CIS 500 — Software Foundations

Midterm II

Answer key

November 12, 2003

Simply typed lambda-calculus

*The following questions refer to the simply typed lambda-calculus with booleans and exceptions. The syntax, typing, and evaluation rules for this system are given on page ♠**??** of the companion handout.*

1. (4 points) In this question, you can use `B` as an abbreviation for the type `Bool` while drawing typing derivation trees. For example, you can draw the derivation tree

$$\dfrac{\dfrac{}{\mathtt{x{:}Bool} \vdash \mathtt{true} : \mathtt{Bool}}\ \textsc{T-True}}{\vdash \lambda\mathtt{x{:}Bool.\ true} : \mathtt{Bool}\to\mathtt{Bool}}\ \textsc{T-Abs}$$

as:

$$\dfrac{\dfrac{}{\mathtt{x{:}B} \vdash \mathtt{true} : \mathtt{B}}\ \textsc{T-True}}{\vdash \lambda\mathtt{x{:}B.\ true} : \mathtt{B}\to\mathtt{B}}\ \textsc{T-Abs}$$

Draw the typing derivation tree for the following lambda-term:

$\lambda\mathtt{x{:}Bool}\to\mathtt{Bool}.\ \lambda\mathtt{y{:}Bool}.\ \mathtt{x\ y}$

Answer:

$$\dfrac{\dfrac{\dfrac{\dfrac{x{:}B\to B \in x{:}B\to B,\, y{:}B}{x{:}B\to B,\, y{:}B \vdash x : B\to B}\ \textsc{T-Var} \qquad \dfrac{y{:}B \in x{:}B\to B,\, y{:}B}{x{:}B\to B,\, y{:}B \vdash y : B}\ \textsc{T-Var}}{x{:}B\to B,\, y{:}B \vdash x\ y : B}\ \textsc{T-App}}{x{:}B\to B \vdash \lambda y{:}B.\ x\ y : B\to B}\ \textsc{T-Abs}}{\vdash \lambda x{:}B\to B.\ \lambda y{:}B.\ x\ y : (B\to B)\to B\to B}\ \textsc{T-Abs}$$

Grading scheme: Three points off for deriving the wrong type, one point off for each tiny error.

2. (5 points) Consider the following terms:

```
t = λx:Bool.
      try
        (if x then
           (λy:Bool. true)
         else
           ((λz:____. z) error))
      with
        error

a = t true

b = t false
```

(a) What type must we put on the binder of `z` (in place of ____), in order for the whole term to be well typed? *Answer:* `Bool→Bool`

(b) What is the type of `t`? *Answer:* `Bool→(Bool→Bool)`

(c) What does `t` evaluate to? *Answer: itself*

(d) What does `a` evaluate to? *Answer:* `(λy:Bool. true)`

(e) What does `b` evaluate to? *Answer:* `error`

Grading scheme: Binary

References

The following questions refer to the simply typed lambda-calculus with references (and `Unit`, `Nat`, `Bool`, and `let`). The syntax, typing, and evaluation rules for this system are given on page ♠?? of the companion handout.

3. (9 points) Evaluating the expression

```
let x = ref (λn:Nat. 0) in
let y = ref (λn:Nat. (!x) n) in
let z = ref (λn:Nat. (!y) n) in
(!z) 3
```

beginning in an empty store yields:

Result: 0 Store: $l_1 \mapsto$ `λn:Nat. 0`
$l_2 \mapsto$ `λn:Nat. (!`l_1`) n`
$l_3 \mapsto$ `λn:Nat. (!`l_2`) n`

Fill in the resulting values and final stores (when started with an empty store) for the following terms:

(a)
```
let x = ref 0 in
let y = ref 1 in
let f = λz:Ref Nat. z := succ(!z) in
(f y); (!x)
```
Answer:

Result: 0 *Store:* $l_1 \mapsto$ *0*
$l_2 \mapsto$ *2*

(b)
```
let x = ref 5 in
let y = x in
let z = ref (λa:Nat. y := a; pred (!x)) in
(!z) (!y)
```
Answer:

Result: 4 *Store:* $l_1 \mapsto$ *5*
$l_2 \mapsto$ *λa:Nat. (*l_1 *:= a; pred (!*l_1*))*

(c)
```
let f = ref (λn:Nat. ref 999) in
f := λn:Nat. if iszero(n) then ref 0
                          else ref ( !((!f) (pred n)) );
(!f) 3
```
Answer:

Result: l_5 *Store:* $l_1 \mapsto$ *λn:Nat. if iszero(n) then (ref 0)*
else ref (!((!l_1*)(pred n)))*
$l_2 \mapsto$ *0*
$l_3 \mapsto$ *0*
$l_4 \mapsto$ *0*
$l_5 \mapsto$ *0*

Grading scheme: One point for the result, two points for the store.

4. (10 points)

Recall the following notations from the book:

$\Gamma \mid \Sigma \vdash \mu$ iff $dom(\Sigma) = dom(\mu)$ and $\Gamma \mid \Sigma \vdash \mu(l) : \Sigma(l)$ for every $l \in \mu$.
$\Sigma \subseteq \Sigma'$ iff $dom(\Sigma) \subseteq dom(\Sigma')$ and we have $\Sigma(l) = \Sigma'(l)$ for every $l \in dom(\Sigma)$.

State the preservation theorem for the simply typed lambda-calculus with references.

Answer: If

$\Gamma \mid \Sigma \vdash t : T$
$\Gamma \mid \Sigma \vdash \mu$
$t \mid \mu \longrightarrow t' \mid \mu'$

then, for some $\Sigma' \supseteq \Sigma$,

$\Gamma \mid \Sigma' \vdash t' : T$
$\Gamma \mid \Sigma' \vdash \mu'$.

Grading scheme:

- *Minus two points if* $\Sigma' \supseteq \Sigma$ *is missing.*
- *Minus for points if heap well-formedness* ($\Gamma \mid \Sigma' \vdash \mu$) *is ignored or the heap is not part of the evaluation relation.*
- *Minus two points if they make up notation and do not define it.*
- *Minus one point if they misstate the type of quantification or place the quantifiers in the wrong place.*
- *Minus four points if they are missing the well-formedness of the term.*
- *Minus one points for smaller notational mistakes (forgetting to put "If" and "then" or missing contexts)*
- *Minus for points if there is no mention of the evaluation relation.*

5. (3 points) Suppose we delete the rule T-REF from the definition of the typing relation. How should the statement of the preservation theorem be changed so that it is true for the modified system?

Answer: No change is needed—the theorem as stated is also true for the new system.

However, if we like, we can make the theorem a bit more precise by replacing μ' *with* μ*, since in the new system there is no way to extend the store by allocating new references.*

Grading scheme: Three points for correct answers, one point for "almost correct" answers.

6. (21 points) There are seven mistakes in the following proof of the progress theorem for the simply-typed lambda calculus with references. Circle each mistake and write an appropriate correction beside it. Several cases (T-Var, T-Abs, T-App, etc.) are not shown and do not need correction. The (correct) inversion and canonical forms lemmas are repeated on the next page, for reference.

THEOREM [PROGRESS]: Suppose $\mathtt{t}$ is a closed, well-typed term — that is, $\emptyset \mid \Sigma \vdash \mathtt{t} : \mathtt{T}$ for some $\mathtt{T}$ and Σ. Then either $\mathtt{t}$ is a value or else, for any store μ such that $\emptyset \mid \Sigma \vdash \mu$, there are some term $\mathtt{t}'$ and store μ' with $\mathtt{t} \mid \mu \longrightarrow \mathtt{t}' \mid \mu'$.

PROOF: By induction on the structure of the term $\mathtt{t}$ *[should be: the derivation of $\emptyset \mid \Sigma \vdash t : T$]*:

Case T-UNIT: $\mathtt{t} = \mathtt{unit}$ $\mathtt{T} = \mathtt{Unit}$

Immediate: $\mathtt{unit}$ is a value.

Case T-LOC: $\mathtt{t} = l$ $\mathtt{T} = \Sigma(l)$ *[should be: $T = \mathit{Ref}\ \Sigma(l)$]*

Can't happen because $\mathtt{t}$ is closed. *[should be: Immediate: l is a value.]*

Case T-REF: $\mathtt{t} = \mathtt{ref}\ \mathtt{t}_1$ $\mathtt{T} = \mathtt{Ref}\ \mathtt{T}_1$ $\emptyset \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_1$

By the induction hypothesis, there are two cases to consider:

(a) $\mathtt{t}_1$ is a value, $\mathtt{v}_1$. Then we are done, since $\mathtt{ref}\ \mathtt{v}_1$ is a value. *[should be: Then rule* E-REFV *yields $ref\ v_1 \mid \mu \longrightarrow l \mid (\mu, l \mapsto v_1)$, for some $l \notin dom(\mu)$]*

(b) $\mathtt{t}_1 \longrightarrow \mathtt{t}_1'$ *[should be: $t_1 \mid \mu \longrightarrow t_1' \mid \mu'$]*: The result then follows by E-REF.

Case T-DEREF: $\mathtt{t} = \mathtt{!}\mathtt{t}_1$ $\emptyset \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Ref}\ \mathtt{T}$

By the induction hypothesis, there are two cases to consider:

(a) $\mathtt{t}_1$ is a value, $\mathtt{v}_1$: By the inversion *[should be: canonical forms]* lemma, $\mathtt{v}_1$ must be a location l. But then, by the canonical forms *[should be: inversion]* lemma, $l \in dom(\Sigma)$, and, since $\emptyset \mid \Sigma \vdash \mu$, we have $l \in dom(\mu)$. The result now follows, since $\mathtt{t}$ can make a step by rule E-DEREFLOC.

(b) $\mathtt{t}_1 \mid \mu \longrightarrow \mathtt{t}_1' \mid \mu'$: The result follows by rule E-ASSIGN *[should be:* E-DEREF*]*.

Grading scheme: There are actually eight *mistakes in the proof–we missed one of them when making the question! As a result, we add three bonus points to this question: you only need to identify any seven of the eight mistakes to have the full score (21 points) for the question.*

-3 for missing a mistake; -2 for a wrong reason; and -2 for mis-identifying a correct part as a mistake.

Students should pay attention to details such as when to use IH, what it means by "can t happen," what are the given conditions (e.g. $t_1 \mid \mu \longrightarrow t_1' \mid \mu'$), and what are the implicit equalities ($\Gamma = \emptyset$). In particular, E-DEREFLOC *is applicable only if $\mu(l) = v$, which is true because $\emptyset \mid \Sigma \vdash \mu$.*

(Note: there are no mistakes to find in the following lemmas — they are given here for reference only.)

LEMMA [CANONICAL FORMS]

(a) If $\mathtt{v}$ is a value of type $\mathtt{Bool}$, then $\mathtt{v}$ is either $\mathtt{true}$ or $\mathtt{false}$.
(b) If $\mathtt{v}$ is a value of type $\mathtt{Nat}$, then $\mathtt{v}$ is a numeric value.
(c) If $\mathtt{v}$ is a value of type $\mathtt{Ref}\ \mathtt{T}_1$, then $\mathtt{v}$ is a location $\mathtt{l}$.
(d) If $\mathtt{v}$ is a value of type $\mathtt{T}_1 \to \mathtt{T}_2$, then $\mathtt{v}$ has the form $\lambda\mathtt{x}:\mathtt{T}_1.\mathtt{t}_2$.

LEMMA [INVERSION]

(a) If $\Gamma \mid \Sigma \vdash \mathtt{x} : \mathtt{R}$, then $\mathtt{x}:\mathtt{R} \in \Gamma$.
(b) If $\Gamma \mid \Sigma \vdash \lambda\mathtt{x}:\mathtt{T}_1.\ \mathtt{t}_2 : \mathtt{R}$, then $\mathtt{R} = \mathtt{T}_1 \to \mathtt{R}_2$ for some $\mathtt{R}_2$ with $\Gamma, \mathtt{x}:\mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{R}_2$.
(c) If $\Gamma \mid \Sigma \vdash \mathtt{t}_1\ \mathtt{t}_2 : \mathtt{R}$, then there is some type $\mathtt{T}_{11}$ such that $\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{R}$ and $\Gamma \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$.
(d) If $\Gamma \mid \Sigma \vdash \mathtt{ref}\ \mathtt{t}_1 : \mathtt{R}$, then there is some type $\mathtt{T}_1$ such that $\mathtt{R} = \mathtt{Ref}\ \mathtt{T}_1$ and $\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_1$.
(e) If $\Gamma \mid \Sigma \vdash\ !\mathtt{t}_1 : \mathtt{R}$, then $\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Ref}\ \mathtt{R}$.
(f) If $\Gamma \mid \Sigma \vdash \mathtt{t}_1 := \mathtt{t}_2 : \mathtt{R}$, then $\mathtt{R} = \mathtt{Unit}$ and there is some type $\mathtt{T}_{11}$ such that $\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Ref}\ \mathtt{T}_{11}$ and $\Gamma \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_{11}$.
(g) If $\Gamma \mid \Sigma \vdash \mathtt{l} : \mathtt{R}$, then there is some type $\mathtt{T}_1$ such that $\Sigma(\mathtt{l}) = \mathtt{T}_1$ and $\mathtt{R} = \mathtt{Ref}\ \mathtt{T}_1$.
(h) and similar cases for numbers, booleans, $\mathtt{let}$, $\mathtt{unit}$, ...

Subtyping

The following questions refer to the simply typed lambda-calculus with subtyping, records, and variants. The syntax, typing, and evaluation rules for this system are given on page ♠**??** *of the companion handout.*

7. (12 points) For each of the following pairs of types, write “less” if the type on the left is a subtype of that on the right, “greater” greater if the type on the left is a supertype of the type on the right, “equivalent” if each type is a subtype of the other, or “incomparable” if neither is a subtype of the other.

a)	({}→{}) → Top *Answer: greater*	Top → Top
b)	(Top→Top) → {} → {} *Answer: less*	(Top→{}) → Top
c)	{a:Top, b:{d:Top}, c:Top} *Answer: equivalent*	{b:{d:Top}, a:Top, c:Top}
d)	{g:Top, f:Top} → {f:Top, g:Top} *Answer: incomparable*	{g:Top} → {f:Top}
e)	<l:Top, m:{n:Top}> → {q:Top, p:Top} *Answer: less*	<m:{n:Top, o:Top}> → {p:Top}
f)	<> → Top *Answer: incomparable*	{}→ Top

Grading scheme: 2 points each. Half credit given for “adjacent” answers (e.g., “equivalent” or “incomparable” instead of “less,” etc.)

8. (16 points) Recall the clause of the canonical forms lemma for function types in the simply typed lambda-calculus with subtyping:

   If $\mathtt{v}$ is a closed value of type $\mathtt{T}_1 \rightarrow \mathtt{T}_2$, then $\mathtt{v}$ has the form $\lambda\mathtt{x}:\mathtt{S}_1.\mathtt{t}_2$.

   Give a detailed proof of the above statement. You may make use of the following property of the subtype relation:

   LEMMA [SUBTYPING INVERSION]: If $\mathtt{S} <: \mathtt{T}_1 \rightarrow \mathtt{T}_2$, then $\mathtt{S}$ has the form $\mathtt{S}_1 \rightarrow \mathtt{S}_2$, with $\mathtt{T}_1 <: \mathtt{S}_1$ and $\mathtt{S}_2 <: \mathtt{T}_2$.

   Answer: By induction a derivation of $\vdash \mathtt{v} : \mathtt{T}_1 \rightarrow \mathtt{T}_2$.

   By inspection of the typing rules, the final rule in a derivation of $\vdash \mathtt{v} : \mathtt{T}_1 \rightarrow \mathtt{T}_2$ *must be either* T-ABS *or* T-SUB. *If it is* T-ABS, *then the desired result is immediate from the premise of the rule.*

   Suppose, then, that the last rule is T-SUB. *From the premises of* T-SUB, *we have* $\vdash \mathtt{v} : \mathtt{S}$ *and* $\mathtt{S} <: \mathtt{T}_1 \rightarrow \mathtt{T}_2$. *From the subtyping inversion lemma, we know that* $\mathtt{S}$ *has the form* $\mathtt{S}_1 \rightarrow \mathtt{S}_2$. *The result now follows from the induction hypothesis.*

   Grading scheme:

   - *No points for completely garbled or incomprehensible answers*
   - *-2 for omitting "by induction on typing derivations" (or something similar)*
   - *-10 for omitting the T-Sub case completely*
   - *-6 for using the induction hypothesis and the subtyping inversion lemma in the wrong order in the T-Sub case*
   - *-6 for including cases for subtyping rules in a proof by induction on the typing relation*
   - *-1 for wrongly implying, in the T-Sub case, that the domain type annotation* $\mathtt{S}_1$ *is the same as the* $\mathtt{S}_1$ *appearing in the arrow type in the rule's premise*
   - *-2 for correct proofs with not enough detail provided*
   - *-1 to -4 for various infelicities and confusions*
   - *several people misread the problem and gave proofs of the subtyping inversion lemma (!); these proofs were graded on their own merits, with a maximum score of 6 points.*

Companion handout

Full definitions of the systems used in the exam

Simply typed lambda calculus with exceptions (and Bool)

Syntax

t ::=		*terms*
	error	*run-time error*
	true	*constant true*
	false	*constant false*
	if t then t else t	*conditional*
	x	*variable*
	λx:T.t	*abstraction*
	t t	*application*
v ::=		*values*
	true	*true value*
	false	*false value*
	λx:T.t	*abstraction value*
T ::=		*types*
	T→T	*type of functions*
	Bool	*type of booleans*
Γ ::=		*contexts*
	∅	*empty context*
	Γ, x:T	*term variable binding*

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

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

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

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

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

$$\texttt{error } t_2 \longrightarrow \texttt{error} \qquad \text{(E-AppErr1)}$$

$$v_1 \texttt{ error} \longrightarrow \texttt{error} \qquad \text{(E-AppErr2)}$$

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

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

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

Typing

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

$$\frac{\mathtt{x}{:}\mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{T}} \qquad \text{(T-VAR)}$$

$$\frac{\Gamma, \mathtt{x}{:}\mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2 : \mathtt{T}_1 {\to} \mathtt{T}_2} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} {\to} \mathtt{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma \vdash \mathtt{t}_1\ \mathtt{t}_2 : \mathtt{T}_{12}} \qquad \text{(T-APP)}$$

$$\Gamma \vdash \mathtt{true} : \mathtt{Bool} \qquad \text{(T-TRUE)}$$

$$\Gamma \vdash \mathtt{false} : \mathtt{Bool} \qquad \text{(T-FALSE)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Bool} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T} \qquad \Gamma \vdash \mathtt{t}_3 : \mathtt{T}}{\Gamma \vdash \mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 : \mathtt{T}} \qquad \text{(T-IF)}$$

$$\Gamma \vdash \mathtt{error} : \mathtt{T} \qquad \text{(T-ERROR)}$$

Simply typed lambda calculus with references
(and `Unit`, `Nat`, `Bool`, and `let`)

Syntax

`t ::=`		*terms*
	`x`	*variable*
	`let x=t in t`	*let binding*
	`unit`	*constant* `unit`
	`λx:T.t`	*abstraction*
	`t t`	*application*
	`ref t`	*reference creation*
	`!t`	*dereference*
	`t:=t`	*assignment*
	`l`	*store location*
	`true`	*constant true*
	`false`	*constant false*
	`if t then t else t`	*conditional*
	`0`	*constant zero*
	`succ t`	*successor*
	`pred t`	*predecessor*
	`iszero t`	*zero test*
`v ::=`		*values*
	`unit`	*constant* `unit`
	`λx:T.t`	*abstraction value*
	`l`	*store location*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`T ::=`		*types*
	`Unit`	*unit type*
	`T→T`	*type of functions*
	`Ref T`	*type of reference cells*
	`Bool`	*type of booleans*
	`Nat`	*type of natural numbers*
μ ::=		*stores*
	$\emptyset$	*empty store*
	$\mu, l = v$	*location binding*
Γ ::=		*contexts*
	$\emptyset$	*empty context*
	Γ, `x:T`	*term variable binding*
Σ ::=		*store typings*
	$\emptyset$	*empty store typing*
	Σ, `l:T`	*location typing*

$\texttt{nv} ::=$ *numeric values*

$\texttt{0}$ *zero value*

$\texttt{succ nv}$ *successor value*

Evaluation $\boxed{\texttt{t} \mid \mu \longrightarrow \texttt{t}' \mid \mu'}$

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

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

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{t}_1\ \texttt{t}_2 \mid \mu \longrightarrow \texttt{t}_1'\ \texttt{t}_2 \mid \mu'} \qquad \text{(E-App1)}$$

$$\frac{\texttt{t}_2 \mid \mu \longrightarrow \texttt{t}_2' \mid \mu'}{\texttt{v}_1\ \texttt{t}_2 \mid \mu \longrightarrow \texttt{v}_1\ \texttt{t}_2' \mid \mu'} \qquad \text{(E-App2)}$$

$$(\lambda\texttt{x:T}_{11}\texttt{.t}_{12})\ \texttt{v}_2 \mid \mu \longrightarrow [\texttt{x} \mapsto \texttt{v}_2]\texttt{t}_{12} \mid \mu \qquad \text{(E-AppAbs)}$$

$$\frac{l \notin \mathit{dom}(\mu)}{\texttt{ref v}_1 \mid \mu \longrightarrow l \mid (\mu, l \mapsto \texttt{v}_1)} \qquad \text{(E-RefV)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{ref t}_1 \mid \mu \longrightarrow \texttt{ref t}_1' \mid \mu'} \qquad \text{(E-Ref)}$$

$$\frac{\mu(l) = \texttt{v}}{\texttt{!}l \mid \mu \longrightarrow \texttt{v} \mid \mu} \qquad \text{(E-DerefLoc)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{!t}_1 \mid \mu \longrightarrow \texttt{!t}_1' \mid \mu'} \qquad \text{(E-Deref)}$$

$$l\texttt{:=v}_2 \mid \mu \longrightarrow \texttt{unit} \mid [l \mapsto \texttt{v}_2]\mu \qquad \text{(E-Assign)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{t}_1\texttt{:=t}_2 \mid \mu \longrightarrow \texttt{t}_1'\texttt{:=t}_2 \mid \mu'} \qquad \text{(E-Assign1)}$$

$$\frac{\texttt{t}_2 \mid \mu \longrightarrow \texttt{t}_2' \mid \mu'}{\texttt{v}_1\texttt{:=t}_2 \mid \mu \longrightarrow \texttt{v}_1\texttt{:=t}_2' \mid \mu'} \qquad \text{(E-Assign2)}$$

$$\texttt{if true then t}_2 \texttt{ else t}_3 \mid \mu \longrightarrow \texttt{t}_2 \mid \mu \qquad \text{(E-IfTrue)}$$

$$\texttt{if false then t}_2 \texttt{ else t}_3 \mid \mu \longrightarrow \texttt{t}_3 \mid \mu \qquad \text{(E-IfFalse)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{if t}_1 \texttt{ then t}_2 \texttt{ else t}_3 \mid \mu \longrightarrow \texttt{if t}_1' \texttt{ then t}_2 \texttt{ else t}_3 \mid \mu'} \qquad \text{(E-If)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{succ t}_1 \mid \mu \longrightarrow \texttt{succ t}_1' \mid \mu'} \qquad \text{(E-Succ)}$$

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

$$\texttt{pred (succ nv}_1\texttt{)} \mid \mu \longrightarrow \texttt{nv}_1 \mid \mu \qquad \text{(E-PredSucc)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{pred t}_1 \mid \mu \longrightarrow \texttt{pred t}_1' \mid \mu} \qquad \text{(E-Pred)}$$

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

$$\texttt{iszero}\ (\texttt{succ}\ \texttt{nv}_1) \mid \mu \longrightarrow \texttt{false} \mid \mu \qquad \text{(E-IszeroSucc)}$$

$$\frac{\texttt{t}_1 \mid \mu \longrightarrow \texttt{t}_1' \mid \mu'}{\texttt{iszero}\ \texttt{t}_1 \mid \mu \longrightarrow \texttt{iszero}\ \texttt{t}_1' \mid \mu'} \qquad \text{(E-IsZero)}$$

Typing $\boxed{\Gamma \mid \Sigma \vdash \texttt{t} : \texttt{T}}$

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

$$\frac{\texttt{x}{:}\texttt{T} \in \Gamma}{\Gamma \mid \Sigma \vdash \texttt{x} : \texttt{T}} \qquad \text{(T-Var)}$$

$$\frac{\Gamma, \texttt{x}{:}\texttt{T}_1 \mid \Sigma \vdash \texttt{t}_2 : \texttt{T}_2}{\Gamma \mid \Sigma \vdash \lambda \texttt{x}{:}\texttt{T}_1.\texttt{t}_2 : \texttt{T}_1 {\to} \texttt{T}_2} \qquad \text{(T-Abs)}$$

$$\frac{\Gamma \mid \Sigma \vdash \texttt{t}_1 : \texttt{T}_{11} {\to} \texttt{T}_{12} \qquad \Gamma \mid \Sigma \vdash \texttt{t}_2 : \texttt{T}_{11}}{\Gamma \mid \Sigma \vdash \texttt{t}_1\ \texttt{t}_2 : \texttt{T}_{12}} \qquad \text{(T-App)}$$

$$\frac{\Sigma(l) = \texttt{T}_1}{\Gamma \mid \Sigma \vdash l : \texttt{Ref}\ \texttt{T}_1} \qquad \text{(T-Loc)}$$

$$\frac{\Gamma \mid \Sigma \vdash \texttt{t}_1 : \texttt{T}_1}{\Gamma \mid \Sigma \vdash \texttt{ref}\ \texttt{t}_1 : \texttt{Ref}\ \texttt{T}_1} \qquad \text{(T-Ref)}$$

$$\frac{\Gamma \mid \Sigma \vdash \texttt{t}_1 : \texttt{Ref}\ \texttt{T}_{11}}{\Gamma \mid \Sigma \vdash \texttt{!}\texttt{t}_1 : \texttt{T}_{11}} \qquad \text{(T-Deref)}$$

$$\frac{\Gamma \mid \Sigma \vdash \texttt{t}_1 : \texttt{Ref}\ \texttt{T}_{11} \qquad \Gamma \mid \Sigma \vdash \texttt{t}_2 : \texttt{T}_{11}}{\Gamma \mid \Sigma \vdash \texttt{t}_1 \texttt{:=} \texttt{t}_2 : \texttt{Unit}} \qquad \text{(T-Assign)}$$

$$\Gamma \vdash \texttt{true} : \texttt{Bool} \qquad \text{(T-True)}$$

$$\Gamma \vdash \texttt{false} : \texttt{Bool} \qquad \text{(T-False)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Bool} \qquad \Gamma \vdash \texttt{t}_2 : \texttt{T} \qquad \Gamma \vdash \texttt{t}_3 : \texttt{T}}{\Gamma \vdash \texttt{if}\ \texttt{t}_1\ \texttt{then}\ \texttt{t}_2\ \texttt{else}\ \texttt{t}_3 : \texttt{T}} \qquad \text{(T-If)}$$

$$\Gamma \vdash 0 : \texttt{Nat} \qquad \text{(T-Zero)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Nat}}{\Gamma \vdash \texttt{succ}\ \texttt{t}_1 : \texttt{Nat}} \qquad \text{(T-Succ)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Nat}}{\Gamma \vdash \texttt{pred}\ \texttt{t}_1 : \texttt{Nat}} \qquad \text{(T-Pred)}$$

$$\frac{\Gamma \vdash \texttt{t}_1 : \texttt{Nat}}{\Gamma \vdash \texttt{iszero}\ \texttt{t}_1 : \texttt{Bool}} \qquad \text{(T-IsZero)}$$

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

Simply typed lambda calculus with subtyping (and records and variants)

Syntax

t	::=		*terms*
		x	*variable*
		λx:T.t	*abstraction*
		t t	*application*
		$\{l_i = t_i\ ^{i \in 1..n}\}$	*record*
		t.l	*projection*
		<l=t>	*tagging*
		case t of $<l_i = x_i> \Rightarrow t_i\ ^{i \in 1..n}$	*case*
v	::=		*values*
		λx:T.t	*abstraction value*
		$\{l_i = v_i\ ^{i \in 1..n}\}$	*record value*
		<l=v>	*tagging*
T	::=		*types*
		$\{l_i : T_i\ ^{i \in 1..n}\}$	*type of records*
		Top	*maximum type*
		T$\to$T	*type of functions*
		$<l_i : T_i\ ^{i \in 1..n}>$	*type of variants*
Γ	::=		*contexts*
		$\emptyset$	*empty context*
		Γ, x:T	*term variable binding*

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

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2} \qquad \text{(E-APP1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'} \qquad \text{(E-APP2)}$$

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

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

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

$$\frac{t_j \longrightarrow t_j'}{\begin{array}{c}\{l_i = v_i\ ^{i \in 1..j-1}, l_j = t_j, l_k = t_k\ ^{k \in j+1..n}\} \\ \longrightarrow \{l_i = v_i\ ^{i \in 1..j-1}, l_j = t_j', l_k = t_k\ ^{k \in j+1..n}\}\end{array}} \qquad \text{(E-RCD)}$$

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

$$\frac{t_0 \longrightarrow t_0'}{\text{case } t_0 \text{ of } <l_i = x_i> \Rightarrow t_i\ ^{i \in 1..n} \longrightarrow \text{case } t_0' \text{ of } <l_i = x_i> \Rightarrow t_i\ ^{i \in 1..n}} \qquad \text{(E-CASE)}$$

$$\frac{t_i \longrightarrow t_i'}{<l_i = t_i> \longrightarrow <l_i = t_i'>} \qquad \text{(E-VARIANT)}$$

Subtyping $\boxed{\mathtt{S <: T}}$

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

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

$$\mathtt{S <: Top} \qquad \text{(S-Top)}$$

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

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

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

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

$$<\mathtt{l_i{:}T_i}^{\ i \in 1..n}> \ <: \ <\mathtt{l_i{:}T_i}^{\ i \in 1..n+k}> \qquad \text{(S-VariantWidth)}$$

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

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

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

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

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

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

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

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

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

$$\frac{\Gamma \vdash \mathtt{t_1 : T_1}}{\Gamma \vdash <\mathtt{l_1{=}t_1}> : <\mathtt{l_1{:}T_1}>} \qquad \text{(T-Variant)}$$

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