Goals Since Day 1

Develop tools to rigorously study what programs mean.

semantics

equivalence, termination, determinism, ...

Develop tools for studying program behavior

inductive defns, structural induction, inference rules

Investigate core PL concepts

types, functions, scope, mutation, iteration

Cruising to Victory

Covered Serious Ground

• Functional Programming
• Formal Definitions, Structural Induction, Semantics
• Various Lambda Calculi
• Types, Progress, Preservation
• Evaluation Contexts and Continuation Passing Style
• Subtyping, Parametric Polymorphism
Developed Sweet Skills

- Writing Formal Proofs
- Language Implementation
- Extending Languages
- Taste for Design Tradeoffs
- Appreciating Deep Connections (e.g. Curry-Howard)
- *Enduring Long Exams*

Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
- Subtyping: LSP, Covariance, Contravariance
- Type Derivations with Parametric Polymorphism
- Course Evaluations

Developed Sweet Skills

- *Keeping a Straight Face*

Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
- Subtyping: LSP, Covariance, Contravariance
- Type Derivations with Parametric Polymorphism
- Course Evaluations
Extensions and Type Safety

Need to establish two properties:

1. **Progress**
   
   If \( \star \vdash e : T \), then either (A) \( e \) is a value or (B) there exists \( e' \) such that \( e \rightarrow e' \).

2. **Preservation**
   
   If \( \star \vdash e : T \) and \( e \rightarrow e' \), then \( \star \vdash e' : T \).

---

**Product Progress**

Case \( \star \vdash (e_1, e_2) : T_1 \times T_2 \):

- inversion provides \( \star \vdash e_1 : T_1 \) and \( \star \vdash e_2 : T_2 \)
- if \( e_1 \) not a value
  - by **IH** and typing \( e_1 \) can step to \( e_1' \)
  - then \( (e_1, e_2) \) can step to \( (e_1', e_2) \)
- else \( e_1 \) a value, if \( e_2 \) not a value
  - by **IH** and typing \( e_2 \) can step to \( e_2' \)
  - then \( (e_1, e_2) \) can step to \( (e_1, e_2') \)
- else \( e_2 \) a value
  - both values, whole thing value, not stuck, done

---

**Preservation**

Proof generally has this shape:

induction on \( \star \vdash e : T \)

base cases either:

(A) not typable in empty context (bogus)
(B) cannot step (bogus)

inductive cases:

- inversion on typing provides types for subexprs
- IH + subexpr type implies they are values or can step
- if subexpression steps, big expression steps
- **NOTE**: canonical forms provides shape of typed values
Product Preservation

Case

\[ (e_1, e_2) : T_1 \times T_2 \text{ and } (e_1, e_2) \rightarrow e' \]

- inversion provides \( e_1 : T_1 \) and \( e_2 : T_2 \)
- case analysis on step

\[- e_1 \rightarrow e_1' \text{ and } e' = (e_1', e_2) \]
  - by IH and typing \( e_1' : T_1 \)
  - then \( (e_1', e_2) \) still has type \( T_1 \times T_2 \)

\[- e_2 \rightarrow e_2' \text{ and } e' = (e_1, e_2') \]
  - by IH and typing \( e_2' : T_2 \)
  - then \( (e_1, e_2') \) still has type \( T_1 \times T_2 \)

Today: Review & Review

• Extending Progress and Preservation Proofs
• Quick Look Back at Evaluation Contexts
• Putting Terms into Continuation Passing Style
• Subtyping: LSP, Covariance, Contravariance
• Type Derivations with Parametric Polymorphism
• Course Evaluations

Evaluation Contexts

Evaluation contexts define where interesting work can happen:

\[
E ::= \text{[]} \mid E \cdot e \mid v \cdot E \mid (E, e) \mid (v, E) \mid E.1 \mid E.2 \\
\mid A(E) \mid B(E) \mid (\text{match } E \text{ with } Ax. \ e_1 \mid By. \ e_2)
\]

\[ e \rightarrow e' \text{ with 1 rule: } \frac{e \xrightarrow{B} e'}{E[e] \rightarrow E[e']} \]

\[ e \xrightarrow{B} e' \text{ does all the “interesting work”:} \]

\[
\begin{align*}
(\lambda x. \ e) \ x & \xrightarrow{B} e[v/x] \\
(v_1, v_2).1 & \xrightarrow{B} v_1 \\
(v_1, v_2).2 & \xrightarrow{B} v_2 \\
\text{match } A(v) \text{ with } Ax. \ e_1 \mid By. \ e_2 & \xrightarrow{B} e_1[v/x] \\
\text{match } B(v) \text{ with } Ay. \ e_1 \mid Bx. \ e_2 & \xrightarrow{B} e_2[v/x]
\end{align*}
\]
Today: Review & Review

- Extending Progress and Preservation Proofs
- **Quick Look Back at Evaluation Contexts**
- Putting Terms into Continuation Passing Style
- Subtyping: LSP, Covariance, Contravariance
- Type Derivations with Parametric Polymorphism
- Course Evaluations

CPS

```
let rec fact n =
  if n = 0 then 1
  else n * fact (n - 1)
```

```
let rec fact’ n k =
  (eq’ n 0 (fun b ->
    (if b then
      (k 1)
    else
      (sub’ n 1 (fun m ->
        (fact’ m (fun p ->
          (mult’ n p k))))))))
```

Everything takes a continuation, all the time!
Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
  - Subtyping: LSP, Covariance, Contravariance
  - Type Derivations with Parametric Polymorphism
- Course Evaluations

Subtyping: Follow LSP

Liskov Substitution Principle:

If \( A \) is a subtype of \( B \) (written \( A <: B \)),
then we can safely use a value of type \( A \)
anywhere a value of type \( B \) is expected.

Subtyping Smaller Parts

- Covariance: same direction as bigger type
- Contravariance: opposite direction of bigger type

???

\[ \tau_1 \rightarrow \tau_2 \leq \tau_3 \rightarrow \tau_4 \]

Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
  - Subtyping: LSP, Covariance, Contravariance
- Type Derivations with Parametric Polymorphism
- Course Evaluations
Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
- Subtyping: LSP, Covariance, Contravariance
  - Type Derivations with Parametric Polymorphism
- Course Evaluations

Typing Bambdas

- Look at AST, look at typing rules, pattern match
  - *Try to think as little as possible*

\[
\begin{align*}
\Delta; \Gamma \vdash x : \Gamma(x) & \quad \Delta; \Gamma \vdash e : \text{int} \\
\Delta; \Gamma, x : \tau_1 \vdash e : \tau_2 & \quad \Delta \vdash \tau_1 \\
\Delta; \Gamma \vdash \lambda x : \tau_1. \ e : \tau_1 \to \tau_2 & \quad \Delta; \Gamma \vdash e_1 : \tau_2 \to \tau_1 \quad \Delta; \Gamma \vdash e_2 : \tau_2 \\
\Delta; \Gamma \vdash e_1 \ e_2 : \tau_1 & \\
\Delta, \alpha; \Gamma \vdash e : \tau_1 & \quad \Delta; \Gamma \vdash \forall \alpha. \tau_1 \\
\Delta; \Gamma \vdash \Lambda \alpha. \ e : \forall \alpha. \tau_1 & \quad \Delta; \Gamma \vdash e[\tau_2] : \tau_1[\tau_2/\alpha] \\
(\Lambda \alpha. \lambda x : \alpha. \ \lambda f : \alpha \to \beta. \ f \ x) [\text{int}] [\text{int}] 3 (\lambda y : \text{int}. \ y + y)
\end{align*}
\]

Today: Review & Review

- Extending Progress and Preservation Proofs
- Quick Look Back at Evaluation Contexts
- Putting Terms into Continuation Passing Style
- Subtyping: LSP, Covariance, Contravariance
  - Type Derivations with Parametric Polymorphism
- Course Evaluations
Thanks!

• Really enjoyed our discussions during lecture
• Learned a lot about teaching vs. giving a lecture
• Y’all are incredibly bright, very promising futures
• Remember tricks:
  • Have one question for each topic.
  • “That’s a great question. What do you think?”

Course Feedback

• Voluntary
• Confidential
• Grade Independent
• No. 2 pencil ONLY on scan forms