Programming Languages and Compilers (CS 421)

Talia Ringer (they/them)

4218 SC, UIUC

https://courses.grainger.illinois.edu/cs421/fa2023/

Based heavily on slides by Elsa Gunter, which were based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

Objectives for Today

- Last class, we covered **mutually recursive** and **nested recursive** datatypes
- We then showed **types** and **type checking**
- Today, we will continue with **types** and **type checking**, starting in a **simply typed** (monomorphic) setting
- This will set the stage for work next week on **typing rules** for a subset of **OCaml**

Objectives for Today

- Last class, we covered **mutually recursive** and **nested recursive** datatypes
- We then showed **types** and **type checking**
- Today, we will continue with **types** and **type checking**, starting in a **simply typed** (monomorphic) setting
- This will set the stage for work next week on **typing rules** for a subset of **OCaml**

Questions from Tuesday?

Judgments, Systems, Derivations

Reminder: Syntax

$$\frac{\textbf{Premises}}{\textbf{Conclusion}}$$

Judgments, Systems, **Derivations**

Reminder: Syntax

$$\frac{\textbf{“I like all chocolate truffles”}}{\textbf{“I like Godiva chocolate truffles”}}$$

Reminder: Syntax

$$\frac{\textbf{Premise 1 \ldots Premise n}}{\textbf{Conclusion}}$$

Judgments, Systems, **Derivations**

Reminder: Syntax

$$\dfrac{\dfrac{\textbf{Premises of Premise 1}}{\textbf{Premise 1}} \quad \textbf{...} \quad \dfrac{\textbf{Premises of Premise n}}{\textbf{Premise n}}}{\textbf{Conclusion}}$$

Reminder: Syntax

Axiom

Judgments, Systems, **Derivations**

Reminder: Syntax

“other people are conscious, too”

Reminder: Syntax

$$\cfrac{\cfrac{\cfrac{}{\textbf{Axiom}}}{\textbf{Premise 1}}\ \ \textbf{...}\ \ \cfrac{\cfrac{}{\textbf{Axiom}}}{\textbf{Premise n}}}{\textbf{Conclusion}}$$

Reminder: Syntax

$$\frac{}{\textbf{Axiom}} \qquad \frac{}{\textbf{Axiom}}$$

$$\frac{\textbf{Premise}}{\textbf{Conclusion}} \qquad \frac{\textbf{Premise}}{\textbf{Conclusion}}$$

Reminder: Syntax

syntax of judgment should go in this box

$$\frac{}{\textbf{Axiom}} \qquad \frac{}{\textbf{Axiom}}$$

$$\frac{\textbf{Premise}}{\textbf{Conclusion}} \qquad \frac{\textbf{Premise}}{\textbf{Conclusion}}$$

Reminder: Syntax

$$\boxed{\Gamma \vdash \mathbf{t} : \mathbf{T}}$$

$$\frac{}{\Gamma \vdash \mathbf{true} : \mathbf{bool}} \qquad \frac{}{\Gamma \vdash \mathbf{false} : \mathbf{bool}}$$

$$\frac{\textbf{Premise}}{\textbf{Conclusion}} \qquad \frac{\textbf{Premise}}{\textbf{Conclusion}}$$

Reminder: Syntax

$$\boxed{\Gamma \vdash \mathbf{t} : \mathbf{T}}$$

$$\frac{}{\Gamma \vdash \mathbf{true} : \mathbf{bool}} \qquad \frac{}{\Gamma \vdash \mathbf{false} : \mathbf{bool}}$$

$$\frac{\Gamma \vdash \mathbf{t1} : \mathbf{bool} \qquad \Gamma \vdash \mathbf{t2} : \mathbf{bool}}{\Gamma \vdash \mathbf{t1}\ \&\&\ \mathbf{t2} : \mathbf{bool}}$$

Type Checking

A “Simple” (Monomorphic) Type System

$$\boxed{\Gamma \vdash t : T}$$

Type Checking

A “Simple” (Monomorphic) Type System

Axioms

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}} \qquad \frac{}{\Gamma \vdash \text{false} : \text{bool}}$$

A “Simple” (Monomorphic) Type System

$$\boxed{\Gamma \vdash t : T}$$

Axioms

$$\frac{}{\Gamma \vdash \mathrm{true} : \mathrm{bool}} \qquad \frac{}{\Gamma \vdash \mathrm{false} : \mathrm{bool}}$$

$$\frac{}{\Gamma \vdash n : \mathrm{int}}$$ (for n integer constant)

A “Simple” (Monomorphic) Type System

$$\boxed{\Gamma \vdash t : T}$$

Axioms

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}} \qquad \frac{}{\Gamma \vdash \text{false} : \text{bool}}$$

$$\frac{}{\Gamma \vdash n : \text{int}}$$ (for n integer constant)

$$\frac{}{\Gamma \vdash v : T}$$ (if $\Gamma(v) = T$)

A “Simple” (Monomorphic) Type System

Axioms

$\boxed{\Gamma \vdash t : T}$

$$\frac{}{\Gamma \vdash \text{true} : \text{bool}}\ \textbf{CONST} \qquad \frac{}{\Gamma \vdash \text{false} : \text{bool}}\ \textbf{CONST}$$

$$\frac{}{\Gamma \vdash n : \text{int}}\ \textbf{CONST}$$ (for n integer constant)

$$\frac{}{\Gamma \vdash v : T}\ \textbf{VAR}$$ (if $\Gamma(v) = T$)

Type Checking

A “Simple” (Monomorphic) Type System

Boolean Connectives

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash \mathbf{t1 : bool} \qquad \Gamma \vdash \mathrm{t2 : bool}}{\Gamma \vdash \mathrm{t1 \; \&\& \; t2 : bool}} \;\textbf{AND}$$

$$\frac{\Gamma \vdash \mathrm{t1 : bool} \qquad \Gamma \vdash \mathrm{t2 : bool}}{\Gamma \vdash \mathrm{t1 \; || \; t2 : bool}}$$

A “Simple” (Monomorphic) Type System

Boolean Connectives

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash t1 : bool \qquad \Gamma \vdash t2 : bool}{\Gamma \vdash t1 \;\&\&\; t2 : bool}\ \text{AND}$$

$$\frac{\Gamma \vdash t1 : bool \qquad \Gamma \vdash t2 : bool}{\Gamma \vdash t1 \;||\; t2 : bool}$$

A “Simple” (Monomorphic) Type System

Boolean Connectives

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : bool \qquad \Gamma \vdash t2 : bool}{\Gamma \vdash t1 \; \&\& \; t2 : bool} \; \text{AND}$$

$$\frac{\Gamma \vdash t1 : bool \qquad \Gamma \vdash t2 : bool}{\Gamma \vdash t1 \; || \; t2 : bool}$$

A “Simple” (Monomorphic) Type System

Boolean Connectives

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : bool \qquad \Gamma \vdash t2 : bool}{\Gamma \vdash t1 \;\&\&\; t2 : bool}\ \textbf{AND}$$

$$\frac{\Gamma \vdash \mathbf{t1 : bool} \qquad \Gamma \vdash \mathbf{t2 : bool}}{\Gamma \vdash \mathbf{t1 \;||\; t2 : bool}}\ \textbf{OR}$$

Example: Type Checking

How do we check this?

$$\frac{}{\{b1 : bool\} \vdash b1 \ \&\& \ false : bool}$$

Type Checking

Example: Type Checking

$$\frac{\boxed{\textbf{Premises?}}}{\{b1 : bool\} \vdash b1 \;\&\&\; false : bool}\ \textbf{AND}$$

Example: Type Checking

$$\frac{\{b1 : \text{bool}\} \vdash b1 : \text{bool} \quad \{b1 : \text{bool}\} \vdash \text{false} : \text{bool}}{\{b1 : \text{bool}\} \vdash b1 \;\&\&\; \text{false} : \text{bool}}\,\textbf{AND}$$

Example: Type Checking

How do we check this?

How do we check this?

$$\dfrac{\dfrac{}{\{b1 : \text{bool}\} \vdash b1 : \text{bool}} \qquad \dfrac{}{\{b1 : \text{bool}\} \vdash \text{false} : \text{bool}}}{\{b1 : \text{bool}\} \vdash b1 \;\&\&\; \text{false} : \text{bool}}\ \textbf{AND}$$

Example: Type Checking

How do we check this?

$$\dfrac{\dfrac{}{\{\text{b1} : \text{bool}\} \vdash \text{b1} : \text{bool}}\text{VAR} \qquad \dfrac{}{\{\text{b1} : \text{bool}\} \vdash \text{false} : \text{bool}}}{\{\text{b1} : \text{bool}\} \vdash \text{b1 \&\& false} : \text{bool}}\text{AND}$$

Example: Type Checking

$$\dfrac{\dfrac{}{\{b1 : \text{bool}\} \vdash b1 : \text{bool}}\,\text{VAR} \qquad \dfrac{}{\{b1 : \text{bool}\} \vdash \text{false} : \text{bool}}\,\text{CONST}}{\{b1 : \text{bool}\} \vdash b1 \ \&\& \ \text{false} : \text{bool}}\,\text{AND}$$

Type Derivation

$$\dfrac{\dfrac{}{\{\mathrm{b1} : \mathrm{bool}\} \vdash \mathrm{b1} : \mathrm{bool}}\,\textbf{VAR} \qquad \dfrac{}{\{\mathrm{b1} : \mathrm{bool}\} \vdash \mathrm{false} : \mathrm{bool}}\,\textbf{CONST}}{\{\mathrm{b1} : \mathrm{bool}\} \vdash \mathrm{b1}\ \&\&\ \mathrm{false} : \mathrm{bool}}\,\textbf{AND}$$

Questions so far?

Type Variables

Type Variables in Rules

Conditionals

Γ ⊢ t : T

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash (\text{if } t1 \text{ then } t2 \text{ else } t3) : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Type Variables in Rules

Conditionals

Γ ⊢ t : T

$$\frac{\Gamma \vdash \mathbf{t1} : \mathbf{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash (\text{if } \mathbf{t1} \text{ then } t2 \text{ else } t3) : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Type Variables in Rules

Conditionals

Γ ⊢ t : T

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash (\text{if } t1 \text{ then } \mathbf{t2} \text{ else } \mathbf{t3}) : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Type Variables in Rules

Conditionals

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : \mathrm{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash (\mathrm{if}\ t1\ \mathrm{then}\ \mathbf{t2}\ \mathrm{else}\ \mathbf{t3}) : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the same **same type**

Type Variables

Type Variables in Rules

Conditionals

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash (\text{if } t1 \text{ then } \mathbf{t2} \text{ else } \mathbf{t3}) : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Type Variables in Rules

Conditionals

Γ ⊢ t : T

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : T \quad \Gamma \vdash t3 : T}{\Gamma \vdash \textbf{(if t1 then t2 else t3)} : T}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the same **same type**

Type Variables

Type Variables in Rules

Conditionals

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash \mathbf{t2 : T} \quad \Gamma \vdash \mathbf{t3 : T}}{\Gamma \vdash \mathbf{(if\ t1\ then\ t2\ else\ t3) : T}}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the same type

Type Variables

Type Variables in Rules

Conditionals

$\boxed{\Gamma \vdash \mathbf{t} : \mathbf{T}}$

$$\frac{\Gamma \vdash t1 : bool \quad \Gamma \vdash \mathbf{t2 : T} \quad \Gamma \vdash \mathbf{t3 : T}}{\Gamma \vdash \mathbf{(if\ t1\ then\ t2\ else\ t3) : T}}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Type Variables in Rules

Conditionals

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : \mathbf{T} \quad \Gamma \vdash t3 : \mathbf{T}}{\Gamma \vdash (\text{if } t1 \text{ then } t2 \text{ else } t3) : \mathbf{T}}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

A “Simple” (Monomorphic) Type System

Conditionals

$$\boxed{\Gamma \vdash \mathbf{t} : \mathbf{T}}$$

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : \mathbf{T} \quad \Gamma \vdash t3 : \mathbf{T}}{\Gamma \vdash (\text{if } t1 \text{ then } t2 \text{ else } t3) : \mathbf{T}}\ \text{COND}$$

- T is a **type variable** (metavariable)
- Can take *any type* at all
- All instances in rule application must get same type
- The then branch, the else branch, and the final term must all have the **same type**

Type Variables

Question

Is this well typed?

$$\frac{???}{\{\ b : \text{bool}\ \} \vdash (\text{if } b \text{ then } 5 \text{ else false}) : ???}$$

Question

Why is this well typed?

$$\frac{\textbf{???}}{\{\ b : \text{bool}\ \} \vdash (\text{if } b \text{ then } 5 \text{ else } 7) : \text{int}}$$

Questions so far?

Functions and Application

Function Application

Function Application

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash f : T1 \rightarrow T2 \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \text{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Functions and Application

Function Application

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash \mathbf{f} : \mathbf{T1} \rightarrow \mathbf{T2} \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

- If you have a function **f** of type **T1→T2**, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Function Application

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash f : T1 \to T2 \quad \Gamma \vdash \mathbf{t1} : \mathbf{T1}}{\Gamma \vdash \mathbf{(f\ t1)} : T2}\ \textbf{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument **t1** of type **T1**, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Functions and Application

Function Application

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \ \vdash f : T1 \rightarrow T2 \quad \Gamma \ \vdash t1 \ : T1}{\Gamma \ \vdash \mathbf{(f\ t1)} : \mathbf{T2}} \text{ APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression **f t1** has type **T2**
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Function Application

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash \mathbf{f} : \mathbf{T1} \to \mathbf{T2} \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, **f** takes a **T1** to a **T2**, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Function Application

Function Application

$$\boxed{\boldsymbol{\Gamma \vdash t : T}}$$

$$\frac{\Gamma \vdash f : \mathbf{T1} \rightarrow T2 \qquad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular **T1** (which we call t1) to f, you'll get a T2

Functions and Application

Function Application

Function Application

$$\boxed{\mathbf{\Gamma \vdash t : T}}$$

$$\frac{\Gamma \vdash f : \mathbf{T1}\rightarrow T2 \quad \Gamma \vdash t1 : \mathbf{T1}}{\Gamma \vdash (f\ \mathbf{t1}) : T2}\ \textbf{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular **T1** (which we call **t1**) to f, you'll get a T2

Functions and Application

Function Application

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash f : T1 \rightarrow \mathbf{T2} \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : \mathbf{T2}}\ \textbf{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a **T2**

Functions and Application

A "Simple" (Monomorphic) Type System

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash f : \mathbf{T1} \rightarrow \mathbf{T2} \quad \Gamma \vdash t1 : \mathbf{T1}}{\Gamma \vdash (f\ t1) : \mathbf{T2}}\ \text{APP}$$

- If you have a function f of type T1→T2, and you apply it to an argument t1 of type T1, the resulting expression f t1 has type T2
- That is, f takes a T1 to a T2, so if you pass a particular T1 (which we call t1) to f, you'll get a T2

Functions and Application

A “Simple” (Monomorphic) Type System

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \ \vdash f : \mathbf{T1 \rightarrow T2} \quad \Gamma \ \vdash t1 \ : T1}{\Gamma \ \vdash (f\ t1) : T2} \ \textbf{APP}$$

Functions

$$\frac{\Gamma, t1 : T1 \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \text{ -> } t2 \ : \mathbf{T1 \rightarrow T2}} \ \textbf{FUN}$$

Functions and Application

A “Simple” (Monomorphic) Type System

Function Application

$$\boxed{\mathbf{\Gamma \vdash t : T}}$$

$$\frac{\Gamma \ \vdash f : \mathbf{T1} \rightarrow \mathbf{T2} \qquad \Gamma \ \vdash t1 \ : T1}{\Gamma \ \vdash (f\ t1) : T2}\ \textbf{APP}$$

Functions

$$\frac{\Gamma, \mathbf{t1 : T1} \vdash t2 : T2}{\Gamma \vdash \text{fun } \mathbf{t1} \text{ -> } t2 \ : \mathbf{T1} \rightarrow T2}\ \textbf{FUN}$$

Functions and Application

A “Simple” (Monomorphic) Type System

Function Application

$\boxed{\mathbf{\Gamma \vdash t : T}}$

$$\frac{\Gamma \vdash f : \mathbf{T1 \rightarrow T2} \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

Functions

$$\frac{\Gamma, t1 : T1 \vdash \mathbf{t2} : \mathbf{T2}}{\Gamma \vdash \text{fun } t1 \text{ -> } \mathbf{t2} : T1 \rightarrow \mathbf{T2}}\ \textbf{FUN}$$

Functions and Application

A “Simple” (Monomorphic) Type System

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash f : T1 \to T2 \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

Functions

Extending the type environment

$$\frac{\mathbf{\Gamma, t1 : T1} \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \text{ -> } t2 : T1 \to T2}\ \textbf{FUN}$$

Functions and Application

A "Simple" (Monomorphic) Type System

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \ \vdash f : T1 \to T2 \quad \Gamma \ \vdash t1 \ : T1}{\Gamma \ \vdash (f\ t1) : T2}\ \textbf{APP}$$

Functions

Extending the type environment

$$\frac{\mathbf{\{t1 : T1\} + \Gamma} \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \text{ -> } t2 \ : T1 \to T2}\ \textbf{FUN}$$

Functions and Application

A “Simple” (Monomorphic) Type System

Function Application

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{\Gamma \vdash f : T1 \rightarrow T2 \quad \Gamma \vdash t1 : T1}{\Gamma \vdash (f\ t1) : T2}\ \textbf{APP}$$

Functions

Extending the type environment

$$\frac{\mathbf{\Gamma, t1 : T1} \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \text{ -> } t2 : T1 \rightarrow T2}\ \textbf{FUN}$$

Functions and Application

A "Simple" (Monomorphic) Type System

$\boxed{\Gamma \vdash t : T}$

Rules describe types, but also how **type environment** Γ may change. Can only do what rule allows!

Functions

$$\frac{\Gamma, \mathbf{t1 : T1} \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \text{ -> } t2 \; : T1 \rightarrow T2} \; \textbf{FUN}$$

Functions and Application

Fun Examples

$$\frac{\Gamma, y : \text{int} \vdash y + 3 \ : \text{int}}{\Gamma \vdash \mathbf{fun\ y\ \text{->}\ y + 3} \ : \text{int} \rightarrow \text{int}}$$

Fun Examples

$$\frac{\Gamma, y : \text{int} \vdash y + 3 : \text{int}}{\Gamma \vdash \text{fun } y \text{ -> } y + 3 : \textbf{int} \rightarrow \textbf{int}}$$

Fun Examples

$$\frac{\Gamma,\ \mathbf{y} : \mathbf{int} \vdash \text{y} + 3 : \text{int}}{\Gamma \vdash \text{fun } \mathbf{y} \text{ -> y} + 3 : \mathbf{int} \rightarrow \text{int}}\ \textbf{FUN}$$

Fun Examples

$$\frac{\Gamma, y : \text{int} \vdash \mathbf{y + 3 : int}}{\Gamma \vdash \text{fun } y \text{ -> } \mathbf{y + 3} : \text{int} \rightarrow \mathbf{int}}\ \textbf{FUN}$$

Fun Examples

$$\frac{\Gamma, y : \text{int} \vdash y + 3 \;:\; \text{int}}{\Gamma \vdash \text{fun } y \text{ -> } y + 3 \;:\; \text{int} \rightarrow \text{int}}\;\textbf{FUN}$$

This is just one step of the derivation

Questions so far?

Let Expressions

(Monomorphic) Let and Let Rec

$\boxed{\Gamma \vdash t : T}$

let

$$\frac{\Gamma \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 \; : T2}{\Gamma \vdash \textbf{(let x = t1 in t2)} : T2}\;\textsf{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \;|\!- \text{(let rec x = t1 in t2)} : T2}$$

Let Expressions

(Monomorphic) Let and Let Rec

$$\boxed{\mathbf{\Gamma \vdash t : T}}$$

let

$$\frac{\Gamma \vdash \mathbf{t1 : T1} \qquad \Gamma, x : T1 \vdash t2 \ : T2}{\Gamma \vdash (\text{let } x = \mathbf{t1} \text{ in } t2) : T2}\ \textbf{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad\qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \ |\text{-}\ (\text{let rec } x = t1 \text{ in } t2) : T2}$$

(Monomorphic) Let and Let Rec

$$\boxed{\mathbf{\Gamma \vdash t : T}}$$

let

$$\frac{\Gamma \vdash t1 : T1 \qquad \mathbf{\Gamma, x : T1 \vdash t2 : T2}}{\Gamma \vdash (\text{let } x = t1 \text{ in } \mathbf{t2}) : T2}\ \text{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \mid\!- (\text{let rec } x = t1 \text{ in } t2) : T2}$$

Let Expressions

(Monomorphic) Let and Let Rec

$\boxed{\Gamma \vdash t : T}$

let

$$\frac{\Gamma \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 \ : \mathbf{T2}}{\Gamma \vdash \mathbf{(let\ x = t1\ in\ t2)} : \mathbf{T2}}\ \text{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \mid\!\!- \text{(let rec x = t1 in t2)} : T2}$$

Let Expressions

(Monomorphic) Let and Let Rec

$$\boxed{\Gamma \vdash t : T}$$

let

$$\frac{\Gamma \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \vdash (\text{let } x = t1 \text{ in } t2) : T2}\ \text{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \mid\!\text{-}\ (\text{let rec } x = t1 \text{ in } t2) : T2}\ \text{LET-REC}$$

Let Expressions

(Monomorphic) Let and Let Rec

$$\boxed{\mathbf{\Gamma \vdash t : T}}$$

let

$$\frac{\mathbf{\Gamma \vdash t1 : T1} \qquad \Gamma, x : T1 \vdash t2 \ : T2}{\Gamma \vdash (\text{let } x = t1 \text{ in } t2) : T2}\ \textbf{LET}$$

let rec

$$\frac{\mathbf{\Gamma, \boxed{x : T1} \vdash t1 : T1} \qquad\qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \;|\text{-}\; (\text{let rec } x = t1 \text{ in } t2) : T2}\ \textbf{LET-REC}$$

Let Expressions

A “Simple” (Monomorphic) Type System

$$\boxed{\Gamma \vdash t : T}$$

let

$$\frac{\Gamma \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma \vdash (\text{let } x = t1 \text{ in } t2) : T2}\ \textbf{LET}$$

let rec

$$\frac{\Gamma, x : T1 \vdash t1 : T1 \qquad \Gamma, x : T1 \vdash t2 : T2}{\Gamma\ |\text{-}\ (\text{let rec } x = t1 \text{ in } t2) : T2}\ \textbf{LET-REC}$$

Let Expressions

Example (pretend we have lists)

Which rule do we apply?

$$\frac{???}{\begin{array}{l}\{\} \vdash \text{let rec one} = 1 :: \text{one in} \\ \qquad \text{let } x = 2 \text{ in} \\ \qquad \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}\end{array}}$$

Example

The let rec rule:

$$\frac{\begin{array}{l}\{\text{one} : \text{int list}\} \vdash \\ (1 :: \text{one}) : \text{int list}\end{array} \qquad \begin{array}{l}\{\text{one} : \text{int list}\} \vdash \\ \quad \text{let } x = 2 \text{ in} \\ \text{fun } y \text{ -> } (x :: y :: \text{one}) : \\ \text{int} \to \text{int list}\end{array}}{\begin{array}{l}\{\} \vdash \text{let rec one} = 1 :: \text{one in} \\ \qquad \text{let } x = 2 \text{ in} \\ \qquad \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \to \text{int list}\end{array}} \;\textbf{LET-REC}$$

Example

Need more room …

$$\dfrac{\begin{array}{l}\{\text{one} : \text{int list}\} \vdash \\ (1 :: \text{one}) : \text{int list}\end{array} \qquad \cdots}{\begin{array}{l}\{\} \vdash \text{let rec one} = 1 :: \text{one in} \\ \qquad \text{let x} = 2 \text{ in} \\ \qquad \text{fun y -> (x :: y :: one)} : \text{int} \rightarrow \text{int list}\end{array}}\ \textbf{LET-REC}$$

Example

Binary operator (saw last class)

$$\dfrac{\dfrac{\{\text{one} : \text{int list}\} \vdash 1 : \text{int} \qquad \{\text{one} : \text{int list}\} \vdash \text{one} : \text{int list}}{\{\text{one} : \text{int list}\} \vdash (1 :: \text{one}) : \text{int list}}\ \textbf{BIN-OP} \qquad \cdots}{\{\} \vdash \text{let rec one = 1 :: one in let x = 2 in fun y -> (x :: y :: one)} : \text{int} \to \text{int list}}\ \textbf{LET-REC}$$

Example

Trivial

CONST

{one : int list} ⊢
1: int

VAR

{one : int list} ⊢
one : int list

BIN-OP

{one : int list} ⊢
(1 :: one) : int list

LET-REC

{} ⊢ let rec one = 1 :: one in
let x = 2 in
fun y -> (x :: y :: one) : int → int list

Let Expressions

Example

To the next slide …

$$\dfrac{\dfrac{\dfrac{}{\{\text{one} : \text{int list}\} \vdash 1 : \text{int}}\,\text{CONST} \qquad \dfrac{}{\{\text{one} : \text{int list}\} \vdash \text{one} : \text{int list}}\,\text{VAR}}{\{\text{one} : \text{int list}\} \vdash (1 :: \text{one}) : \text{int list}}\,\text{BIN-OP}}{\begin{array}{l}\{\} \vdash \text{let rec one = 1 :: one in} \\ \qquad \text{let x = 2 in} \\ \qquad \text{fun y -> (x :: y :: one)} : \text{int} \to \text{int list}\end{array}}\,\text{LET-REC}$$

Example

… from the last slide

$$\frac{???}{\{\text{one} : \text{int list}\} \vdash \text{let } x = 2 \text{ in fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}$$

Example

What rule?

$$\frac{\textbf{???}}{\{\text{one} : \text{int list}\} \vdash \text{let } x = 2 \text{ in fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}$$

Let Expressions

Example

The let rule

$$\frac{\{\text{one} : \text{int list}\} \vdash 2 : \text{int} \qquad \begin{array}{l}\{\text{x} : \text{int}, \text{one} : \text{int list}\} \vdash \\ \quad \text{fun y -> (x :: y :: one)} \\ : \text{int} \rightarrow \text{int list}\end{array}}{\begin{array}{l}\{\text{one} : \text{int list}\} \vdash \\ \quad \text{let x = 2 in fun y -> (x :: y :: one)} \\ : \text{int} \rightarrow \text{int list}\end{array}}\ \textbf{LET}$$

Example

Need more room …

$$\dfrac{\{\text{one} : \text{int list}\} \vdash 2 : \text{int}}{\{\text{one} : \text{int list}\} \vdash \text{let x = 2 in fun y -> (x :: y :: one)} : \text{int} \rightarrow \text{int list}}\ \textbf{LET}$$

Example

What next?

$$\dfrac{\dfrac{???}{\{\text{one} : \text{int list}\} \vdash 2 : \text{int}}}{\{\text{one} : \text{int list}\} \vdash \text{let x = 2 in fun y -> (x :: y :: one)} : \text{int} \to \text{int list}}\ \text{LET}$$

Example

Trivial

CONST

{one : int list} ⊢ 2 : int

LET

{one : int list} ⊢ let x = 2 in fun y -> (x :: y :: one) : int → int list

Example

To the next slide …

CONST

$$\frac{}{\{one : int\ list\} \vdash 2 : int}$$

LET

$$\frac{\{one : int\ list\} \vdash 2 : int}{\{one : int\ list\} \vdash let\ x = 2\ in\ fun\ y \text{ -> } (x :: y :: one) : int \to int\ list}$$

Let Expressions

Example

… from the last slide

???

{x : int, one : int list} ⊢

fun y -> (x :: y :: one)

: int → int list

Example

What rule?

$$\frac{???}{\{x : \text{int}, \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}$$

Let Expressions

Example

Fun!

$$\dfrac{\{y : \text{int},\ x : \text{int},\ \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}{\{x : \text{int},\ \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\ \textbf{FUN}$$

Example

Binary operator (saw last class)

$$\dfrac{\dfrac{\begin{array}{l}\{y : \text{int}, x : \text{int},\\ \ \text{one} : \text{int list}\} \vdash\\ \ \ x : \text{int}\end{array} \qquad \begin{array}{l}\{y : \text{int}, x : \text{int},\\ \text{one} : \text{int list}\} \vdash\\ (y :: \text{one}) : \text{int list}\end{array}}{\begin{array}{l}\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash\\ (x :: y :: \text{one}) : \text{int list}\end{array}}\ \textbf{BIN-OP}}{\begin{array}{l}\{x : \text{int}, \text{one} : \text{int list}\} \vdash\\ \quad \text{fun } y \text{ -> } (x :: y :: \text{one})\\ : \text{int} \rightarrow \text{int list}\end{array}}\ \textbf{FUN}$$

Example

Trivial

$$\dfrac{\dfrac{\dfrac{}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash x : \text{int}}\,\text{VAR} \qquad \{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (y :: \text{one}) : \text{int list}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}\,\text{BIN-OP}}{\{x : \text{int}, \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\,\text{FUN}$$

Let Expressions

Example

What rule?

$$\dfrac{\dfrac{\dfrac{}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash x : \text{int}}\,\text{VAR} \qquad \dfrac{???}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (y :: \text{one}) : \text{int list}}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}\,\text{BIN-OP}}{\{x : \text{int}, \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\,\text{FUN}$$

Example

Binary operator

{y : int, …} ⊢ y : int   {… , one : int list} ⊢ one : int list   **BIN-OP**

VAR

{y : int, x : int, one : int list} ⊢ x : int

{y : int, x : int, one : int list} ⊢ (y :: one) : int list   **BIN-OP**

{y : int, x : int, one : int list} ⊢ (x :: y :: one) : int list   **FUN**

{x : int, one : int list} ⊢ fun y -> (x :: y :: one) : int → int list

Let Expressions

Example

Trivial

$$\dfrac{\dfrac{\dfrac{}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash x : \text{int}}\,\text{VAR} \qquad \dfrac{\dfrac{}{\{y : \text{int}, \ldots\} \vdash y : \text{int}}\,\text{VAR} \quad \dfrac{}{\{\ldots , \text{one} : \text{int list}\} \vdash \text{one} : \text{int list}}\,\text{VAR}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (y :: \text{one}) : \text{int list}}\,\text{BIN-OP}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}\,\text{BIN-OP}}{\{x : \text{int}, \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\,\text{FUN}$$

Let Expressions

Example

QED

$$
\dfrac{
 \dfrac{
   \dfrac{}{\{\text{one} : \text{int list}\} \vdash 1 : \text{int}}\;\textbf{CONST}
   \qquad
   \dfrac{}{\{\text{one} : \text{int list}\} \vdash \text{one} : \text{int list}}\;\textbf{VAR}
 }{\{\text{one} : \text{int list}\} \vdash (1 :: \text{one}) : \text{int list}}\;\textbf{BIN-OP}
}{\begin{array}{l}\{\} \vdash \text{let rec one} = 1 :: \text{one in} \\ \qquad \text{let } x = 2 \text{ in} \\ \qquad \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}\end{array}}\;\textbf{LET-REC}
$$

Example

QED

$$\dfrac{\dfrac{}{\{\text{one} : \text{int list}\} \vdash 2 : \text{int}}\,\text{CONST}}{\{\text{one} : \text{int list}\} \vdash \text{let } x = 2 \text{ in fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\,\text{LET}$$

Example

QED

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash x : \text{int}}\ \text{VAR} \qquad \dfrac{\dfrac{}{\{y : \text{int}, \ldots\} \vdash y : \text{int}}\ \text{VAR} \quad \dfrac{}{\{\ldots , \text{one} : \text{int list}\} \vdash \text{one} : \text{int list}}\ \text{VAR}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (y :: \text{one}) : \text{int list}}\ \text{BIN-OP}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}\ \text{BIN-OP}}{\{x : \text{int}, \text{one} : \text{int list}\} \vdash \text{fun } y \text{ -> } (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}}\ \text{FUN}}{}$$

Questions so far?

The Hidden Glory

Big Picture: Type Checking

- It's **a lot of work** to write this out by hand
- Thankfully, you normally don't have to! Just **implement type checking** once, and let the compiler do it for you!
- This is a class though so you'll have to do it **manually** sometimes just to make sure you understand it well enough (sorry…)
- And anyways, it turns out knowing how to write **proofs** this way is ***way more general*** than one might imagine …

Hidden Glory

Big Picture: Curry-Howard

- **Type systems ⇔ logical systems**
- **Types ⇔ propositions**
- **Terms ⇔ proofs** (of the propositions that their types represent)
- **Type checking ⇔ proof checking**
- So with a fancy enough type system, you can write pretty much any proof you want this this way and **have a computer check it automatically**
- And if you don't feel like writing it by hand, you can write **automation** that writes it for/with you

Hidden Glory

Big Picture: Curry-Howard

Modus ponens

$$\frac{\mathbf{A} \Rightarrow \mathbf{B} \qquad \mathbf{A}}{\mathbf{B}}$$

Function Application

$$\frac{\Gamma \vdash f : \mathbf{A} \rightarrow \mathbf{B} \qquad \Gamma \vdash a : \mathbf{A}}{\Gamma \mid\!\!- \; f\, a : \mathbf{B}}$$

Hidden Glory

Big Picture: Curry-Howard

- **Type systems** ⇔ **logical systems**
- **Types** ⇔ **propositions**
- **Terms** ⇔ **proofs** (of the propositions that their types represent)
- **Type checking** ⇔ **proof checking**
- So with a fancy enough type system, you can write pretty much any proof you want this this way and **have a computer check it automatically**
- And if you don’t feel like writing it by hand, you can write **automation** that writes it for/with you

Hidden Glory

Big Picture: Curry-Howard

- **Type systems ⇔ logical systems**
- **Types ⇔ propositions**
- **Terms ⇔ proofs** (of the propositions that their types repre
- **Type chec

Ask me in office hours!!!
This is my research area!

- So with a fancy enough type system, you can write pretty much any proof you want this this way and **have a computer check it automatically**
- And if you don’t feel like writing it by hand, you can write **automation** that writes it for/with you

Hidden Glory

Questions?

Next Class: Polymorphism

Next Class

- **I will be away next week!**
  - I will miss office hours and both lectures.
  - **Prof. Elsa Gunter will cover lectures**.
  - This is my final planned absence.
- **Quiz 3** on **MP5** is next Tuesday
- All deadlines can be found on **course website**
- Use **office hours** and **class forums** for help