λ-calculus

Rules for evaluation, typing and subtyping

Booleans

Syntax

t	::=		*terms:*
		`true`	*constant true*
		`false`	*constant false*
		`if t then t else t`	*conditional*
v	::=		*values:*
		`true`	*true value*
		`false`	*false value*

Evaluation $t \longrightarrow t'$

$$\texttt{if true then } t_2 \texttt{ else } t_3 \longrightarrow t_2 \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then } t_2 \texttt{ else } t_3 \longrightarrow t_3 \qquad \text{(E-IfFalse)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{if } t_1 \texttt{ then } t_2 \texttt{ else } t_3 \longrightarrow \texttt{if } t_1' \texttt{ then } t_2 \texttt{ else } t_3} \qquad \text{(E-If)}$$

Arithmetic expressions

New syntactic forms

t	::=	...	*terms:*
		`0`	*constant zero*
		`succ t`	*successor*
		`pred t`	*predecessor*
		`iszero t`	*zero test*
v	::=	...	*values:*
		`nv`	*numeric value*
nv	::=		*numeric values:*
		`0`	*zero value*
		`succ nv`	*successor value*

New evaluation rules $t \longrightarrow t'$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{succ } t_1 \longrightarrow \texttt{succ } t_1'} \qquad \text{(E-Succ)}$$

$$\texttt{pred 0} \longrightarrow \texttt{0} \qquad \text{(E-PredZero)}$$

$$\texttt{pred (succ } nv_1\texttt{)} \longrightarrow nv_1 \qquad \text{(E-PredSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{pred } t_1 \longrightarrow \texttt{pred } t_1'} \qquad \text{(E-Pred)}$$

$$\texttt{iszero 0} \longrightarrow \texttt{true} \qquad \text{(E-IszeroZero)}$$

$$\texttt{iszero (succ } nv_1\texttt{)} \longrightarrow \texttt{false} \qquad \text{(E-IszeroSucc)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{iszero } t_1 \longrightarrow \texttt{iszero } t_1'} \qquad \text{(E-IsZero)}$$

Untyped lambda-calculus

Syntax

t	::=		*terms:*
		x	*variable*
		λx.t	*abstraction*
		t t	*application*
v	::=		*values:*
		λx.t	*abstraction value*

Evaluation $t \longrightarrow t'$

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \quad \text{(E-App2)}$$

$$(\lambda x.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-AppAbs)}$$

Typing rules for Booleans

New syntactic forms

T	::=		*types:*
		Bool	*type of booleans*

New typing rules $t : T$

$$\text{true} : \text{Bool} \quad \text{(T-True)}$$

$$\text{false} : \text{Bool} \quad \text{(T-False)}$$

$$\frac{t_1 : \text{Bool} \qquad t_2 : T \qquad t_3 : T}{\text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T} \quad \text{(T-If)}$$

Typing rules for numbers

New syntactic forms

T	::=	...	*types:*
		Nat	*type of natural numbers*

New typing rules $t : T$

$$0 : \text{Nat} \quad \text{(T-Zero)}$$

$$\frac{t_1 : \text{Nat}}{\text{succ } t_1 : \text{Nat}} \quad \text{(T-Succ)}$$

$$\frac{t_1 : \text{Nat}}{\text{pred } t_1 : \text{Nat}} \quad \text{(T-Pred)}$$

$$\frac{t_1 : \text{Nat}}{\text{iszero } t_1 : \text{Bool}} \quad \text{(T-IsZero)}$$

Pure simply typed lambda-calculus

Syntax

t	::=		*terms:*
		x	*variable*
		$\lambda x{:}T.t$	*abstraction*
		$t\ t$	*application*
v	::=		*values:*
		$\lambda x{:}T.t$	*abstraction value*
T	::=		*types:*
		$T \to T$	*type of functions*
Γ	::=		*contexts:*
		$\emptyset$	*empty context*
		$\Gamma, x{:}T$	*term variable binding*

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

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \quad \text{(E-App2)}$$

$$(\lambda x{:}T_{11}.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-AppAbs)}$$

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

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

$$\frac{\Gamma, x{:}T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x{:}T_1.t_2 : T_1 \to T_2} \quad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \quad \text{(T-App)}$$

Unit type

New syntactic forms

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

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

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

New derived forms

$$t_1;t_2 \stackrel{\text{def}}{=} (\lambda x{:}\texttt{Unit}.t_2)\ t_1 \quad \text{where } x \notin FV(t_2)$$

Ascription

New syntactic forms

t	::=	...	*terms:*
		$t\ \texttt{as}\ T$	*ascription*

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

$$v_1\ \texttt{as}\ T \longrightarrow v_1 \quad \text{(E-Ascribe)}$$

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

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

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

Let binding

New syntactic forms

t	::=	...	*terms:*
		`let x=t in t`	*let binding*

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

$$\texttt{let x=}v_1\texttt{ in }t_2 \longrightarrow [x \mapsto v_1]t_2 \quad \text{(E-LetV)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{let x=}t_1\texttt{ in }t_2 \longrightarrow \texttt{let x=}t_1'\texttt{ in }t_2} \quad \text{(E-Let)}$$

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

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma, x{:}T_1 \vdash t_2 : T_2}{\Gamma \vdash \texttt{let x=}t_1\texttt{ in }t_2 : T_2} \quad \text{(T-Let)}$$

Pairs

New syntactic forms

t	::=	...	*terms:*
		`{t,t}`	*pair*
		`t.1`	*first projection*
		`t.2`	*second projection*
v	::=	...	*values:*
		`{v,v}`	*pair value*
T	::=	...	*types:*
		$T_1 \times T_2$	*product type*

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

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

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

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

$$\frac{t_1 \longrightarrow t_1'}{t_1.2 \longrightarrow t_1'.2} \quad \text{(E-Proj2)}$$

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

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

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

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash \{t_1,t_2\} : T_1 \times T_2} \quad \text{(T-Pair)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \times T_{12}}{\Gamma \vdash t_1.1 : T_{11}} \quad \text{(T-Proj1)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \times T_{12}}{\Gamma \vdash t_1.2 : T_{12}} \quad \text{(T-Proj2)}$$

Tuples

New syntactic forms

t	::=	...	*terms:*
		$\{t_{i\ i\in 1..n}\}$	*tuple*
		$t.i$	*projection*
v	::=	...	*values:*
		$\{v_{i\ i\in 1..n}\}$	*tuple value*
T	::=	...	*types:*
		$\{T_{i\ i\in 1..n}\}$	*tuple type*

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

$$\{v_{i\ i\in 1..n}\}.j \longrightarrow v_j \qquad \text{(E-ProjTuple)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.i \longrightarrow t_1'.i} \qquad \text{(E-Proj)}$$

$$\frac{t_j \longrightarrow t_j'}{\{v_{i\ i\in 1..j-1}, t_j, t_{k\ k\in j+1..n}\} \longrightarrow \{v_{i\ i\in 1..j-1}, t_j', t_{k\ k\in j+1..n}\}} \qquad \text{(E-Tuple)}$$

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

$$\frac{\text{for each } i \quad \Gamma \vdash t_i : T_i}{\Gamma \vdash \{t_{i\ i\in 1..n}\} : \{T_{i\ i\in 1..n}\}} \qquad \text{(T-Tuple)}$$

$$\frac{\Gamma \vdash t_1 : \{T_{i\ i\in 1..n}\}}{\Gamma \vdash t_1.j : T_j} \qquad \text{(T-Proj)}$$

Records

New syntactic forms

t	::=	...	*terms:*
		$\{l_i{=}t_{i\ i\in 1..n}\}$	*record*
		$t.l$	*projection*
v	::=	...	*values:*
		$\{l_i{=}v_{i\ i\in 1..n}\}$	*record value*
T	::=	...	*types:*
		$\{l_i{:}T_{i\ i\in 1..n}\}$	*type of records*

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

$$\{l_i{=}v_{i\ i\in 1..n}\}.l_j \longrightarrow v_j \qquad \text{(E-ProjRcd)}$$

$$\frac{t_1 \longrightarrow t_1'}{t_1.l \longrightarrow t_1'.l} \qquad \text{(E-Proj)}$$

$$\frac{t_j \longrightarrow t_j'}{\{l_i{=}v_{i\ i\in 1..j-1}, l_j{=}t_j, l_k{=}t_{k\ k\in j+1..n}\} \longrightarrow \{l_i{=}v_{i\ i\in 1..j-1}, l_j{=}t_j', l_k{=}t_{k\ k\in j+1..n}\}} \qquad \text{(E-Rcd)}$$

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

$$\frac{\text{for each } i \quad \Gamma \vdash t_i : T_i}{\Gamma \vdash \{l_i{=}t_{i\ i\in 1..n}\} : \{l_i{:}T_{i\ i\in 1..n}\}} \qquad \text{(T-Rcd)}$$

$$\frac{\Gamma \vdash t_1 : \{l_i{:}T_{i\ i\in 1..n}\}}{\Gamma \vdash t_1.l_j : T_j} \qquad \text{(T-Proj)}$$

Sums

New syntactic forms

t	::=	...	*terms:*
		`inl t`	*tagging (left)*
		`inr t`	*tagging (right)*
		`case t of inl x⇒t | inr x⇒t`	*case*
v	::=	...	*values:*
		`inl v`	*tagged value (left)*
		`inr v`	*tagged value (right)*
T	::=	...	*types:*
		`T+T`	*sum type*

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

$$\texttt{case (inl } v_0 \texttt{)}\ \texttt{of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_0]t_1 \qquad \text{(E-CaseInl)}$$

$$\texttt{case (inr } v_0 \texttt{)}\ \texttt{of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_0]t_2 \qquad \text{(E-CaseInr)}$$

$$\frac{t_0 \longrightarrow t_0'}{\texttt{case } t_0 \texttt{ of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 \longrightarrow \texttt{case } t_0' \texttt{ of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2} \qquad \text{(E-Case)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inl } t_1 \longrightarrow \texttt{inl } t_1'} \qquad \text{(E-Inl)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inr } t_1 \longrightarrow \texttt{inr } t_1'} \qquad \text{(E-Inr)}$$

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

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash \texttt{inl } t_1 : T_1 + T_2} \qquad \text{(T-Inl)}$$

$$\frac{\Gamma \vdash t_1 : T_2}{\Gamma \vdash \texttt{inr } t_1 : T_1 + T_2} \qquad \text{(T-Inr)}$$

$$\frac{\Gamma \vdash t_0 : T_1 + T_2 \qquad \Gamma, x_1 : T_1 \vdash t_1 : T \qquad \Gamma, x_2 : T_2 \vdash t_2 : T}{\Gamma \vdash \texttt{case } t_0 \texttt{ of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 : T} \qquad \text{(T-Case)}$$

Sums (with unique typing)

New syntactic forms

t	::=	...	*terms:*
		`inl t as T`	*tagging (left)*
		`inr t as T`	*tagging (right)*
v	::=	...	*values:*
		`inl v as T`	*tagged value (left)*
		`inr v as T`	*tagged value (right)*

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

$$\texttt{case (inl } v_0 \texttt{ as } T_0 \texttt{)}\ \texttt{of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 \longrightarrow [x_1 \mapsto v_0]t_1 \qquad \text{(E-CaseInl)}$$

$$\texttt{case (inr } v_0 \texttt{ as } T_0 \texttt{)}\ \texttt{of inl } x_1 \Rightarrow t_1 \mid \texttt{inr } x_2 \Rightarrow t_2 \longrightarrow [x_2 \mapsto v_0]t_2 \qquad \text{(E-CaseInr)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inl } t_1 \texttt{ as } T_2 \longrightarrow \texttt{inl } t_1' \texttt{ as } T_2} \qquad \text{(E-Inl)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{inr } t_1 \texttt{ as } T_2 \longrightarrow \texttt{inr } t_1' \texttt{ as } T_2} \qquad \text{(E-Inr)}$$

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

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash \texttt{inl } t_1 \texttt{ as } T_1 + T_2 : T_1 + T_2} \qquad \text{(T-Inl)}$$

$$\frac{\Gamma \vdash t_1 : T_2}{\Gamma \vdash \texttt{inr } t_1 \texttt{ as } T_1 + T_2 : T_1 + T_2} \qquad \text{(T-Inr)}$$

Variants

New syntactic forms

t	::=	...	*terms:*
		$\langle l{=}t \rangle \text{ as } T$	*tagging*
		$\text{case } t \text{ of } \langle l_i{=}x_i \rangle \Rightarrow t_i \ ^{i \in 1..n}$	*case*
T	::=	...	*types:*
		$\langle l_i : T_i \ ^{i \in 1..n} \rangle$	*type of variants*

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

$$\text{case } (\langle l_j{=}v_j \rangle \text{ as } T) \text{ of } \langle l_i{=}x_i \rangle \Rightarrow t_i \ ^{i \in 1..n} \longrightarrow [x_j \mapsto v_j]t_j$$

(E-CaseVariant)

$$\frac{t_0 \longrightarrow t_0'}{\text{case } t_0 \text{ of } \langle l_i{=}x_i \rangle \Rightarrow t_i \ ^{i \in 1..n} \longrightarrow \text{case } t_0' \text{ of } \langle l_i{=}x_i \rangle \Rightarrow t_i \ ^{i \in 1..n}}$$

(E-Case)

$$\frac{t_i \longrightarrow t_i'}{\langle l_i{=}t_i \rangle \text{ as } T \longrightarrow \langle l_i{=}t_i' \rangle \text{ as } T}$$

(E-Variant)

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

$$\frac{\Gamma \vdash t_j : T_j}{\Gamma \vdash \langle l_j{=}t_j \rangle \text{ as } \langle l_i : T_i \ ^{i \in 1..n} \rangle : \langle l_i : T_i \ ^{i \in 1..n} \rangle}$$

(T-Variant)

$$\frac{\Gamma \vdash t_0 : \langle l_i : T_i \ ^{i \in 1..n} \rangle \quad \text{for each } i \quad \Gamma, x_i : T_i \vdash t_i : T}{\Gamma \vdash \text{case } t_0 \text{ of } \langle l_i{=}x_i \rangle \Rightarrow t_i \ ^{i \in 1..n} : T}$$

(T-Case)

General recursion

New syntactic forms

t	::=	...	*terms:*
		$\text{fix } t$	*fixed point of t*

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

$$\text{fix } (\lambda x{:}T_1.t_2) \longrightarrow [x \mapsto (\text{fix } (\lambda x{:}T_1.t_2))]t_2$$

(E-FixBeta)

$$\frac{t_1 \longrightarrow t_1'}{\text{fix } t_1 \longrightarrow \text{fix } t_1'}$$

(E-Fix)

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

$$\frac{\Gamma \vdash t_1 : T_1 \to T_1}{\Gamma \vdash \text{fix } t_1 : T_1}$$

(T-Fix)

New derived forms

$$\text{letrec } x : T_1 = t_1 \text{ in } t_2 \stackrel{\text{def}}{=} \text{let } x = \text{fix } (\lambda x : T_1 . t_1) \text{ in } t_2$$

Lists

New syntactic forms

t	::=	...	*terms:*
		`nil[T]`	*empty list*
		`cons[T] t t`	*list constructor*
		`isnil[T] t`	*test for empty list*
		`head[T] t`	*head of a list*
		`tail[T] t`	*tail of a list*
v	::=	...	*values:*
		`nil[T]`	*empty list*
		`cons[T] v v`	*list constructor*
T	::=	...	*types:*
		`List T`	*type of lists*

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

$$\frac{t_1 \longrightarrow t_1'}{\texttt{cons[T]}\ t_1\ t_2 \longrightarrow \texttt{cons[T]}\ t_1'\ t_2} \quad \text{(E-Cons1)}$$

$$\frac{t_2 \longrightarrow t_2'}{\texttt{cons[T]}\ v_1\ t_2 \longrightarrow \texttt{cons[T]}\ v_1\ t_2'} \quad \text{(E-Cons2)}$$

$$\texttt{isnil[S] (nil[T])} \longrightarrow \texttt{true} \quad \text{(E-IsnilNil)}$$

$$\texttt{isnil[S] (cons[T]}\ v_1\ v_2\texttt{)} \longrightarrow \texttt{false} \quad \text{(E-IsnilCons)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{isnil[T]}\ t_1 \longrightarrow \texttt{isnil[T]}\ t_1'} \quad \text{(E-Isnil)}$$

$$\texttt{head[S] (cons[T]}\ v_1\ v_2\texttt{)} \longrightarrow v_1 \quad \text{(E-HeadCons)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{head[T]}\ t_1 \longrightarrow \texttt{head[T]}\ t_1'} \quad \text{(E-Head)}$$

$$\texttt{tail[S] (cons[T]}\ v_1\ v_2\texttt{)} \longrightarrow v_2 \quad \text{(E-TailCons)}$$

$$\frac{t_1 \longrightarrow t_1'}{\texttt{tail[T]}\ t_1 \longrightarrow \texttt{tail[T]}\ t_1'} \quad \text{(E-Tail)}$$

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

$$\Gamma \vdash \texttt{nil [}T_1\texttt{]} : \texttt{List}\ T_1 \quad \text{(T-Nil)}$$

$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash t_2 : \texttt{List}\ T_1}{\Gamma \vdash \texttt{cons[}T_1\texttt{]}\ t_1\ t_2 : \texttt{List}\ T_1} \quad \text{(T-Cons)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{List}\ T_{11}}{\Gamma \vdash \texttt{isnil[}T_{11}\texttt{]}\ t_1 : \texttt{Bool}} \quad \text{(T-Isnil)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{List}\ T_{11}}{\Gamma \vdash \texttt{head[}T_{11}\texttt{]}\ t_1 : T_{11}} \quad \text{(T-Head)}$$

$$\frac{\Gamma \vdash t_1 : \texttt{List}\ T_{11}}{\Gamma \vdash \texttt{tail[}T_{11}\texttt{]}\ t_1 : \texttt{List}\ T_{11}} \quad \text{(T-Tail)}$$

Simply typed lambda-calculus with subtyping

Syntax

t	::=		*terms:*
		x	*variable*
		$\lambda x{:}T.t$	*abstraction*
		$t\ t$	*application*
v	::=		*values:*
		$\lambda x{:}T.t$	*abstraction value*
T	::=		*types:*
		Top	*maximum type*
		$T \to T$	*type of functions*
Γ	::=		*contexts:*
		$\varnothing$	*empty context*
		$\Gamma, x{:}T$	*term variable binding*

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

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \quad \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \quad \text{(E-App2)}$$

$$(\lambda x{:}T_{11}.t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12} \quad \text{(E-AppAbs)}$$

Subtyping $\boxed{S <: T}$

$$S <: S \quad \text{(S-Refl)}$$

$$\frac{S <: U \qquad U <: T}{S <: T} \quad \text{(S-Trans)}$$

$$S <: \mathsf{Top} \quad \text{(S-Top)}$$

$$\frac{T_1 <: S_1 \qquad S_2 <: T_2}{S_1 \to S_2 <: T_1 \to T_2} \quad \text{(S-Arrow)}$$

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

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

$$\frac{\Gamma, x{:}T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x{:}T_1.t_2 : T_1 \to T_2} \quad \text{(T-Abs)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \quad \text{(T-App)}$$

$$\frac{\Gamma \vdash t : S \qquad S <: T}{\Gamma \vdash t : T} \quad \text{(T-Sub)}$$

Records and subtyping

New subtyping rules $\boxed{S <: T}$

$$\{l_i{:}T_i\ ^{i \in 1..n+k}\} <: \{l_i{:}T_i\ ^{i \in 1..n}\} \quad \text{(S-RcdWidth)}$$

$$\frac{\text{for each } i \quad S_i <: T_i}{\{l_i{:}S_i\ ^{i \in 1..n}\} <: \{l_i{:}T_i\ ^{i \in 1..n}\}} \quad \text{(S-RcdDepth)}$$

$$\frac{\{k_j{:}S_j\ ^{j \in 1..n}\} \text{ is a permutation of } \{l_i{:}T_i\ ^{i \in 1..n}\}}{\{k_j{:}S_j\ ^{j \in 1..n}\} <: \{l_i{:}T_i\ ^{i \in 1..n}\}} \quad \text{(S-RcdPerm)}$$

Bottom type

New syntactic forms

T	::=	...	*types:*
		Bot	*minimum type*

New subtyping rules $\boxed{S <: T}$

$$\mathsf{Bot} <: T \quad \text{(S-Bot)}$$

Subtyping variants

New syntactic forms

$t ::= \ldots$ — *terms:*

$\langle l = t \rangle$ (no as) — *tagging*

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

$$\frac{\Gamma \vdash t_1 : T_1}{\Gamma \vdash \langle l_1 = t_1 \rangle : \langle l_1 : T_1 \rangle} \quad \text{(T-VARIANT)}$$

New subtyping rules $\boxed{S <: T}$

$$\langle l_i : T_i \; {}^{i \in 1..n} \rangle \;<:\; \langle l_i : T_i \; {}^{i \in 1..n+k} \rangle$$

(S-VARIANTWIDTH)

$$\frac{\text{for each } i \quad S_i <: T_i}{\langle l_i : S_i \; {}^{i \in 1..n} \rangle \;<:\; \langle l_i : T_i \; {}^{i \in 1..n} \rangle}$$

(S-VARIANTDEPTH)

$$\frac{\langle k_j : S_j \; {}^{j \in 1..n} \rangle \text{ is a permutation of } \langle l_i : T_i \; {}^{i \in 1..n} \rangle}{\langle k_j : S_j \; {}^{j \in 1..n} \rangle \;<:\; \langle l_i : T_i \; {}^{i \in 1..n} \rangle}$$

(S-VARIANTPERM)