

# Lecture 10: The Simply Typed $\lambda$ -Calculus

## Typechecker implementation and some extra types

Polyvios Pratikakis

Computer Science Department, University of Crete

Type Systems and Programming Languages



## Last time

- Function types:  $T \rightarrow T'$ ,  $\rightarrow$  is a *type constructor*
- Add booleans as a *base type*: inductive type language  $T$
- Add type declarations for function arguments  $\lambda x : T.e$
- Typing relation  $\Gamma \vdash e : T$
- ...uses *context*  $\Gamma$  to type free variables ( $\alpha$ -renamed to be unique)
- Also called typing *judgement*
- Defined recursively (with type rules)



# Typed $\lambda$ -calculus definition

Term language  $e ::= e e \mid \lambda x : T. e \mid x$   
                   $\mid \text{true} \mid \text{false} \mid \text{if } e \text{ then } e \text{ else } e$   
Values  $v ::= \lambda x : T. e \mid \text{true} \mid \text{false}$   
Type language  $T ::= T \rightarrow T \mid \text{Bool}$

$$[\text{T-ABS}] \frac{\Gamma, x : T_1 \vdash e_2 : T_2}{\Gamma \vdash \lambda x : T_1. e_2 : T_1 \rightarrow T_2}$$

$$[\text{T-VAR}] \frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$[\text{T-APP}] \frac{\Gamma \vdash e_1 : T \rightarrow T' \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 e_2 : T'}$$

$$[\text{T-IF}] \frac{\Gamma \vdash e_1 : \text{Bool} \quad \Gamma \vdash e_2 : T \quad \Gamma \vdash e_3 : T}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T}$$

$$[\text{T-TRUE}] \frac{}{\Gamma \vdash \text{true} : \text{Bool}}$$

$$[\text{T-FALSE}] \frac{}{\Gamma \vdash \text{false} : \text{Bool}}$$



# Metatheory

- Progress theorem

- ▶ If  $\vdash e : T$  then either  $e$  is a value, or it can take a step  $e \rightarrow e'$  to some  $e'$
- ▶ Proof by induction on  $\Gamma \vdash e : T$

- Preservation theorem

- ▶ If  $\vdash e : T$  and  $e \rightarrow e'$  then  $e' : T$
- ▶ Proof by induction on  $e \rightarrow e'$ , using lemmas
- ▶ Permutation lemma
  - ★ If  $\Gamma \vdash e : T$  and  $\Gamma'$  is a permutation of  $\Gamma$ , then  $\Gamma' \vdash e : T$
  - ★ Proof by induction on  $\Gamma \vdash e : T$
- ▶ Weakening lemma (by induction on  $\Gamma \vdash e : T$ )
  - ★ If  $\Gamma \vdash e : T$  and  $x \notin \text{dom}(\Gamma)$ , then  $\Gamma, x : T' \vdash e : T$
- ▶ Substitution lemma (by induction on  $\Gamma, x : T' \vdash e : T$ )
  - ★ If  $\Gamma, x : T' \vdash e : T$  and  $\Gamma \vdash e' : T'$ , then  $\Gamma \vdash e[e'/x] : T$



# Typecheck functions

- Lambda term:

$$[\text{T-ABS}] \frac{\Gamma, x : T_1 \vdash e_2 : T_2}{\Gamma \vdash \lambda x : T_1. e_2 : T_1 \rightarrow T_2}$$

- If the program is a lambda term of the form  $\lambda x : T_1. e_2$
- Add  $x : T_1$  to the input environment  $\Gamma$  and using that, find the type of  $e_2$ 
  - ▶ Before adding  $x : T_1$  to  $\Gamma$ , make sure it is not already bound
- If the type of  $e_2 : T_2$ , then return the type  $T_1 \rightarrow T_2$



# Typecheck variables

- Variable:

$$[\text{T-VAR}] \frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

- If the program is a variable  $x$
- Check in the input environment  $\Gamma$  for any binding of  $x$ :  $(x : T) \in \Gamma$
- If there is one, return type  $T$
- If not, the variable  $x$  is undeclared in the program: type error



# Typecheck application

- If the program is an application  $e_1 e_2$

$$[\text{T-APP}] \frac{\Gamma \vdash e_1 : T \rightarrow T' \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 e_2 : T'}$$

- Find the type  $T_1$  of  $e_1$  using the input environment  $\Gamma$
- Find the type  $T_2$  of  $e_2$  using the same input environment  $\Gamma$
- Check whether  $T_1$  is a function type
- If it is, check that the argument type matches  $T_2$
- If it does, return the result type of the function  $T_1$
  
- ...and so on, for the other cases
- If you cannot apply a rule to compute a type: type error



## A small revision

- Prove substitution lemma (one case each)
- If  $\Gamma, x : T' \vdash e : T$  and  $\Gamma \vdash e' : T'$ , then  $\Gamma \vdash e[e'/x] : T$
- Proof by induction on  $\Gamma, x : T' \vdash e : T$





# Adding more types

- The unit type
- Pairs (the product type)
- Binary unions (the sum type)



# The unit type

- Syntax

$$\begin{aligned} e &::= \dots \mid () \mid e_1; e_2 \\ v &::= \dots \mid () \\ T &::= \dots \mid Unit \end{aligned}$$

- Typing

$$\begin{array}{c} [T-UNIT] \\ \hline \Gamma \vdash () : Unit \end{array} \qquad \begin{array}{c} \Gamma \vdash e_1 : Unit \\ \Gamma \vdash e_2 : T \\ [T-SEQ] \\ \hline \Gamma \vdash e_1; e_2 : T \end{array}$$

- Semantics

$$\frac{}{() ; e_2 \rightarrow e_2} \qquad \frac{e_1 \rightarrow e'_1}{e_1 ; e_2 \rightarrow e'_1 ; e_2}$$


# Pairs: the product type

- Syntax

$$\begin{aligned}e &::= \dots \mid e.1 \mid e.2 \mid (e, e) \\v &::= \dots \mid (v, v) \\T &::= \dots \mid T \times T\end{aligned}$$

- Typing

$$[\text{T-PAIR}] \frac{\Gamma \vdash e_1 : T_1 \quad \Gamma \vdash e_2 : T_2}{\Gamma \vdash (e_1, e_2) : T_1 \times T_2}$$

$$[\text{T-FST}] \frac{\Gamma \vdash e : T \times T'}{\Gamma \vdash e.1 : T} \quad [\text{T-SND}] \frac{\Gamma \vdash e : T \times T'}{\Gamma \vdash e.2 : T'}$$



# Pairs: the product type (cont'd)

- Semantics

$$\frac{}{(v_1, v_2).1 \rightarrow v_1} \qquad \frac{}{(v_1, v_2).2 \rightarrow v_2}$$
$$\frac{e_1 \rightarrow e'_1}{(e_1, e_2) \rightarrow (e'_1, e_2)} \qquad \frac{e_2 \rightarrow e'_2}{(v_1, e_2) \rightarrow (v_1, e'_2)}$$
$$\frac{e \rightarrow e'}{e.1 \rightarrow e'.1} \qquad \frac{e \rightarrow e'}{e.2 \rightarrow e'.2}$$

- Adapt progress and preservation ...



# Binary unions: the sum type

- Syntax

$$\begin{aligned} e &::= \dots \mid \text{inl } e \mid \text{inr } e \mid \text{case } e \text{ of inl } x \Rightarrow e \mid \text{inr } x \Rightarrow e \\ v &::= \dots \mid \text{inl } v \mid \text{inr } v \\ T &::= \dots \mid T + T \end{aligned}$$

- Typing

$$\begin{array}{c} \Gamma \vdash e : T_1 \\ \text{[T-INL]} \frac{}{\Gamma \vdash \text{inl } e : T_1 + T_2} \end{array} \qquad \begin{array}{c} \Gamma \vdash e : T_2 \\ \text{[T-INR]} \frac{}{\Gamma \vdash \text{inr } e : T_1 + T_2} \end{array}$$

$$\text{[T-CASE]} \frac{\begin{array}{c} \Gamma \vdash e : T_1 + T_2 \\ \Gamma, x_1 : T_1 \vdash e_1 : T \\ \Gamma, x_2 : T_2 \vdash e_2 : T \end{array}}{\Gamma \vdash \text{case } e \text{ of inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2 : T}$$



# Binary unions: the sum type (cont'd)

- Semantics

$$\frac{e \rightarrow e'}{\text{inl } e \rightarrow \text{inl } e'} \qquad \frac{e \rightarrow e'}{\text{inr } e \rightarrow \text{inr } e'}$$

$$\frac{e \rightarrow e'}{\text{case } e \text{ of inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2 \rightarrow \text{case } e' \text{ of inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2}$$

$$\frac{}{\text{case (inl } v) \text{ of inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2 \rightarrow e_1[v/x_1]}$$

$$\frac{}{\text{case (inr } v) \text{ of inl } x_1 \Rightarrow e_1 \mid \text{inr } x_2 \Rightarrow e_2 \rightarrow e_2[v/x_2]}$$

