CIS 500 — Software Foundations
Midterm II

Answer key

November 8, 2006

Instructions

- This is a closed-book exam.
- You have 80 minutes to answer all of the questions. The entire exam is worth 80 points for students in section 002 and 90 points for students in section 001 (there is one PhD-section-only problem).
- Questions vary significantly in difficulty, and the point value of a given question is not always exactly proportional to its difficulty. Do not spend too much time on any one question.
- Partial credit will be given. All correct answers are short. The back side of each page and the companion handout may be used as scratch paper.
- Good luck!

References

The following problems concern the simply typed lambda calculus with references. This system is summarized on page 4 of the companion handout.

1. (5 points) Give a well-typed term whose evaluation (beginning in the empty store) will produce the following store when evaluation terminates:

$$(l_1 \mapsto \lambda\texttt{x:Nat. (!}l_2\texttt{) x}, \\ l_2 \mapsto \lambda\texttt{x:Nat. (!}l_1\texttt{) x})$$

   Answer:

```
let a = ref (λx:Nat. x) in
let b = ref (λx:Nat. (!a) x) in
  a := (λx:Nat. (!b) x)
```

   Grading scheme: -1 or -2 for small errors: for example, a small fragment was not properly typed. -3 or -4 for more major errors: for example, allocating too many locations or badly mangled syntax for your program.

2. (8 points)

   (a) Give a well-typed term whose evaluation (beginning in the empty store) will produce the following store when evaluation terminates.

$$\mu = (l_1 \mapsto \texttt{5}, \\ l_2 \mapsto l_1, \\ l_3 \mapsto l_2)$$

   Answer:

```
let a = ref 5 in
let b = ref a in
let c = ref b in
  unit
```

   (b) Give a store typing Σ corresponding to this store (i.e., such that $\emptyset|\Sigma \vdash \mu$).
   Answer:

$$\Sigma = (l_1 \mapsto \textit{Nat}, \\ l_2 \mapsto \textit{Ref Nat}, \\ l_3 \mapsto \textit{Ref (Ref Nat)})$$

   Grading scheme: Part (a): -1 or -2 for small errors: for example, slightly mangled syntax; -2 or -3 for more major errors: for example, not allocating the right number of locations. Part (b): -1 for getting the type of a given location wrong.

3. (8 points) Is there a well-typed term whose evaluation (beginning in the empty store) will produce the following store when evaluation terminates?

$$\begin{aligned}\mu = (&l_1 \mapsto l_2,\\ &l_2 \mapsto l_3,\\ &l_3 \mapsto l_1)\end{aligned}$$

If so, give it. If not, explain briefly why no such term exists.

Answer: No such term exists. There are two different ways to see this:

- *Suppose it did. Then the preservation theorem would tell us that there is some store typing Σ (extending the empty store typing) with respect to which the above store is well typed. But, in such a store typing, we would have to have $\Sigma(l_1)$ =* `Ref` *$(\Sigma(l_2))$ =* `Ref (Ref` *$(\Sigma(l_3))$*`)` *=* `Ref (Ref (Ref` *$(\Sigma(l_1))$*`)))`*—in other words, the typing assigned to l_1 by Σ would have to include itself as a proper sub-phrase. Since all types are finite in size, this cannot be.*
- *Observe that the very first reference cell allocated by a well-typed program must be to a non-*`Ref` *type, since at that point there are no locations that could be used as the initial value of the cell. Furthermore, the typing rules guarantee that any subsequent assignment to this reference cell must be a value of the same (non-*`Ref`*) type. So it is not possible for a well-typed program to produce a final store that does not contain at least one cell storing something other than a location.*

Grading scheme:

- *Wrong answer: 0*
- *Correct answer but completely wrong justification: 2*
- *Vaguely correct answer... "in the right spirit": 4-5*
- *Correct answers with small errors: 6-8*

Exceptions

This problem concerns the simply typed lambda calculus with exceptions carrying numeric values—i.e., the system defined in TAPL Section 14.3, where the "exception type" $\mathtt{T}_{exn}$ is taken to be `Nat`. This system is summarized on page 1 of the companion handout.

4. (9 points) For each of the following terms, first check whether the term is well typed. If it is, write its type (if the term has multiple types, pick any one of them) and give the final result of evaluating the term (which will be either a value or `raise nv` for some numeric value `nv`). If it is not, write *ill typed*.

(a)
```
raise (if raise 1 then raise 2 else raise 3)
```
Answer: We can give this term any type. It evaluates to `raise 1`*.*

(b)
```
try
  succ (raise 4)
with
  (λx:Nat. true)
```
Answer: ill-typed

(c)
```
(try
   (λx:Nat. raise 5)
 with
   (λx:Nat. x))
6
```
Answer: ill-typed

Grading scheme: 3 points for each part. The first part gets 1 point if the result is correct and 2 if the type is correct.

Subtyping

The following problems concern the simply typed lambda calculus with subtyping (and records and variants). This system is summarized on page 7 of the companion handout.

5. (8 points) Draw a derivation tree for the following subtyping statement:

$$\texttt{\{a:Top, b:\{\}}\to\texttt{\{\}, c:\{x:Nat\}\}} \;<:\; \texttt{\{b:\{\}}\to\texttt{Top, c:\{\}\}}$$

Answer:

$$\frac{D_1 \quad D_3}{\{a{:}Top,\ b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\} \;<:\; \{b{:}\{\}\to Top,\ c{:}\{\}\}}\ \text{S-Trans}$$

where derivation D_1 is

$$\frac{\dfrac{}{\{a{:}Top,\ b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\} \;<:\; \{b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\},\ a{:}Top\}}\ \text{S-RcdPerm} \qquad D_2}{\{a{:}Top,\ b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\} \;<:\; \{b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\}}\ \text{S-Trans}$$

and where derivation D_2 is

$$\frac{}{\{b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\},\ a{:}Top\} \;<:\; \{b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\}}\ \text{S-RcdWidth}$$

and where derivation D_3 is

$$\frac{\dfrac{\dfrac{}{\{\} <: \{\}}\ \text{S-Refl} \qquad \dfrac{}{\{\} <: Top}\ \text{S-Top}}{\{\}\to\{\} \;<:\; \{\}\to Top}\ \text{S-Arrow} \qquad \dfrac{}{\{x{:}Nat\} <: \{\}}\ \text{S-RcdWidth}}{\{b{:}\{\}\to\{\},\ c{:}\{x{:}Nat\}\} \;<:\; \{b{:}\{\}\to Top,\ c{:}\{\}\}}\ \text{S-RcdDepth}$$

Grading scheme: (Approximately) one point off for each missing/misused rule.

6. (20 points)

Recall the following properties of the simply typed lambda-calculus with subtyping:

- *Progress*: If $\vdash \texttt{t : T}$, then either $\texttt{t}$ is a value or else $\texttt{t} \longrightarrow \texttt{t}'$ for some $\texttt{t}'$.
- *Preservation*: If $\Gamma \vdash \texttt{t : T}$ and $\texttt{t} \longrightarrow \texttt{t}'$, then $\Gamma \vdash \texttt{t}' \texttt{ : T}$.

Each part of this exercise suggests a different way of changing the language. (These changes are not cumulative: each part starts from the original language.) In each part, indicate (by circling TRUE or FALSE) whether each property remains true or becomes false after the suggested change. If a property becomes false, give a counterexample.

(a) Suppose we add the following typing rule:

$$\frac{\Gamma \vdash \texttt{t} : \texttt{S}_1 \rightarrow \texttt{S}_2 \qquad \texttt{S}_1 <: \texttt{S}_2 \qquad \texttt{S}_2 <: \texttt{S}_1 \qquad \texttt{S}_2 <: \texttt{T}_2}{\Gamma \vdash \texttt{t} : \texttt{T}_1 \rightarrow \texttt{T}_2}$$

Progress: *Answer: True*

Preservation: *Answer: True*

(b) Suppose we add the following evaluation rule:

$$\texttt{\{\}} \longrightarrow (\lambda \texttt{x:Top. x})$$

Progress: *Answer: True*

Preservation: *Answer: False: for example, {} has type {} but steps to $(\lambda$x:Top. x), which does not have type {}.*

(c) Suppose we add the following subtyping rule:

<> <: {}

Progress: *Answer: True*
Preservation: *Answer: True*

(d) Suppose we add the following subtyping rule:

{} <: <>

Progress: *Answer: False: for example,* case {} of <foo=x> ⇒ x *is well typed but stuck.*
Preservation: *Answer: True*

Grading scheme: 4 points for the first and third parts, 6 for the second and fourth, evenly divided between progress and preservation.

7. (22 points) Fill in the missing steps in the proof of the subtyping inversion lemma for arrow types from Chapter 15. Your wording does not need to exactly match what is in the book or lecture notes, but every step required in the proof (use of an assumption, use of the induction hypothesis, or use of a subtyping rule) must be mentioned explicitly.

   Lemma: If $S <: T_1 \to T_2$, then S has the form $S_1 \to S_2$, with $T_1 <: S_1$ and $S_2 <: T_2$.

   Proof: By induction on subtyping derivations. By inspection of the subtyping rules, it is clear that the final rule in the derivation of $S <: T_1 \to T_2$ must be S-Refl, S-Trans, or S-Arrow.

   Case S-Refl: $S = T_1 \to T_2$

   Answer: Both $T_1 <: T_1$ and $T_2 <: T_2$ follow by reflexivity.

   Case S-Trans: $S <: U$ $\quad U <: T_1 \to T_2$

   Answer: If the final rule is S-Trans*, then we have subderivations with conclusions $S <: U$ and $U <: T_1 \to T_2$ for some type U. Applying the induction hypothesis to the second subderivation, we see that U has the form $U_1 \to U_2$, with $T_1 <: U_1$ and $U_2 <: T_2$. Now, since we know that U is an arrow type, we can apply the induction hypothesis to the first subderivation to obtain $S = S_1 \to S_2$ with $U_1 <: S_1$ and $S_2 <: U_2$. Finally, we can use* S-Trans *twice to reassemble the facts we have established, obtaining $T_1 <: S_1$ (from $T_1 <: U_1$ and $U_1 <: S_1$) and $S_2 <: T_2$ (from $S_2 <: U_2$ and $U_2 <: T_2$).*

   Case S-Arrow: $S = S_1 \to S_2$ $\quad T_1 <: S_1$ $\quad S_2 <: T_2$

   Answer: Immediate.

   Grading scheme: 6 points for reflexivity and arrow cases; 10 for transitivity case. 1 or 2 points off for extraneous stuff; 4 for putting the two uses of the IH in the wrong order; 2 for generally correct answers but mangled explanations; 4 or 5 for more serious errors.

8. (10 points) **(For students in the PhD section only.)**

Section 15.5 in the book discusses two ways of combining subtyping with references. The first uses just the `Ref` type constructor, with a simple subtyping rule:

$$\frac{\texttt{S}_1 <: \texttt{T}_1 \qquad \texttt{T}_1 <: \texttt{S}_1}{\texttt{Ref S}_1 <: \texttt{Ref T}_1} \qquad \text{(S-REF)}$$

The second, more refined, treatment introduces two new type constructors, `Source` and `Sink`—intuitively, `Source T` is thought of as a capability to read values of type `T` from a cell (but which does not permit assignment), while `Sink T` is a capability to write to a cell. `Ref T` is intuitively a combination of these two capabilities, giving permission both to read and to write.

The typing rule for reference creation returns a `Ref` (it is unchanged from Chapter 13), while the rules for dereferencing and assignment are changed to demand only the appropriate capability. The details of these rules are not important for this question, but they are reproduced, for reference, on 9 of the companion handout.

The subtyping relation is extended with a rules stating that the `Source` constructor is contravariant, the `Sink` constructor is covariant, and the `Ref` constructor can be promoted to either `Source` or `Sink`.

$$\frac{\texttt{S}_1 <: \texttt{T}_1}{\texttt{Source S}_1 <: \texttt{Source T}_1} \qquad \text{(S-SOURCE)}$$

$$\frac{\texttt{T}_1 <: \texttt{S}_1}{\texttt{Sink S}_1 <: \texttt{Sink T}_1} \qquad \text{(S-SINK)}$$

$$\texttt{Ref T}_1 <: \texttt{Source T}_1 \qquad \text{(S-REFSOURCE)}$$

$$\texttt{Ref T}_1 <: \texttt{Sink T}_1 \qquad \text{(S-REFSINK)}$$

If we know that `S <: T` and we know something about the shape of `T`, the subtype inversion lemma gives us information about the shape of `S` and the subtype relationships that must hold between the sub-expressions of `S` and `T`. For example, question 7 above asked you to prove the arrow case.

Fill in appropriate statements for the cases of the subtyping inversion lemma for the constructors `Ref`, `Source`, and `Sink`. You do not need to give proofs.

(a) If $\texttt{S} <: \texttt{Ref T}_1$, then *Answer:* $S = \textbf{Ref}\ S_1$ *for some* S_1 *with* $S_1 <: T_1$ *and* $T_1 <: S_1$.

(b) If $\texttt{S} <: \texttt{Source T}_1$, then *Answer: either: (1)* $S = \textbf{Ref}\ S_1$ *for some* S_1 *with* $S_1 <: T_1$*, or (2)* $S = \textbf{Source}\ S_1$ *for some* S_1 *with* $S_1 <: T_1$.

(c) If $\texttt{S} <: \texttt{Sink T}_1$, then *Answer: either: (1)* $S = \textbf{Ref}\ S_1$ *for some* S_1 *with* $T_1 <: S_1$*, or (2)* $S = \textbf{Sink}\ S_1$ *for some* S_1 *with* $T_1 <: S_1$.

Grading scheme: Answers varied considerably, so the grading was basically done on a case by case basis. But here were some common mistakes:

- *Missing* ***Ref*** *cases in parts 2 and 3: -3*
- *Too many constraints in parts 2 and 3: -2*

Companion handout

Full definitions of the systems used in the exam

Simply-typed lambda calculus with error handling (and numbers and booleans), using Nat as the T_{exn} type

Syntax

t ::=		*terms*
	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*
	x	*variable*
	λx:T.t	*abstraction*
	t t	*application*
	raise t	*raise exception*
	try t with t	*handle exceptions*
v ::=		*values*
	true	*true value*
	false	*false value*
	nv	*numeric value*
	λx:T.t	*abstraction value*
nv ::=		*numeric values*
	0	*zero value*
	succ nv	*successor value*
T ::=		*types*
	Bool	*type of booleans*
	Nat	*type of natural numbers*
	T→T	*type of functions*
Γ ::=		*type environments*
	∅	*empty type env.*

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\texttt{(raise v}_{11}\texttt{)}\ \texttt{t}_2 \longrightarrow \texttt{raise v}_{11} \qquad \text{(E-AppRaise1)}$$

$$\texttt{v}_1\ \texttt{(raise v}_{21}\texttt{)} \longrightarrow \texttt{raise v}_{21} \qquad \text{(E-AppRaise2)}$$

$$\frac{\texttt{t}_1 \longrightarrow \texttt{t}_1'}{\texttt{raise t}_1 \longrightarrow \texttt{raise t}_1'} \qquad \text{(E-Raise)}$$

$$\texttt{raise (raise v}_{11}\texttt{)} \longrightarrow \texttt{raise v}_{11} \qquad \text{(E-RaiseRaise)}$$

$$\texttt{if raise v}_{11}\texttt{ then t}_2\texttt{ else t}_3 \longrightarrow \texttt{raise v}_{11} \qquad \text{(E-IfRaise)}$$

$$\texttt{try v}_1\texttt{ with t}_2 \longrightarrow \texttt{v}_1 \qquad \text{(E-TryV)}$$

$$\begin{array}{c}\texttt{try raise v}_{11}\texttt{ with t}_2 \\ \longrightarrow \texttt{t}_2\ \texttt{v}_{11}\end{array} \qquad \text{(E-TryRaise)}$$

$$\frac{\texttt{t}_1 \longrightarrow \texttt{t}_1'}{\texttt{try t}_1\texttt{ with t}_2 \longrightarrow \texttt{try t}_1'\texttt{ with t}_2} \qquad \text{(E-Try)}$$

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

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

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

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

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

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

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

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

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

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

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

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{Nat}}{\Gamma \vdash \mathtt{raise}\ \mathtt{t}_1 : \mathtt{T}} \qquad \text{(T-Exn)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{Nat} \to \mathtt{T}}{\Gamma \vdash \mathtt{try}\ \mathtt{t}_1\ \mathtt{with}\ \mathtt{t}_2 : \mathtt{T}} \qquad \text{(T-Try)}$$

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

Syntax

`t` ::=		*terms*
	`unit`	*constant* `unit`
	`x`	*variable*
	λ`x: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*
	λ`x:T.t`	*abstraction value*
	l	*store location*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
`T` ::=		*types*
	`Unit`	*unit type*
	`T`$\to$`T`	*type of functions*
	`Ref T`	*type of reference cells*
	`Bool`	*type of booleans*
	`Nat`	*type of natural numbers*
μ ::=		*stores*
	$\emptyset$	*empty store*
	$\mu, l = \mathtt{v}$	*location binding*
Γ ::=		*type environments*
	$\emptyset$	*empty type env.*
	Γ, `x:T`	*term variable binding*
Σ ::=		*store typings*
	$\emptyset$	*empty store typing*
	Σ, l`:T`	*location typing*
`nv` ::=		*numeric values*
	`0`	*zero value*
	`succ nv`	*successor value*

Evaluation $\boxed{\mathtt{t}|\mu \longrightarrow \mathtt{t}'|\mu'}$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\mathtt{pred}\ \mathtt{0}|\mu \longrightarrow \mathtt{0}|\mu \qquad \text{(E-PredZero)}$$

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Syntax

$\texttt{t} ::=$		*terms*
	$\texttt{x}$	*variable*
	$\lambda\texttt{x:T.t}$	*abstraction*
	$\texttt{t t}$	*application*
	$\texttt{\{l}_i\texttt{=t}_i\ ^{i \in 1..n}\texttt{\}}$	*record*
	$\texttt{t.l}$	*projection*
	$\texttt{<l=t>}$ (no $\texttt{as}$)	*tagging*
	$\texttt{case t of <l}_i\texttt{=x}_i\texttt{>}\Rightarrow\texttt{t}_i\ ^{i \in 1..n}$	*case*
$\texttt{v} ::=$		*values*
	$\lambda\texttt{x:T.t}$	*abstraction value*
	$\texttt{\{l}_i\texttt{=v}_i\ ^{i \in 1..n}\texttt{\}}$	*record value*
$\texttt{T} ::=$		*types*
	$\texttt{\{l}_i\texttt{:T}_i\ ^{i \in 1..n}\texttt{\}}$	*type of records*
	$\texttt{Top}$	*maximum type*
	$\texttt{T}\to\texttt{T}$	*type of functions*
	$\texttt{<l}_i\texttt{:T}_i\ ^{i \in 1..n}\texttt{>}$	*type of variants*
$\Gamma ::=$		*type environments*
	$\emptyset$	*empty type env.*

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

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

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

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

$$\texttt{\{l}_i\texttt{=v}_i\ ^{i \in 1..n}\texttt{\}.l}_j \longrightarrow \texttt{v}_j \quad \text{(E-ProjRcd)}$$

$$\texttt{case (<l}_j\texttt{=v}_j\texttt{> as T) of <l}_i\texttt{=x}_i\texttt{>}\Rightarrow\texttt{t}_i\ ^{i \in 1..n} \longrightarrow [\texttt{x}_j \mapsto \texttt{v}_j]\texttt{t}_j \quad \text{(E-CaseVariant)}$$

$$\frac{\texttt{t}_0 \longrightarrow \texttt{t}_0'}{\texttt{case t}_0\texttt{ of <l}_i\texttt{=x}_i\texttt{>}\Rightarrow\texttt{t}_i\ ^{i \in 1..n} \longrightarrow \texttt{case t}_0'\texttt{ of <l}_i\texttt{=x}_i\texttt{>}\Rightarrow\texttt{t}_i\ ^{i \in 1..n}} \quad \text{(E-Case)}$$

$$\frac{\texttt{t}_i \longrightarrow \texttt{t}_i'}{\texttt{<l}_i\texttt{=t}_i\texttt{> as T} \longrightarrow \texttt{<l}_i\texttt{=t}_i'\texttt{> as T}} \quad \text{(E-Variant)}$$

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

$$\frac{\text{for each } i \quad \Gamma \vdash \texttt{t}_i : \texttt{T}_i}{\Gamma \vdash \texttt{\{l}_i\texttt{=t}_i\ ^{i \in 1..n}\texttt{\}} : \texttt{\{l}_i\texttt{:T}_i\ ^{i \in 1..n}\texttt{\}}} \quad \text{(T-Rcd)}$$

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

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

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

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

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

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_1}{\Gamma \vdash \langle \mathtt{l}_1\mathtt{=t}_1 \rangle : \langle \mathtt{l}_1\mathtt{:T}_1 \rangle} \qquad \text{(T-Variant)}$$

$$\frac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n} \rangle \\ \text{for each } i \quad \Gamma, \mathtt{x}_i\mathtt{:T}_i \vdash \mathtt{t}_i : \mathtt{T}\end{array}}{\Gamma \vdash \mathtt{case}\ \mathtt{t}_0\ \mathtt{of}\ \langle \mathtt{l}_i\mathtt{=x}_i \rangle \Rightarrow \mathtt{t}_i{}^{\,i\in 1..n} : \mathtt{T}} \qquad \text{(T-Case)}$$

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

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

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

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

$$\frac{\mathtt{T}_1 <: \mathtt{S}_1 \qquad \mathtt{S}_2 <: \mathtt{T}_2}{\mathtt{S}_1 \to \mathtt{S}_2 <: \mathtt{T}_1 \to \mathtt{T}_2} \qquad \text{(S-Arrow)}$$

$$\{\mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n+k}\} <: \{\mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n}\} \qquad \text{(S-RcdWidth)}$$

$$\frac{\text{for each } i \quad \mathtt{S}_i <: \mathtt{T}_i}{\{\mathtt{l}_i\mathtt{:S}_i{}^{\,i\in 1..n}\} <: \{\mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n}\}} \qquad \text{(S-RcdDepth)}$$

$$\frac{\{\mathtt{k}_j\mathtt{:S}_j{}^{\,j\in 1..n}\} \text{ is a permutation of } \{\mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n}\}}{\{\mathtt{k}_j\mathtt{:S}_j{}^{\,j\in 1..n}\} <: \{\mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n}\}} \qquad \text{(S-RcdPerm)}$$

$$\langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n} \rangle \quad <: \quad \langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n+k} \rangle \qquad \text{(S-VariantWidth)}$$

$$\frac{\text{for each } i \quad \mathtt{S}_i <: \mathtt{T}_i}{\langle \mathtt{l}_i\mathtt{:S}_i{}^{\,i\in 1..n} \rangle \quad <: \quad \langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n} \rangle} \qquad \text{(S-VariantDepth)}$$

$$\frac{\langle \mathtt{k}_j\mathtt{:S}_j{}^{\,j\in 1..n} \rangle \text{ is a permutation of } \langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n} \rangle}{\langle \mathtt{k}_j\mathtt{:S}_j{}^{\,j\in 1..n} \rangle \quad <: \quad \langle \mathtt{l}_i\mathtt{:T}_i{}^{\,i\in 1..n} \rangle} \qquad \text{(S-VariantPerm)}$$

Typing rules for `Source`, `Sink`, and `Ref` constructors

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

$$\frac{\Gamma|\Sigma \vdash \mathtt{t}_1 : \mathtt{Source}\ \mathtt{T}_{11}}{\Gamma|\Sigma \vdash\ !\mathtt{t}_1 : \mathtt{T}_{11}} \qquad \text{(T-Deref)}$$

$$\frac{\Gamma|\Sigma \vdash \mathtt{t}_1 : \mathtt{Sink}\ \mathtt{T}_{11} \qquad \Gamma|\Sigma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma|\Sigma \vdash \mathtt{t}_1 \mathtt{:=} \mathtt{t}_2 : \mathtt{Unit}} \qquad \text{(T-Assign)}$$