# Midterm 1 is next Wednesday

- ♦ Today's lecture will not be covered by the midterm.
- ♦ Next Monday, review class.
- ♦ Old exams and review questions on webpage.
- $\blacklozenge\,$  No recitation sections next week.
- ♦ New office hours next week, watch news group for details.

CIS 500, 6 October

# Where we've been:

- ♦ Inductive definitions
  - ${\ }{\ }$  abstract syntax
  - inference rules
- $\blacklozenge$  Proofs by structural induction
- ♦ Operational semantics
- ♦ The lambda-calculus
- ♦ Typing rules and type soundness

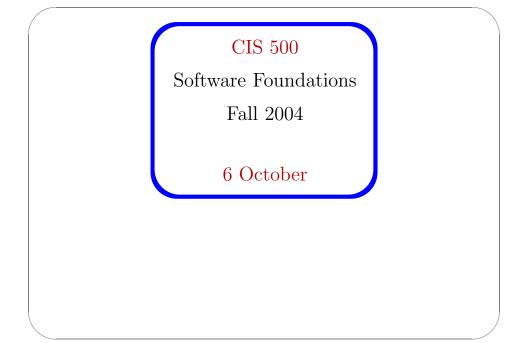
#### Where we're going:

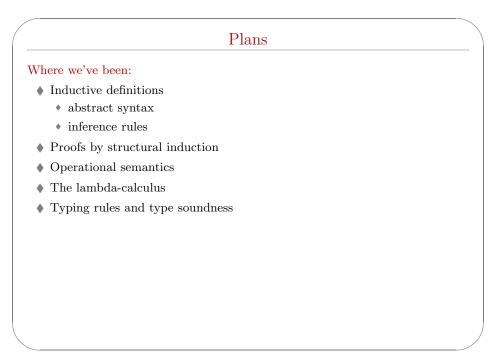
- ♦ "Simple types" for the lambda-calculus
- Formalizing more features of real-world languages (records, datatypes, references, exceptions, etc.)

Plans

- ♦ Subtyping
- Objects

CIS 500, 6 October





1

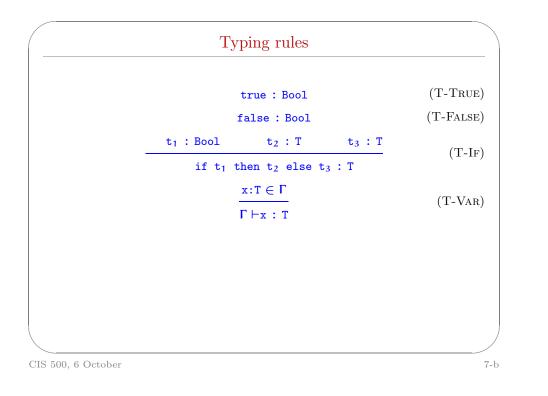
2

3-a

Lambda-calculu	s with booleans	Typing rules	
; ::=	terms		
x	variable	true : Bool	(T-TRUE)
$\lambda$ x.t	abstraction	false : Bool	(T-False
t t	application	$t_1$ : Bool $t_2$ : T $t_3$ : T	
true	constant true		(T-IF)
false	constant false	if $t_1$ then $t_2$ else $t_3$ : T	
if t then t else t	conditional		
<i>t</i> ::=	values		
$\lambda x.t$	abstraction value		
true	true value		
false	false value		
		<	
500, 6 October	5 Cl	IS 500, 6 October	

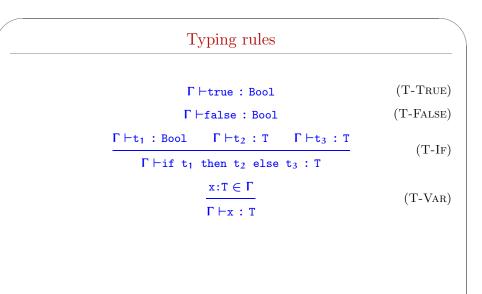
The Simply Typed Lambda-Calculus

	"Simpl	e Types"
<b>T</b> ::=		types
H	Bool	type of booleans
1	$\Gamma \rightarrow T$	types of functions



Typing rules	
Γ⊢true : Bool Γ⊢false : Bool	(T-True) (T-False)
$\frac{\Gamma \vdash t_1 : Bool}{\Gamma \vdash t_2 : T} \qquad \Gamma \vdash t_3 : T$	(T-IF)
$\Gamma \vdash  ext{if }  extsf{t}_1  extsf{ then }  extsf{t}_2  extsf{ else }  extsf{t}_3  extsf{ : }  extsf{T}$ $ extsf{x} :  extsf{T} \in \Gamma$	(1 11)
$\Gamma \vdash \mathbf{x} : \mathbf{T}$	(T-VAR)
$\frac{\Gamma, \mathbf{x}: \mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x}: \mathtt{T}_1 \cdot \mathtt{t}_2 : \mathtt{T}_1 \rightarrow \mathtt{T}_2}$	(T-Abs)
	_
October	7-0

Typing rulestrue : Bool(T-TRUE)false : Bool(T-FALSE) $t_1 : Bool$  $t_2 : T$  $t_3 : T$ if  $t_1$  then  $t_2$  else  $t_3 : T$ (T-IF)if  $t_1$  then  $t_2$  else  $t_3 : T$ (T-VAR)x : T(T-VAR)



7-a

CIS 500, 6

# Typing Derivations

What derivations justify the following typing statements?

- $\models \vdash (\lambda x: Bool.x)$ true : Bool
- f:Bool→Bool ⊢ f (if false then true else false) : Bool
- ♦ f:Bool $\rightarrow$ Bool $\vdash \lambda x$ :Bool. f (if x then false else x) : Bool $\rightarrow$ Bool

# Properties of $\lambda_{\rightarrow}$

As before, the fundamental property of the type system we have just defined is soundness with respect to the operational semantics.

1. Progress: A closed, well-typed term is not stuck

If  $\vdash t : T$ , then either t is a value or else  $t \longrightarrow t'$  for some t'.

2. Preservation: Types are preserved by one-step evaluation

If  $\Gamma \vdash t$ : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$ : T.

CIS 500, 6 October

Typing rules (T-TRUE)  $\Gamma \vdash true : Bool$ (T-FALSE)  $\Gamma \vdash false : Bool$  $\Gamma \vdash t_1 : Bool \quad \Gamma \vdash t_2 : T \quad \Gamma \vdash t_3 : T$ (T-IF) $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T$  $x:T \in \Gamma$ (T-VAR) Г⊢х : Т  $\Gamma$ , x:T<sub>1</sub>  $\vdash$ t<sub>2</sub> : T<sub>2</sub> (T-ABS)  $\Gamma \vdash \lambda x: T_1 . t_2 : T_1 \rightarrow T_2$  $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \qquad \Gamma \vdash t_2 : T_{11}$ (T-APP)  $\Gamma \vdash t_1 t_2 : T_{12}$ 

CIS 500, 6 October

Properties of  $\lambda_{\rightarrow}$ 

As before, the fundamental property of the type system we have just defined is soundness with respect to the operational semantics.

8

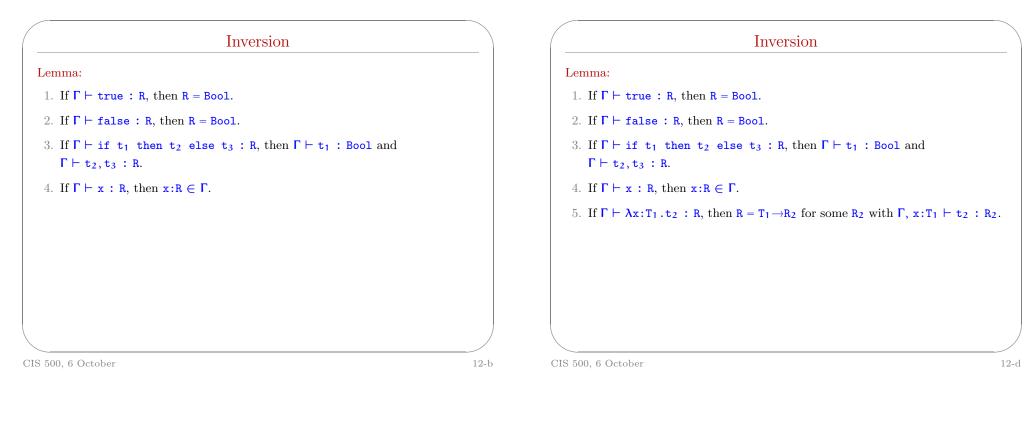
9-a

Proving progress	Inversion
Same steps as before • inversion lemma for typing relation • canonical forms lemma	Lemma: 1. If $\Gamma \vdash \text{true}$ : R, then R = Bool. 2. If $\Gamma \vdash \text{false}$ : R, then R = Bool.
♦ progress theorem	3. If $\Gamma \vdash \text{if } t_1$ then $t_2$ else $t_3 : R$ , then $\Gamma \vdash t_1 :$ Bool and $\Gamma \vdash t_2, t_3 : R$ .
CIS 500, 6 October 10-a	a CIS 500, 6 October 12

# Proving progress

Same steps as before...

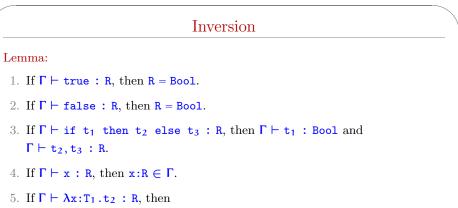
Typing rules again (for reference) (T-TRUE)  $\Gamma \vdash \texttt{true} : \texttt{Bool}$  $\Gamma \vdash false : Bool$ (T-FALSE)  $\Gamma \vdash t_1 : Bool$   $\Gamma \vdash t_2 : T$  $\Gamma \vdash t_3 : T$ (T-IF) $\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T$  $\mathtt{x}\!:\!\mathtt{T}\in\Gamma$ (T-VAR)  $\Gamma \vdash_{X} : T$  $\Gamma$ , x:T<sub>1</sub>  $\vdash$ t<sub>2</sub> : T<sub>2</sub> (T-Abs)  $\Gamma \vdash \lambda x: T_1.t_2 : T_1 \rightarrow T_2$  $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \qquad \Gamma \vdash t_2 : T_{11}$ (T-APP)  $\Gamma \vdash t_1 t_2 : T_{12}$ 



# Inversion

Lemma:

- 1. If  $\Gamma \vdash \text{true} : R$ , then R = Bool.
- 2. If  $\Gamma \vdash false : R$ , then R = Bool.
- 3. If  $\Gamma \vdash \text{if } t_1$  then  $t_2$  else  $t_3 : R$ , then  $\Gamma \vdash t_1 :$  Bool and  $\Gamma \vdash t_2, t_3 : R$ .
- 4. If  $\Gamma \vdash \mathbf{x} : \mathbf{R}$ , then



# Inversion Canonical Forms Lemma: Lemma: 1. If $\Gamma \vdash \text{true} : \mathbb{R}$ , then $\mathbb{R} = \text{Bool}$ . 1. If $\mathbf{v}$ is a value of type **Bool**, then 2. If $\Gamma \vdash$ false : R, then R = Bool. 3. If $\Gamma \vdash \text{if } t_1$ then $t_2$ else $t_3 : R$ , then $\Gamma \vdash t_1 : Bool and$ $\Gamma \vdash t_2, t_3 : \mathbb{R}.$ 4. If $\Gamma \vdash \mathbf{x} : \mathbf{R}$ , then $\mathbf{x} : \mathbf{R} \in \Gamma$ . 5. If $\Gamma \vdash \lambda x: T_1 \cdot t_2 : R$ , then $R = T_1 \rightarrow R_2$ for some $R_2$ with $\Gamma$ , $x: T_1 \vdash t_2 : R_2$ . 6. If $\Gamma \vdash t_1 \quad t_2 : R$ , then there is some type $T_{11}$ such that $\Gamma \vdash t_1 : T_{11} \rightarrow R$ and $\Gamma \vdash t_2 : T_{11}$ . CIS 500, 6 October 12-f CIS 500, 6 October 13-a

Inversion
Lemma:
1. If $\Gamma \vdash \text{true}$ : R, then R = Bool.
2. If $\Gamma \vdash false : R$ , then $R = Bool$ .
3. If $\Gamma \vdash \text{if } t_1$ then $t_2$ else $t_3 : \mathbb{R}$ , then $\Gamma \vdash t_1 : \text{Bool and}$ $\Gamma \vdash t_2, t_3 : \mathbb{R}$ .
4. If $\Gamma \vdash \mathbf{x} : \mathbf{R}$ , then $\mathbf{x}: \mathbf{R} \in \Gamma$ .
5. If $\Gamma \vdash \lambda x: T_1 \cdot t_2 : R$ , then $R = T_1 \rightarrow R_2$ for some $R_2$ with $\Gamma$ , $x: T_1 \vdash t_2 : R_2$ .
6. If $\Gamma \vdash t_1  t_2 : \mathbf{R}$ , then

	Canonical Forms	
Lemma:		

# Canonical Forms Progress Lemma: Theorem: Suppose t is a closed, well-typed term (that is, $\vdash t : T$ for some T). Then either t is a value or else there is some t' with $t \longrightarrow t'$ . 1. If v is a value of type Bool, then v is either true or false. 2. If **v** is a value of type $T_1 \rightarrow T_2$ , then **Proof:** By induction CIS 500, 6 October 13-c CIS 500, 6 October 14

# Canonical Forms

Lemma:

1. If v is a value of type Bool, then v is either true or false.

Canonical Forms

#### Lemma:

- 1. If v is a value of type Bool, then v is either true or false.
- 2. If v is a value of type  $T_1 \rightarrow T_2$ , then v has the form  $\lambda x: T_1.t_2$ .

#### Progress

Theorem: Suppose t is a closed, well-typed term (that is,  $\vdash t$ : T for some T). Then either t is a value or else there is some t' with  $t \longrightarrow t'$ .

**Proof:** By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because t is closed). The abstraction case is immediate, since abstractions are values.

Progress

Theorem: Suppose t is a closed, well-typed term (that is,  $\vdash$  t : T for some T).

Then either t is a value or else there is some t' with  $t \longrightarrow t'$ .

**Proof:** By induction on typing derivations.

#### Progress

Theorem: Suppose t is a closed, well-typed term (that is,  $\vdash t$ : T for some T). Then either t is a value or else there is some t' with  $t \longrightarrow t'$ .

**Proof:** By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because t is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where  $t = t_1 t_2$  with  $\vdash t_1 : T_{11} \rightarrow T_{12}$  and  $\vdash t_2 : T_{11}$ . By the induction hypothesis, either  $t_1$  is a value or else it can make a step of evaluation, and likewise  $t_2$ .

CIS 500, 6 October

14-b

# Progress

Theorem: Suppose t is a closed, well-typed term (that is,  $\vdash t : T$  for some T). Then either t is a value or else there is some t' with  $t \longrightarrow t'$ .

**Proof:** By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because t is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where  $t = t_1 \ t_2$  with  $\vdash t_1 : T_{11} \rightarrow T_{12}$  and  $\vdash t_2 : T_{11}$ .

CIS 500, 6 October

CIS 500, 6 October

14-d

#### **Proving Preservation**

Theorem: If  $\Gamma \vdash t$ : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$ : T.

**Proof:** By induction

CIS 500, 6 October

15

# **Proving Preservation**

Theorem: If  $\Gamma \vdash t$  : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$  : T. Proof: By induction on typing derivations. [Which case is the hard one?] Case T-APP: Given  $t = t_1 t_2$   $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$   $\Gamma \vdash t_2 : T_{11}$   $T = T_{12}$ Show  $\Gamma \vdash t' : T_{12}$ 

CIS 500, 6 October

# Progress

Theorem: Suppose t is a closed, well-typed term (that is,  $\vdash t : T$  for some T). Then either t is a value or else there is some t' with  $t \longrightarrow t'$ .

**Proof:** By induction on typing derivations. The cases for boolean constants and conditions are the same as before. The variable case is trivial (because t is closed). The abstraction case is immediate, since abstractions are values.

Consider the case for application, where  $t = t_1 \ t_2$  with  $\vdash t_1 : T_{11} \rightarrow T_{12}$  and  $\vdash t_2 : T_{11}$ . By the induction hypothesis, either  $t_1$  is a value or else it can make a step of evaluation, and likewise  $t_2$ . If  $t_1$  can take a step, then rule E-APP1 applies to t. If  $t_1$  is a value and  $t_2$  can take a step, then rule E-APP2 applies. Finally, if both  $t_1$  and  $t_2$  are values, then the canonical forms lemma tells us that  $t_1$  has the form  $\lambda x: T_{11} \cdot t_{12}$ , and so rule E-APPABS applies to t.

# **Proving Preservation**

Theorem: If  $\Gamma \vdash t$ : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$ : T.

Proof: By induction on typing derivations. [Which case is the hard one?] 15-b

# **Proving Preservation**

Theorem: If  $\Gamma \vdash t$ : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$ : T. Proof: By induction on typing derivations. [Which case is the hard one?] Case T-APP: Given  $t = t_1 t_2$   $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$   $\Gamma \vdash t_2 : T_{11}$   $T = T_{12}$ Show  $\Gamma \vdash t' : T_{12}$ By the inversion lemma for evaluation, there are three subcases... Subcase:  $t_1 = \lambda x: T_{11}$ .  $t_{12}$   $t_2$  a value  $v_2$   $t' = [x \mapsto v_2]t_{12}$ CIS 500, 6 October

15-d

# Proving Preservation

Theorem: If  $\Gamma \vdash t$ : T and t  $\longrightarrow$  t', then  $\Gamma \vdash t'$ : T.

Proof: By induction on typing derivations. [Which case is the hard one?]

Case T-APP: Given  $t = t_1 t_2$ 

```
\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}\Gamma \vdash t_2 : T_{11}
```

# $T = T_{12}$

Show 
$$\Gamma \vdash t' : T_{12}$$

By the inversion lemma for evaluation, there are three subcases...

## The "Substitution Lemma"

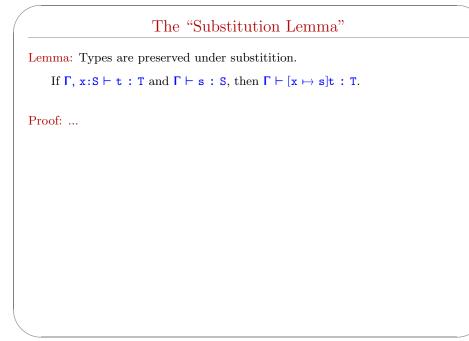
Lemma: Types are preserved under substitution. If  $\Gamma$ ,  $x:S \vdash t$  : T and  $\Gamma \vdash s$  : S, then  $\Gamma \vdash [x \mapsto s]t$  : T.

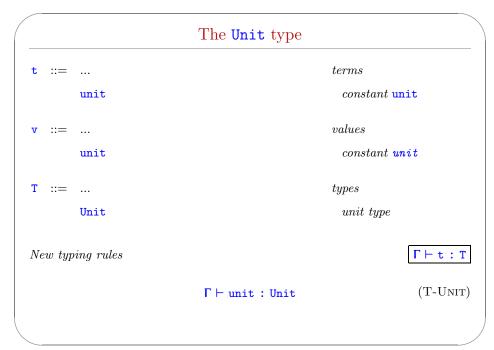
CIS 500, 6 October

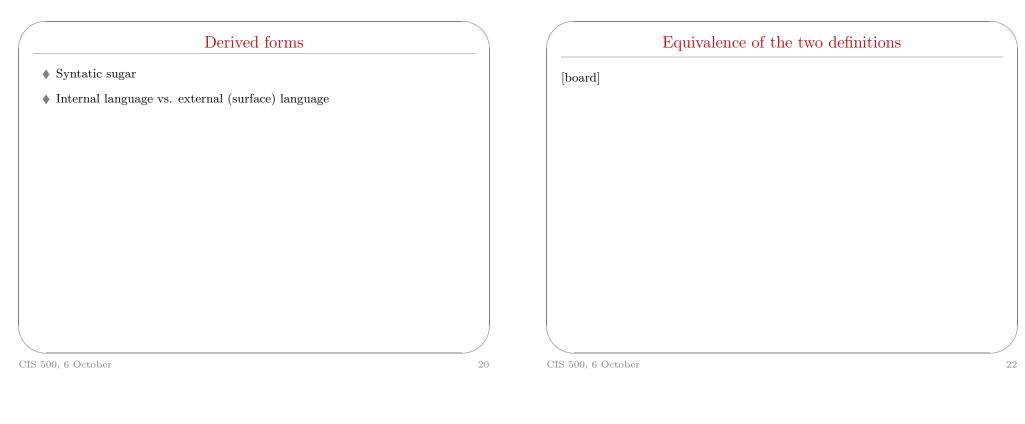
 $\begin{array}{c} Proving \ Preservation\\ \hline \\ Theorem: \ If \ \Gamma \vdash t : T \ and \ t \longrightarrow t', \ then \ \Gamma \vdash t' : T.\\ Proof: \ By induction \ on \ typing \ derivations.\\ \hline \\ [Which \ case \ is \ the \ hard \ one?]\\ \hline \\ Case \ T-APP: \ Given \ t = t_1 \ t_2 \\ \Gamma \vdash t_1 : \ T_{11} \longrightarrow T_{12} \\ \Gamma \vdash t_2 : \ T_{11} \\ T = T_{12} \\ Show \quad \Gamma \vdash t' : \ T_{12}\\ \hline \\ By \ the \ inversion \ lemma \ for \ evaluation, \ there \ are \ three \ subcases...\\ \hline \\ Subcase: \ t_1 = \lambda x: T_{11}. \ t_{12} \\ t_2 \ a \ value \ v_2 \\ t' = [x \mapsto v_2]t_{12}\\ \hline \\ Uh \ oh. \end{array}$ 

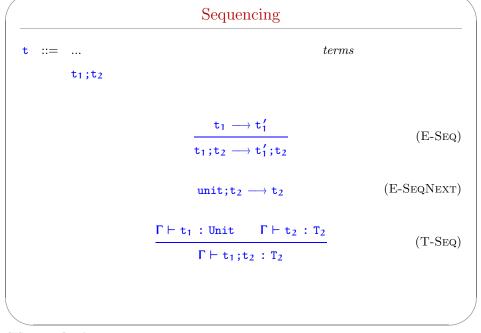
16

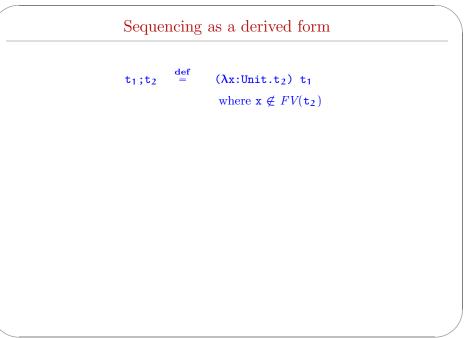


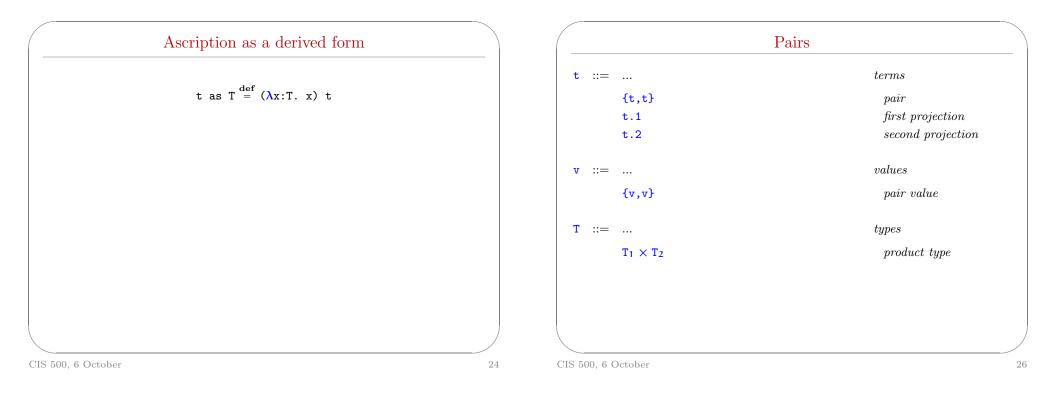


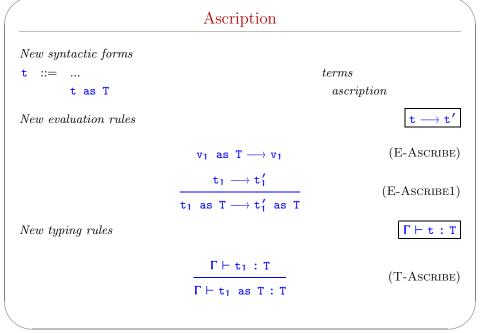


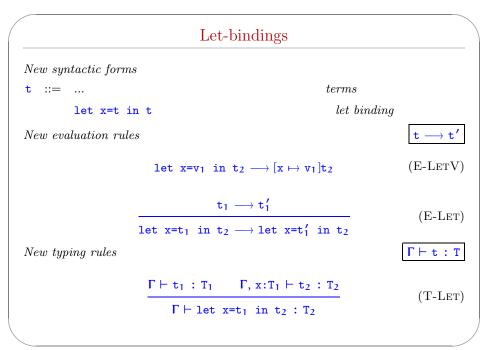






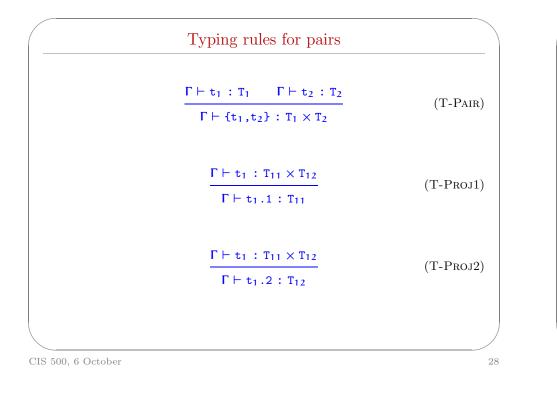


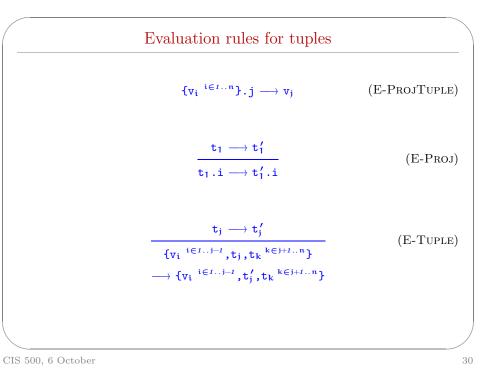




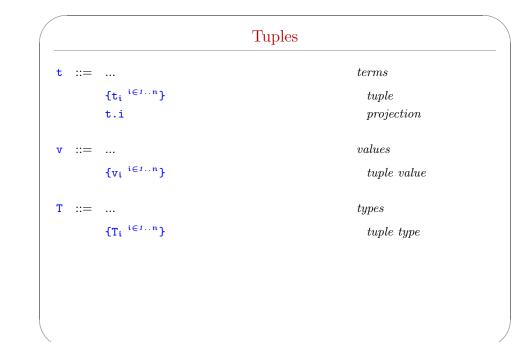
23

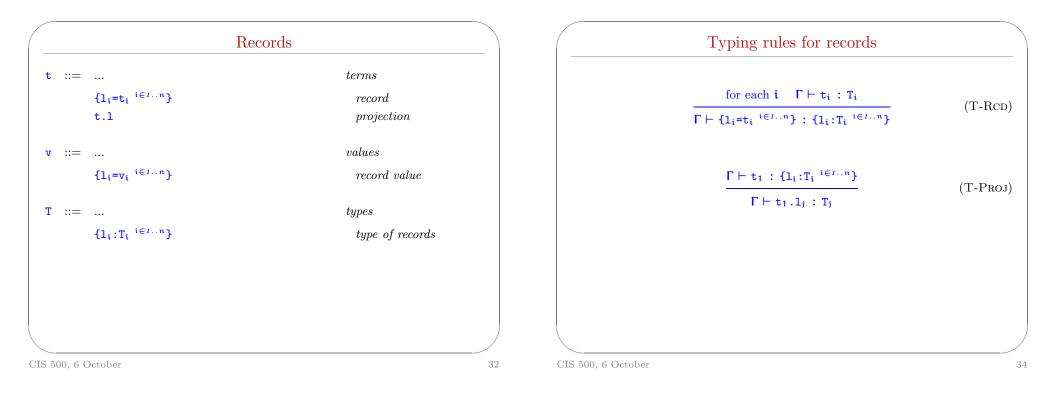
CIS 500, 6 October





 Evaluation rules for pairs	
$\{v_1, v_2\}.1 \longrightarrow v_1$	(E-PAIRBETA1)
$\{v_1, v_2\}.2 \longrightarrow v_2$	(E-PAIRBETA2)
$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1.1 \longrightarrow \mathtt{t}_1'.1}$	(E-Proj1)
$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1.2 \longrightarrow \mathtt{t}_1'.2}$	(E-Proj2)
$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\{\mathtt{t}_1, \mathtt{t}_2\} \longrightarrow \{\mathtt{t}_1', \mathtt{t}_2\}}$	(E-Pair1)
$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\{\mathtt{v}_1, \mathtt{t}_2\} \longrightarrow \{\mathtt{v}_1, \mathtt{t}_2'\}}$	(E-Pair2)
$\{\mathtt{v}_1,\mathtt{t}_2\}\longrightarrow\{\mathtt{v}_1,\mathtt{t}_2'\}$	





Typing rules for tuples	
$\frac{\text{for each } i  \Gamma \vdash t_i : T_i}{\Gamma \vdash \{t_i^{i \in I \dots n}\} : \{T_i^{i \in I \dots n}\}}$	(T-Tuple)
$\frac{\Gamma \vdash t_1 : \{T_i^{i \in In}\}}{\Gamma \vdash t_1 . j : T_j}$	(T-Proj)

E-ProjRcd)
(E-Proj)
(E-RCD)

# Intro vs. elim forms

An introduction form for a given type gives us a way of constructing elements of this type.

An elimination form for a type gives us a way of using elements of this type. What typing rules are introduction forms? What are elimination forms?

# Propositions as Types

Logic	PROGRAMMING LANGUAGES
propositions	types
proposition $P \supset Q$	$\operatorname{type} P{\rightarrow} {\tt Q}$
proposition $P \wedge Q$	$\operatorname{type} \mathtt{P} \times \mathtt{Q}$
proof of proposition ${\bf P}$	term t of type P
proposition $\mathbf{P}$ is provable	type ${\tt P}$ is inhabited (by some term)

CIS 500, 6 October

36

C15 500, 0 OCIODCI

Discussion

# The Curry-Howard Correspondence

In constructive logics, a proof of **P** must provide evidence for **P**.

• "law of the excluded middle" —  $\mathbf{P} \lor \neg \mathbf{P}$  — not recognized.

A proof of  $P \wedge Q$  is a pair of evidence for P and evidence for Q.

A proof of  $P \supset Q$  is a procedure for transforming evidence for P into evidence for Q.

CIS 500, 6 October

38

propositions	types
	0 P 00
$\operatorname{proposition}  P \supset  Q$	type $P \rightarrow Q$
proposition $\mathbf{P} \wedge \mathbf{Q}$	type $P \times Q$
proof of proposition <b>P</b>	term t of type P
proposition $\mathbf{P}$ is provable	type $P$ is inhabited (by some term)
proof simplification	evaluation
(a.k.a. "cut elimination")	

# Typability

An untyped  $\lambda$ -term m is said to be typable if there is some term t in the simply typed lambda-calculus, some type T, and some context  $\Gamma$  such that erase(t) = m and  $\Gamma \vdash t : T$ .

Cf. type reconstruction in OCaml.

CIS 500, 6 October

38-b

Logic	Programming languages
propositions	types
proposition $P \supset Q$	$\operatorname{type} P{\rightarrow} Q$
proposition $\mathbf{P} \wedge \mathbf{Q}$	$\operatorname{type} \mathtt{P} \times \mathtt{Q}$
proof of proposition <b>P</b>	term t of type P
proposition $\mathbf{P}$ is provable	type P is inhabited (by some term)
	evaluation

Erasure				
	erase(x)	=	x	
	$erase(\lambda x:T_1. t_2)$	=	$\lambda x. erase(t_2)$	
	$erase(t_1 t_2)$	=	$erase(t_1) erase(t_2)$	

CIS 500, 6 October

40