

Foundations of Certified Programming Language and Compiler Design

Dr.-Ing. Sebastian Ertel

Composable Operating Systems Group, Barkhausen Institute

Outline



ecture	Logic Propositional and first-order logic	Formalisms	PL
2	Tropositional and mot order logic		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



Syntax:



Syntax:

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



Syntax:





Syntax:



Semantics:



Syntax:



Semantics:

$$\begin{array}{c} \boxed{t\longrightarrow t'} \\ \\ \frac{t_1\longrightarrow t_1'}{t_1\,t_2\longrightarrow t_1'\,t_2} \end{array} \text{ E-App1} \end{array}$$



Syntax:

Semantics:

$$\begin{array}{c} \boxed{t\longrightarrow t'} \\ \\ \frac{t_1\longrightarrow t_1'}{t_1\;t_2\longrightarrow t_1'\;t_2} \;\; \text{E-App1} & \quad \frac{t_2\longrightarrow t_2'}{v_1\;t_2\longrightarrow v_1\;t_2'} \;\; \text{E-App2} \end{array}$$



Syntax:

Semantics:

$$\begin{array}{c|c} \hline t \longrightarrow t' \\ \\ \hline \frac{t_1 \longrightarrow t_1'}{t_1 \ t_2 \ \longrightarrow t_1' \ t_2} \ \text{E-App1} & \hline \frac{t_2 \longrightarrow t_2'}{v_1 \ t_2 \ \longrightarrow v_1 \ t_2'} \ \text{E-App2} & \hline \\ \hline (\lambda x \colon T \cdot t_{12}) \ v_2 \ \longrightarrow \ [x \mapsto v_2] t_{12} \end{array} \text{E-AppAbs}$$



Syntax:



Typing



Syntax:



Typing



$$\frac{x:T\in\Gamma}{\Gamma\vdash x:T} \text{ T-Var}$$



Syntax:



Typing

$$\Gamma \vdash t : T$$

$$\frac{x:T\in\Gamma}{\Gamma\vdash x:T} \text{ T-Var }$$

$$\frac{\Gamma, x: T_1 \vdash t_2: T_2}{\Gamma \vdash \lambda x: T_1.t_2: T_1 \rightarrow T_2} \ \ \text{T-Abs}$$



Syntax:

Typing

$$\Gamma \vdash t : T$$

$$\frac{x:T\in\Gamma}{\Gamma\vdash x:T} \text{ T-Var }$$

$$\frac{\Gamma,x:T_1\vdash t_2:T_2}{\Gamma\vdash \lambda x:T_1.t_2:T_1\to T_2} \ \ {}^{\text{T-ABS}}$$

$$\frac{\Gamma \vdash t_1: T_{11} \rightarrow T_{12} \quad \Gamma \vdash t_2: T_{11}}{\Gamma \vdash t_1 \ t_2: T_{12}} \quad \text{T-App}$$



• Typing derviations:



Typing derviations:

$$\begin{array}{c|c} \underline{x: \mathsf{Bool} \in x: \mathsf{Bool}} & \mathsf{T-VAR} \\ \underline{x: \mathsf{Bool} \vdash x: \mathsf{Bool}} & \mathsf{T-VAR} \\ \hline \lambda x: \mathsf{Bool}.x: \mathsf{Bool} \to \mathsf{Bool} & \mathsf{T-ABS} & \vdash \mathsf{true}: \mathsf{Bool} \\ \vdash (\lambda x: \mathsf{Bool}.x) \ \mathsf{true}: \mathsf{Bool} & \mathsf{T-APP} \end{array}$$



Typing derviations:

$$\begin{array}{c} \underline{x: \mathsf{Bool} \in x: \mathsf{Bool}} \\ \underline{x: \mathsf{Bool} \vdash x: \mathsf{Bool}} \\ \underline{\lambda x: \mathsf{Bool}.x: \mathsf{Bool} \to \mathsf{Bool}} \end{array} \begin{array}{c} \mathsf{T-VAR} \\ \hline \\ \vdash \mathsf{true}: \mathsf{Bool} \\ \hline \\ \vdash (\lambda x: \mathsf{Bool}.x) \ \mathsf{true}: \mathsf{Bool} \end{array} \begin{array}{c} \mathsf{T-TRUE} \\ \hline \\ \mathsf{T-APP} \end{array}$$

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



Typing derviations:

$$\frac{x: \texttt{Bool} \in x: \texttt{Bool}}{x: \texttt{Bool} \vdash x: \texttt{Bool}} \text{ T-VAR} \\ \frac{\lambda x: \texttt{Bool} \vdash x: \texttt{Bool}}{\vdash (\lambda x: \texttt{Bool} \rightarrow \texttt{Bool})} \text{ T-ABS} \\ \frac{\vdash \texttt{true}: \texttt{Bool}}{\vdash (\lambda x: \texttt{Bool} \cdot x) \texttt{ true}: \texttt{Bool}} \text{ T-APP}$$

· Check: Show the derivation tree for

 $f: \mathtt{Bool} \to \mathtt{Bool} \vdash f (\mathtt{if false then true else false}) : \mathtt{Bool}$

```
\frac{f: \texttt{Bool} \to \texttt{Bool} \in \Gamma}{\Gamma \vdash f: \texttt{Bool} \to \texttt{Bool}} \  \, \text{T-VAR} \quad \frac{\overbrace{\texttt{false}: \texttt{Bool}} \quad \overline{\texttt{T-FALSE}} \quad \overline{\texttt{true}: \texttt{Bool}} \quad \overline{\texttt{T-TRUE}} \quad \overline{\texttt{false}: \texttt{Bool}}}{\Gamma \vdash \text{if false then true else false}: \underline{\texttt{Bool}}} \quad \overline{\texttt{T-APP}} \quad \overline{
```



Typing derviations:

$$\frac{x: \texttt{Bool} \in x: \texttt{Bool}}{x: \texttt{Bool} \vdash x: \texttt{Bool}} \xrightarrow{\texttt{T-VAR}} \frac{x: \texttt{Bool} \vdash x: \texttt{Bool}}{\texttt{Ax}: \texttt{Bool}.x: \texttt{Bool} \to \texttt{Bool}} \xrightarrow{\texttt{T-ABS}} \frac{\texttt{T-TRUE}}{\vdash \texttt{true}: \texttt{Bool}} \xrightarrow{\texttt{T-APP}}$$

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

$$\frac{f: \texttt{Bool} \to \texttt{Bool} \in \Gamma}{\Gamma \vdash f: \texttt{Bool} \to \texttt{Bool}} \xrightarrow{\mathsf{T-VAR}} \frac{\overline{\texttt{false} : \texttt{Bool}}}{\Gamma \vdash f: \texttt{false}} \xrightarrow{\mathsf{T-FALSE}} \frac{\overline{\texttt{true} : \texttt{Bool}}}{\overline{\texttt{true} : \texttt{Bool}}} \xrightarrow{\mathsf{T-TRUE}} \frac{\mathsf{T-FALSE}}{\mathsf{T-IF}} \xrightarrow{\mathsf{T-APP}} \frac{\mathsf{T-APP}}{\mathsf{T-APP}}$$

• Check: Find a context Γ for f x y: Bool.



Typing derviations:

$$\frac{x: \texttt{Bool} \in x: \texttt{Bool}}{x: \texttt{Bool} \vdash x: \texttt{Bool}} \xrightarrow{\texttt{T-VAR}} \frac{x: \texttt{Bool} \vdash x: \texttt{Bool}}{\texttt{Ax}: \texttt{Bool}.x: \texttt{Bool} \to \texttt{Bool}} \xrightarrow{\texttt{T-ABS}} \frac{\texttt{T-TRUE}}{\vdash \texttt{true}: \texttt{Bool}} \xrightarrow{\texttt{T-APP}}$$

Check: Show the derivation tree for

 $f: \mathtt{Bool} o \mathtt{Bool} \vdash f \ (\mathtt{if} \ \mathtt{false} \ \mathtt{then} \ \mathtt{true} \ \mathtt{else} \ \mathtt{false}) : \mathtt{Bool}$

$$\frac{f: \texttt{Bool} \to \texttt{Bool} \in \Gamma}{\Gamma \vdash f: \texttt{Bool} \to \texttt{Bool}} \xrightarrow{\mathsf{T-VAR}} \frac{\overline{\texttt{false} : \texttt{Bool}}}{\Gamma \vdash \mathsf{if}} \xrightarrow{\mathsf{false} : \texttt{Bool}} \frac{\mathsf{T-FALSE}}{\mathsf{true} : \texttt{Bool}} \xrightarrow{\mathsf{T-TRUE}} \frac{\overline{\texttt{false} : \texttt{Bool}}}{\mathsf{T-HF}} \xrightarrow{\mathsf{T-APP}} \frac{\mathsf{T-FALSE}}{\mathsf{T-HF}}$$

• Check: Find a context Γ for $f \ x \ y$: Bool.

```
\Gamma = f: \mathtt{Bool} 	o \mathtt{Bool} 	o \mathtt{Bool}, \qquad f: \mathtt{Nat} 	o \mathtt{Nat} 	o \mathtt{Bool}, \qquad f: \mathtt{T} 	o \mathtt{T} 	o \mathtt{Bool} x: \mathtt{Bool}, y: \mathtt{Bool} x: \mathtt{Nat}, y: \mathtt{Nat} x: \mathtt{T}, y: \mathtt{T}
```



Lemmas:



Lemmas:

Inversion of the typing relation.



Lemmas:

Inversion of the typing relation. Canonical forms.



Lemmas:

Inversion of the typing relation. Canonical forms.

Theorems:



Lemmas:

Inversion of the typing relation.

Canonical forms.

Theorems:

Uniqueness of types.



Lemmas:

Inversion of the typing relation.

Canonical forms.

Theorems:

Uniqueness of types.



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 $t\longrightarrow t'$.



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 $\Gamma \vdash t' : T$



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 $\Gamma \vdash t' : T$

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



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 (Normalization)

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

Theorem (Preservation)

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

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:



• Base types: Nat, Bool, String, Float, ...



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



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



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

```
T ::= \dots types: 
 | A base type
```



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

$$T ::= \dots$$
 types:
 | A base type

• Consider: $\lambda x : A \cdot x : A \rightarrow A$



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

$$T ::= \dots$$
 types:
 | A base type

- Consider: $\lambda x : A. \ x : A \rightarrow A$
- 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:

Unit



New syntactic forms:



New syntactic forms:

New Typing Rules:

Unit



New syntactic forms:

New Typing Rules:

```
\boxed{\Gamma \vdash t : T} \boxed{\frac{}{\text{unit} : \text{Unit}}} \ \text{T-Unit}
```



• Document your code ... with types!



Document your code ... with types!
 New syntactic forms:



Document your code ... with types!
 New syntactic forms:

```
t ::= \dots terms:

| t \text{ as } T | ascription
```



Document your code ... with types!
 New syntactic forms:

```
t ::= \dots terms:

| t \text{ as } T | ascription
```

New Evaluation Rules:



Document your code ... with types!
 New syntactic forms:

$$t ::= \dots$$
 terms:
 $| t \text{ as } T |$ ascription

New Evaluation Rules:

$$t \longrightarrow t'$$

$$\overline{v_1 \text{ as } T_1 \longrightarrow v_1}$$
 E-Ascribe1

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



Document your code ... with types!
 New syntactic forms:

```
t ::= \dots terms:

| t \text{ as } T | ascription
```

New Evaluation Rules:

$$t \longrightarrow t'$$

$$\overline{v_1 \text{ as } T_1 \longrightarrow v_1}$$
 E-Ascribe1

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

New Typing Rules:



Document your code ... with types!
 New syntactic forms:

```
t ::= \dots terms:
```

New Evaluation Rules:

$$t \longrightarrow t'$$

$$\overline{v_1 \text{ as } T_1 \longrightarrow v_1}$$
 E-Ascribe1

$$rac{t_1 \longrightarrow t_1'}{t_1 ext{ as } T_1 \longrightarrow t_1' ext{ as } T_1} ext{ E-Ascribe2}$$

New Typing Rules:

$$\Gamma \vdash t : T$$

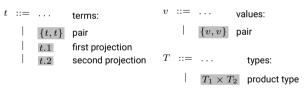
$$rac{\Gamma dash t_1:T_1}{\Gamma dash t_1 ext{ as } T_1:T_1}$$
 T-Ascribe



New syntactic forms:



New syntactic forms:





New syntactic forms:



New Evaluation Rules:



New syntactic forms:

New Evaluation Rules:

$$t \longrightarrow t'$$

$$\overline{\{v_1,v_2\}.1\longrightarrow v_1}$$
 E-PairBeta1
$$\underbrace{t_2\longrightarrow t_2'}_{\text{F-ProJ2}}$$
 E-ProJ2

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

$$rac{t_1 \longrightarrow t_1'}{\{t_1,t_2\} \longrightarrow \{t_1',t_2\}}$$
 E-Pair1

$$rac{t_1 \longrightarrow t_1'}{t_1.1 \longrightarrow t_1'.1}$$
 E-ProJ1

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



New syntactic forms:



New Typing Rules:

$$\frac{\Gamma \vdash t:T}{\Gamma \vdash \{t_1,t_2\}:T_1\times T_2} \quad \text{T-Pair} \qquad \frac{\Gamma \vdash t:T_1\times T_2}{\Gamma \vdash t.1:T_1} \quad \text{T-ProJ1} \qquad \frac{\Gamma \vdash t:T_1\times T_2}{\Gamma \vdash t.2:T_2} \quad \text{T-ProJ2}$$



New syntactic forms:



New syntactic forms:





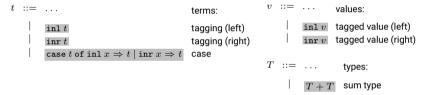
New syntactic forms:

New Evaluation Rules:

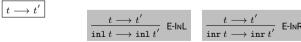


12

New syntactic forms:



New Evaluation Rules:



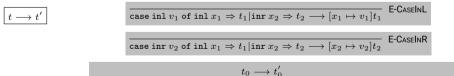


E-CASE

New syntactic forms:



New Evaluation Rules:



case t_0 of inl $x_1\Rightarrow t_1|$ inr $x_2\Rightarrow t_2\longrightarrow$ case t_0' of inl $x_1\Rightarrow t_1|$ inr $x_2\Rightarrow t_2$



New syntactic forms:

```
t ::= ...
                                                                                        values:
                                                 terms:
                                                 tagging (left)
                                                                                       tagged value (left)
                                                                                        tagged value (right)
                                                 tagging (right)
         case t of inl x \Rightarrow t \mid \text{inr } x \Rightarrow t
                                                 case
                                                                     T ::= ...
                                                                                          types:
                                                                                          sum type
```

New Typing Rules:

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

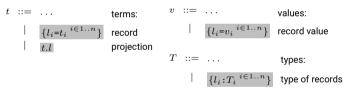
$$\frac{\Gamma \vdash t_0:T_1+T_2 \quad \Gamma, x_1:T_1\vdash t_1:T \quad \Gamma, x_2:T_2\vdash t_2:T}{\Gamma \vdash \text{case } t_0 \text{ of inl } x_1\Rightarrow t_1|\text{inr } x_2\Rightarrow t_2:T} \text{ T-CASE}$$



New syntactic forms:

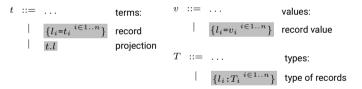


New syntactic forms:





New syntactic forms:



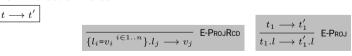
New Evaluation Rules:



New syntactic forms:

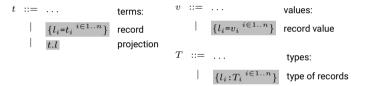


New Evaluation Rules:





New syntactic forms:

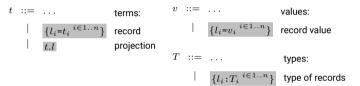


New Evaluation Rules:

$$\frac{t\longrightarrow t'}{\{l_i=v_i\ ^{i\in 1...j-1},l_j=t_j,l_k=t_k\ ^{k\in j+1..n}\}\longrightarrow \{l_i=v_i\ ^{i\in 1...j-1},l_j=t'_i,l_k=t_k\ ^{k\in j+1..n}\}}$$



New syntactic forms:



New Typing Rules:

$$\Gamma \vdash t : T$$

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

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



Valid types for inl 5 are Nat + Nat, Nat + Bool etc.



Valid types for inl 5 are Nat + Nat, Nat + Bool etc. New syntactic forms:



Valid types for inl 5 are Nat + Nat, Nat + Bool etc. New syntactic forms:



 $\label{lem:valid_types} \textbf{Valid} \ \textbf{types} \ \textbf{for inl} \ \ \textbf{5} \ \textbf{are} \ \textbf{Nat} + \textbf{Nat}, \textbf{Nat} + \textbf{Bool} \ \textbf{etc}.$

New syntactic forms:

New Evaluation Rules:



 $\label{lem:valid_types} \textbf{Valid} \ \textbf{types} \ \textbf{for inl} \ \ \textbf{5} \ \textbf{are} \ \textbf{Nat} + \textbf{Nat}, \textbf{Nat} + \textbf{Bool} \ \textbf{etc}.$

New syntactic forms:

New Evaluation Rules:

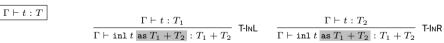
```
\frac{t \longrightarrow t'}{\mathsf{case}\,(\mathsf{inl}\,v_1 \,\mathsf{las}\,T)\mathsf{of}\,\mathsf{inl}\,x_1 \Rightarrow t_1|\mathsf{inr}\,x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_1]t_1} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,x_1 \Rightarrow t_1|\mathsf{inr}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_1 \Rightarrow t_1|\mathsf{inr}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{Inl}\,\mathsf{R} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{E\text{-}Case} \mathsf{Inl}\,x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_2]t_2} \,\,\mathsf{E\text{-}Case} \mathsf{E\text{-}Case} \mathsf{E\text{-}Ca
```



Valid types for inl 5 are Nat + Nat, Nat + Bool etc.

New syntactic forms:

New Typing Rules:

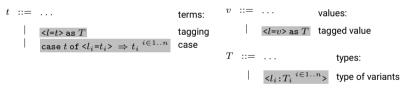




New syntactic forms:

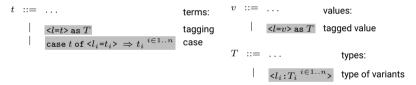


New syntactic forms:





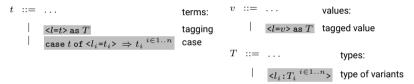
New syntactic forms:



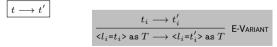
New Evaluation Rules:



New syntactic forms:

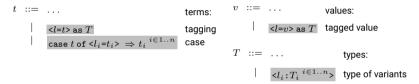


New Evaluation Rules:

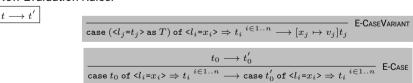




New syntactic forms:

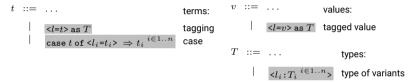


New Evaluation Rules:

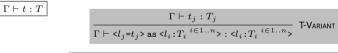




New syntactic forms:



New Typing Rules:



$$\frac{\Gamma \vdash t_0 : <\!\! l_i \!:\! T_i \stackrel{i \in 1 \dots n}{>} \quad \text{for each } i \quad \Gamma, x_i : T_i \vdash t_i : T}{\Gamma \vdash \mathsf{case} \ t_0 \ \mathsf{of} <\!\! l_i \!=\!\! x_i \!>} \Rightarrow t_i \stackrel{i \in 1 \dots n}{=} : T$$
 T-CASE

Examples



STLC++

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).
```