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

98-317 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

Sentinel values

"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)

Data races

Data races

- One mutable memory location

Data races

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

Data races

- 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

- $e : \tau$, "e has type τ"

Through judgements

- $e : \tau$, "e has type τ"
- $3 : \mathsf{int}$

Through judgements

- $e : \tau$, "e has type τ"
- $3 : \mathsf{int}$
- "foo" : str

The structure of an inference rule

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

Some example inference rules

Some example inference rules

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

Some example inference rules

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

$$\frac{}{\text{“...”} : \mathsf{str}}$$

Rules with more premises

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

Rules with more premises

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

$$\frac{e_1 : \mathsf{str} \qquad e_2 : \mathsf{str}}{e_1 \wedge e_2 : \mathsf{str}}$$

What to do about this?

$$\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$$

What to do about this?

$$\mathsf{let}\ x = e_1\ \mathsf{in}\ e_2$$

$$\mathsf{let}\ x = 2 + 3\ \mathsf{in}\ x + x \Rightarrow 10$$

Variables, scope, and context

- Before: $e : \tau$

Variables, scope, and context

- Before: $e : \tau$
- Now: $\Gamma \vdash e : \tau$

The old rules, upgraded

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

$$\frac{}{\Gamma \vdash \text{``...''} : \mathsf{str}}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{int} \qquad \Gamma \vdash e_2 : \mathsf{int}}{\Gamma \vdash e_1 + e_2 : \mathsf{int}}$$

$$\frac{\Gamma \vdash e_1 : \mathsf{str} \qquad \Gamma \vdash e_2 : \mathsf{str}}{\Gamma \vdash e_1 \wedge e_2 : \mathsf{str}}$$

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

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}$$

What's the type?

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

A proof tree

$$\dfrac{\dfrac{\dfrac{}{\cdot \vdash \text{“foo”} : \mathsf{str}}}{\cdot \vdash \mathsf{len}(\text{“foo”}) : \mathsf{int}} \qquad \dfrac{\dfrac{}{\cdot, x : \mathsf{int} \vdash x : \mathsf{int}} \qquad \dfrac{\dfrac{}{3 \in \mathbb{Z}}}{\cdot, x : \mathsf{int} \vdash 3 : \mathsf{int}}}{\cdot, x : \mathsf{int} \vdash x + 3 : \mathsf{int}}}{\cdot \vdash \mathsf{let}\ x = \mathsf{len}(\text{“foo”})\ \mathsf{in}\ x + 3 : \mathsf{int}}$$