Typechecker generation via equational reasoning

Cameron Wong

CS-252R final project

2022-12-01

Driving question

- Can we systematically generate a typechecker from a collection of inference rules?

Typechecking, the problem statement

- Let Exp be an arbitrary expression language with typing judgment

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

- Two obvious ways to define a "typechecking" function:

$$\texttt{check} : \texttt{Ctx} \to \texttt{Exp} \to \texttt{Ty} \to \texttt{Bool} \tag{1}$$

$$\texttt{synth} : \texttt{Ctx} \to \texttt{Exp} \to \texttt{Option Ty} \tag{2}$$

Typechecking, the problem statement

- What's our correctness theorem?
- "$\Gamma \vdash e : \tau$ if $\texttt{synth}\ \Gamma\ e \equiv \texttt{Some}\ \tau$"

Theorem Prover Background

- How to represent inference rules in the host language?
- In a proof assistant, typically done with an inductive family:

$$
\begin{array}{l}
\mathbf{data}\ \_ \vdash \_ : \_ \ : \ \mathtt{Ctx} \to \mathtt{Exp} \to \mathtt{Ty} \to \ \mathtt{Set} \\
\quad \mathtt{typ\text{-}nat} \ : \ \forall\{\Gamma\ \ \mathtt{n}\} \to \Gamma \vdash \mathtt{Val}\ \ n : \mathtt{Nat} \\
\quad \mathtt{typ\text{-}add} \ : \ \forall\{\Gamma\ \ e_1\ \ e_2\} \to \\
\qquad \Gamma \vdash e_1 : \mathtt{Nat} \to \Gamma \vdash e_1 : \mathtt{Nat} \to \ \text{-- } \textit{premises} \\
\qquad \Gamma \vdash e_1 + e_2 : \mathtt{Nat} \ \ \text{-- } \textit{conclusion}
\end{array}
$$

Typechecking, the problem statement, in Agda

- Correctness theorem, in Agda (some noise elided):

$$\texttt{synth-correct : } \Gamma \vdash e : \tau \to \texttt{synth } \Gamma \ e \equiv \texttt{ Some } \tau$$

The Plan

- Use equational reasoning to derive `synth-correct` and `synth` simultaneously
- Advantage is that we can perform induction on the judgment $\Gamma \vdash e : \tau$ instead of just structurally on e

Source Language

$$\tau := \texttt{Nat} \mid \texttt{Bool} \mid \tau \to \tau$$
$$e := \mathit{true} \mid \mathit{false} \mid \overline{n} \mid e + e \mid \lambda(x : \tau).e$$

$$\frac{}{\Gamma \vdash \mathit{true} : \texttt{Bool}}\ \textsc{Typ-true} \qquad \frac{}{\Gamma \vdash \mathit{false} : \texttt{Bool}}\ \textsc{Typ-false}$$

$$\frac{}{\Gamma \vdash \overline{n} : \texttt{Nat}}\ \textsc{Typ-nat} \qquad \frac{\Gamma \vdash e_1 : \texttt{Nat} \qquad \Gamma \vdash e_2 : \texttt{Nat}}{\Gamma \vdash e_1 + e_2 : \texttt{Nat}}\ \textsc{Typ-add}$$

$$\frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda(x : \tau).e : \tau \to \tau'}\ \textsc{Typ-abs}$$

The Derivation

- Proceed by induction on the derivation of $\Gamma \vdash e : \tau$

The Derivation

Case $\textsc{Typ-true}$:

- $e = \mathit{true}$, $\tau = \texttt{Bool}$
- Need proof of $\texttt{synth}\ \Gamma\ \mathit{true} \equiv \texttt{Some Bool}$
- Currently, no clause of `synth` for *true*!
  - So define $\texttt{synth}\ \mathit{true} = \texttt{Some Bool}$

The Derivation

Case $\textsc{Typ-true}$:

```
synth-correct TYP-TRUE = begin
    synth Γ true
  ≡ ⟨Define clause: synth Γ true = Some Bool⟩
    Some Bool
  □
```

The Derivation

Case TYP-ADD:

- $e = e_1 + e_2$, $\tau = \texttt{Nat}$
- Have $\Gamma \vdash e_1 : \texttt{Nat}$, $\Gamma \vdash e_2 : \texttt{Nat}$
- Need a proof of $\texttt{synth}\ \Gamma\ e_1 + e_2 \equiv \texttt{Nat}$
- Currently, no clause of `synth` for $e_1 + e_2$, so define $\texttt{synth}\ e_1 + e_2 = \texttt{Nat}$

The Derivation

Case Typ-add:

- Wait, what?

Problem Statement, revisited

- Need to forbid degenerate typecheckers!
- More precisely, need to explicitly talk about typechecker failure

Problem Statement, revisited

- New correctness theorem:
  - "$\Gamma \vdash e : \tau$ if and only if $\texttt{synth}\ \Gamma\ e \equiv \texttt{Some}\ \tau$"