barkhauseninstitut.org

Foundations of Certified Programming Language and Compiler Design

Dr.-Ing. Sebastian Ertel

Composable Operating Systems Group, Barkhausen Institute

Outline

Lecture	Logic	Formalisms	PL
1	Propositional and first-order logic		
2			Functional programming
3		Syntax and Semantics	
4			The untyped lambda calculus
5		Types	
6			The typed lambda calculus
7			Polymorphism
8		Curry-Howard	
9			Higher-order types
10			Dependent types

The simply typed lambda calculus

Syntax:

$$
\begin{array}{llll}
t & ::= & & \text{terms:} \\
 & | & x & \text{variable} \\
 & | & \lambda x : T.t & \text{abstraction} \\
 & | & t\ t & \text{application} \\
v & ::= & & \text{values:} \\
 & | & \lambda x : T.t & \text{abstraction value}
\end{array}
$$

$$
\begin{array}{llll}
T & ::= & & \text{types:} \\
 & | & T \to T & \text{type of functions} \\
\Gamma & ::= & & \text{contexts:} \\
 & | & \varnothing & \text{empty context} \\
 & | & \Gamma, x : T & \text{term variable binding}
\end{array}
$$

Semantics:

$$\boxed{t \longrightarrow t'}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{E-App1} \qquad \frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'}\ \text{E-App2} \qquad \frac{}{(\lambda x : T.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12}}\ \text{E-AppAbs}$$

The simply typed lambda calculus

Syntax:

$$
\begin{array}{lcll}
t & ::= & & \text{terms:} \\
 & | & x & \text{variable} \\
 & | & \lambda x : T.t & \text{abstraction} \\
 & | & t\ t & \text{application} \\
v & ::= & & \text{values:} \\
 & | & \lambda x : T.t & \text{abstraction value}
\end{array}
$$

$$
\begin{array}{lcll}
T & ::= & & \text{types:} \\
 & | & T \to T & \text{type of functions} \\
\Gamma & ::= & & \text{contexts:} \\
 & | & \emptyset & \text{empty context} \\
 & | & \Gamma, x : T & \text{term variable binding}
\end{array}
$$

Typing

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \ \text{T-VAR} \qquad \frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t_2 : T_1 \to T_2} \ \text{T-ABS} \qquad \frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \quad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \ \text{T-APP}$$

Derivation Trees

- Typing derviations:

$$\dfrac{\dfrac{\dfrac{x : \texttt{Bool} \in x : \texttt{Bool}}{x : \texttt{Bool} \vdash x : \texttt{Bool}}\ \textsc{T-Var}}{\lambda x : \texttt{Bool}.x : \texttt{Bool} \to \texttt{Bool}}\ \textsc{T-Abs} \qquad \dfrac{}{\vdash \texttt{true} : \texttt{Bool}}\ \textsc{T-True}}{\vdash (\lambda x : \texttt{Bool}.x)\ \texttt{true} : \texttt{Bool}}\ \textsc{T-App}$$

- Check: Show the derivation tree for $f : \texttt{Bool} \to \texttt{Bool} \vdash f\ (\texttt{if false then true else false}) : \texttt{Bool}$

$$\dfrac{\dfrac{f : \texttt{Bool} \to \texttt{Bool} \in \Gamma}{\Gamma \vdash f : \texttt{Bool} \to \texttt{Bool}}\ \textsc{T-Var} \qquad \dfrac{\dfrac{}{\texttt{false} : \texttt{Bool}}\ \textsc{T-False} \quad \dfrac{}{\texttt{true} : \texttt{Bool}}\ \textsc{T-True} \quad \dfrac{}{\texttt{false} : \texttt{Bool}}\ \textsc{T-False}}{\Gamma \vdash \texttt{if false then true else false} : \texttt{Bool}}\ \textsc{T-If}}{f : \texttt{Bool} \to \texttt{Bool} \vdash f\ (\texttt{if false then true else false}) : \texttt{Bool}}\ \textsc{T-App}$$

- Check: Find a context Γ for $f\ x\ y : \texttt{Bool}$.

$$\Gamma = \quad \begin{array}{lll} f : \texttt{Bool} \to \texttt{Bool} \to \texttt{Bool}, & f : \texttt{Nat} \to \texttt{Nat} \to \texttt{Bool}, & f : \texttt{T} \to \texttt{T} \to \texttt{Bool} \\ x : \texttt{Bool}, y : \texttt{Bool} & x : \texttt{Nat}, y : \texttt{Nat} & x : \texttt{T}, y : \texttt{T} \end{array}$$

Properties

Lemmas:

- Inversion of the typing relation.
- Canonical forms.

Theorems:

- Uniqueness of types.

Theorem (Progress)

Suppose t is a closed, well-typed term, i.e., $\vdash t : T$. Then either t is a value or there exists some t' such that $t \longrightarrow t'$.

Theorem (Preservation)

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

Theorem (Normalization)

If $\vdash t : T$, then t is normalizable.

What is the type of $(\lambda x.x\ x)\ (\lambda x.x\ x)$?

Extensions

- Currently, we cannot implement the STLC because we are missing the base case for our types.
- Let's study extensions:

Extensions
Base Types

- Base types: `Nat`, `Bool`, `String`, `Float`, ...
- Let's add some uninterpreted/unkown base types without any primitive operations.

  New syntactic forms:

$$\begin{array}{lll} T & ::= & \ldots \\ & | & \boxed{\mathtt{A}} \end{array}$$

$T ::= \ldots$	types:
$\mid \boxed{\mathtt{A}}$	base type

- Consider: $\lambda x : A.\, x : A \to A$
- We could assume that $A = \mathtt{Nat}$ is some number: $(\lambda x : A.\, x : A \to A)\ 5$

Extensions

- But where do these values come from and what actually is `Nat`?
- We do not want to add unknown values/types?!
- We would like to have a closed system to reason about it.

Unit

New syntactic forms:

$t ::= \dots$	terms:	$v ::= \dots$	values:
$\mid$ `unit`	constant `unit`	$\mid$ `unit`	constant `unit`
$T ::= \dots$	types:		
$\mid$ `Unit`	unit type		

New Typing Rules:

$$\boxed{\Gamma \vdash t : T}$$

$$\frac{}{\texttt{unit} : \texttt{Unit}} \quad \text{T-Unit}$$

Ascription

- Document your code ... with types!

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \dots & \text{terms:} \\
 & | & t \text{ as } T & \text{ascription}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{}{v_1 \text{ as } T_1 \longrightarrow v_1} \quad \text{(E-Ascribe1)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1 \text{ as } T_1 \longrightarrow t_1' \text{ as } T_1} \quad \text{(E-Ascribe2)}$$

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash t_1 \text{ as } T_1 : T_1} \quad \text{(T-Ascribe)}$$

Product Types a.k.a. Pairs a.k.a. Tuples (with two elements)

New syntactic forms:

$$
\begin{array}{rclll}
t & ::= & \ldots & & \text{terms:} \\
 & | & \{t, t\} & & \text{pair} \\
 & | & t.1 & & \text{first projection} \\
 & | & t.2 & & \text{second projection}
\end{array}
$$

$$
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
 & | & \{v, v\} & \text{pair}
\end{array}
$$

$$
\begin{array}{rcll}
T & ::= & \ldots & \text{types:} \\
 & | & T_1 \times T_2 & \text{product type}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{}{\{v_1, v_2\}.1 \longrightarrow v_1} \quad \text{E-PairBeta1}$$

$$\frac{}{\{v_1, v_2\}.2 \longrightarrow v_2} \quad \text{E-PairBeta2}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.1 \longrightarrow t_1'.1} \quad \text{E-Proj1}$$

$$\frac{t_2 \longrightarrow t_2'}{t_2.2 \longrightarrow t_2'.2} \quad \text{E-Proj2}$$

$$\frac{t_1 \longrightarrow t_1'}{\{t_1, t_2\} \longrightarrow \{t_1', t_2\}} \quad \text{E-Pair1}$$

$$\frac{t_2 \longrightarrow t_2'}{\{v_1, t_2\} \longrightarrow \{v_1, t_2'\}} \quad \text{E-Pair2}$$

Product Types a.k.a. Pairs a.k.a. Tuples (with two elements)

New syntactic forms:

t	$::=$	$\ldots$	terms:
	$\mid$	$\{t, t\}$	pair
	$\mid$	$t.1$	first projection
	$\mid$	$t.2$	second projection

v	$::=$	$\ldots$	values:
	$\mid$	$\{v, v\}$	pair

T	$::=$	$\ldots$	types:
	$\mid$	$T_1 \times T_2$	product type

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t_1 : T_1 \quad \Gamma \vdash t_2 : T_2}{\Gamma \vdash \{t_1, t_2\} : T_1 \times T_2} \text{ T-PAIR}$$

$$\frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash t.1 : T_1} \text{ T-PROJ1}$$

$$\frac{\Gamma \vdash t : T_1 \times T_2}{\Gamma \vdash t.2 : T_2} \text{ T-PROJ2}$$

Sum Types

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \ldots & \text{terms:} \\
 & | & \texttt{inl}\ t & \text{tagging (left)} \\
 & | & \texttt{inr}\ t & \text{tagging (right)} \\
 & | & \texttt{case}\ t\ \texttt{of}\ \texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ x \Rightarrow t & \text{case}
\end{array}
$$

$$
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
 & | & \texttt{inl}\ v & \text{tagged value (left)} \\
 & | & \texttt{inr}\ v & \text{tagged value (right)}
\end{array}
$$

$$
\begin{array}{rcll}
T & ::= & \ldots & \text{types:} \\
 & | & T + T & \text{sum type}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{t \longrightarrow t'}{\texttt{inl}\ t \longrightarrow \texttt{inl}\ t'}\ \text{E-InL} \qquad \frac{t \longrightarrow t'}{\texttt{inr}\ t \longrightarrow \texttt{inr}\ t'}\ \text{E-InR}$$

Sum Types

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \ldots & \text{terms:} \\
& | & \texttt{inl}\ t & \text{tagging (left)} \\
& | & \texttt{inr}\ t & \text{tagging (right)} \\
& | & \texttt{case}\ t\ \texttt{of}\ \texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ x \Rightarrow t & \text{case}
\end{array}
$$

$$
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
& | & \texttt{inl}\ v & \text{tagged value (left)} \\
& | & \texttt{inr}\ v & \text{tagged value (right)}
\end{array}
$$

$$
\begin{array}{rcll}
T & ::= & \ldots & \text{types:} \\
& | & T + T & \text{sum type}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{}{\texttt{case}\ \texttt{inl}\ v_1\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_1]t_1} \quad \text{E-CaseInL}$$

$$\frac{}{\texttt{case}\ \texttt{inr}\ v_2\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \quad \text{E-CaseInR}$$

$$\frac{t_0 \longrightarrow t_0'}{\texttt{case}\ t_0\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow \texttt{case}\ t_0'\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2} \quad \text{E-Case}$$

Sum Types

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \ldots & \text{terms:} \\
& | & \texttt{inl}\ t & \text{tagging (left)} \\
& | & \texttt{inr}\ t & \text{tagging (right)} \\
& | & \texttt{case}\ t\ \texttt{of}\ \texttt{inl}\ x \Rightarrow t \mid \texttt{inr}\ x \Rightarrow t & \text{case}
\end{array}
$$

$$
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
& | & \texttt{inl}\ v & \text{tagged value (left)} \\
& | & \texttt{inr}\ v & \text{tagged value (right)}
\end{array}
$$

$$
\begin{array}{rcll}
T & ::= & \ldots & \text{types:} \\
& | & T + T & \text{sum type}
\end{array}
$$

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t : T_1}{\Gamma \vdash \texttt{inl}\ t : T_1 + T_2}\ \text{T-INL} \qquad \frac{\Gamma \vdash t : T_2}{\Gamma \vdash \texttt{inl}\ t : T_1 + T_2}\ \text{T-INR}$$

$$\frac{\Gamma \vdash t_0 : T_1 + T_2 \quad \Gamma, x_1 : T_1 \vdash t_1 : T \quad \Gamma, x_2 : T_2 \vdash t_2 : T}{\Gamma \vdash \texttt{case}\ t_0\ \texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 : T}\ \text{T-CASE}$$

Labeled Product Types a.k.a. Records

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \ldots & \text{terms:} \\
 & | & \{l_i{=}t_i\ ^{i\in 1..n}\} & \text{record} \\
 & | & t.l & \text{projection}
\end{array}
$$

$$
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
 & | & \{l_i{=}v_i\ ^{i\in 1..n}\} & \text{record value}
\end{array}
$$

$$
\begin{array}{rcll}
T & ::= & \ldots & \text{types:} \\
 & | & \{l_i{:}T_i\ ^{i\in 1..n}\} & \text{type of records}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$
\frac{}{\{l_i{=}v_i\ ^{i\in 1..n}\}.l_j \longrightarrow v_j} \quad \text{E-ProjRcd}
$$

$$
\frac{t_1 \longrightarrow t_1'}{t_1.l \longrightarrow t_1'.l} \quad \text{E-Proj}
$$

Labeled Product Types a.k.a. Records

New syntactic forms:

t	::=	$\ldots$	terms:
	$\mid$	$\{l_i{=}t_i \;^{i\in 1..n}\}$	record
	$\mid$	$t.l$	projection

v	::=	$\ldots$	values:
	$\mid$	$\{l_i{=}v_i \;^{i\in 1..n}\}$	record value
T	::=	$\ldots$	types:
	$\mid$	$\{l_i{:}T_i \;^{i\in 1..n}\}$	type of records

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{t_j \longrightarrow t'_j}{\{l_i{=}v_i \;^{i\in 1..j-1}, l_j{=}t_j, l_k{=}t_k \;^{k\in j+1..n}\} \longrightarrow \{l_i{=}v_i \;^{i\in 1..j-1}, l_j{=}t'_j, l_k{=}t_k \;^{k\in j+1..n}\}} \quad \text{E-RCD}$$

Labeled Product Types a.k.a. Records

New syntactic forms:

t	$::=$	$\ldots$	terms:
	$\mid$	$\{l_i{=}t_i\ ^{i\in 1..n}\}$	record
	$\mid$	$t.l$	projection

v	$::=$	$\ldots$	values:
	$\mid$	$\{l_i{=}v_i\ ^{i\in 1..n}\}$	record value

T	$::=$	$\ldots$	types:
	$\mid$	$\{l_i{:}T_i\ ^{i\in 1..n}\}$	type of records

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$\frac{\text{for each } i \quad \Gamma \vdash t_i : T_i}{\Gamma \vdash \{l_i{=}t_i\ ^{i\in 1..n}\} : \{l_i{:}T_i\ ^{i\in 1..n}\}} \ \text{T-RCD}$$

$$\frac{\Gamma \vdash t_1 : \{l_i{:}T_i\ ^{i\in 1..j-1}, l_j{:}T_j, l_k{:}T_k\ ^{k\in j+1..n}\}}{\Gamma \vdash \{t_1.l_j\} : T_j} \ \text{T-PROJ}$$

Sum Types (again)
Uniqueness of Typing

Valid types for $\texttt{inl 5}$ are $\texttt{Nat} + \texttt{Nat}$, $\texttt{Nat} + \texttt{Bool}$ etc.

New syntactic forms:

$t ::= \dots$	terms:	$v ::= \dots$	values:
$\mid\ \texttt{inl}\ t\ \texttt{as}\ T$	tagging (left)	$\mid\ \texttt{inl}\ v\ \texttt{as}\ T$	tagged value (left)
$\mid\ \texttt{inr}\ t\ \texttt{as}\ T$	tagging (right)	$\mid\ \texttt{inr}\ v\ \texttt{as}\ T$	tagged value (right)

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$\frac{}{\texttt{case}\ (\texttt{inl}\ v_1\ \texttt{as}\ T)\,\texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_1]t_1} \quad \text{E-CaseInL}$$

$$\frac{}{\texttt{case}\ (\texttt{inr}\ v_2\ \texttt{as}\ T)\,\texttt{of}\ \texttt{inl}\ x_1 \Rightarrow t_1 | \texttt{inr}\ x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \quad \text{E-CaseInR}$$

Sum Types (again)
Uniqueness of Typing

Valid types for $\texttt{inl 5}$ are $\texttt{Nat} + \texttt{Nat}$, $\texttt{Nat} + \texttt{Bool}$ etc.

New syntactic forms:

$$
\begin{array}{rcll}
t & ::= & \ldots & \text{terms:} \\
& | & \texttt{inl}\ t\ \texttt{as}\ T & \text{tagging (left)} \\
& | & \texttt{inr}\ t\ \texttt{as}\ T & \text{tagging (right)}
\end{array}
\qquad
\begin{array}{rcll}
v & ::= & \ldots & \text{values:} \\
& | & \texttt{inl}\ v\ \texttt{as}\ T & \text{tagged value (left)} \\
& | & \texttt{inr}\ v\ \texttt{as}\ T & \text{tagged value (right)}
\end{array}
$$

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$
\frac{\Gamma \vdash t : T_1}{\Gamma \vdash \texttt{inl}\ t\ \texttt{as}\ T_1 + T_2 : T_1 + T_2}\ \text{T-InL}
\qquad
\frac{\Gamma \vdash t : T_2}{\Gamma \vdash \texttt{inl}\ t\ \texttt{as}\ T_1 + T_2 : T_1 + T_2}\ \text{T-InR}
$$

Labeled Sum Types a.k.a. Variants a.k.a. Algebraic Datatypes

New syntactic forms:

$$
\begin{array}{lll}
t & ::= & \ldots \\
 & | & \texttt{<}l\texttt{=}t\texttt{>}\ \texttt{as}\ T \\
 & | & \texttt{case}\ t\ \texttt{of}\ \texttt{<}l_i\texttt{=}t_i\texttt{>} \Rightarrow t_i\ {}^{i\in 1..n}
\end{array}
$$

terms:
tagging
case

$$
\begin{array}{lll}
v & ::= & \ldots \\
 & | & \texttt{<}l\texttt{=}v\texttt{>}\ \texttt{as}\ T
\end{array}
$$

values:
tagged value

$$
\begin{array}{lll}
T & ::= & \ldots \\
 & | & \texttt{<}l_i : T_i\ {}^{i\in 1..n}\texttt{>}
\end{array}
$$

types:
type of variants

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$
\frac{t_i \longrightarrow t_i'}{\texttt{<}l_i\texttt{=}t_i\texttt{>}\ \texttt{as}\ T \longrightarrow \texttt{<}l_i\texttt{=}t_i'\texttt{>}\ \texttt{as}\ T}\ \textsc{E-Variant}
$$

Labeled Sum Types a.k.a. Variants a.k.a. Algebraic Datatypes

New syntactic forms:

$$
\begin{array}{lllr}
t & ::= & \ldots & \text{terms:} \\
 & | & \texttt{<}l\texttt{=}t\texttt{>}\ \texttt{as}\ T & \text{tagging} \\
 & | & \texttt{case}\ t\ \texttt{of}\ \texttt{<}l_i\texttt{=}t_i\texttt{>}\ \Rightarrow t_i\ ^{i\in 1..n} & \text{case}
\end{array}
$$

$$
\begin{array}{lllr}
v & ::= & \ldots & \text{values:} \\
 & | & \texttt{<}l\texttt{=}v\texttt{>}\ \texttt{as}\ T & \text{tagged value}
\end{array}
$$

$$
\begin{array}{lllr}
T & ::= & \ldots & \text{types:} \\
 & | & \texttt{<}l_i\texttt{:}T_i\ ^{i\in 1..n}\texttt{>} & \text{type of variants}
\end{array}
$$

New Evaluation Rules:

$\boxed{t \longrightarrow t'}$

$$
\frac{}{\texttt{case}\ (\texttt{<}l_j\texttt{=}t_j\texttt{>}\ \texttt{as}\ T)\ \texttt{of}\ \texttt{<}l_i\texttt{=}x_i\texttt{>} \Rightarrow t_i\ ^{i\in 1..n} \longrightarrow [x_j \mapsto v_j]t_j} \quad \text{E-CaseVariant}
$$

$$
\frac{t_0 \longrightarrow t_0'}{\texttt{case}\ t_0\ \texttt{of}\ \texttt{<}l_i\texttt{=}x_i\texttt{>} \Rightarrow t_i\ ^{i\in 1..n} \longrightarrow \texttt{case}\ t_0'\ \texttt{of}\ \texttt{<}l_i\texttt{=}x_i\texttt{>} \Rightarrow t_i\ ^{i\in 1..n}} \quad \text{E-Case}
$$

Labeled Sum Types a.k.a. Variants a.k.a. Algebraic Datatypes

New syntactic forms:

$t ::= \ldots$	terms:	$v ::= \ldots$	values:
$\mid$ `<`l`=`t`>` `as` T	tagging	$\mid$ `<`l`=`v`>` `as` T	tagged value
$\mid$ `case` t `of` `<`l_i`=`t_i`>` $\Rightarrow t_i\ ^{i \in 1..n}$	case		
		$T ::= \ldots$	types:
		$\mid \texttt{<}l_i : T_i\ ^{i \in 1..n}\texttt{>}$	type of variants

New Typing Rules:

$\boxed{\Gamma \vdash t : T}$

$$\frac{\Gamma \vdash t_j : T_j}{\Gamma \vdash \texttt{<}l_j\texttt{=}t_j\texttt{>}\ \texttt{as}\ \texttt{<}l_i : T_i\ ^{i \in 1..n}\texttt{>} : \texttt{<}l_i : T_i\ ^{i \in 1..n}\texttt{>}} \quad \text{T-VARIANT}$$

$$\frac{\Gamma \vdash t_0 : \texttt{<}l_i : T_i\ ^{i \in 1..n}\texttt{>} \qquad \text{for each } i \quad \Gamma, x_i : T_i \vdash t_i : T}{\Gamma \vdash \texttt{case}\ t_0\ \texttt{of}\ \texttt{<}l_i\texttt{=}x_i\texttt{>} \Rightarrow t_i\ ^{i \in 1..n} : T} \quad \text{T-CASE}$$

Examples

STLC++

```
Bool = <false:Unit, true:Unit>
false = <false=unit> as Bool
true = <true=unit> as Bool

Week = <weekday:Unit, weekend:Unit>
weekday = <weekday=unit> as Week
weekend = <weekend=unit> as Week
weekendYet = λ x:Week.
   case x of
     <weekday=unit> ⇒ false
     <weekend=unit> ⇒ true

OptionNat = <none:Unit, some:Nat>
NatList = <nil:Unit, cons:{Nat, NatList}>
Nat = <zero:Unit, succ:Nat>
```

Coq

```
Inductive bool := False | True.

Inductive week = Weekday | Weekend.

Inductive optionNat = None | Some (_:nat_
Inductive natList = Nil | Cons (_:nat) (_:natList).
Inductive nat = Zero | Succ (_:nat).
```