What are type rules?

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{ + e_1 e_2 \} : \text{num} \]

An example - the rule for +
What are type rules?

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{ + e_1 e_2 \} : \text{num} \]

An example - the rule for +

• This is just one of a set of inference rules.

• Together the set of rules define the type judgment, which is a relation that assigns types to expressions.
What are type rules?

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{ + \; e_1 \; e_2 \} : \text{num} \]

An example - the rule for +

• This is just one of a set of inference rules.

• Together the set of rules define the type judgment, which is a relation that assigns types to expressions.

• Fine, but what does that mean...
Inference rules

A

B

C

The general form of an inference rule
Inference rules

A
  
  
  
  B

C

The general form of a inference rule

• A and B are premises (not necessarily two of them)

• C is the conclusion
Inference rules

The general form of an inference rule:

- A and B are *premises* (not necessarily two of them)
- C is the *conclusion*
- This is a *rule*, which says:
- If I know A and B, then I can conclude C
Inference rules

\[
\begin{array}{c}
A \\
\hline
A \land B
\end{array}
\]

An example — logical “and”
Inference rules

\[
\begin{array}{cc}
A & B \\
\hline \\
A \land B
\end{array}
\]

An example — logical “and”

- If I know \(A\), and I know \(B\), then I can conclude \(A \land B\)
Inference rules

\[
\begin{array}{c}
A \quad B \\
\hline
A \land B
\end{array}
\]

An example — logical “and”

• If I know \( A \), and I know \( B \), then I can conclude \( A \land B \)

• How would I know \( A \) and \( B \)?
Inference rules

\[ A \land B \]

An example — logical “and”

- If I know \( A \), and I know \( B \), then I can conclude \( A \land B \)

- How would I know \( A \) and \( B \)?

- I used some rule to conclude they were true
Inference rules

\[ A \land B \]

An example — logical “and”

• If I know \( A \), and I know \( B \), then I can conclude \( A \land B \)

• Some other \( \land \) rules:

\[ A \land B \]

\[ A \land B \]

\[ A \]

\[ B \]
Inference rules

\[
\begin{array}{c}
A \\
\hline
A \land B \\
B
\end{array}
\]

An example — logical “and”

• If I know \(A\), and I know \(B\), then I can conclude \(A \land B\)

\[
\begin{array}{c}
A \land B \\
A
\end{array}
\quad
\begin{array}{c}
A \land B \\
B
\end{array}
\]

• Add some more rules and you have a system for deciding the truth of logical sentences
A TFAE Rule

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{ + e_1 \ e_2 \} : \text{num} \]
A TFAE Rule

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \frac{}{\Gamma \vdash \{+ e_1 e_2\} : \text{num}} \]

One of a set of rules that define the type judgment

\[ \Gamma \vdash e : \tau \]
A TFAE Rule

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{ + e_1 e_2 \} : \text{num} \]

One of a set of rules that define the *type judgment*

\[ \Gamma \vdash e : \tau \]

• Which means...

• With the type bindings in \( \Gamma \), I can conclude that \( e \) has the type \( \tau \)
A TFAE Rule

\[ \Gamma \vdash e_1 : \text{num} \quad \Gamma \vdash e_2 : \text{num} \]

\[ \Gamma \vdash \{+ e_1 e_2\} : \text{num} \]

One of a set of rules that define the type judgment

\[ \Gamma \vdash e : \tau \]

• Which means...

• With the type bindings in \(\Gamma\), I can conclude that \(e\) has the type \(\tau\)

• \(\Gamma\) is the type environment and is just a map from \(<\text{id}>\) to \(\tau\) (type)
TFAE Rules

\[ \frac{\Gamma \vdash \text{<num>} : \text{num}}{\Gamma \vdash \text{true} : \text{bool}} \quad \frac{\Gamma \vdash \text{false} : \text{bool}}{\Gamma \vdash \text{false} : \text{bool}} \]

\[ \frac{\Gamma \vdash \text{e}_1 : \text{num} \quad \Gamma \vdash \text{e}_2 : \text{num}}{\Gamma \vdash \{+ \text{e}_1 \text{e}_2\} : \text{num}} \]

\[ \frac{\Gamma \vdash \text{e}_1 : \text{bool} \quad \Gamma \vdash \text{e}_2 : \tau_0 \quad \Gamma \vdash \text{e}_3 : \tau_0}{\Gamma \vdash \{\text{if e}_1 \text{e}_2 \text{e}_3\} : \tau_0} \]

\[ \frac{\Gamma \vdash \text{e} : \tau_0}{\Gamma[\text{<id>←}\tau_1] \vdash \text{e} : \tau_0} \]

\[ \frac{\Gamma \vdash \{\text{fun} \{\text{id} : \tau_1\} \text{e}\} : (\tau_1 \rightarrow \tau_0)}{\Gamma \vdash \text{e}_0 : (\tau_1 \rightarrow \tau_0) \quad \Gamma \vdash \text{e}_1 : \tau_1}{\Gamma \vdash \{\text{e}_0 \text{e}_1\} : \tau_0} \]
Type derivations

1 : \textit{num} \quad 2 : \textit{num}

\begin{array}{c}
\hline \\
\{+ 1 2\} : \textit{num} \\
\hline \\
\{+ \{+ 1 2\} 3\} : \textit{num} \\
\end{array}

\begin{itemize}
\item We can conclude that an expression has some type if we can come up with a derivation using the type rules.
\end{itemize}
Type derivations

- We can conclude that an expression has some type if we can come up with a derivation using the type rules.

- Great, but given some expression, how can we find the right derivation, and its type
Type derivations

1 : num  2 : num

\[\{+ 1 2\} : num \quad 3 : num\]

\[\{+ \{+ 1 2\} 3\} : num\]

• We can conclude that an expression has some type if we can come up with a derivation using the type rules.

• Great, but given some expression, how can we find the right derivation, and its type

• And what if it doesn’t have a type...
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \text{if true} \ \{+ \ 1 \ 2\} \ 3 \} : \tau \ ? \]
Finding a type

\[ \Gamma \vdash e : \tau \]

Let's try to find a type for this expression

\[ [] \vdash \{ \text{if } \text{true } \{ + \ 1 \ 2 \} \ 3 \} : \tau \]

• What is the type of \( \{ \text{if } \text{true } \{ + \ 1 \ 2 \} \ 3 \} \)?

• Is there some \( \tau \) that will satisfy the type judgment?
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \text{if true} \{ + 1 2 \} 3 \} : \tau \]

Let’s try a rule:

\[
\begin{align*}
\Gamma \vdash e_1 : \text{num} & \quad \Gamma \vdash e_2 : \text{num} \\
\hline
\Gamma \vdash \{ + e_1 \ e_2 \} : \text{num}
\end{align*}
\]
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \textbf{if } \text{true} \ \{ + \ 1 \ 2 \} \ 3 \} : \tau \ ? \]

Let’s try a rule:

\[
\begin{array}{c}
\Gamma \vdash e_1 : \text{num} \\
\Gamma \vdash e_2 : \text{num}
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash \{ + e_1 e_2 \} : \text{num}
\end{array}
\]

This one won’t work (the expressions don’t match)...

Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \text{if true} \{ + 1 \ 2 \} \ 3 \} : \tau \ ? \]

Try again:

\[
\begin{align*}
\Gamma & \vdash e_1 : \text{bool} & \Gamma & \vdash e_2 : \tau_0 & \Gamma & \vdash e_3 : \tau_0 \\
\hline
\Gamma & \vdash \{ \text{if} \ e_1 \ e_2 \ e_3 \} : \tau_0
\end{align*}
\]
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \text{if } \text{true } \{ + 1 \ 2 \} \ 3 \} : \tau ? \]

Try again:

\[
\begin{align*}
\Gamma & \vdash e_1 : \text{bool} & \Gamma & \vdash e_2 : \tau_0 & \Gamma & \vdash e_3 : \tau_0 \\
\hline
\Gamma & \vdash \{ \text{if } e_1 \ e_2 \ e_3 \} : \tau_0
\end{align*}
\]

This works, but we still don’t have a full derivation...
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[ [] \vdash \{ \text{if true } \{+ 1 2\} 3 \} : \tau \ ? \]

Try again:

\[ \begin{align*}
\Gamma & \vdash e_1 : \text{bool} & \Gamma & \vdash e_2 : \tau_0 & \Gamma & \vdash e_3 : \tau_0 \\
\end{align*} \]

\[ \Gamma \vdash \{ \text{if } e_1 \ e_2 \ e_3 \} : \tau_0 \]

So we have to try again...
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[
\{} \text{if true \{ + 1 2\} 3} \text{ : \tau} \?
\]

\[
\begin{align*}
\Gamma & \vdash e_1 : bool & \Gamma & \vdash e_2 : \tau_0 & \Gamma & \vdash e_3 : \tau_0
\end{align*}
\]

\[
\Gamma \vdash \{\text{if } e_1 \ e_2 \ e_3\} : \tau_0
\]

- In general, we are stuck doing an expensive search where we try every rule for every expression (with backtracking).
Finding a type

\[ \Gamma \vdash e : \tau \]

Let’s try to find a type for this expression

\[
[ ] \vdash \{ \text{if true \{+ 1 2\} 3} \} : \tau ?
\]

\[
\begin{align*}
\Gamma \vdash e_1 : bool & \quad \Gamma \vdash e_2 : \tau_0 & \quad \Gamma \vdash e_3 : \tau_0 \\
\hline
\Gamma \vdash \{ \text{if } e_1 \ e_2 \ e_3 \} : \tau_0
\end{align*}
\]

• In general, we are stuck doing an expensive search where we try every rule for every expression (with backtracking).

• But actually the type rules have some nice properties, so things aren’t really that difficult...
TFAE Rules

\[
\begin{align*}
\Gamma \vdash <\text{num}> : \text{num} & \quad \quad [\ldots <\text{id}> \leftarrow \tau \ldots] \vdash <\text{id}> : \tau \\
\Gamma \vdash \text{true} : \text{bool} & \quad \quad \Gamma \vdash \text{false} : \text{bool} \\
\Gamma \vdash e_1 : \text{num} & \quad \quad \Gamma \vdash e_2 : \text{num} \\
\hline \notag \\
\Gamma \vdash \{+ e_1 e_2\} : \text{num} \\
\Gamma \vdash e_1 : \text{bool} & \quad \quad \Gamma \vdash e_2 : \tau_0 & \quad \quad \Gamma \vdash e_3 : \tau_0 \\
\hline \notag \\
\Gamma \vdash \{\text{if } e_1 e_2 e_3\} : \tau_0 \\
\Gamma[<\text{id}> \leftarrow \tau_1] \vdash e : \tau_0 \\
\hline \notag \\
\Gamma \vdash \{\text{fun } \{<\text{id}> : \tau_1\} e\} : (\tau_1 \rightarrow \tau_0) \\
\Gamma \vdash e_0 : (\tau_1 \rightarrow \tau_0) & \quad \quad \Gamma \vdash e_1 : \tau_1 \\
\hline \notag \\
\Gamma \vdash \{e_0 e_1\} : \tau_0
\end{align*}
\]
Properties of TFAE types

\[ \Gamma \vdash e : \tau \]

- There is only one rule that applies to any TFAE expression
Properties of TFAE types

\[ \Gamma \vdash e : \tau \]

• There is only one rule that applies to any TFAE expression
• So there is only one (possible) type derivation for any expression
Properties of TFAE types

\[ \Gamma \vdash e : \tau \]

• For any rule, \( \Gamma \) and \( e \) always determine \( \tau \)
Properties of TFAE types

\[ \Gamma \vdash e : \tau \]

- For any rule, \( \Gamma \) and \( e \) always determine \( \tau \)
- Think of \( \Gamma \) and \( e \) as inputs — they give us the necessary information for recursive calls (premises).
- Think of \( \tau \) as an output — the premises give us what we need to know to construct the result type
Properties of TFAE types

\[ \Gamma \vdash e : \tau \]

- For any rule, \( \Gamma \) and \( e \) always determine \( \tau \)
- Think of \( \Gamma \) and \( e \) as inputs — they give us the necessary information for recursive calls (premises).
- Think of \( \tau \) as an output — the premises give us what we need to know to construct the result type
- So we can easily turn the type judgment into a function:

```plaintext
; type-check \( \Gamma \ e \rightarrow \tau \)
;```