(Why | How to) study types(? | .)

 $98\mathchar`-317$ Hype for Types

August 27, 2019

Why study types?

Why study types?

▶ They are interesting in their own right

Why study types?

- ▶ They are interesting in their own right
- ▶ They provide benefits to programmers

"If it compiles, it works."

"If it compiles, it works."

▶ It would be nice if this were true

"If it compiles, it works."

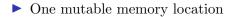
- ▶ It would be nice if this were true
- ▶ Types bring us closer to this

Easy bugs prevented by using types

Undefined variable

▶ Incorrect argument passed a function

"Array.prototype.indexOf returns the first index at which a given element can be found in the array, or -1 if it is not present." (MDN)



- ▶ One mutable memory location
- ▶ More than one concurrent writer to this memory location

- ▶ One mutable memory location
- ▶ More than one concurrent writer to this memory location
- Prevent statically by tracking mutability

How to study types.

How to study types.

How should a programming language be defined?

Through code

"s is a program in the language L if P(s) doesn't output any errors."

Through a written spec

"s is a program in the language L if the spec S defines the semantics of s."

Through judgements

 $\blacktriangleright e: \tau$, "e has type τ "

Through judgements

e:τ, "e has type τ"
3: int

Through judgements

e:τ, "e has type τ"
3: int
"foo": str

The structure of an inference rule

 $\frac{J_1 \qquad J_2 \qquad \dots \qquad J_n}{J}$

Some example inference rules

Some example inference rules

 $\frac{n\in\mathbb{Z}}{n:\mathsf{int}}$

Some example inference rules

$\frac{n\in\mathbb{Z}}{n:\mathsf{int}}$

"..." : str

Rules with more premises

 $\frac{e_1:\mathsf{int}}{e_1+e_2:\mathsf{int}}$

Rules with more premises

e_1	: int	$e_2:int$
	$e_1 + e_2$: int
e_1	: str	$e_2:str$
	$e_1 \wedge e_2$: str

What to do about this?

let $x = e_1$ in e_2

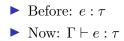
What to do about this?

let
$$x = e_1$$
 in e_2
let $x = 2 + 3$ in $x + x \Rightarrow 10$

Variables, scope, and context



Variables, scope, and context



The old rules, upgraded



 $\Gamma \vdash$ "..." : str

The let rule

$\frac{\Gamma \vdash e_1:\tau_1 \qquad \Gamma, x:\tau_1 \vdash e_2:\tau_2}{\Gamma \vdash \mathsf{let} \ x = e_1 \ \mathsf{in} \ e_2:\tau_2}$

The variable rule

$\Gamma, x:\tau \vdash x:\tau$

What's the type?

$$\cdot \vdash \mathsf{let} \ x = \mathsf{len}(\,\text{``foo''}\,) \ \mathsf{in} \ x + 3 : \tau$$

A proof tree

