hisham Muhammad <hisham@konghq.com>







0:21 / 22:25

minimalistic

experimental

emerging







untyped: no types at all

assembly, un(i)typed lambda calculus

typed: types exist!

string and number are different things (even if you can do "1" + 2)

dynamically typed: values have types, variables don't

Lua, Scheme, JavaScript, Python, Ruby, PHP, etc.

statically typed:

values have types, variables have types

C, Java, Go, C#, Rust, Haskell, etc.

Python → mypy, pytype
Ruby → Sorbet
PHP → Hack

JavaScript → TypeScript
Racket → Typed Racket
etc.

Lua?

adding types (or anything!) makes a language larger

- conceptually
- and in implementation

adding types (or anything!) makes a language larger

-conceptually

- and in implementation

adding types (or anything!) makes a language larger

-conceptually

- and in implementation

if the language grows too much, it doesn't feel like Lua anymore

if the language grows too much, it doesn't feel like Lua anymore

if the type checker is too simplistic, it doesn't feel like Lua anymore

but we want both:

a small language that fits in your head

a type checker that catches when you make a silly typo

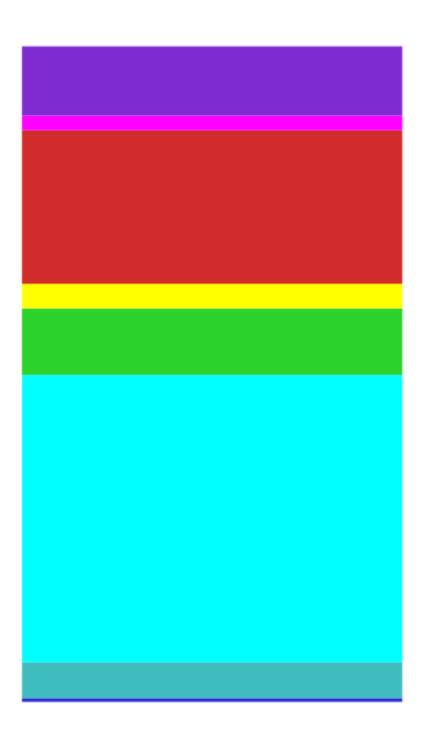
the challenge: to find the sweet spot between minimalism and functionality

tl

minimal implementation in the Lua spirit:

Lua: 297 kB tarball pure standard C, no dependencies

tl: single file, currently 4806 lines pure Lua, no dependencies



- lexer
- lexer pretty-printer
- parser
- AST traversal
- AST pretty-printer
- type checker
- standard library types
- loader

no dependencies: drop tl.lua in your Lua project and off you go

tl check file.tl →

tl gen file.tl → file.lua

tl run file.tl

two modes:

.tl ("strict" mode)

.lua ("lax" mode)

```
function f(x)
    return x
end
local z = f(0)
```

```
function f(x: number): number
  return x
end
local z = f(0)
```

tl reports errors and unknowns separately

type checker: the bulk of the compiler

```
function keys(t: {string: string}): {string}
  local ks = {}
  for k, v in pairs(t) do
      table.insert(ks, k)
  end
  return ks
end
```

types of tables

what is a Lua table?

maps, like {string:boolean}

maps, like {string:boolean}
array, like {string}

maps, like {string:boolean}
array, like {string}
record, like Point

maps, like {string:boolean}
 array, like {string}
 record, like Point
 array-record, like Node

maps, like {string:boolean}
 array, like {string}
 record, like Point
 array-record, like Node
 array-map? not yet

nominal records

```
Point = record
```

x: number

y: number

end

no inheritance or interfaces/traits (for now?)

with dynamic types, it's trivial to write very generic code

```
function keys(t: {`K: `V}): {`K}
  local ks = {}
  for k, v in pairs(t) do
     table.insert(ks, k)
  end
  return ks
end
```

prioritizing practical needs over a feature checklist

yay, types! now what?

which errors are left?

```
oops.lua:279: attempt to index a nil value (field '?')
stack traceback:
  oops.lua:279: in function 'oh_no'
  oops.lua:12: in function 'not_again'
  oops.lua:490: in function 'main'
  [C]: in ?
```

tl (and Lua): any variable may be nil

option types?

Maybe in Haskell,
Result in Rust,
etc...

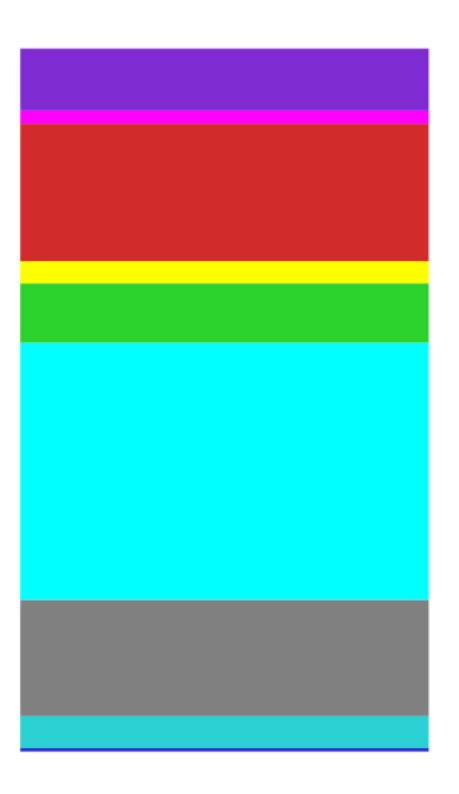
trickier for Lua:

every t[x] returns an option type? nah

...have the compiler detect it?







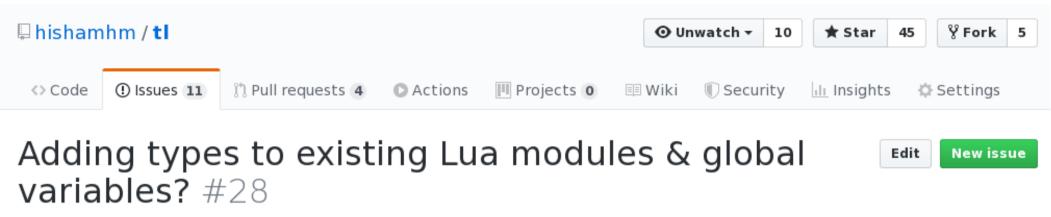
- lexer
- lexer pretty-printer
- parser
- AST traversal
- AST pretty-printer
- type checker
- flow analysis
- standard library types
- loader

dug out of the rabbit hole!

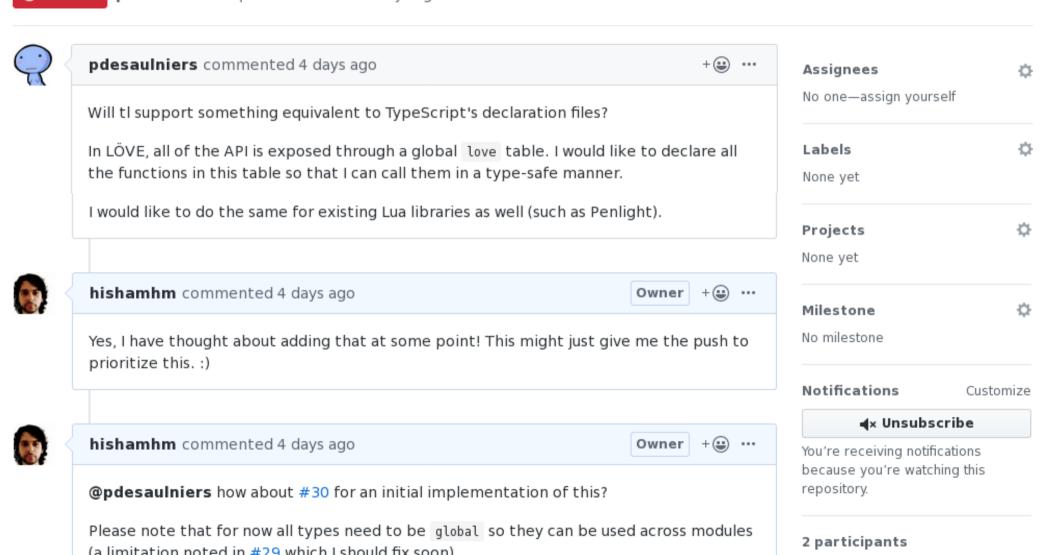
...by the FOSDEM deadline and by user feedback!

practical issues!

11 Open ✓ 16 Closed Author ▼ Label ▼ Projects ▼ Milestones ▼ Assignee ▼	Sort ▼
Union types? #40 opened 2 days ago by pdesaulniers	P 1
Method definition on record imported from declaration file does not throw an error? #39 opened 2 days ago by pdesaulniers	□ 2
Function overloading in record definitions #36 opened 3 days ago by pdesaulniers	Г 3
How to load declaration files that do not correspond to Lua modules #35 opened 3 days ago by hishamhm	□ 2
Convenient way of generating Lua files for every tl file? #31 opened 3 days ago by pdesaulniers	
missing support for exported types #29 opened 4 days ago by hishamhm	
name idea(s) #25 opened on Nov 24, 2019 by akavel	₽1
Would be nice to have more info about the project and it's goals #24 opened on Nov 22, 2019 by ryanford-frontend	₽1
in Lua mode, warn on assignment of literal with extra fields to a record type	





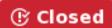


definition files

require("socket")

when typechecking, load socket.d.tl when running, load socket.lua

Function overloading in record definitions #36



③ Closed pdesaulniers opened this issue 3 days ago · 4 comments · Fixed by #38



```
pdesaulniers commented 3 days ago • edited ▼
```

Some functions in LÖVE have multiple overloads. For instance, love.graphics.print.

Right now, it seems like tl only checks the last overload:

```
global love graphics = record
        print: function(text: string, x: number, y: number, r: number, sx: number, sy: number,
        print: function(coloredtext: {any}, x: number, y: number, r: number, sx: number, s
end
global love = record
        graphics: love graphics
end
```

```
require("love")
function love.draw()
        love.graphics.print("Hello lol", 100, 100)
end
```

```
main.tl:4:22: argument 1: got string "Hello lol", expected {any}
```

Lua has no function overloading! but it's common to fake it

challenge:

```
love.graphics.print(\{\{1,1,1,1\}, "Hello", \{1,0,0,1\}, "World"\})
```

SYNOPSIS love.graphics.print(coloredtext, x, y, angle, sx, sy, ox, oy, kx, ky) ARGUMENTS table coloredtext A table containing colors and strings to add to the object, in the form of {color1, string1, color2, string2, ...} table color1 A table containing red, green, blue, and optional alpha components to use as a color for the next string in the table, in the form of {red, green, blue, alpha}. string stringl A string of text which has a color specified by the previous color. table color2 A table containing red, green, blue, and optional alpha components to use as a color for the next string in the table, in the form of {red, green, blue, alpha}. string string2 A string of text which has a color specified by the previous color. tables and strings ... Additional colors and strings. number x (θ) The position of the text on the x-axis. number y (θ) The position of the text on the y-axis. number angle (0) The orientation of the text in radians. number sx (1) Scale factor on the x-axis. number sy (sx) Scale factor on the y-axis. number ox (θ) Origin offset on the x-axis. number oy (θ) Origin offset on the y-axis. number kx (0) Shearing / skew factor on the x-axis. number ky (θ) Shearing / skew factor on the y-axis.

1. any

1. any

2. table

- **1.** any
- 2. table
- **3.** {any}

- **1.** any
- 2. table
- **3.** {any}
- 4. {string or {number}}

- **1.** any
- 2. table
- **3.** {any}
- 4. {string or {number}}
- 5. {[i%2==1]:{number},[i%2==0]:string}

```
1. any
                 2. table
                 3. {any}
        4. {string or {number}}
5. { [i%2==1]: {number}, [i%2==0]: string}
    6. {[i\%2==1]:({number}|len==4),
            [i\%2==0]:string\}
```

```
1. any
                 2. table
                 3. {any}
        4. {string or {number}}
5. { [i\%2==1]: {number}, [i\%2==0]: string}
    6. \{[i\%2==1]: (\{number\} | len==4),
            [i\%2==0]:string
   7. \{[i\%2==1]:(\{number\}|len==4),
      [i\%2==0]:string\}|len\%2==0)
```

```
1. any
                 2. table
                 3. {any}
        4. {string or {number}}
5. {[i%2==1]:{number},[i%2==0]:string}
    6. {[i\%2==1]:({number}|len==4),
            [i\%2==0]:string\}
   7. \{[i\%2==1]:(\{number\}|len==4),
      [i\%2==0]:string\}|len\%2==0)
    8. \{[i\%2==1]:(\{[0-1]\}|len==4),
      [i\%2==0]:string{|len\%2==0|
```

```
1. any
                 2. table
                 3. {any}
        4. {string or {number}}
5. {[i\%2==1]:{number},[i\%2==0]:string}
    6. {[i\%2==1]:({number}|len==4),
            [i\%2==0]:string\}
   7. \{[i\%2==1]:(\{number\}|len==4),
      [i\%2==0]:string\}|len\%2==0)
    8. \{[i\%2==1]:(\{[0-1]\}|len==4),
      [i\%2==0]:string{|len\%2==0|
```

```
local ColorText = record
   r: number
   g: number
   b: number
   a: number
   text: string
end
function my typed print(colortext: {ColorText})
end
my typed print({
   \{r = 1, g = 1, b = 1, a = 1, text = "Hello"\},
   \{r = 1, g = 0, b = 0, a = 0, text = "World"\}
```

types in Lua — did they deliver?

is it easier to maintain an application?

types in Lua — did they deliver?

is it easier to maintain an application? YES!

so, in closing

http://github.com/hishamhm/tl release 0.1.0

luarocks install tl

(still looking for a better name!)

Lua and types: join us!

thank you