Theory of Types and Programming Languages Fall 2022

Week 6

Plan

PREVIOUSLY:

1. type safety as *progress* and *preservation*
2. typed arithmetic expressions
3. simply typed lambda calculus (STLC)
   - 3.1 Progress
   - 3.2 Inversion Lemma
   - 3.3 Canonical Forms Lemma

TODAY:

1. STLC, continued
   - 1.1 Preservation for STLC
   - 1.2 Substitution Lemma
   - 1.3 Weakening and Permutation
2. Extensions to STLC

NEXT: state, recursion, polymorphism, etc.

Preservation for STLC

Preservation for STLC

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

Proof: By induction

Preservation for STLC

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

Proof: By induction on typing derivations.

Which case is the hard one??

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-App: Given $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$

$\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$

$\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$

$\mathtt{T} = \mathtt{T}_{12}$

Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given

$$\begin{aligned} &\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2 \\ &\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \rightarrow \mathtt{T}_{12} \\ &\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11} \\ &\mathtt{T} = \mathtt{T}_{12} \end{aligned}$$

Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

There are three subcases for such evaluation...

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given

$$\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$$
$$\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$$
$$\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$$
$$\mathtt{T} = \mathtt{T}_{12}$$

Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

There are three subcases for such evaluation...

Subcase: $\mathtt{t}_1 = \lambda \mathtt{x}{:}\mathtt{T}_{11}.\ \mathtt{t}_{12}$

$\mathtt{t}_2$ a value $\mathtt{v}_2$

$\mathtt{t}' = [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12}$

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given $\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$

$\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$

$\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$

$\mathtt{T} = \mathtt{T}_{12}$

Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

There are three subcases for such evaluation...

Subcase: $\mathtt{t}_1 = \lambda \mathtt{x}{:}\mathtt{T}_{11}.\ \mathtt{t}_{12}$

$\mathtt{t}_2$ a value $\mathtt{v}_2$

$\mathtt{t}' = [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12}$

Uh oh.

The "Substitution Lemma"

Lemma: Types are preserved under substititition.

That is, if $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

The "Substitution Lemma"

Lemma: Types are preserved under substititon.

That is, if $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: ...

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin dom(\Gamma)$, then $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathit{dom}(\Gamma)$, then $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.

Permutation tells us that the order of assumptions in (the list) Γ does not matter.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and Δ is a permutation of Γ, then $\Delta \vdash \mathtt{t} : \mathtt{T}$.

Weakening and Permutation

Two other lemmas will be useful.

Weakening tells us that we can *add assumptions* to the context without losing any true typing statements.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and $\mathtt{x} \notin \mathit{dom}(\Gamma)$, then $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.

Moreover, the latter derivation has the same depth as the former.

Permutation tells us that the order of assumptions in (the list) Γ does not matter.

Lemma: If $\Gamma \vdash \mathtt{t} : \mathtt{T}$ and Δ is a permutation of Γ, then $\Delta \vdash \mathtt{t} : \mathtt{T}$.

Moreover, the latter derivation has the same depth as the former.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

I.e., "Types are preserved under substitition."

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.
Proceed by cases on the final typing rule used in the derivation.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.
Proceed by cases on the final typing rule used in the derivation.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.
Proceed by cases on the final typing rule used in the derivation.

Case T-APP:

$$\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$$
$$\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$$
$$\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t}_2 : \mathtt{T}_2$$
$$\mathtt{T} = \mathtt{T}_1$$

By the induction hypothesis,
$\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$ and $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2 : \mathtt{T}_2$.

By T-APP, $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_2 : \mathtt{T}$
i.e., $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}](\mathtt{t}_1\ \mathtt{t}_2) : \mathtt{T}$.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.
Proceed by cases on the final typing rule used in the derivation.

Case T-VAR: $\mathtt{t} = \mathtt{z}$
with $\mathtt{z}{:}\mathtt{T} \in (\Gamma, \mathtt{x}{:}\mathtt{S})$

There are two sub-cases to consider, depending on whether $\mathtt{z}$ is $\mathtt{x}$ or another variable.

- If $\mathtt{z} = \mathtt{x}$, then $[\mathtt{x} \mapsto \mathtt{s}]\mathtt{z} = \mathtt{s}$. The required result is then $\Gamma \vdash \mathtt{s} : \mathtt{S}$, which is among the assumptions of the lemma.
- Otherwise, $[\mathtt{x} \mapsto \mathtt{s}]\mathtt{z} = \mathtt{z}$, and the desired result is immediate.

The "Substitution Lemma"

Lemma: If $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$, then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Proof: By induction on the derivation of $\Gamma, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t} : \mathtt{T}$.
Proceed by cases on the final typing rule used in the derivation.

Case T-ABS: $\mathtt{t} = \lambda\mathtt{y}{:}\mathtt{T}_2.\mathtt{t}_1 \qquad \mathtt{T} = \mathtt{T}_2 \to \mathtt{T}_1$
$\Gamma, \mathtt{x}{:}\mathtt{S}, \mathtt{y}{:}\mathtt{T}_2 \vdash \mathtt{t}_1 : \mathtt{T}_1$

By our conventions on choice of bound variable names, we may assume $\mathtt{x} \neq \mathtt{y}$ and $\mathtt{y} \notin FV(\mathtt{s})$.

- Using *permutation* on the given subderivation, we obtain $\Gamma, \mathtt{y}{:}\mathtt{T}_2, \mathtt{x}{:}\mathtt{S} \vdash \mathtt{t}_1 : \mathtt{T}_1$.
- Using *weakening* on the other given derivation ($\Gamma \vdash \mathtt{s} : \mathtt{S}$), we obtain $\Gamma, \mathtt{y}{:}\mathtt{T}_2 \vdash \mathtt{s} : \mathtt{S}$.
- Now, by the induction hypothesis, $\Gamma, \mathtt{y}{:}\mathtt{T}_2 \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_1$.
- By T-ABS, $\Gamma \vdash \lambda\mathtt{y}{:}\mathtt{T}_2.\ [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t}_1 : \mathtt{T}_2 \to \mathtt{T}_1$, i.e. (by the definition of substitution), $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}](\lambda\mathtt{y}{:}\mathtt{T}_2.\ \mathtt{t}_1) : \mathtt{T}_2 \to \mathtt{T}_1$.

Preservation for STLC

Going back to preservation...

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given
$$\texttt{t} = \texttt{t}_1\ \texttt{t}_2$$
$$\Gamma \vdash \texttt{t}_1 : \texttt{T}_{11} \to \texttt{T}_{12}$$
$$\Gamma \vdash \texttt{t}_2 : \texttt{T}_{11}$$
$$\texttt{T} = \texttt{T}_{12}$$

Show $\Gamma \vdash \texttt{t}' : \texttt{T}_{12}$

There are three subcases for such evaluation...

Subcase: $\texttt{t}_1 = \lambda \texttt{x}{:}\texttt{T}_{11}.\ \texttt{t}_{12}$

$\texttt{t}_2$ a value $\texttt{v}_2$

$\texttt{t}' = [\texttt{x} \mapsto \texttt{v}_2]\texttt{t}_{12}$

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given
$$\begin{aligned} &\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2 \\ &\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12} \\ &\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11} \\ &\mathtt{T} = \mathtt{T}_{12} \end{aligned}$$
Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

There are three subcases for such evaluation...

Subcase:
$\mathtt{t}_1 = \lambda \mathtt{x}{:}\mathtt{T}_{11}.\ \mathtt{t}_{12}$
$\mathtt{t}_2$ a value $\mathtt{v}_2$
$\mathtt{t}' = [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12}$

By inversion, we have $\Gamma, \mathtt{x}{:}\mathtt{T}_{11} \vdash \mathtt{t}_{12} : \mathtt{T}_{12}$.

Preservation for STLC

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

Proof: By induction on typing derivations.

Case T-APP: Given
$$\mathtt{t} = \mathtt{t}_1\ \mathtt{t}_2$$
$$\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12}$$
$$\Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$$
$$\mathtt{T} = \mathtt{T}_{12}$$
Show $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$

There are three subcases for such evaluation...

Subcase: $\mathtt{t}_1 = \lambda \mathtt{x}{:}\mathtt{T}_{11}.\ \mathtt{t}_{12}$
$\mathtt{t}_2$ a value $\mathtt{v}_2$
$\mathtt{t}' = [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t}_{12}$

By inversion, we have $\Gamma, \mathtt{x}{:}\mathtt{T}_{11} \vdash \mathtt{t}_{12} : \mathtt{T}_{12}$.

By the substitution lemma, this gives us $\Gamma \vdash \mathtt{t}' : \mathtt{T}_{12}$.

Summary: Preservation

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

Lemmas to prove:

- Weakening
- Permutation
- Substitution preserves types
- Reduction preserves types (i.e., preservation)

Review: Type Systems

To define and verify a type system, you must:

1. Define types
2. Specify typing rules
3. Prove soundness: *progress* and *preservation*

Two Typing Topics

Erasure

$$
\begin{array}{lcl}
\mathit{erase}(\mathtt{x}) & = & \mathtt{x} \\
\mathit{erase}(\lambda\mathtt{x}:\mathtt{T_1.}\ \mathtt{t_2}) & = & \lambda\mathtt{x.}\ \mathit{erase}(\mathtt{t_2}) \\
\mathit{erase}(\mathtt{t_1}\ \mathtt{t_2}) & = & \mathit{erase}(\mathtt{t_1})\ \mathit{erase}(\mathtt{t_2})
\end{array}
$$

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.

Extensions to STLC

Base types

Up to now, we've formulated "base types" (e.g. `Nat`) by adding them to the syntax of types, extending the syntax of terms with associated constants (`zero`) and operators (`succ`, etc.) and adding appropriate typing and evaluation rules. We can do this for as many base types as we like.

For more theoretical discussions (as opposed to programming) we can often ignore the term-level inhabitants of base types, and just treat these types as uninterpreted constants.
E.g., suppose `B` and `C` are some base types. Then we can ask (without knowing anything more about `B` or `C`) whether there are any types `S` and `T` such that the term

```
(λf:S. λg:T. f g) (λx:B. x)
```

is well typed.

The $\texttt{Unit}$ type

$\texttt{t}$	$::=$	$\dots$	*terms*
		$\texttt{unit}$	*constant* $\texttt{unit}$
$\texttt{v}$	$::=$	$\dots$	*values*
		$\texttt{unit}$	*constant* $\textit{unit}$
$\texttt{T}$	$::=$	$\dots$	*types*
		$\texttt{Unit}$	*unit type*

New typing rules $\boxed{\Gamma \vdash \texttt{t} : \texttt{T}}$

$$\Gamma \vdash \texttt{unit} : \texttt{Unit} \qquad \text{(T-UNIT)}$$

Sequencing

$$\begin{array}{lcll} \texttt{t} & ::= & \ldots & \textit{terms} \\ & & \texttt{t}_1\texttt{;}\texttt{t}_2 & \end{array}$$

Sequencing

$$\texttt{t} \quad ::= \quad \ldots \qquad\qquad \textit{terms}$$
$$\qquad\qquad \texttt{t}_1\texttt{;t}_2$$

$$\frac{\texttt{t}_1 \longrightarrow \texttt{t}_1'}{\texttt{t}_1\texttt{;t}_2 \longrightarrow \texttt{t}_1'\texttt{;t}_2} \qquad \text{(E-Seq)}$$

$$\texttt{unit;t}_2 \longrightarrow \texttt{t}_2 \qquad \text{(E-SeqNext)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Unit} \qquad \Gamma \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \vdash \texttt{t}_1\texttt{;t}_2 : \texttt{T}_2} \qquad \text{(T-Seq)}$$

Derived forms

- Syntatic sugar
- Internal language vs. external (surface) language

Sequencing as a derived form

$$\mathtt{t}_1\mathtt{;t}_2 \quad \stackrel{\text{def}}{=} \quad (\lambda\mathtt{x}\mathtt{:Unit.t}_2)\ \mathtt{t}_1$$
$$\text{where } \mathtt{x} \notin FV(\mathtt{t}_2)$$

Equivalence of the two definitions

[board]

Ascription

New syntactic forms

$$\begin{array}{llll} \mathtt{t} & ::= & \ldots & \textit{terms} \\ & & \mathtt{t\ as\ T} & \quad \textit{ascription} \end{array}$$

New evaluation rules $\boxed{\mathtt{t} \longrightarrow \mathtt{t}'}$

$$\mathtt{v_1\ as\ T} \longrightarrow \mathtt{v_1} \qquad \text{(E-Ascribe)}$$

$$\frac{\mathtt{t_1} \longrightarrow \mathtt{t_1'}}{\mathtt{t_1\ as\ T} \longrightarrow \mathtt{t_1'\ as\ T}} \qquad \text{(E-Ascribe1)}$$

New typing rules $\boxed{\Gamma \vdash \mathtt{t} : \mathtt{T}}$

$$\frac{\Gamma \vdash \mathtt{t_1} : \mathtt{T}}{\Gamma \vdash \mathtt{t_1\ as\ T} : \mathtt{T}} \qquad \text{(T-Ascribe)}$$

Ascription as a derived form

$$\texttt{t as T} \stackrel{\text{def}}{=} (\lambda\texttt{x:T. x})\ \texttt{t}$$

Let-bindings

New syntactic forms

$$\begin{array}{llll} \texttt{t} & ::= & \ldots & \textit{terms} \\ & & \texttt{let x=t in t} & \textit{let binding} \end{array}$$

New evaluation rules $\boxed{\texttt{t} \longrightarrow \texttt{t}'}$

$$\texttt{let x=v}_1\texttt{ in t}_2 \longrightarrow [\texttt{x} \mapsto \texttt{v}_1]\texttt{t}_2 \qquad \text{(E-LetV)}$$

$$\frac{\texttt{t}_1 \longrightarrow \texttt{t}_1'}{\texttt{let x=t}_1\texttt{ in t}_2 \longrightarrow \texttt{let x=t}_1'\texttt{ in t}_2} \qquad \text{(E-Let)}$$

New typing rules $\boxed{\Gamma \vdash \texttt{t} : \texttt{T}}$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{T}_1 \qquad \Gamma, \texttt{x}{:}\texttt{T}_1 \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \vdash \texttt{let x=t}_1\texttt{ in t}_2 : \texttt{T}_2} \qquad \text{(T-Let)}$$

Pairs

`t`	$::=$	...	*terms*
		`{t,t}`	*pair*
		`t.1`	*first projection*
		`t.2`	*second projection*
`v`	$::=$	...	*values*
		`{v,v}`	*pair value*
`T`	$::=$	...	*types*
		$\mathtt{T}_1 \times \mathtt{T}_2$	*product type*

Evaluation rules for pairs

$$\{\mathtt{v}_1,\mathtt{v}_2\}.1 \longrightarrow \mathtt{v}_1 \qquad \text{(E-PairBeta1)}$$

$$\{\mathtt{v}_1,\mathtt{v}_2\}.2 \longrightarrow \mathtt{v}_2 \qquad \text{(E-PairBeta2)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1.1 \longrightarrow \mathtt{t}_1'.1} \qquad \text{(E-Proj1)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1.2 \longrightarrow \mathtt{t}_1'.2} \qquad \text{(E-Proj2)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\{\mathtt{t}_1,\mathtt{t}_2\} \longrightarrow \{\mathtt{t}_1',\mathtt{t}_2\}} \qquad \text{(E-Pair1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\{\mathtt{v}_1,\mathtt{t}_2\} \longrightarrow \{\mathtt{v}_1,\mathtt{t}_2'\}} \qquad \text{(E-Pair2)}$$

Typing rules for pairs

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_1 \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \{\mathtt{t}_1,\mathtt{t}_2\} : \mathtt{T}_1 \times \mathtt{T}_2} \qquad \text{(T-Pair)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \times \mathtt{T}_{12}}{\Gamma \vdash \mathtt{t}_1\mathtt{.1} : \mathtt{T}_{11}} \qquad \text{(T-Proj1)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \times \mathtt{T}_{12}}{\Gamma \vdash \mathtt{t}_1\mathtt{.2} : \mathtt{T}_{12}} \qquad \text{(T-Proj2)}$$

Tuples

$\mathtt{t}$	$::=$	$\ldots$	*terms*
		$\{\mathtt{t}_i\ ^{i\in 1..n}\}$	*tuple*
		$\mathtt{t.i}$	*projection*
$\mathtt{v}$	$::=$	$\ldots$	*values*
		$\{\mathtt{v}_i\ ^{i\in 1..n}\}$	*tuple value*
$\mathtt{T}$	$::=$	$\ldots$	*types*
		$\{\mathtt{T}_i\ ^{i\in 1..n}\}$	*tuple type*

Evaluation rules for tuples

$$\{\mathtt{v}_i \ ^{i\in 1..n}\}.\mathtt{j} \longrightarrow \mathtt{v}_j \qquad \text{(E-ProjTuple)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1.\mathtt{i} \longrightarrow \mathtt{t}_1'.\mathtt{i}} \qquad \text{(E-Proj)}$$

$$\frac{\mathtt{t}_j \longrightarrow \mathtt{t}_j'}{\begin{array}{c}\{\mathtt{v}_i \ ^{i\in 1..j-1}, \mathtt{t}_j, \mathtt{t}_k \ ^{k\in j+1..n}\} \\ \longrightarrow \{\mathtt{v}_i \ ^{i\in 1..j-1}, \mathtt{t}_j', \mathtt{t}_k \ ^{k\in j+1..n}\}\end{array}} \qquad \text{(E-Tuple)}$$

Typing rules for tuples

$$\frac{\text{for each } i \quad \Gamma \vdash \mathtt{t}_i : \mathtt{T}_i}{\Gamma \vdash \{\mathtt{t}_i{}^{\,i \in 1..n}\} : \{\mathtt{T}_i{}^{\,i \in 1..n}\}} \qquad \text{(T-Tuple)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \{\mathtt{T}_i{}^{\,i \in 1..n}\}}{\Gamma \vdash \mathtt{t}_1\mathtt{.j} : \mathtt{T}_j} \qquad \text{(T-Proj)}$$