Type Inference

Figuring out an expression’s type
without help from the programmer

Type Checking
vs
Type Synthesis
vs
Type Inference

The difference between Checking and Synthesis:

The inputs/outputs

Checking: program and type -> Boolean

Synthesis : program -> type

An example

$$\texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : \qquad ?$$

Recap: Type Checking Algorithm

$$\overline{\vdash \underbrace{\texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha)}_{\text{INPUT}} : \underbrace{(\alpha \to \beta) \to \alpha \to \beta}_{\text{INPUT}}}$$

Recap: Type Checking Algorithm

$$\frac{?}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}$$

Recap: Type Checking Algorithm

$$\dfrac{\dfrac{?}{(f : \alpha \to \beta) \vdash \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : \alpha \to \beta}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}$$

Recap: Type Checking Algorithm

$$\cfrac{\cfrac{\cfrac{?\qquad\qquad\qquad\qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ (x : \alpha) : \beta}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : \alpha \to \beta}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}$$

Recap: Type Checking Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{?}{(f : \alpha \to \beta), (x : \alpha) \vdash f : \alpha \to \beta} \qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ (x : \alpha) : \beta}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : \alpha \to \beta}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}$$

Recap: Type Checking Algorithm

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{!}{(f : \alpha \to \beta), (x : \alpha) \vdash f : \alpha \to \beta}
      \qquad ?
    }{(f : \alpha \to \beta), (x : \alpha) \vdash f\ (x : \alpha) : \beta}
  }{(f : \alpha \to \beta) \vdash \mathtt{fn}\ x \Rightarrow f\ (x : \alpha) : \alpha \to \beta}
}{\vdash \mathtt{fn}\ f \Rightarrow \mathtt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}
$$

Recap: Type Checking Algorithm

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{(f : \alpha \to \beta), (x : \alpha) \vdash f : \alpha \to \beta} \qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ (x : \alpha) : \beta}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : \alpha \to \beta}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ (x : \alpha) : (\alpha \to \beta) \to \alpha \to \beta}
$$

Recap: Type Checking Algorithm

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \dfrac{?}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ (x:\alpha):\beta}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ x\Rightarrow f\ (x:\alpha):\alpha\to\beta}}{\vdash \texttt{fn}\ f\Rightarrow \texttt{fn}\ x\Rightarrow f\ (x:\alpha):(\alpha\to\beta)\to\alpha\to\beta}
$$

Recap: Type Checking Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta}\qquad\dfrac{!}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ (x:\alpha):\beta}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ x\Rightarrow f\ (x:\alpha):\alpha\to\beta}}{\vdash \texttt{fn}\ f\Rightarrow \texttt{fn}\ x\Rightarrow f\ (x:\alpha):(\alpha\to\beta)\to\alpha\to\beta}$$

Recap: Type Checking Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \quad \dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ (x:\alpha):\beta}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ x\Rightarrow f\ (x:\alpha):\alpha\to\beta}}{\vdash \texttt{fn}\ f\Rightarrow \texttt{fn}\ x\Rightarrow f\ (x:\alpha):(\alpha\to\beta)\to\alpha\to\beta}$$

Output: SUCCESS

Type Synthesis Algorithm

$$\frac{}{\vdash \underbrace{\texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x}_{\text{INPUT}} : \underbrace{\qquad\qquad\qquad\qquad\qquad\qquad}_{\text{OUTPUT}}}$$

Type Synthesis Algorithm

$$\frac{?}{\vdash \texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \qquad ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{?}{(f : \alpha \to \beta) \vdash \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}}{\vdash \texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{? \qquad\qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ x : ?}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}}{\vdash \texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \qquad ?}$$

Type Synthesis Algorithm

$$
\dfrac{\dfrac{\dfrac{\dfrac{?}{(f : \alpha \to \beta), (x : \alpha) \vdash f : \quad ?} \qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ x : ?}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}}{\vdash \texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}
$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{!}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta}\qquad ?}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x:\ ?}}{(f:\alpha\to\beta)\vdash \mathtt{fn}\ (x:\alpha)\Rightarrow f\ x:\ ?}}{\vdash \mathtt{fn}\ (f:\alpha\to\beta)\Rightarrow \mathtt{fn}\ (x:\alpha)\Rightarrow f\ x:\ ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f : \alpha \to \beta), (x : \alpha) \vdash f : \alpha \to \beta} \qquad ?}{(f : \alpha \to \beta), (x : \alpha) \vdash f\ x : ?}}{(f : \alpha \to \beta) \vdash \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \quad ?}}{\vdash \texttt{fn}\ (f : \alpha \to \beta) \Rightarrow \texttt{fn}\ (x : \alpha) \Rightarrow f\ x : \qquad ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \dfrac{?}{(f:\alpha\to\beta),(x:\alpha)\vdash x : ?}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x : ?}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : ?}}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \dfrac{!}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x : ?}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : \quad ?}}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : \qquad ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\overline{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \overline{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x : \text{?}}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : \quad \text{?}}}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x : \qquad \text{?}}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \quad \dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x:\beta}}{\dfrac{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x:\quad ?}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x:\quad ?}}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \quad \dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{\dfrac{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x:\beta}{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x:\alpha\to\beta}}}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x: \quad ?}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \dfrac{}{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x:\beta}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ (x:\alpha)\Rightarrow f\ x:\alpha\to\beta}}{\vdash \texttt{fn}\ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ (x:\alpha)\Rightarrow f\ x:(\alpha\to\beta)\to\alpha\to\beta}$$

Type Synthesis Algorithm

$$\dfrac{\dfrac{\dfrac{\overline{(f:\alpha\to\beta),(x:\alpha)\vdash f:\alpha\to\beta} \qquad \overline{(f:\alpha\to\beta),(x:\alpha)\vdash x:\alpha}}{(f:\alpha\to\beta),(x:\alpha)\vdash f\ x:\beta}}{(f:\alpha\to\beta)\vdash \texttt{fn}\ \ (x:\alpha)\Rightarrow f\ x:\alpha\to\beta}}{\vdash \texttt{fn}\ \ (f:\alpha\to\beta)\Rightarrow \texttt{fn}\ \ (x:\alpha)\Rightarrow f\ x:(\alpha\to\beta)\to\alpha\to\beta}$$

Output: $(\alpha\to\beta)\to\alpha\to\beta$

The difference between Synthesis and Inference

Type annotations in the programming language

If we can do type inference, then the programmer doesn’t need to annotate types

Let’s try to feel out an algorithm for inference

Starting point: it’ll be a lot like synthesis

Trying to infer types

$$\frac{}{\vdash \underbrace{\mathtt{fn}\ f \Rightarrow \mathtt{fn}\ x \Rightarrow f\ x}_{\text{INPUT}} : ?}$$

Trying to infer types

What type do we put in the context for f?

UHHH IDK….. LETS GUESS!

$$\frac{}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

$$\frac{?}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

$$\dfrac{\dfrac{?}{(f : ?_1) \vdash \texttt{fn}\ x \Rightarrow f\ x :}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

$$\dfrac{\dfrac{\dfrac{?}{(f :?_1), (x :?_2) \vdash f\ x :}}{(f :?_1) \vdash \texttt{fn}\ x \Rightarrow f\ x :}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

Now what?

We need $?_1$ to be an arrow type

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f :?_1), (x :?_2) \vdash f :?_1} \qquad ?}{(f :?_1), (x :?_2) \vdash f\ x :}}{(f :?_1) \vdash \texttt{fn}\ x \Rightarrow f\ x :}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

Make new guess variables $?_3$ and $?_4$

"Force" the equality $?_1 = ?_3 \rightarrow ?_4$

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f :?_1), (x :?_2) \vdash f :?_1} \qquad ?}{(f :?_1), (x :?_2) \vdash f\ x :}}{(f :?_1) \vdash \texttt{fn}\ x \Rightarrow f\ x :}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

Make new guess variables $?_3$ and $?_4$

"Force" the equality $?_1 = ?_3 \rightarrow ?_4$

$$\cfrac{\cfrac{\cfrac{\cfrac{}{(f :?_3 \rightarrow ?_4), (x :?_2) \vdash f :?_3 \rightarrow ?_4} \qquad ?}{(f :?_3 \rightarrow ?_4), (x :?_2) \vdash f\ x : ?}}{(f :?_3 \rightarrow ?_4) \vdash \mathtt{fn}\ x \Rightarrow f\ x : ?}}{\vdash \mathtt{fn}\ f \Rightarrow \mathtt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_2) \vdash f :?_3 \to ?_4} \qquad ?}{(f :?_3 \to ?_4), (x :?_2) \vdash f\ x : ?}}{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x : ?}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

$$\dfrac{\dfrac{\dfrac{\overline{(f :?_3 \to ?_4), (x :?_2) \vdash f :?_3 \to ?_4} \qquad \overline{(f :?_3 \to ?_4), (x :?_2) \vdash x :?_2}}{(f :?_3 \to ?_4), (x :?_2) \vdash f\ x : ?}}{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x : ?}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : ?}$$

Trying to infer types

"Force" the equality $?_2 = ?_3$

$$\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_2) \vdash f :?_3 \to ?_4} \qquad \dfrac{}{(f :?_3 \to ?_4), (x :?_2) \vdash x :?_2}}{\dfrac{(f :?_3 \to ?_4), (x :?_2) \vdash f\ x : ?}{\dfrac{(f :?_3 \to ?_4) \vdash \mathtt{fn}\ x \Rightarrow f\ x : ?}{\vdash \mathtt{fn}\ f \Rightarrow \mathtt{fn}\ x \Rightarrow f\ x : ?}}}$$

Trying to infer types

$$\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash f :?_3 \to ?_4} \qquad \dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash x :?_3}}{\dfrac{(f :?_3 \to ?_4), (x :?_3) \vdash f\ x : ?}{\dfrac{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x : \quad ?}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : \qquad ?}}}$$

Trying to infer types

$$\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash f :?_3 \to ?_4} \quad \dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash x :?_3}}{\dfrac{(f :?_3 \to ?_4), (x :?_3) \vdash f\ x :?_4}{\dfrac{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x : \quad ?}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : \quad ?}}}$$

Trying to infer types

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash f :?_3 \to ?_4} \quad \dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash x :?_3}}{(f :?_3 \to ?_4), (x :?_3) \vdash f\ x :?_4}}{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x :?_3 \to ?_4}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : \qquad ?}
$$

Trying to infer types

$$\dfrac{\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash f :?_3 \to ?_4} \quad \dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash x :?_3}}{(f :?_3 \to ?_4), (x :?_3) \vdash f\ x :?_4}}{\dfrac{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x :?_3 \to ?_4}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : (?_3 \to ?_4) \to ?_3 \to ?_4}}$$

Trying to infer types

$$\dfrac{\dfrac{\dfrac{\dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash f :?_3 \to ?_4} \qquad \dfrac{}{(f :?_3 \to ?_4), (x :?_3) \vdash x :?_3}}{(f :?_3 \to ?_4), (x :?_3) \vdash f\ x :?_4}}{(f :?_3 \to ?_4) \vdash \texttt{fn}\ x \Rightarrow f\ x :?_3 \to ?_4}}{\vdash \texttt{fn}\ f \Rightarrow \texttt{fn}\ x \Rightarrow f\ x : (?_3 \to ?_4) \to ?_3 \to ?_4}$$

Output: $(\alpha \to \beta) \to \alpha \to \beta$

I think we need to talk about this "forcing equality" stuff…

It seems like it could be complicated, so let's be more formal about it.

Unification: A problem in math/CS

- Input: a pair of terms
- Output: a list of mappings from term variables to terms, such that applying the mappings to the terms results in the same term

Example of unification:

- Input: $\alpha \doteq \beta \times \gamma$

Example of unification:

- Input: $\alpha \doteq \beta \times \gamma$
- Output: $[\alpha \mapsto \beta \times \gamma]$

Example of unification:

- Input: $\alpha \doteq \beta \times \gamma$
- Output: $[\alpha \mapsto \beta \times \gamma]$
- Result of applying the mappings:

$$\beta \times \gamma \doteq \beta \times \gamma$$

Is unification always possible?

Is unification always possible?

- Input: $\alpha \times \beta \doteq \gamma + \delta$

Is unification always possible?

- Input: $\alpha \times \beta \doteq \gamma + \delta$
- Cannot unify; terms have different head symbols.
- No, unification is not always possible.

How else can unification fail?

How else can unification fail?

- Input: $\alpha \doteq \alpha \times \beta$

How else can unification fail?

- Input: $\alpha \doteq \alpha \times \beta$

- Cannot unify; circularity

- There are two ways for unification to fail:

  head symbol conflict and circularity.

Is unification always unique?

Is unification always unique?

- Input: $\alpha \doteq \beta$

Is unification always unique?

- Input: $\alpha \doteq \beta$

- Many valid outputs:

$$[\alpha \mapsto \beta],$$

$$\beta \doteq \beta$$

Is unification always unique?

- Input: $\alpha \doteq \beta$

- Many valid outputs:

$$[\alpha \mapsto \beta], [\beta \mapsto \alpha],$$

$$\beta \doteq \beta \qquad \alpha \doteq \alpha$$

Is unification always unique?

- Input: $\alpha \doteq \beta$
- Many valid outputs:

$$[\alpha \mapsto \beta], [\beta \mapsto \alpha], [\alpha \mapsto \gamma, \beta \mapsto \gamma]$$

$$\beta \doteq \beta \qquad \alpha \doteq \alpha \qquad \gamma \doteq \gamma$$

Coming up with an algorithm for unification

Case 1: one of the terms is a variable

$$\alpha \doteq \alpha \quad \Rightarrow$$

$$\alpha \doteq \alpha \times \beta \quad \Rightarrow$$

$$\alpha \doteq \beta \times \gamma \quad \Rightarrow$$

Case 1: one of the terms is a variable

$$\alpha \doteq \alpha \quad \Rightarrow \quad []$$

$$\alpha \doteq \alpha \times \beta \quad \Rightarrow$$

$$\alpha \doteq \beta \times \gamma \quad \Rightarrow$$

If the other term is the same variable, output []

Case 1: one of the terms is a variable

$$\alpha \doteq \alpha \quad \Rightarrow \quad [\,]$$

$$\alpha \doteq \alpha \times \beta \quad \Rightarrow$$

$$\alpha \doteq \beta \times \gamma \quad \Rightarrow$$

If the other term is the same variable, output []

Otherwise, if the other term contains the variable, circularity error.

Case 1: one of the terms is a variable

$$\alpha \doteq \alpha \quad \Rightarrow \quad []$$

$$\alpha \doteq \alpha \times \beta \quad \Rightarrow$$

$$\alpha \doteq \beta \times \gamma \quad \Rightarrow \quad [\alpha \mapsto \beta \times \gamma]$$

If the other term is the same variable, output []

Otherwise, if the other term contains the variable, circularity error.

Otherwise, just map the variable to the other term.

Case 2: Terms have same head symbols

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow$$

Case 2: Terms have same head symbols

$$\alpha \doteq \beta$$

$$\alpha + \alpha \doteq \beta + \gamma$$

1. Unify the first subterms.

Case 2: Terms have same head symbols

$$\alpha \doteq \beta \Rightarrow [\alpha \mapsto \beta]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow$$

1. Unify the first subterms.

Case 2: Terms have same head symbols

$$\alpha \doteq \beta \quad \Rightarrow [\alpha \mapsto \beta]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \ldots]$$

1. Unify the first subterms.

Case 2: Terms have same head symbols

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \ldots]$$

1. Unify the first subterms.

Case 2: Terms have same head symbols

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \ldots]$$

1. Unify the first subterms.
2. Unify the second subterms?

Case 2: Terms have same head symbols

$$\alpha \doteq \gamma \Rightarrow [\alpha \mapsto \gamma]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \alpha \mapsto \gamma]$$

1. Unify the first subterms.
2. Unify the second subterms? Wait… that doesn't make sense

Case 2: Terms have same head symbols

$$\alpha \doteq \gamma \Rightarrow [\alpha \mapsto \gamma]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \alpha \mapsto \gamma]$$

1. Unify the first subterms.
2. Unify the second subterms? Wait… that doesn’t make sense

Solution: Apply the mappings from unifying the first subterms to the second subterms before unifying them.

Case 2: Terms have same head symbols

$$\beta \doteq \gamma \Rightarrow$$

$$[\alpha \mapsto \beta] \qquad [\alpha \mapsto \beta]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \ldots]$$

1. Unify the first subterms.
2. Apply the mappings from unifying the first subterms to the second subterms, then unify the second subterms.

Case 2: Terms have same head symbols

$$
\begin{tikzcd}
\beta \doteq \gamma \arrow[r, Rightarrow] & {[\beta \mapsto \gamma]} \\
\alpha + \alpha \doteq \beta + \gamma \arrow[u, "{[\alpha \mapsto \beta]}"] \arrow[u, bend right, "{[\alpha \mapsto \beta]}"'] \arrow[r, Rightarrow] & {[\alpha \mapsto \beta, \ldots]}
\end{tikzcd}
$$

1. Unify the first subterms.
2. Apply the mappings from unifying the first subterms to the second subterms, then unify the second subterms.

Case 2: Terms have same head symbols

$$\beta \doteq \gamma \quad \Rightarrow [\beta \mapsto \gamma]$$

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \beta \mapsto \gamma]$$

1. Unify the first subterms.
2. Apply the mappings from unifying the first subterms to the second subterms, then unify the second subterms.

Case 2: Terms have same head symbols

$$\alpha + \alpha \doteq \beta + \gamma \Rightarrow [\alpha \mapsto \beta, \beta \mapsto \gamma]$$

1. Unify the first subterms.
2. Apply the mappings from unifying the first subterms to the second subterms, then unify the second subterms.

Case 3: Terms have different head symbols

$$\alpha \times \beta \doteq \gamma + \delta \Rightarrow$$

Case 3: Terms have different head symbols

$$\alpha \times \beta \doteq \gamma + \delta$$

Head symbol conflict.

Tips on using unification for type inference

- DON'T CARE ABOUT EFFICIENCY
  - Type inference can be done in linear time (with respect to program size) but that's **not fun**
- Focus on maintaining invariants about which type variables can appear where
- Keep the difference between expression terms and type terms straight
  - Unification happens to type terms and maps type variables to other types
  - Expression terms (including expression variables) have types, which are type terms which contain type variables.

Tips on using unification for type inference

- At any point during the execution of the algorithm, every type variable will fall into one of two categories:
    - Still allowed to appear in our guess types
    - Has been “unified away” and should no longer show up in any types in our context
- A variable moves from the first category to the second category when the unification algorithm maps it to something
- Take care to apply the variable mappings returned by unification, so that variables that have been “unified away” don’t reappear or get mapped twice

Tips on using unification for type inference

- Consider using a dictionary to store the type that each “unified away” variable is currently mapped to
- Your inference algorithm will then be passing *two* dictionaries around: one representing the context which maps expression variables to types, and one representing the mapping from “unified away” type variables to types
- You’ll want to maintain an invariant like: if a variable has been “unified away”, it cannot appear in any type mapped to by either dictionary
- Every time you call the unification algorithm, apply the mappings (in left-to-right order) to the type variable dictionary

Tips on using unification for type inference

- Whenever you want to force a type to be of a certain form generate new (not used anywhere else) type variables and unify
  - e.g. say we want to force a type τ to be an arrow type.
  - Create fresh type variables α and β, then unify:

$$\tau \doteq \alpha \to \beta$$

Questions?