



---

# 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

---

**Premises**

---

**Conclusion**

Judgments, Systems, **Derivations**



## Reminder: Syntax

---

**“I like all chocolate truffles”**

---

**“I like Godiva chocolate truffles”**

Judgments, Systems, **Derivations**

**Premise 1 ... Premise n**

**Conclusion**

Judgments, Systems, **Derivations**



**Premises of**    **Premises of**  
**Premise 1**    **Premise n**  
**Premise 1 ... Premise n**  
**Conclusion**



# Reminder: Syntax

---

---

## **Axiom**

Judgments, Systems, **Derivations**



# Reminder: Syntax

---

---

**“other people are  
conscious, too”**

Judgments, Systems, **Derivations**



# Reminder: Syntax

---

**Axiom**

**Axiom**

**Premise 1 ... Premise n**

**Conclusion**

Judgments, Systems, **Derivations**



# Reminder: Syntax

---

**Axiom**

**Axiom**

**Premise**

**Premise**

**Conclusion**

**Conclusion**

Judgments, **Systems**, **Derivations**



# Reminder: Syntax

---

syntax of judgment  
should go in this box

Axiom

Axiom

Premise

Premise

Conclusion

Conclusion

Judgments, Systems, Derivations



# Reminder: Syntax

---

$\Gamma \vdash t : T$

$\Gamma \vdash \text{true} : \text{bool}$

$\Gamma \vdash \text{false} : \text{bool}$

**Premise**

**Conclusion**

**Premise**

**Conclusion**

**Judgments, Systems, Derivations**



# Reminder: Syntax

---

$$\Gamma \vdash t : T$$
$$\frac{}{\Gamma \vdash \text{true} : \text{bool}}$$
$$\frac{}{\Gamma \vdash \text{false} : \text{bool}}$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : \text{bool}}{\Gamma \vdash t_1 \ \&\& \ t_2 : \text{bool}}$$

## Judgments, Systems, Derivations





# Type Checking

---



# A “Simple” (Monomorphic) Type System

---

$$\Gamma \vdash t : T$$

**Type Checking**



# A "Simple" (Monomorphic) Type System

---

## Axioms

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



# A "Simple" (Monomorphic) Type System

---

## Axioms

$$\Gamma \vdash t : T$$
$$\frac{}{\Gamma \vdash \text{true} : \text{bool}}$$
$$\frac{}{\Gamma \vdash \text{false} : \text{bool}}$$
$$\frac{}{\Gamma \vdash n : \text{int}}$$

(for  $n$  integer constant)



# A "Simple" (Monomorphic) Type System

---

## Axioms

$$\Gamma \vdash t : T$$
$$\frac{}{\Gamma \vdash \text{true} : \text{bool}}$$
$$\frac{}{\Gamma \vdash \text{false} : \text{bool}}$$
$$\frac{}{\Gamma \vdash n : \text{int}}$$

(for  $n$  integer constant)

$$\frac{}{\Gamma \vdash v : T}$$

(if  $\Gamma(v) = T$ )

# A "Simple" (Monomorphic) Type System

## Axioms

$$\Gamma \vdash t : T$$
$$\frac{}{\Gamma \vdash \text{true} : \text{bool}} \text{CONST}$$
$$\frac{}{\Gamma \vdash \text{false} : \text{bool}} \text{CONST}$$
$$\frac{}{\Gamma \vdash n : \text{int}} \text{CONST} \quad (\text{for } n \text{ integer constant})$$
$$\frac{}{\Gamma \vdash v : T} \text{VAR} \quad (\text{if } \Gamma(v) = T)$$

# A "Simple" (Monomorphic) Type System

## Boolean Connectives

$$\Gamma \vdash t : T$$

$$\frac{\Gamma \vdash \mathbf{t1} : \mathbf{bool} \quad \Gamma \vdash t2 : \mathbf{bool}}{\Gamma \vdash t1 \ \&\& \ t2 : \mathbf{bool}} \text{ AND}$$

$$\frac{\Gamma \vdash t1 : \mathbf{bool} \quad \Gamma \vdash t2 : \mathbf{bool}}{\Gamma \vdash t1 \ || \ t2 : \mathbf{bool}}$$

Type Checking

# A "Simple" (Monomorphic) Type System

## Boolean Connectives

$$\Gamma \vdash t : T$$

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

$$\frac{\Gamma \vdash t1 : \text{bool} \quad \Gamma \vdash t2 : \text{bool}}{\Gamma \vdash t1 \ || \ t2 : \text{bool}}$$



# A "Simple" (Monomorphic) Type System

## Boolean Connectives

$$\Gamma \vdash t : T$$

$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : \text{bool}}{\Gamma \vdash t_1 \ \&\& \ t_2 : \text{bool}} \text{ AND}$$

$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : \text{bool}}{\Gamma \vdash t_1 \ || \ t_2 : \text{bool}}$$

# A "Simple" (Monomorphic) Type System

## Boolean Connectives

$$\Gamma \vdash t : T$$

$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : \text{bool}}{\Gamma \vdash t_1 \ \&\& \ t_2 : \text{bool}} \text{ AND}$$

$$\frac{\Gamma \vdash \mathbf{t1} : \text{bool} \quad \Gamma \vdash \mathbf{t2} : \text{bool}}{\Gamma \vdash \mathbf{t1} \ || \ \mathbf{t2} : \text{bool}} \text{ OR}$$



# Example: Type Checking

---

**How do we check this?**

---

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



# Example: Type Checking

---

**Premises?**

{b1 : bool} ⊢ b1 && false : bool AND



# Example: Type Checking

---

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

# Example: Type Checking

How do we check this?

How do we check this?

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

# Example: Type Checking

How do we check this?

$$\frac{\frac{}{\{b1 : \text{bool}\} \vdash b1 : \text{bool}} \text{VAR} \quad \frac{}{\{b1 : \text{bool}\} \vdash \text{false} : \text{bool}}}{\{b1 : \text{bool}\} \vdash b1 \ \&\& \ \text{false} : \text{bool}} \text{AND}$$

Type Checking



# Example: Type Checking

---

$$\frac{\frac{}{\{b1 : \text{bool}\} \vdash b1 : \text{bool}} \text{VAR} \quad \frac{}{\{b1 : \text{bool}\} \vdash \text{false} : \text{bool}} \text{CONST}}{\{b1 : \text{bool}\} \vdash b1 \ \&\& \ \text{false} : \text{bool}} \text{AND}$$





# Type Derivation

---

$$\frac{\frac{}{\{b1 : \text{bool}\} \vdash b1 : \text{bool}} \text{VAR} \quad \frac{}{\{b1 : \text{bool}\} \vdash \text{false} : \text{bool}} \text{CONST}}{\{b1 : \text{bool}\} \vdash b1 \ \&\& \ \text{false} : \text{bool}} \text{AND}$$



Questions so far?

---



# Type Variables

---

# Type Variables in Rules

## Conditionals

$$\Gamma \vdash t : T$$

$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } t_2 \text{ else } t_3) : T}$$

- 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

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

- 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

$$\Gamma \vdash t : T$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } \mathbf{t_2} \text{ else } \mathbf{t_3}) : T}$$

- 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

$$\Gamma \vdash t : T$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } \mathbf{t_2} \text{ else } \mathbf{t_3}) : T}$$

- 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

$$\Gamma \vdash t : T$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } \mathbf{t_2} \text{ else } \mathbf{t_3}) : T}$$

- 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

$$\Gamma \vdash t : T$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } t_2 \text{ else } t_3) : T}$$

- 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

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

- 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

$$\Gamma \vdash t : T$$
$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash \mathbf{t_2} : T \quad \Gamma \vdash \mathbf{t_3} : T}{\Gamma \vdash (\mathbf{\text{if } t_1 \text{ then } t_2 \text{ else } t_3}) : T}$$

- 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

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

- 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

# A "Simple" (Monomorphic) Type System

## Conditionals

$$\Gamma \vdash t : T$$

$$\frac{\Gamma \vdash t_1 : \text{bool} \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T}{\Gamma \vdash (\text{if } t_1 \text{ then } t_2 \text{ else } t_3) : T} \text{COND}$$

- 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?**

???

{ b : bool } ⊢ (if b then 5 else false) : ???

**Type Variables**



# Question

---

**Why is this well typed?**

???

---

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



Questions so far?

---





# Functions and Application

---

# Function Application

## Function Application

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , and you apply it to an argument  $\mathbf{t1}$  of type  $\mathbf{T1}$ , the resulting expression  $\mathbf{f t1}$  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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

- If you have a function  $f$  of type  $T1 \rightarrow T2$ , 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

$$\Gamma \vdash t : T$$

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

## Functions

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

## Functions and Application

# A "Simple" (Monomorphic) Type System

## Function Application

$$\Gamma \vdash t : T$$

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

## Functions

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

## Functions and Application

# A "Simple" (Monomorphic) Type System

## Function Application

 $\Gamma \vdash t : T$ 

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

## Functions

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

## Functions and Application

# A "Simple" (Monomorphic) Type System

## Function Application

$$\Gamma \vdash t : T$$

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

## Functions

Extending the type environment

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

## Functions and Application

# A "Simple" (Monomorphic) Type System

## Function Application

$$\Gamma \vdash t : T$$

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

## Functions

Extending the type environment

$$\frac{\{t1 : T1\} + \Gamma \vdash t2 : T2}{\Gamma \vdash \text{fun } t1 \rightarrow t2 : T1 \rightarrow T2} \text{ FUN}$$

## Functions and Application

# A "Simple" (Monomorphic) Type System

## Function Application

$$\Gamma \vdash t : T$$

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

## Functions

Extending the type environment

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

## Functions and Application



# A "Simple" (Monomorphic) Type System

$$\Gamma \vdash t : T$$

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

## Functions

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

## Functions and Application



# Fun Examples

---

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



# Fun Examples

---

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



# Fun Examples

---

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



# Fun Examples

---

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

## Functions and Application



# Fun Examples

---

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

This is just one step of the derivation



Questions so far?

---



# Let Expressions

---



# (Monomorphic) Let and Let Rec

 $\Gamma \vdash t : T$ 

**let**

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\mathbf{let\ } x = t_1 \mathbf{ in\ } t_2) : T_2} \text{LET}$$

**let rec**

$$\frac{\Gamma, x : T_1 \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\mathbf{let\ rec\ } x = t_1 \mathbf{ in\ } t_2) : T_2}$$

**Let Expressions**

# (Monomorphic) Let and Let Rec

$$\Gamma \vdash t : T$$

**let**

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

**let rec**

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

**Let Expressions**

# (Monomorphic) Let and Let Rec

$$\Gamma \vdash t : T$$

**let**

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let } x = t_1 \text{ in } t_2) : T_2} \text{LET}$$

**let rec**

$$\frac{\Gamma, x : T_1 \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let rec } x = t_1 \text{ in } t_2) : T_2}$$

**Let Expressions**

# (Monomorphic) Let and Let Rec

**let**

$\Gamma \vdash t : T$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let } x = t_1 \text{ in } t_2) : T_2} \text{LET}$$

**let rec**

$$\frac{\Gamma, x : T_1 \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let rec } x = t_1 \text{ in } t_2) : T_2}$$

**Let Expressions**

# (Monomorphic) Let and Let Rec

 $\Gamma \vdash t : T$ 

**let**

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

**let rec**

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

**Let Expressions**

# (Monomorphic) Let and Let Rec

**let**

$\Gamma \vdash t : T$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let } x = t_1 \text{ in } t_2) : T_2} \text{LET}$$

**let rec**

$$\frac{\Gamma, x : T_1 \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let rec } x = t_1 \text{ in } t_2) : T_2} \text{LET-REC}$$

**Let Expressions**

# A "Simple" (Monomorphic) Type System

$$\Gamma \vdash t : T$$

**let**

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let } x = t_1 \text{ in } t_2) : T_2} \text{LET}$$

**let rec**

$$\frac{\Gamma, x : T_1 \vdash t_1 : T_1 \quad \Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash (\text{let rec } x = t_1 \text{ in } t_2) : T_2} \text{LET-REC}$$

**Let Expressions**

# Example (pretend we have lists)

**Which rule do we apply?**

???

---

$\{ \} \vdash$  let rec one = 1 :: one in  
let x = 2 in  
fun y -> (x :: y :: one) : int  $\rightarrow$  int list

**Let Expressions**



# Example

## The let rec rule:

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

## Let Expressions

# Example

**Need more room ...**

$\{\text{one} : \text{int list}\} \vdash$

$(1 :: \text{one}) : \text{int list}$

---

$\{\} \vdash \text{let rec one} = 1 :: \text{one} \text{ in}$

$\text{let } x = 2 \text{ in}$

$\text{fun } y \rightarrow (x :: y :: \text{one}) : \text{int} \rightarrow \text{int list}$

LET-REC

**Let Expressions**

# Example

## Binary operator (saw last class)

$$\frac{\{one : int\ list\} \vdash 1 : int \quad \{one : int\ list\} \vdash one : int\ list}{\text{BIN-OP}}$$
$$\{one : int\ list\} \vdash (1 :: one) : int\ list$$

LET-REC

$$\{\} \vdash \text{let rec one} = 1 :: one \text{ in}$$
$$\quad \text{let } x = 2 \text{ in}$$
$$\quad \text{fun } y \rightarrow (x :: y :: one) : int \rightarrow int\ list$$

## Let Expressions

# Example

**Trivial**

$$\frac{\frac{}{\{one : int\ list\} \vdash 1 : int} \text{CONST} \quad \frac{}{\{one : int\ list\} \vdash one : int\ list} \text{VAR}}{} \text{BIN-OP}$$
$$\frac{\{one : int\ list\} \vdash (1 :: one) : int\ list}{}$$

LET-REC

$$\{\} \vdash \text{let rec one} = 1 :: one \text{ in}$$
$$\quad \text{let } x = 2 \text{ in}$$
$$\quad \text{fun } y \rightarrow (x :: y :: one) : int \rightarrow int\ list$$

## Let Expressions

# Example

To the next slide ...

$$\frac{\text{CONST}}{\{one : int\ list\} \vdash 1 : int} \quad \frac{\text{VAR}}{\{one : int\ list\} \vdash one : int\ list} \quad \text{BIN-OP}$$
$$\{one : int\ list\} \vdash (1 :: one) : int\ list$$
$$\{\} \vdash \text{let rec one} = 1 :: one \text{ in}$$
$$\text{let } x = 2 \text{ in}$$
$$\text{fun } y \rightarrow (x :: y :: one) : int \rightarrow int\ list$$

LET-REC

## Let Expressions

# Example

... from the last slide

???

$\{one : \text{int list}\} \vdash$

$\text{let } x = 2 \text{ in fun } y \rightarrow (x :: y :: one)$

$: \text{int} \rightarrow \text{int list}$

**Let Expressions**

# Example

**What rule?**

???

---

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

**Let Expressions**

# Example

## The let rule

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

## Let Expressions



# Example

**Need more room ...**

$\{one : \text{int list}\} \vdash$

$2 : \text{int}$

LET

$\{one : \text{int list}\} \vdash$

$\text{let } x = 2 \text{ in fun } y \rightarrow (x :: y :: one)$

$: \text{int} \rightarrow \text{int list}$

## Let Expressions

# Example

**What next?**

???

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

$2 : int$

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

$let\ x = 2\ in\ fun\ y \rightarrow (x :: y :: one)$

$: int \rightarrow int\ list$

LET

## Let Expressions

# Example

**Trivial**

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

---

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

LET

## Let Expressions

# Example

To the next slide ...

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

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

LET

## Let Expressions

# Example

... from the last slide

???

---

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

**Let Expressions**

# Example

**What rule?**

???

---

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

**Let Expressions**

# Example

**Fun!**

$\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash$   
 $(x :: y :: \text{one}) : \text{int list}$

**FUN**

---

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

**Let Expressions**

# Example

## Binary operator (saw last class)

$$\{y : \text{int}, x : \text{int}, \\ \text{one} : \text{int list}\} \vdash \\ x : \text{int}$$
$$\{y : \text{int}, x : \text{int}, \\ \text{one} : \text{int list}\} \vdash \\ (y :: \text{one}) : \text{int list}$$

**BIN-OP**

$$\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash \\ (x :: y :: \text{one}) : \text{int list}$$

**FUN**

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

## Let Expressions



# Example

**Trivial**

**VAR**

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $x : \text{int}$

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $(y :: \text{one}) : \text{int list}$

**BIN-OP**

$\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash$   
 $(x :: y :: \text{one}) : \text{int list}$

**FUN**

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

**Let Expressions**

# Example

**What rule?**

???

VAR

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $x : \text{int}$

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $(y :: \text{one}) : \text{int list}$

BIN-OP

$\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash$   
 $(x :: y :: \text{one}) : \text{int list}$

FUN

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

## Let Expressions

# Example

**Binary operator**

$$\frac{\{y : \text{int}, \dots\} \quad \{\dots, \text{one} : \text{int list}\} \quad \vdash y : \text{int} \quad \vdash \text{one} : \text{int list}}{\text{BIN-OP}}$$

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $x : \text{int}$

$\{y : \text{int}, x : \text{int},$   
 $\text{one} : \text{int list}\} \vdash$   
 $(y :: \text{one}) : \text{int list}$   
**BIN-OP**

$\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash$   
 $(x :: y :: \text{one}) : \text{int list}$   
**FUN**

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

**Let Expressions**

# Example

**Trivial**

$$\frac{\frac{\text{VAR}}{\{y : \text{int}, \dots\}} \quad \frac{\text{VAR}}{\{\dots, \text{one} : \text{int list}\}}}{\frac{\vdash y : \text{int} \quad \vdash \text{one} : \text{int list}}{\text{BIN-OP}}}$$

$$\frac{\frac{\text{VAR}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\}} \vdash x : \text{int} \quad \frac{\text{BIN-OP}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (y :: \text{one}) : \text{int list}}}{\text{BIN-OP}}$$

$$\frac{\text{FUN}}{\frac{\text{BIN-OP}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\} \vdash (x :: y :: \text{one}) : \text{int list}}}$$

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

## Let Expressions

# Example

**QED**

$$\frac{\frac{\text{CONST}}{\{one : int\ list\} \vdash 1 : int} \quad \frac{\text{VAR}}{\{one : int\ list\} \vdash one : int\ list}}{\text{BIN-OP}}}{\frac{\text{LET-REC}}{\{ \} \vdash \text{let rec one} = 1 :: one \text{ in} \\ \text{let } x = 2 \text{ in} \\ \text{fun } y \rightarrow (x :: y :: one) : int \rightarrow int\ list}}$$

**Let Expressions**

# Example

**QED**

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

LET

## Let Expressions

# Example

**QED**

$$\frac{\frac{\text{VAR}}{\{y : \text{int}, \dots\}} \quad \frac{\text{VAR}}{\{\dots, \text{one} : \text{int list}\}}}{\frac{\text{VAR}}{\vdash y : \text{int}} \quad \frac{\text{BIN-OP}}{\vdash \text{one} : \text{int list}}}}$$
$$\frac{\frac{\text{VAR}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\}} \vdash x : \text{int}}{\frac{\text{BIN-OP}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\}} \vdash (y :: \text{one}) : \text{int list}}}}$$
$$\frac{\frac{\text{FUN}}{\{y : \text{int}, x : \text{int}, \text{one} : \text{int list}\}} \vdash (x :: y :: \text{one}) : \text{int list}}{\text{FUN}}}}$$
$$\begin{aligned} & \{x : \text{int}, \text{one} : \text{int list}\} \vdash \\ & \quad \text{fun } y \text{ -> } (x :: y :: \text{one}) \\ & \quad : \text{int} \rightarrow \text{int list} \end{aligned}$$

**Let Expressions**



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 ...



# Big Picture: Curry-Howard

---

- **Type systems**  $\Leftrightarrow$  **logical systems**
- **Types**  $\Leftrightarrow$  **propositions**
- **Terms**  $\Leftrightarrow$  **proofs** (of the propositions that their types represent)
- **Type checking**  $\Leftrightarrow$  **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



# Big Picture: Curry-Howard

---

## Modus ponens

$$\frac{A \Rightarrow B \quad A}{B}$$

## Function Application

$$\frac{\Gamma \vdash f : A \rightarrow B \quad \Gamma \vdash a : A}{\Gamma \vdash f a : B}$$



# Big Picture: Curry-Howard

---

- **Type systems**  $\Leftrightarrow$  **logical systems**
- **Types**  $\Leftrightarrow$  **propositions**
- **Terms**  $\Leftrightarrow$  **proofs** (of the propositions that their types represent)
- **Type checking**  $\Leftrightarrow$  **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



# Big Picture: Curry-Howard

---

- Type systems  $\Leftrightarrow$  logical systems
- Types  $\Leftrightarrow$  propositions
- Terms  $\Leftrightarrow$  proofs (of the propositions that their types represent)
- Type checked
- 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

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



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