# When does a program typecheck?

Esther Wang

*@esdrwang*
*Airbnb*

# In a statically typed language

- Does typecheck:

```
(\x -> x) 0
```

- Does not typecheck:

```
0 (\x -> x)
```

# Topics

- Introduction
- Formalizing type systems
- Type soundness

# Type systems are not arbitrary

- Guarantee “good behavior” by ruling out certain classes of runtime errors
- Provide safety

# Formalizing type systems

# Formalizing type systems

- Typing judgement

$$\Gamma \vdash e : \tau$$

# Formalizing type systems

- Typing judgement

$$\Gamma \vdash e : \tau$$

- Typing rule

$$\frac{\Gamma_1 \vdash e_1 : \tau_1 \qquad \ldots \qquad \Gamma_n \vdash e_n : \tau_n}{\Gamma \vdash e : \tau}$$

# Simply-typed lambda calculus

$$
\begin{aligned}
e ::= &\\
&\quad x\\
&\quad \lambda x : \tau \ .\ e\\
&\quad e\ e\\
\tau ::= &\\
&\quad \tau \rightarrow \tau
\end{aligned}
$$

# Simply-typed lambda calculus

$$
\begin{aligned}
e ::= \\
& x \\
& \lambda x : \tau\ .\ e \\
& e\ e \\
\tau ::= \\
& \tau \to \tau
\end{aligned}
$$

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

# Simply-typed lambda calculus

$$
\begin{aligned}
e ::= \quad & \\
& x \\
& \lambda x : \tau \ . \ e \\
& e \ e \\
\tau ::= \quad & \\
& \tau \to \tau
\end{aligned}
$$

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

$$\frac{\Gamma, \ x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 \ . \ e : \tau_1 \to \tau_2}$$

# Simply-typed lambda calculus

$$
\begin{aligned}
e ::= \\
&\quad x \\
&\quad \lambda x : \tau \ . \ e \\
&\quad e \ e \\
\tau ::= \\
&\quad \tau \to \tau
\end{aligned}
$$

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

$$\frac{\Gamma, \ x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \lambda x : \tau_1 \ . \ e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma \vdash e_2 : \tau_1 \to \tau_2}{\Gamma \vdash e_2 \ e_1 : \tau_2}$$

# Example

$$\dfrac{\dfrac{\dfrac{x : \tau \in x : \tau}{x : \tau \vdash x : \tau}}{\vdash \lambda x : \tau \ .\ x : \tau \to \tau} \qquad \dfrac{}{\vdash 0 : Int}}{\vdash (\lambda x : \tau \ .\ x)\ 0 : Int}$$

# Type soundness

# Type soundness

- **Progress:** a well-typed term will either be a value, or can be stepped
- **Preservation:** a well-typed term is still well-typed and has the same type after a single evaluation step

# Type soundness

- Can be proved given a formal type system and a semantics
- The program will never behave in an unspecified way

# Conclusion

A program typechecks when the formal type system can guarantee that it will be well-behaved.

# Questions?