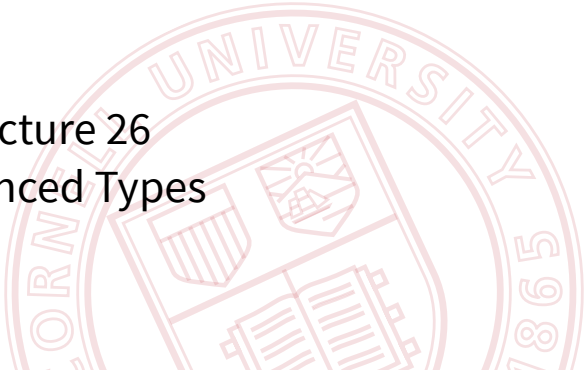
# CS 4110
# Programming Languages & Logics




Lecture 26
Advanced Types

# Plan

We'll finish proving the soundness of a simple type system for $\lambda$-calculus.

Then, we'll extend our *type system* with features commonly found in real-world languages: products, sums, and references.

# Simply-Typed Lambda Calculus

## Syntax

| | |
|---|---|
| expressions | $e ::= x \mid \lambda x{:}\tau.\, e \mid e_1\, e_2 \mid n \mid e_1 + e_2 \mid ()$ |
| values | $v ::= \lambda x{:}\tau.\, e \mid n \mid ()$ |
| types | $\tau ::= \textbf{int} \mid \textbf{unit} \mid \tau_1 \to \tau_2$ |

## Dynamic Semantics

$$E ::= [\cdot] \mid E\, e \mid v\, E \mid E + e \mid v + E$$

$$\frac{e \to e'}{E[e] \to E[e']} \qquad \frac{}{(\lambda x{:}\tau.\, e)\, v \to e\{v/x\}} \qquad \frac{n = n_1 + n_2}{n_1 + n_2 \to n}$$

# Simply-Typed Lambda Calculus

## Static Semantics

$$\frac{}{\Gamma \vdash n : \mathbf{int}}\ \textsc{T-Int} \qquad \frac{}{\Gamma \vdash () : \mathbf{unit}}\ \textsc{T-Unit}$$

$$\frac{\Gamma \vdash e_1 : \mathbf{int} \quad \Gamma \vdash e_2 : \mathbf{int}}{\Gamma \vdash e_1 + e_2 : \mathbf{int}}\ \textsc{T-Add}$$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}\ \textsc{T-Var} \qquad \frac{\Gamma, x : \tau \vdash e : \tau'}{\Gamma \vdash \lambda x : \tau.\, e : \tau \to \tau'}\ \textsc{T-Abs}$$

$$\frac{\Gamma \vdash e_1 : \tau \to \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\, e_2 : \tau'}\ \textsc{T-App}$$

# Properties

**Theorem (Type soundness)**

*If* $\vdash e : \tau$ *and* $e \to^* e'$ *and* $e' \not\to$ *then* $e'$ *is a value and* $\vdash e' : \tau$.

# Properties

**Theorem (Type soundness)**

*If* $\vdash e : \tau$ *and* $e \to^* e'$ *and* $e' \not\to$ *then* $e'$ *is a value and* $\vdash e' : \tau$.

**Lemma (Preservation)**

*If* $\vdash e : \tau$ *and* $e \to e'$ *then* $\vdash e' : \tau$.

# Properties

**Theorem (Type soundness)**

*If* $\vdash e : \tau$ *and* $e \to^* e'$ *and* $e' \not\to$ *then* $e'$ *is a value and* $\vdash e' : \tau$.

**Lemma (Preservation)**

*If* $\vdash e : \tau$ *and* $e \to e'$ *then* $\vdash e' : \tau$.

**Lemma (Progress)**

*If* $\vdash e : \tau$ *then either* $e$ *is a value or there exists an* $e'$ *such that* $e \to e'$.

# Extra Lemmas for Preservation

**Lemma (Substitution)**

*If* $x : \tau' \vdash e : \tau$ *and* $\vdash v : \tau'$ *then* $\vdash e\{v/x\} : \tau$.

**Lemma (Context)**

*If* $\vdash E[e] : \tau$ *and* $\vdash e : \tau'$ *and* $\vdash e' : \tau'$ *then* $\vdash E[e'] : \tau$.

# Extra Lemma for Progress

## Lemma (Canonical Forms)

*If* $\vdash v : \tau$, *then*

1. *If* $\tau$ *is* ***int****, then v is a constant, i.e., some c.*
2. *If* $\tau$ *is* $\tau_1 \to \tau_2$, *then v is an abstraction, i.e.,* $\lambda x : \tau_1 . e$ *for some x and e.*

# Products (Pairs)

## Syntax

$$e ::= \cdots \mid (e_1, e_2) \mid \#1\, e \mid \#2\, e$$
$$v ::= \cdots \mid (v_1, v_2)$$

# Products (Pairs)

## Syntax

$$e ::= \cdots \mid (e_1, e_2) \mid \#1\, e \mid \#2\, e$$
$$v ::= \cdots \mid (v_1, v_2)$$

## Semantics

$$E ::= \cdots \mid (E, e) \mid (v, E) \mid \#1\, E \mid \#2\, E$$

$$\frac{}{\#1\,(v_1, v_2) \to v_1} \qquad \frac{}{\#2\,(v_1, v_2) \to v_2}$$

# Product Types

$$\tau_1 \times \tau_2$$

# Product Types

$$\tau_1 \times \tau_2$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$$

# Product Types

$$\tau_1 \times \tau_2$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (e_1, e_2) : \tau_1 \times \tau_2}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \#1\, e : \tau_1}$$

$$\frac{\Gamma \vdash e : \tau_1 \times \tau_2}{\Gamma \vdash \#2\, e : \tau_2}$$

# Sums (Tagged Unions)

## Syntax

$$e ::= \cdots \mid \mathsf{inl}_{\tau_1+\tau_2}\, e \mid \mathsf{inr}_{\tau_1+\tau_2}\, e \mid (\mathsf{case}\, e_1\, \mathsf{of}\, e_2 \mid e_3)$$
$$v ::= \cdots \mid \mathsf{inl}_{\tau_1+\tau_2}\, v \mid \mathsf{inr}_{\tau_1+\tau_2}\, v$$

# Sums (Tagged Unions)

## Syntax

$$e ::= \cdots \mid \mathsf{inl}_{\tau_1+\tau_2}\, e \mid \mathsf{inr}_{\tau_1+\tau_2}\, e \mid (\mathsf{case}\, e_1\, \mathsf{of}\, e_2 \mid e_3)$$
$$v ::= \cdots \mid \mathsf{inl}_{\tau_1+\tau_2}\, v \mid \mathsf{inr}_{\tau_1+\tau_2}\, v$$

## Semantics

$$E ::= \cdots \mid \mathsf{inl}_{\tau_1+\tau_2}\, E \mid \mathsf{inr}_{\tau_1+\tau_2}\, E \mid (\mathsf{case}\, E\, \mathsf{of}\, e_2 \mid e_3)$$

$$\frac{}{\mathsf{case}\, \mathsf{inl}_{\tau_1+\tau_2}\, v\, \mathsf{of}\, e_2 \mid e_3 \to e_2\, v} \qquad \frac{}{\mathsf{case}\, \mathsf{inr}_{\tau_1+\tau_2}\, v\, \mathsf{of}\, e_2 \mid e_3 \to e_3\, v}$$

# Sum Types

$$\tau ::= \cdots \mid \tau_1 + \tau_2$$

# Sum Types

$$\tau ::= \cdots \mid \tau_1 + \tau_2$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \mathsf{inl}_{\tau_1+\tau_2}\, e : \tau_1 + \tau_2}$$

$$\frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \mathsf{inr}_{\tau_1+\tau_2}\, e : \tau_1 + \tau_2}$$

# Sum Types

$$\tau ::= \cdots \mid \tau_1 + \tau_2$$

$$\frac{\Gamma \vdash e : \tau_1}{\Gamma \vdash \mathsf{inl}_{\tau_1+\tau_2}\, e : \tau_1 + \tau_2}$$

$$\frac{\Gamma \vdash e : \tau_2}{\Gamma \vdash \mathsf{inr}_{\tau_1+\tau_2}\, e : \tau_1 + \tau_2}$$

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma \vdash e_1 : \tau_1 \to \tau \quad \Gamma \vdash e_2 : \tau_2 \to \tau}{\Gamma \vdash \mathsf{case}\ e\ \mathsf{of}\ e_1 \mid e_2 : \tau}$$

# Example

$$
\begin{array}{l}
\text{let } f = \lambda a : \mathbf{int} + (\mathbf{int} \to \mathbf{int}). \\
\qquad \text{case } a \text{ of } (\lambda y : \mathbf{int}.\, y + 1) \mid (\lambda g : \mathbf{int} \to \mathbf{int}.\, g\, 35) \text{ in} \\
\text{let } h = \lambda x : \mathbf{int}.\, x + 7 \text{ in} \\
f\, (\mathrm{inr}_{\mathbf{int}+(\mathbf{int}\to\mathbf{int})}\, h)
\end{array}
$$

# References

## Syntax

$$e ::= \cdots \mid \mathsf{ref}\, e \mid !e \mid e_1 := e_2 \mid \ell$$
$$v ::= \cdots \mid \ell$$

# References

## Syntax

$$e ::= \cdots \mid \mathsf{ref}\, e \mid !e \mid e_1 := e_2 \mid \ell$$
$$v ::= \cdots \mid \ell$$

## Semantics

$$E ::= \cdots \mid \mathsf{ref}\, E \mid !E \mid E := e \mid v := E$$

$$\frac{\ell \notin dom(\sigma)}{\langle \sigma, \mathsf{ref}\, v \rangle \to \langle \sigma[\ell \mapsto v], \ell \rangle} \qquad \frac{\sigma(\ell) = v}{\langle \sigma,\ !\ell \rangle \to \langle \sigma, v \rangle} \qquad \frac{}{\langle \sigma, \ell := v \rangle \to \langle \sigma[\ell \mapsto v], v \rangle}$$

# Reference Types

$$\tau ::= \cdots \mid \tau \ \textbf{ref}$$

# Reference Types

$$\tau ::= \cdots \mid \tau \ \textbf{ref}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ref}\, e : \tau \ \textbf{ref}}$$

# Reference Types

$$\tau ::= \cdots \mid \tau \ \textbf{ref}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ref}\, e : \tau \ \textbf{ref}}$$

$$\frac{\Gamma \vdash e : \tau \ \textbf{ref}}{\Gamma \vdash !e : \tau}$$

# Reference Types

$$\tau ::= \cdots \mid \tau \ \textbf{ref}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathsf{ref}\, e : \tau \ \textbf{ref}}$$

$$\frac{\Gamma \vdash e : \tau \ \textbf{ref}}{\Gamma \vdash !e : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau \ \textbf{ref} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 := e_2 : \tau}$$

# Question

Is this type system sound?

# Question

Is this type system sound?

Well... what is the type of a location $\ell$?

# Question

Is this type system sound?

Well... what is the type of a location $\ell$? (Oops!)

# Store Typings

Let $\Sigma$ range over partial functions from locations to types.

# Store Typings

Let $\Sigma$ range over partial functions from locations to types.

$$\frac{\Gamma, \Sigma \vdash e : \tau}{\Gamma, \Sigma \vdash \mathsf{ref}\, e : \tau\ \mathbf{ref}}$$

# Store Typings

Let $\Sigma$ range over partial functions from locations to types.

$$\frac{\Gamma, \Sigma \vdash e : \tau}{\Gamma, \Sigma \vdash \text{ref}\, e : \tau\ \textbf{ref}}$$

$$\frac{\Gamma, \Sigma \vdash e : \tau\ \textbf{ref}}{\Gamma, \Sigma \vdash\ !e : \tau}$$

# Store Typings

Let $\Sigma$ range over partial functions from locations to types.

$$\frac{\Gamma, \Sigma \vdash e : \tau}{\Gamma, \Sigma \vdash \mathsf{ref}\, e : \tau\ \mathbf{ref}}$$

$$\frac{\Gamma, \Sigma \vdash e : \tau\ \mathbf{ref}}{\Gamma, \Sigma \vdash\ !e : \tau}$$

$$\frac{\Gamma, \Sigma \vdash e_1 : \tau\ \mathbf{ref} \quad \Gamma, \Sigma \vdash e_2 : \tau}{\Gamma, \Sigma \vdash e_1 := e_2 : \tau}$$

# Store Typings

Let $\Sigma$ range over partial functions from locations to types.

$$\frac{\Gamma, \Sigma \vdash e : \tau}{\Gamma, \Sigma \vdash \mathsf{ref}\, e : \tau\ \mathbf{ref}}$$

$$\frac{\Gamma, \Sigma \vdash e : \tau\ \mathbf{ref}}{\Gamma, \Sigma \vdash\ !e : \tau}$$

$$\frac{\Gamma, \Sigma \vdash e_1 : \tau\ \mathbf{ref} \quad \Gamma, \Sigma \vdash e_2 : \tau}{\Gamma, \Sigma \vdash e_1 := e_2 : \tau}$$

$$\frac{\Sigma(\ell) = \tau}{\Gamma, \Sigma \vdash \ell : \tau\ \mathbf{ref}}$$

# Reference Types Metatheory

## Definition

Store $\sigma$ is *well-typed* with respect to typing context $\Gamma$ and store typing $\Sigma$, written $\Gamma, \Sigma \vdash \sigma$ , if $dom(\sigma) = dom(\Sigma)$ and for all $\ell \in dom(\sigma)$ we have $\Gamma, \Sigma \vdash \sigma(\ell) : \Sigma(\ell)$.

# Reference Types Metatheory

## Definition

Store $\sigma$ is *well-typed* with respect to typing context $\Gamma$ and store typing $\Sigma$, written $\Gamma, \Sigma \vdash \sigma$ , if $dom(\sigma) = dom(\Sigma)$ and for all $\ell \in dom(\sigma)$ we have $\Gamma, \Sigma \vdash \sigma(\ell) : \Sigma(\ell)$.

## Theorem (Type soundness)

*If* $\cdot, \Sigma \vdash e : \tau$ *and* $\cdot, \Sigma \vdash \sigma$ *and* $\langle e, \sigma \rangle \to^* \langle e', \sigma' \rangle$ *and* $\langle e', \sigma' \rangle \not\to$*, then* $e'$ *is a value.*

# Reference Types Metatheory

## Definition

Store $\sigma$ is *well-typed* with respect to typing context $\Gamma$ and store typing $\Sigma$, written $\Gamma, \Sigma \vdash \sigma$ , if $dom(\sigma) = dom(\Sigma)$ and for all $\ell \in dom(\sigma)$ we have $\Gamma, \Sigma \vdash \sigma(\ell) : \Sigma(\ell)$.

## Theorem (Type soundness)

*If* $\cdot, \Sigma \vdash e : \tau$ *and* $\cdot, \Sigma \vdash \sigma$ *and* $\langle e, \sigma \rangle \to^* \langle e', \sigma' \rangle$ *and* $\langle e', \sigma' \rangle \not\to$*, then* $e'$ *is a value.*

## Lemma (Preservation)

*If* $\Gamma, \Sigma \vdash e : \tau$ *and* $\Gamma, \Sigma \vdash \sigma$ *and* $\langle e, \sigma \rangle \to \langle e', \sigma' \rangle$ *then there exists some* $\Sigma' \supseteq \Sigma$ *such that* $\Gamma, \Sigma' \vdash e' : \tau$ *and* $\Gamma, \Sigma' \vdash \sigma'$.

# Landin's Knot

Using references, we (re)gain the ability define recursive functions!

$$\textbf{let } r = \text{ref } \lambda x : \textbf{int}.\ 0 \textbf{ in}$$

# Landin's Knot

Using references, we (re)gain the ability define recursive functions!

$$\textbf{let } r = \text{ref } \lambda x\!:\!\textbf{int}.\ 0 \textbf{ in}$$
$$\textbf{let } f = (\lambda x\!:\!\textbf{int}.\ \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } x \times (!r)\ (x - 1)) \textbf{ in}$$

# Landin's Knot

Using references, we (re)gain the ability define recursive functions!

$$\begin{array}{l}\textbf{let } r = \text{ref } \lambda x\!:\!\textbf{int}.\ 0 \textbf{ in} \\ \textbf{let } f = (\lambda x\!:\!\textbf{int}.\ \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } x \times (!r)\ (x-1)) \textbf{ in} \\ \textbf{let } a = (r := f) \textbf{ in}\end{array}$$

# Landin's Knot

Using references, we (re)gain the ability define recursive functions!

$$
\begin{array}{l}
\textbf{let } r = \text{ref } \lambda x\!:\!\textbf{int}.\ 0 \textbf{ in} \\
\textbf{let } f = (\lambda x\!:\!\textbf{int}.\ \textbf{if } x = 0 \textbf{ then } 1 \textbf{ else } x \times (!r)\ (x-1)) \textbf{ in} \\
\textbf{let } a = (r := f) \textbf{ in} \\
f\ 5
\end{array}
$$

# Fixed Points

## Syntax

$$e ::= \cdots \mid \mathsf{fix}\, e$$

# Fixed Points

## Syntax

$$e ::= \cdots \mid \mathsf{fix}\, e$$

## Semantics

$$E ::= \cdots \mid \mathsf{fix}\, E$$

$$\frac{}{\mathsf{fix}\ \lambda x{:}\tau.\, e \to e\{(\mathsf{fix}\ \lambda x{:}\tau.\, e)/x\}}$$

# Fixed Points

## Syntax

$$e ::= \cdots \mid \mathsf{fix}\, e$$

## Semantics

$$E ::= \cdots \mid \mathsf{fix}\, E$$

$$\frac{}{\mathsf{fix}\ \lambda x{:}\tau.\, e \to e\{(\mathsf{fix}\ \lambda x{:}\tau.\, e)/x\}}$$

The typing rule for fix is left as an exercise...

# Exceptions

## Syntax

$$e ::= \cdots \mid \textbf{error} \mid \textbf{try}\ e\ \textbf{with}\ e$$

# Exceptions

## Syntax

$$e ::= \cdots \mid \textbf{error} \mid \textbf{try}\ e\ \textbf{with}\ e$$

## Semantics

$$\frac{e_1 \to e_1'}{\textbf{try}\ e_1\ \textbf{with}\ e_2 \to \textbf{try}\ e_1'\ \textbf{with}\ e_2} \qquad \frac{}{E[\textbf{error}] \to \textbf{error}}$$

# Exceptions

## Syntax

$$e ::= \cdots \mid \textbf{error} \mid \textbf{try}\ e\ \textbf{with}\ e$$

## Semantics

$$\frac{e_1 \to e_1'}{\textbf{try}\ e_1\ \textbf{with}\ e_2 \to \textbf{try}\ e_1'\ \textbf{with}\ e_2} \qquad \frac{}{E[\textbf{error}] \to \textbf{error}} \qquad \frac{}{\textbf{try error with}\ e \to e}$$

# Exceptions

## Syntax

$$e ::= \cdots \mid \textbf{error} \mid \textbf{try}\ e\ \textbf{with}\ e$$

## Semantics

$$\frac{e_1 \to e_1'}{\textbf{try}\ e_1\ \textbf{with}\ e_2 \to \textbf{try}\ e_1'\ \textbf{with}\ e_2} \qquad \frac{}{E[\textbf{error}] \to \textbf{error}} \qquad \frac{}{\textbf{try error with}\ e \to e}$$

$$\frac{}{\textbf{try}\ v\ \textbf{with}\ e \to v}$$

# Exception Types

We don’t need to add any new types...

$$\frac{}{\Gamma \vdash \textbf{error} : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \textbf{try}\ e_1\ \textbf{with}\ e_2 : \tau}$$

# Exception Metatheory

## Lemma (Progress)

*If* $\vdash e : \tau$ *then either*

- *e is a value or*
- *e is* ***error*** *or*
- *there exists* $e'$ *such that* $e \to e'$.

# Exception Metatheory

## Lemma (Progress)

*If* $\vdash e : \tau$ *then either*

- *e is a value or*
- *e is* ***error*** *or*
- *there exists* $e'$ *such that* $e \to e'$.

## Theorem (Soundness)

*If* $\vdash e : \tau$ *and* $e \to^* e'$ *and* $e' \not\to$ *then either*

- *e is a value or*
- *e is* ***error***