CIS 500 — Software Foundations

Final Exam

Answer key

December 20, 2004

True/False questions

For each of the following statements, circle T if the sentence is true or F otherwise.

1. (10 points)

   (a) $\boxed{\text{T}}$ F The untyped lambda calculus can encode any computable function on the natural numbers.

   (b) T $\boxed{\text{F}}$ The simply-typed lambda calculus (not including fix) can encode any computable function on the natural numbers.

   (c) $\boxed{\text{T}}$ F The simply-typed lambda calculus (not including fix) with references can encode any computable function on the natural numbers.

   (d) $\boxed{\text{T}}$ F Featherweight Java can encode any computable function on the natural numbers.

   (e) $\boxed{\text{T}}$ F If the preservation theorem is true for a language, removing a typing rule may cause it to become untrue.

   (f) T $\boxed{\text{F}}$ If the progress theorem is true for a language, removing a typing rule may cause it to become untrue.

   (g) T $\boxed{\text{F}}$ The only way to prove that the preservation theorem holds for a language is by induction on the structure of the typing derivation.

   (h) $\boxed{\text{T}}$ F For a given syntax and operational semantics, it is always possible to devise a set of typing rules such that the preservation and progress theorems hold.

   (i) $\boxed{\text{T}}$ F Featherweight Java has the uniqueness of types property.

   (j) T $\boxed{\text{F}}$ The evaluation relation must be deterministic (i.e. for any term there should be only way to evaluate it) to prove the progress theorem.

   Grading scheme: Binary. 1 pt each.

Untyped lambda-calculus

The following questions refer to the untyped lambda-calculus. The syntax and evaluation rules for this system are given on page 1 of the companion handout.

2. (12 points)

Consider the following definition of the multi-step evaluation relation, $\mathtt{t} \longrightarrow^{*} \mathtt{t}'$:

$$\mathtt{t} \longrightarrow^{*} \mathtt{t} \qquad \text{(EV-DONE)}$$

$$\frac{\mathtt{t} \longrightarrow \mathtt{t}' \qquad \mathtt{t}' \longrightarrow^{*} \mathtt{t}''}{\mathtt{t} \longrightarrow^{*} \mathtt{t}''} \qquad \text{(EV-STEP)}$$

(a) Is the multi-step evaluation relation a partial function? In other words, for any $\mathtt{t}$ does there exist *at most* one $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow^{*} \mathtt{t}'$? If yes, briefly say why. If no, give a counterexample.

Answer: No, because $(\lambda x.x)(\lambda y.y) \longrightarrow^{*} (\lambda x.x)$ *and* $(\lambda x.x)(\lambda y.y) \longrightarrow^{*} (\lambda x.x)(\lambda y.y)$

Grading scheme: 3 points. 1 point partial credit for answering no, but providing a wrong counter-example.

(b) For any $\mathtt{t}$, does there exist *at least* one $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow^{*} \mathtt{t}'$? If yes, briefly say why. If no, give a counterexample.

Answer: Yes, because the relation is reflexive. Grading scheme: 3 points. 1 point partial credit for answering yes, but providing the wrong reason.

(c) Show that the multi-step evaluation relation is transitive. In other words, prove that if $\mathtt{t} \longrightarrow^{*} \mathtt{t}'$ and $\mathtt{t}' \longrightarrow^{*} \mathtt{t}''$ then $\mathtt{t} \longrightarrow^{*} \mathtt{t}''$. Be explicit about each step of the proof, but do not include any irrelevant information.

Answer: Proof is by induction on the structure of the derivation $t \longrightarrow^{*} t'$.

- *Case EV-DONE. In this case,* $t = t'$. *As* $t' \longrightarrow^{*} t''$ *by assumption, then* $t \longrightarrow^{*} t''$.
- *Case EV-STEP. In this case* $t \longrightarrow t_1$ *and* $t_1 \longrightarrow^{*} t'$. *By induction* $t_1 \longrightarrow^{*} t''$. *By EV-STEP,* $t \longrightarrow^{*} t''$.

Grading scheme: 6 points total.

3. (10 points)

The following is yet another encoding of numbers in the untyped lambda calculus.

$\mathtt{s}_0 = \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{z}$

$\mathtt{s}_1 = \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s}\ \mathtt{s}_0$

$\mathtt{s}_2 = \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s}\ \mathtt{s}_1$

$\mathtt{s}_3 = \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s}\ \mathtt{s}_2$

In general, $\mathtt{s}_{n+1} = \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s}\ \mathtt{s}_n$.

Below, circle the correct implementation of the following functions. Some of these implementations use the following definitions from TAPL chapter 5:

$\mathtt{pair} = \lambda\mathtt{f}.\ \lambda\mathtt{s}.\ \lambda\mathtt{b}.\ \mathtt{b\ f\ s}$

$\mathtt{fst} = \lambda\mathtt{p}.\ \mathtt{p}\ (\lambda\mathtt{x}.\ \lambda\mathtt{y}.\ \mathtt{x})$

$\mathtt{snd} = \lambda\mathtt{p}.\ \mathtt{p}\ (\lambda\mathtt{x}.\ \lambda\mathtt{y}.\ \mathtt{y})$

$\mathtt{fix} = \lambda\mathtt{f}.\ (\lambda\mathtt{x}.\ \mathtt{f}\ (\lambda\mathtt{y}.\ \mathtt{x\ x\ y}))\ (\lambda\mathtt{x}.\ \mathtt{f}\ (\lambda\mathtt{y}.\ \mathtt{x\ x\ y}))$

(a) The successor function, where $\mathtt{sscc}\ \mathtt{s}_n = \mathtt{s}_{n+1}$.

i. $\lambda\mathtt{x}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{x\ s\ z}$

ii. $\lambda\mathtt{x}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{x\ z\ s}$

iii. $\lambda\mathtt{x}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s\ x}$

iv. $\lambda\mathtt{x}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s}\ (\mathtt{x\ z\ s})$

v. $\lambda\mathtt{x}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{s\ x}\ (\mathtt{x\ z\ s})$

Answer: iii.

(b) The predecessor function, where $\mathtt{sprd}\ \mathtt{s}_0 = \mathtt{s}_0$ and $\mathtt{sprd}\ \mathtt{s}_{n+1} = \mathtt{s}_n$.

i. $\lambda\mathtt{x}.\ \mathtt{x}\ \mathtt{s}_0\ (\lambda\mathtt{y}.\mathtt{y})$

ii. $\lambda\mathtt{x}.\ \mathtt{x}\ (\lambda\mathtt{y}.\mathtt{y})\ (\lambda\mathtt{z}.\mathtt{z})$

iii. $\lambda\mathtt{x}.\ \mathtt{snd}\ (\mathtt{x}\ (\mathtt{pair}\ \mathtt{s}_0\ \mathtt{s}_0))\ (\lambda\mathtt{p}.\ \mathtt{pair}\ (\mathtt{snd\ p})\ (\mathtt{sscc}\ (\mathtt{snd\ p})))$

iv. $\lambda\mathtt{x}.\ \mathtt{fst}\ (\lambda\mathtt{p}.\ \mathtt{pair}\ (\mathtt{snd\ p})\ (\mathtt{sscc}\ (\mathtt{snd\ p})))\ (\mathtt{x}\ (\mathtt{pair}\ \mathtt{s}_0\ \mathtt{s}_0))$

v. $\lambda\mathtt{x}.\ \mathtt{fst}\ (\mathtt{x}\ (\mathtt{pair}\ \mathtt{s}_0\ \mathtt{s}_0))\ (\lambda\mathtt{p}.\ \mathtt{pair}\ (\mathtt{snd\ p})\ (\mathtt{sscc}\ (\mathtt{snd\ p})))$

Answer: i.

(c) The addition function, where $\mathtt{splus}\ \mathtt{s}_m\ \mathtt{s}_n = \mathtt{s}_{n+m}$.

i. $\lambda\mathtt{m}.\ \lambda\mathtt{n}.\ \mathtt{m\ n}\ (\lambda\mathtt{z}.\mathtt{z})$

ii. $\lambda\mathtt{m}.\ \lambda\mathtt{n}.\ \lambda\mathtt{z}.\ \lambda\mathtt{s}.\ \mathtt{m}\ (\mathtt{n\ z\ s})\ \mathtt{s}$

iii. $\lambda\mathtt{m}.\ \lambda\mathtt{n}.\ \mathtt{n\ m}\ (\lambda\mathtt{x}.\ \mathtt{sscc})$

iv. $\mathtt{fix}\ (\lambda\mathtt{plus}.\ \lambda\mathtt{m}.\ \lambda\mathtt{n}.\ \mathtt{n\ m}\ (\mathtt{plus}\ (\mathtt{sscc\ m})))$

v. $\lambda\mathtt{m}.\ \lambda\mathtt{n}.\ \mathtt{fix}\ (\lambda\mathtt{plus}.\ \mathtt{n\ m}\ (\mathtt{sscc}\ (\lambda\mathtt{n}.\ \mathtt{plus\ m}\ (\mathtt{sprd\ n}))))$

Answer: iv.

Grading scheme: 3pts for a and b. 4 points for c.

4. (12 points) Circle the normal forms of the following untyped lambda calculus terms. If a term has no normal form, circle *NOTHING*. Recall that the normal form of a term $\mathtt{t}$ is some term $\mathtt{t}'$ such that $\mathtt{t} \longrightarrow^* \mathtt{t}'$ and $\mathtt{t}' \not\longrightarrow$.

(a) `(λx. λy. x y) (λz. λw. w)`

i. `λy. (λz. λw. w) y`
ii. `(λx. λy. x y) (λz. λw. w)`
iii. `λx. x (λz. λw. w)`
iv. `λy. λw. w`
v. NOTHING

Answer: i.

(b) `(λx. λy. x)(λx. y)`

i. `(λx. λy. x)(λx. y)`
ii. `(λx. λy. (λx. y))`
iii. `(λy. λx. y)`
iv. `(λw. λx. y)`
v. NOTHING

Answer: iv.

(c) `(λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))) (λg. g) (λh.h)`

i. `(λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))) (λg. g)`
ii. `λh.h`
iii. `(λx. (λg.g) (λy. x x y)) (λx. (λg.g) (λy. x x y))`
iv. `λx. (λg.g) (λy. x x y)`
v. NOTHING

Answer: v.

(d) `(λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))) (λg. λy. y) (λh.h)`

i. `(λf. (λx. f (λy. x x y)) (λx. f (λy. x x y))) (λg. λy. y)`
ii. `λh.h`
iii. `(λx. (λg.λy.y) (λy. x x y)) (λx. (λg. λy. y) (λy. x x y))`
iv. `λx. (λg.λy.y) (λy. x x y)`
v. NOTHING

Answer: ii.

Grading scheme: 3 points each.

Simply typed lambda-calculus

The following questions refer to the simply typed lambda-calculus (with recursion and base type `Bool`*). The syntax, typing, and evaluation rules for this system are given on page 2 of the companion handout.*

We can define the *big-step* evaluation relation for the simply typed lambda-calculus with recursion and booleans using the following rules:

$$\mathtt{v} \Downarrow \mathtt{v} \qquad \text{(B-VALUE)}$$

$$\frac{\mathtt{t}_1 \Downarrow \lambda \mathtt{x}{:}\mathtt{T}.\mathtt{t} \qquad \mathtt{t}_2 \Downarrow \mathtt{v}_2 \qquad [\mathtt{x} \mapsto \mathtt{v}_2]\mathtt{t} \Downarrow \mathtt{v}}{\mathtt{t}_1\ \mathtt{t}_2 \Downarrow \mathtt{v}} \qquad \text{(B-APP)}$$

$$\frac{\mathtt{t}_1 \Downarrow \mathtt{true} \qquad \mathtt{t}_2 \Downarrow \mathtt{v}}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \Downarrow \mathtt{v}} \qquad \text{(B-IFTRUE)}$$

$$\frac{\mathtt{t}_1 \Downarrow \mathtt{false} \qquad \mathtt{t}_3 \Downarrow \mathtt{v}}{\mathtt{if}\ \mathtt{t}_1\ \mathtt{then}\ \mathtt{t}_2\ \mathtt{else}\ \mathtt{t}_3 \Downarrow \mathtt{v}} \qquad \text{(B-IFFALSE)}$$

$$\frac{\mathtt{t} \Downarrow \lambda \mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_1 \qquad [\mathtt{x} \mapsto \mathtt{fix}\ (\lambda \mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_1)]\mathtt{t}_1 \Downarrow \mathtt{v}}{\mathtt{fix}\ \mathtt{t} \Downarrow \mathtt{v}} \qquad \text{(B-FIX)}$$

Jen Kennings, an eager assistant professor thought that it would be really great if the following proposition were true about these rules:

> *Proposition:* If $\emptyset \vdash \mathtt{t} : \mathtt{T}$ then there exists a $\mathtt{v}$ such that $\mathtt{t} \Downarrow \mathtt{v}$ and $\emptyset \vdash \mathtt{v} : \mathtt{T}$.

5. (10 points)

   (a) Unfortunately, this proposition is not true. Write down a counter-example where it fails. (i.e., find some closed, well typed term $\mathtt{t}$ such that either there is no $\mathtt{v}$ such that $\mathtt{t} \Downarrow \mathtt{v}$, or there is a such a $\mathtt{v}$, but it doesn't type check with the same type.)

   Answer: Any term that doesn't terminate.

   Grading scheme: 3 points. 1 point partial credit for any answer mentioning fix in any way.

(b) Not realizing that this proposition is false, professor Kennings started trying to prove it by induction on the typing derivation. However, she made a serious mistake in one of the first three cases, shown below. **Briefly describe her error in one or two sentences.** Note, professor Kennings hasn't yet attempted the cases for T-True, T-False, T-If or T-Fix, so those cases are not shown.

Proposition: If $\emptyset \vdash \mathtt{t} : \mathtt{T}$ then there exists a $\mathtt{v}$ such that $\mathtt{t} \Downarrow \mathtt{v}$ and $\emptyset \vdash \mathtt{v} : \mathtt{T}$.

Proof: Proof is induction on the typing derivation $\emptyset \vdash \mathtt{t} : \mathtt{T}$.

Case T-VAR*:* $\mathtt{t=x} \qquad \mathtt{x:T} \in \emptyset$

This case is impossible as the context is assumed to be empty.

Case T-ABS*:* $\mathtt{t=}\lambda\mathtt{x:T_1.t_2} \qquad \mathtt{T=T_1} \to \mathtt{T_2} \qquad \mathtt{x:T_1} \vdash \mathtt{t_2: T_2}$

This case is simple as $\lambda\mathtt{x:T_1.t_2} \Downarrow \lambda\mathtt{x:T_1.t_2}$ by B-VALUE, and $\emptyset \vdash \lambda\mathtt{x:T_1.t_2} : \mathtt{T}$ by assumption.

Case T-APP*:* $\mathtt{t=t_1\ t_2} \qquad \emptyset \vdash \mathtt{t_1 : T_1} \to \mathtt{T} \qquad \emptyset \vdash \mathtt{t_2 : T_1}$

By induction, there exists a $\mathtt{v_1}$ such that $\mathtt{t_1} \Downarrow \mathtt{v_1}$ and $\emptyset \vdash \mathtt{v_1} : \mathtt{T_1}\to\mathtt{T}$.

Also by induction, there exists a $\mathtt{v_2}$ such that $\mathtt{t_2} \Downarrow \mathtt{v_2}$ and $\emptyset \vdash \mathtt{v_2} : \mathtt{T_1}$.

By canonical forms, $\mathtt{v_1}$ is $\lambda\mathtt{x:T_1.t_{11}}$ and by inversion of the typing relation, $\mathtt{x:T_1} \vdash \mathtt{t_{11}} : \mathtt{T}$.

By substitution, $\emptyset \vdash [\mathtt{x} \mapsto \mathtt{v_2}]\mathtt{t_{11}} : \mathtt{T}$.

By induction, there exists a $\mathtt{v}$ such that $[\mathtt{x} \mapsto \mathtt{v_2}]\mathtt{t_{11}} \Downarrow \mathtt{v}$ and $\emptyset \vdash \mathtt{v} : \mathtt{T}$.

Finally, by the evaluation rule B-APP, $\mathtt{t_1\ t_2} \Downarrow \mathtt{v}$ and we've already shown that $\emptyset \vdash \mathtt{v} : \mathtt{T}$.

$\square$

Note: In her proof attempt above, professor Kennings referred to the following lemmas about the typed lambda-calculus with booleans and recursion. These lemmas are true, but she may or may not have used them correctly.

LEMMA (INVERSION OF THE TYPING RELATION):

i. If $\Gamma \vdash \lambda\mathtt{x:T_1.t_2} : \mathtt{R}$ then $\mathtt{R=T_1}\to\mathtt{R_2}$ for some $\mathtt{R_2}$ with $\Gamma, \mathtt{x:T_1} \vdash \mathtt{t_2} : \mathtt{R_2}$.

LEMMA (CANONICAL FORMS): If $\mathtt{v}$ is a value of type $\mathtt{T_1} \to \mathtt{T_2}$ then $\mathtt{v=}\lambda\mathtt{x:T_1.t_2}$.

LEMMA (SUBSTITUTION): If $\Gamma, \mathtt{x:S} \vdash \mathtt{t} : \mathtt{T}$ and $\Gamma \vdash \mathtt{s} : \mathtt{S}$ then $\Gamma \vdash [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}$.

Answer: In the T-App case, the induction hypothesis does not apply to $\emptyset \vdash [\mathtt{x} \mapsto \mathtt{v_2}]\mathtt{t_{11}} : T$ *because it is not a subderivation of the typing derivation.*

Grading scheme: 7 points. 2 points partial credit for identifying the T-App case as the culprit. 2 points deducted for insufficiently explicit answers.

Simply typed lambda-calculus with algorithmic subtyping

The following questions refer to the pure simply typed lambda-calculus with algorithmic subtyping (with just `Top`*—no booleans or records). The syntax, typing and evaluation rule this system are given on page 4 of the companion handout.*

6. (30 points)

The preservation theorem for the system with algorithmic typing can be stated as:

THEOREM (PRESERVATION): If $\Gamma \mapsto \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$ then $\Gamma \mapsto \mathtt{t}' : \mathtt{S}$ where $\mapsto \mathtt{S} <: \mathtt{T}$.

(a) Unlike the system with declarative typing, we **cannot** state the preservation theorem for this system as

If $\Gamma \mapsto \mathtt{t} : \mathtt{T}$ and $\mathtt{t} \longrightarrow \mathtt{t}'$ then $\Gamma \mapsto \mathtt{t}' : \mathtt{T}$.

This version of the theorem is false. Demonstrate why with a counter-example (i.e. find some term `t` that steps to some `t'` that cannot be given the *same* type with the algorithmic typing rules).

Answer: The term $(\lambda x{:}\mathtt{Top}.\ x)(\lambda y{:}\mathtt{Top}.y)$ *must be given the type* `Top`*. However, it single-steps to* $(\lambda y{:}\mathtt{Top}.y)$ *which has type* $\mathtt{Top}{\rightarrow}\mathtt{Top}$ *under the algorithmic typing rules.*

Grading scheme: 5 points. No deduction for answers that weren't technically in the language (involving booleans or numbers). One point deducted for right idea, but example that doesn't type check.

(b) Complete a precise and detailed proof of the preservation theorem on the next page. This proof is by induction on the **evaluation relation** $t \longrightarrow t'$. The case for E-APP1 has been done for you, but you need to do the cases for E-APP2 and E-APPABS. Note that the system we are considering in this problem, defined on page 4 of the handout, includes just `Top` and $\rightarrow$; your proof need not deal with other type constructors such as records. Do not include any extraneous information (true or false) in your proof. If needed, you may refer [without proof] to the lemmas stated below.

i. LEMMA (SUBTYPING RELATION INVERSION): If $\mapsto \mathtt{S} <: \mathtt{T}_1 \rightarrow \mathtt{T}_2$ then $\mathtt{S} = \mathtt{S}_1 \rightarrow \mathtt{S}_2$ with $\mapsto \mathtt{T}_1 <: \mathtt{S}_1$ and $\mapsto \mathtt{S}_2 <: \mathtt{T}_2$.

ii. LEMMA (REFLEXIVITY OF SUBTYPING): $\mapsto \mathtt{S} <: \mathtt{S}$.

iii. LEMMA (TRANSITIVITY OF SUBTYPING): If $\mapsto \mathtt{S} <: \mathtt{T}$ and $\mapsto \mathtt{T} <: \mathtt{U}$ then $\mapsto \mathtt{S} <: \mathtt{U}$.

iv. LEMMA (TYPING RELATION INVERSION):

A. If $\Gamma \mapsto \mathtt{t}_1\mathtt{t}_2 : \mathtt{T}$ then $\Gamma \mapsto \mathtt{t}_1 : \mathtt{T}_{11}{\rightarrow}\mathtt{T}$ and $\Gamma \mapsto \mathtt{t}_2 : \mathtt{T}_2$ and $\mapsto \mathtt{T}_2 <: \mathtt{T}_{11}$.

B. If $\Gamma \mapsto \lambda\mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2 : \mathtt{T}$ then $\mathtt{T} = \mathtt{T}_1 \rightarrow \mathtt{T}_2$ and $\Gamma, \mathtt{x}{:}\mathtt{T}_1 \mapsto \mathtt{t}_2 : \mathtt{T}_2$.

v. LEMMA (SUBSTITUTION): If $\Gamma, \mathtt{x}{:}\mathtt{S} \mapsto \mathtt{t} : \mathtt{T}$ and $\Gamma \mapsto \mathtt{s} : \mathtt{S}'$ with $\mapsto \mathtt{S}' <: \mathtt{S}$ then $\Gamma \mapsto [\mathtt{x} \mapsto \mathtt{s}]\mathtt{t} : \mathtt{T}'$ where $\mapsto \mathtt{T}' <: \mathtt{T}$.

THEOREM (PRESERVATION): If $\Gamma \Vdash \texttt{t} : \texttt{T}$ and $\texttt{t} \longrightarrow \texttt{t}'$ then $\Gamma \Vdash \texttt{t}' : \texttt{S}$ where $\Vdash \texttt{S} <: \texttt{T}$.

Proof: By induction on the evaluation relation, $\texttt{t} \longrightarrow \texttt{t}'$.

Case E-APP1*:* $\quad \texttt{t} = \texttt{t}_1\ \texttt{t}_2 \qquad \texttt{t}' = \texttt{t}_1'\ \texttt{t}_2 \qquad \texttt{t}_1 \longrightarrow \texttt{t}_1'$

By inversion of the typing relation, $\Gamma \Vdash \texttt{t}_1 : \texttt{T}_{11}{\to}\texttt{T}$ and $\Gamma \Vdash \texttt{t}_2 : \texttt{T}_2$ and $\Vdash \texttt{T}_2 <: \texttt{T}_{11}$.
By the induction hypothesis, $\Gamma \Vdash \texttt{t}_1' : \texttt{S}$ for some $\Vdash \texttt{S} <: \texttt{T}_{11}{\to}\texttt{T}$.
By inversion of the subtyping relation, $\texttt{S} = \texttt{S}_1 \to \texttt{S}_2$ where $\Vdash \texttt{T}_{11} <: \texttt{S}_1$ and $\Vdash \texttt{S}_2 <: \texttt{T}$.
By the transitivity of subtyping, $\Vdash \texttt{T}_2 <: \texttt{S}_1$.
By TA-APP, $\Gamma \Vdash \texttt{t}_1'\ \texttt{t}_2 : \texttt{S}_2$, where $\Vdash \texttt{S}_2 <: \texttt{T}$ as required.

Case E-APP2*:* $\quad \texttt{t} = \texttt{v}_1\ \texttt{t}_2 \qquad \texttt{t}' = \texttt{v}_1\ \texttt{t}_2' \qquad \texttt{t}_2 \longrightarrow \texttt{t}_2'$

Answer:
By inversion, $\Gamma \Vdash v_1 : T_{11}{\to}T$ *and* $\Gamma \Vdash t_2 : T_2$ *and* $\Vdash T_2 <: T_{11}$.
By induction, $\Gamma \Vdash t_2' : S$ *for some* $\Vdash S <: T_2$.
By transitivity, $\Vdash S <: T_{11}$.
By TA-APP, then $\Gamma \Vdash v_1\ t_2' : T$.
By reflexivity of subtyping, $\Vdash T <: T$ *as required.*
Grading scheme: Roughly 3 points per line. In particular, -3 for missing the use of reflexivity in the last line or transitivity in the third line.

Case E-APPABS*:* $\quad \texttt{t} = (\lambda \texttt{x}{:}\texttt{T}_{11}\texttt{.}\texttt{t}_{12})\ \texttt{v}_2 \qquad \texttt{t}' = [\texttt{x} \mapsto \texttt{v}_2]\texttt{t}_{12}$

Answer:
By inversion, $\Gamma \Vdash \lambda x{:}T_{11}.t_{12} : S{\to}T$ *and* $\Gamma \Vdash v_2 : T_2$ *and* $\Vdash T_2 <: S$.
By inversion again, $\Gamma, x{:}T_{11} \Vdash t_{12} : T$ *and* $S{\to}T = T_{11}{\to}T$*. Note that this means that* $T_2 <: T_{11}$.
By substitution, $\Gamma \Vdash [x \mapsto v_2]t_{12} : T'$ *where* $\Vdash T' <: T$.
Grading scheme: Roughly 3 points per line. -2 for using the first inversion lemma incorrectly, and saying that $\lambda x{:}T_{11}.t_{12}$ *has type* $T_{11}{\to}T$*. We don't know that the type of the argument is* T_{11} *until we use inversion again.*

$\square$

Simply typed lambda-calculus with subtyping, records, and references

The following questions refer to the simply typed lambda-calculus with subtyping, records, and references (and base types `Nat`*,* `Bool`*, and* `Unit`*). The syntax, typing, and evaluation rules for this system are given on page 5 of the companion handout.*

7. (9 points)

(a) List all syntactically different supertypes of `{a:Top,b:Top}`. Note: `S` and `T` are syntactically different types if they are written differently, even though it may be the case that `S<:T` and `T<:S`.

Answer: There are six. `{a:Top, b:Top}`, `{b:Top, a:Top}`, `{a:Top}`, `{b:Top}`, `{}` *and* `Top`*.*

Grading scheme: 1 point for 1-2 answers, 2 points for 3-4 answers, and 3 points for 5-6 answers.

(b) Is there an infinite *descending* chain in the subtype relation—that is, an infinite sequence of types S_0, S_1, etc. such that each S_{i+1} is a subtype of S_i? Note: Trivial chains don't count—each S_i must be different from all other types in the chain. If so, give an example. If not, describe why.

Answer: Yes, let S_0 = `{}`
S_1 = `{a:Top}`
S_2 = `{a:Top, b:Top}`

Grading scheme: 3 points. Partial credit for answering yes, but providing an incorrect example.

(c) Is there an infinite *ascending* chain in the subtype relation? Again, trivial chains don't count—each S_i must be different from all other types in the chain.If so, give an example. If not, describe why.

Answer: Yes, let $T_0 = S_0 \to \texttt{Top}$, $T_1 = S_1 \to \texttt{Top}$*, etc.*

Grading scheme: 3 points. Partial credit for answering yes, but providing an incorrect example.

8. (15 points)

   What is the minimal (or principal) type of the following expressions in the simply-typed lambda-calculus with subtyping, records and references? If a term does not type check, write NONE.

   (a) `λx:(Ref Bool) → Bool → Nat. x (ref true)`
   Answer: *`((Ref Bool) → Bool → Nat) → Bool → Nat`*

   (b) `(λx:{a:Ref Top}. x) {a=ref (λy:Top. y)}`
   Answer: NONE

   (c) `(λx:{a:Nat}→Top. x {a=2}) (λy:{a:Top}. y.a)`
   Answer: *`Top`*

   (d)
```
if true then λx:Ref Top. { y={b=!x}, d=!x }
         else λx:Ref Top. { y={a=2, b=3} }
```
   Answer: *`(Ref Top) → {y:{b:Top}}`*

   (e)
```
if true then λx:Ref Top. !x
         else λx:Nat. x
```
   Answer: *`Top`*

   Grading scheme: 3 points each. No partial credit for missing parens around the function argument in part (a).

Featherweight Java

The following questions refer to the Featherweight Java language. The syntax, typing, and evaluation rules for this system are given on page 9 of the companion handout.

9. (12 points)

   Consider extending Featherweight Java with *functional field update*. Functional field update allows programmers to easily create new objects that differ from existing objects only in the value of a single field.

   We formalize this extension by adding one new expression form to the syntax of FJ:

$$\begin{array}{lll} \mathtt{t} ::= & \mathtt{...} & \\ & \mathtt{t.f} \mathrel{<=} \mathtt{t} & \textit{functional field update} \end{array}$$

   The computation rule for functional field update returns a new object where the value of field $\mathtt{f}_i$ has been replaced with the new value $\mathtt{v}'$.

$$\frac{\mathtt{fields(C)} = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\mathtt{new\ C(v_1,\ ...,\ v_n).f}_i \mathrel{<=} \mathtt{v}' \longrightarrow \mathtt{new\ C(v_1,\ ...,\ v_{i-1},\ v',\ v_{i+1},\ ...,\ v_n)}} \quad \text{(E-UPDATE)}$$

   The two congruence rules specify the order of evaluation.

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1\mathtt{.f} \mathrel{<=} \mathtt{t}_2 \longrightarrow \mathtt{t}_1'\mathtt{.f} \mathrel{<=} \mathtt{t}_2} \quad \text{(E-UPDATE-RECV)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{v.f} \mathrel{<=} \mathtt{t}_2 \longrightarrow \mathtt{v.f} \mathrel{<=} \mathtt{t}_2'} \quad \text{(E-UPDATE-ARG)}$$

   For example, given the following class table

```
class A extends Object {
  Object x;
  Object y;
  Object z;
}

class B extends Object {
}
```

   A possible evaluation sequence is:

   `(new A(new Object(), new Object(), new Object()).y <= new B()).y`

   $\longrightarrow$ `new A(new Object(), new B(), new Object()).y`

   $\longrightarrow$ `new B()`

(a) Fill in the preconditions of the typing rule for functional field update so that the above example type checks and the preservation and progress theorems of FJ still hold. Furthermore, the type C must be the minimal type for the expression. (You do not need to do any proofs of these properties.)

$$\frac{\textit{Answer:}\quad \Gamma \vdash t : C \qquad \mathit{fields}(C) = \overline{C}\ \overline{f} \qquad \Gamma \vdash t_i : T_i \qquad T_i <: C_i}{\Gamma \vdash \mathtt{t.f_i} \mathrel{<=} \mathtt{t_i} : \mathtt{C}} \quad \text{(T-UPDATE)}$$

Answer: Also possible to replace $\mathit{fields}(C) = \overline{C}\ \overline{f}$ *with* $\Gamma \vdash t.f_i : C_i$

Grading scheme: 2pts per premise. -3 for common error, $\Gamma \vdash t_i : C$.

(b) Recall the statement of the progress theorem for FJ:

> THEOREM (PROGRESS): Suppose t is a closed, well-typed normal form. Then either (1) t is a value, or (2) for some evaluation context E, we can express t as $E[(C)(\mathtt{new}\ D(\overline{v}))]$, with $D \not<: C$.

This theorem relies on the following definition of *evaluation contexts* for FJ.

E ::=

- [ ]
- E.f
- $E.m(\overline{t})$
- $v.m(\overline{v}, E, \overline{t})$
- $\mathtt{new}\ C(\overline{v}, E, \overline{t})$
- (C)E

What new evaluation contexts are required for functional field update?

Answer: $E.f \mathrel{<=} t$ *and* $v.f \mathrel{<=} E$

Grading scheme: 2 pts per answer

Companion handout

Full definitions of the systems used in the exam

Untyped lambda-calculus

Syntax

$\mathtt{t}$ ::=		*terms*
	$\mathtt{x}$	*variable*
	$\lambda\mathtt{x}.\mathtt{t}$	*abstraction*
	$\mathtt{t}\ \mathtt{t}$	*application*
$\mathtt{v}$ ::=		*values*
	$\lambda\mathtt{x}.\mathtt{t}$	*abstraction value*

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

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

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

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

Simply typed lambda calculus (with $\mathtt{Bool}$ and recursion)

Syntax

$\mathtt{t}$ ::=		*terms*
	$\mathtt{true}$	*constant true*
	$\mathtt{false}$	*constant false*
	$\mathtt{if}\ \mathtt{t}\ \mathtt{then}\ \mathtt{t}\ \mathtt{else}\ \mathtt{t}$	*conditional*
	$\mathtt{x}$	*variable*
	$\lambda\mathtt{x}\!:\!\mathtt{T}.\mathtt{t}$	*abstraction*
	$\mathtt{t}\ \mathtt{t}$	*application*
	$\mathtt{fix}\ \mathtt{t}$	*fixed point of* $\mathtt{t}$
$\mathtt{v}$ ::=		*values*
	$\mathtt{true}$	*true value*
	$\mathtt{false}$	*false value*
	$\lambda\mathtt{x}\!:\!\mathtt{T}.\mathtt{t}$	*abstraction value*
$\mathtt{T}$ ::=		*types*
	$\mathtt{Bool}$	*type of booleans*
	$\mathtt{T}\to\mathtt{T}$	*type of functions*
Γ ::=		*contexts*
	$\emptyset$	*empty context*
	$\Gamma, \mathtt{x}\!:\!\mathtt{T}$	*term variable binding*

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

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

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

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

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

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

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

$$\mathtt{fix}(\lambda\mathtt{x}\!:\!\mathtt{T}_1.\mathtt{t}_2) \longrightarrow [\mathtt{x} \mapsto \mathtt{fix}\ (\lambda\mathtt{x}\!:\!\mathtt{T}_1.\mathtt{t}_2)]\mathtt{t}_2 \qquad \text{(E-FixBeta)}$$

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{fix}\ \mathtt{t}_1 \longrightarrow \mathtt{fix}\ \mathtt{t}_1'} \qquad \text{(E-Fix)}$$

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

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

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

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

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

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

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

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_1 \to \mathtt{T}_1}{\Gamma \vdash \mathtt{fix}\ \mathtt{t}_1 : \mathtt{T}_1} \qquad \text{(T-Fix)}$$

Pure simply typed lambda calculus with subtyping (no records) — algorithmic rules

Syntax

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

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

$$\frac{\mathtt{t}_1 \longrightarrow \mathtt{t}_1'}{\mathtt{t}_1\ \mathtt{t}_2 \longrightarrow \mathtt{t}_1'\ \mathtt{t}_2} \quad \text{(E-APP1)}$$

$$\frac{\mathtt{t}_2 \longrightarrow \mathtt{t}_2'}{\mathtt{v}_1\ \mathtt{t}_2 \longrightarrow \mathtt{v}_1\ \mathtt{t}_2'} \quad \text{(E-APP2)}$$

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

Algorithmic subtyping $\boxed{\mapsto \mathtt{S} <: \mathtt{T}}$

$$\mapsto \mathtt{S} <: \mathtt{Top} \quad \text{(SA-TOP)}$$

$$\frac{\mapsto \mathtt{T}_1 <: \mathtt{S}_1 \qquad \mapsto \mathtt{S}_2 <: \mathtt{T}_2}{\mapsto \mathtt{S}_1{\to}\mathtt{S}_2 <: \mathtt{T}_1{\to}\mathtt{T}_2} \quad \text{(SA-ARROW)}$$

Algorithmic typing $\boxed{\Gamma \mapsto \mathtt{t} : \mathtt{T}}$

$$\frac{\mathtt{x}{:}\mathtt{T} \in \Gamma}{\Gamma \mapsto \mathtt{x} : \mathtt{T}} \quad \text{(TA-VAR)}$$

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

$$\frac{\begin{array}{cc}\Gamma \mapsto \mathtt{t}_1 : \mathtt{T}_1 & \mathtt{T}_1 = \mathtt{T}_{11}{\to}\mathtt{T}_{12} \\ \Gamma \mapsto \mathtt{t}_2 : \mathtt{T}_2 & \mapsto \mathtt{T}_2 <: \mathtt{T}_{11}\end{array}}{\Gamma \mapsto \mathtt{t}_1\ \mathtt{t}_2 : \mathtt{T}_{12}} \quad \text{(TA-APP)}$$

Simply typed lambda calculus with subtyping (and records, references, recursion, booleans, numbers)

Syntax

t ::=		*terms*
	`x`	*variable*
	λ`x:T.t`	*abstraction*
	`t t`	*application*
	$\{l_i=t_i\ ^{i\in 1..n}\}$	*record*
	`t.l`	*projection*
	`unit`	*constant* `unit`
	`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*
	`let x=t in t`	*let binding*
	`fix t`	*fixed point of* `t`
v ::=		*values*
	λ`x:T.t`	*abstraction value*
	$\{l_i=v_i\ ^{i\in 1..n}\}$	*record value*
	`unit`	*constant* `unit`
	l	*store location*
	`true`	*true value*
	`false`	*false value*
	`nv`	*numeric value*
T ::=		*types*
	$\{l_i{:}T_i\ ^{i\in 1..n}\}$	*type of records*
	`Top`	*maximum type*
	$T \to T$	*type of functions*
	`Unit`	*unit type*
	`Ref T`	*type of reference cells*
	`Bool`	*type of booleans*
	`Nat`	*type of natural numbers*
Γ ::=		*contexts*
	$\emptyset$	*empty context*
	Γ, `x:T`	*term variable binding*
μ ::=		*stores*
	$\emptyset$	*empty store*

		$\mu, l = v$	*location binding*
Σ	::=		*store typings*
		$\emptyset$	*empty store typing*
		$\Sigma, l{:}T$	*location typing*
nv	::=		*numeric values*
		0	*zero value*
		succ nv	*successor value*

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

$$\frac{t_1 \mid \mu \longrightarrow t_1' \mid \mu'}{t_1\ t_2 \mid \mu \longrightarrow t_1'\ t_2 \mid \mu'} \quad \text{(E-APP1)}$$

$$\frac{t_2 \mid \mu \longrightarrow t_2' \mid \mu'}{v_1\ t_2 \mid \mu \longrightarrow v_1\ t_2' \mid \mu'} \quad \text{(E-APP2)}$$

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

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

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

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

$$\frac{l \notin dom(\mu)}{\texttt{ref}\ v_1 \mid \mu \longrightarrow l \mid (\mu, l \mapsto v_1)} \quad \text{(E-REFV)}$$

$$\frac{t_1 \mid \mu \longrightarrow t_1' \mid \mu'}{\texttt{ref}\ t_1 \mid \mu \longrightarrow \texttt{ref}\ t_1' \mid \mu'} \quad \text{(E-REF)}$$

$$\frac{\mu(l) = v}{!l \mid \mu \longrightarrow v \mid \mu} \quad \text{(E-DEREFLOC)}$$

$$\frac{t_1 \mid \mu \longrightarrow t_1' \mid \mu'}{!t_1 \mid \mu \longrightarrow !t_1' \mid \mu'} \quad \text{(E-DEREF)}$$

$$l{:=}v_2 \mid \mu \longrightarrow \texttt{unit} \mid [l \mapsto v_2]\mu \quad \text{(E-ASSIGN)}$$

$$\frac{t_1 \mid \mu \longrightarrow t_1' \mid \mu'}{t_1{:=}t_2 \mid \mu \longrightarrow t_1'{:=}t_2 \mid \mu'} \quad \text{(E-ASSIGN1)}$$

$$\frac{t_2 \mid \mu \longrightarrow t_2' \mid \mu'}{v_1{:=}t_2 \mid \mu \longrightarrow v_1{:=}t_2' \mid \mu'} \quad \text{(E-ASSIGN2)}$$

$$\texttt{if true then}\ t_2\ \texttt{else}\ t_3 \mid \mu \longrightarrow t_2 \mid \mu \quad \text{(E-IFTRUE)}$$

$$\texttt{if false then}\ t_2\ \texttt{else}\ t_3 \mid \mu \longrightarrow t_3 \mid \mu \quad \text{(E-IFFALSE)}$$

$$\frac{t_1 \mid \mu \longrightarrow t_1' \mid \mu'}{\texttt{if}\ t_1\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \mid \mu \longrightarrow \texttt{if}\ t_1'\ \texttt{then}\ t_2\ \texttt{else}\ t_3 \mid \mu'} \quad \text{(E-IF)}$$

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

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

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

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

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

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

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

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

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

$$\begin{array}{c}\mathtt{fix}\ (\lambda\mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2)|\,\mu \\ \longrightarrow [\mathtt{x} \mapsto (\mathtt{fix}\ (\lambda\mathtt{x}{:}\mathtt{T}_1.\mathtt{t}_2))]\mathtt{t}_2|\,\mu\end{array} \qquad \text{(E-FixBeta)}$$

$$\frac{\mathtt{t}_1|\,\mu \longrightarrow \mathtt{t}_1'|\,\mu'}{\mathtt{fix}\ \mathtt{t}_1|\,\mu \longrightarrow \mathtt{fix}\ \mathtt{t}_1'|\,\mu} \qquad \text{(E-Fix)}$$

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

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

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

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

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

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

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

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

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

$$\frac{\text{for each } i \quad \Gamma \mid \Sigma \vdash \mathtt{t}_i : \mathtt{T}_i}{\Gamma \mid \Sigma \vdash \{\mathtt{l}_i\mathtt{=}\mathtt{t}_i \;^{i\in 1..n}\} : \{\mathtt{l}_i\mathtt{:}\mathtt{T}_i \;^{i\in 1..n}\}} \tag{T-Rcd}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \{\mathtt{l}_i\mathtt{:}\mathtt{T}_i \;^{i\in 1..n}\}}{\Gamma \mid \Sigma \vdash \mathtt{t}_1\mathtt{.}\mathtt{l}_j : \mathtt{T}_j} \tag{T-Proj}$$

$$\frac{\mathtt{x}\mathtt{:}\mathtt{T} \in \Gamma}{\Gamma \mid \Sigma \vdash \mathtt{x} : \mathtt{T}} \tag{T-Var}$$

$$\frac{\Gamma, \mathtt{x}\mathtt{:}\mathtt{T}_1 \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \mid \Sigma \vdash \lambda\mathtt{x}\mathtt{:}\mathtt{T}_1\mathtt{.}\mathtt{t}_2 : \mathtt{T}_1 \to \mathtt{T}_2} \tag{T-Abs}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12} \qquad \Gamma \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma \mid \Sigma \vdash \mathtt{t}_1\;\mathtt{t}_2 : \mathtt{T}_{12}} \tag{T-App}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t} : \mathtt{S} \qquad \mathtt{S} <: \mathtt{T}}{\Gamma \mid \Sigma \vdash \mathtt{t} : \mathtt{T}} \tag{T-Sub}$$

$$\Gamma \mid \Sigma \vdash \mathtt{unit} : \mathtt{Unit} \tag{T-Unit}$$

$$\frac{\Sigma(\mathtt{l}) = \mathtt{T}_1}{\Gamma \mid \Sigma \vdash \mathtt{l} : \mathtt{Ref}\;\mathtt{T}_1} \tag{T-Loc}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_1}{\Gamma \mid \Sigma \vdash \mathtt{ref}\;\mathtt{t}_1 : \mathtt{Ref}\;\mathtt{T}_1} \tag{T-Ref}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Ref}\;\mathtt{T}_{11}}{\Gamma \mid \Sigma \vdash \mathtt{!}\mathtt{t}_1 : \mathtt{T}_{11}} \tag{T-Deref}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Ref}\;\mathtt{T}_{11} \qquad \Gamma \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma \mid \Sigma \vdash \mathtt{t}_1 \mathtt{:=} \mathtt{t}_2 : \mathtt{Unit}} \tag{T-Assign}$$

$$\Gamma \mid \Sigma \vdash \mathtt{true} : \mathtt{Bool} \tag{T-True}$$

$$\Gamma \mid \Sigma \vdash \mathtt{false} : \mathtt{Bool} \tag{T-False}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Bool} \qquad \Gamma \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T} \qquad \Gamma \mid \Sigma \vdash \mathtt{t}_3 : \mathtt{T}}{\Gamma \mid \Sigma \vdash \mathtt{if}\;\mathtt{t}_1\;\mathtt{then}\;\mathtt{t}_2\;\mathtt{else}\;\mathtt{t}_3 : \mathtt{T}} \tag{T-If}$$

$$\Gamma \mid \Sigma \vdash \mathtt{0} : \mathtt{Nat} \tag{T-Zero}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Nat}}{\Gamma \mid \Sigma \vdash \mathtt{succ}\;\mathtt{t}_1 : \mathtt{Nat}} \tag{T-Succ}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Nat}}{\Gamma \mid \Sigma \vdash \mathtt{pred}\;\mathtt{t}_1 : \mathtt{Nat}} \tag{T-Pred}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{Nat}}{\Gamma \mid \Sigma \vdash \mathtt{iszero}\;\mathtt{t}_1 : \mathtt{Bool}} \tag{T-IsZero}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_1 \qquad \Gamma, \mathtt{x}\mathtt{:}\mathtt{T}_1 \mid \Sigma \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \mid \Sigma \vdash \mathtt{let}\;\mathtt{x}\mathtt{=}\mathtt{t}_1\;\mathtt{in}\;\mathtt{t}_2 : \mathtt{T}_2} \tag{T-Let}$$

$$\frac{\Gamma \mid \Sigma \vdash \mathtt{t}_1 : \mathtt{T}_1 \to \mathtt{T}_1}{\Gamma \mid \Sigma \vdash \mathtt{fix}\;\mathtt{t}_1 : \mathtt{T}_1} \tag{T-Fix}$$

Featherweight Java

Syntax

`CL ::=`		*class declarations*
	$\texttt{class C extends C } \{\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K } \overline{\texttt{M}}\}$	
`K ::=`		*constructor declarations*
	$\texttt{C(}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{) \{super(}\overline{\texttt{f}}\texttt{); this.}\overline{\texttt{f}}\texttt{=}\overline{\texttt{f}}\texttt{;\}}$	
`M ::=`		*method declarations*
	$\texttt{C m(}\overline{\texttt{C}}\ \overline{\texttt{x}}\texttt{) \{return t;\}}$	
`t ::=`		*terms*
	`x`	*variable*
	`t.f`	*field access*
	$\texttt{t.m(}\overline{\texttt{t}}\texttt{)}$	*method invocation*
	$\texttt{new C(}\overline{\texttt{t}}\texttt{)}$	*object creation*
	`(C) t`	*cast*
`v ::=`		*values*
	$\texttt{new C(}\overline{\texttt{v}}\texttt{)}$	*object creation*

Subtyping $\boxed{\texttt{C<:D}}$

$$\texttt{C} <: \texttt{C}$$

$$\frac{\texttt{C} <: \texttt{D} \qquad \texttt{D} <: \texttt{E}}{\texttt{C} <: \texttt{E}}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D \{...\}}}{\texttt{C} <: \texttt{D}}$$

Field lookup $\boxed{\mathit{fields}(\texttt{C}) = \overline{\texttt{C}}\ \overline{\texttt{f}}}$

$$\mathit{fields}(\texttt{Object}) = \bullet$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K } \overline{\texttt{M}}\} \qquad \mathit{fields}(\texttt{D}) = \overline{\texttt{D}}\ \overline{\texttt{g}}}{\mathit{fields}(\texttt{C}) = \overline{\texttt{D}}\ \overline{\texttt{g}}, \overline{\texttt{C}}\ \overline{\texttt{f}}}$$

Method type lookup $\boxed{\mathit{mtype}(\texttt{m}, \texttt{C}) = \overline{\texttt{C}} \rightarrow \texttt{C}}$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K } \overline{\texttt{M}}\} \qquad \texttt{B m (}\overline{\texttt{B}}\ \overline{\texttt{x}}\texttt{) \{return t;\}} \in \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C}) = \overline{\texttt{B}} \rightarrow \texttt{B}}$$

$$\frac{CT(\texttt{C}) = \texttt{class C extends D } \{\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K } \overline{\texttt{M}}\} \qquad \texttt{m} \text{ is not defined in } \overline{\texttt{M}}}{\mathit{mtype}(\texttt{m}, \texttt{C}) = \mathit{mtype}(\texttt{m}, \texttt{D})}$$

Method body lookup $\boxed{\mathit{mbody}(\texttt{m}, \texttt{C}) = (\overline{\texttt{x}}, \texttt{t})}$

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ \{\overline{C}\ \overline{f};\ K\ \overline{M}\}} \quad \mathtt{B\ m\ (\overline{B}\ \overline{x})\ \{return\ t;\}} \in \overline{\mathtt{M}}}{mbody(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{t})}$$

$$\frac{CT(\mathtt{C}) = \mathtt{class\ C\ extends\ D\ \{\overline{C}\ \overline{f};\ K\ \overline{M}\}} \quad \mathtt{m} \text{ is not defined in } \overline{\mathtt{M}}}{mbody(\mathtt{m}, \mathtt{C}) = mbody(\mathtt{m}, \mathtt{D})}$$

Valid method overriding $\boxed{override(\mathtt{m}, \mathtt{D}, \overline{\mathtt{C}} \to \mathtt{C}_0)}$

$$\frac{mtype(\mathtt{m}, \mathtt{D}) = \overline{\mathtt{D}} \to \mathtt{D}_0 \text{ implies } \overline{\mathtt{C}} = \overline{\mathtt{D}} \text{ and } \mathtt{C}_0 = \mathtt{D}_0}{override(\mathtt{m}, \mathtt{D}, \overline{\mathtt{C}} \to \mathtt{C}_0)}$$

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

$$\frac{fields(\mathtt{C}) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\mathtt{(new\ C(\overline{v})).f}_i \longrightarrow \mathtt{v}_i} \quad \text{(E-ProjNew)}$$

$$\frac{mbody(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{t}_0)}{\mathtt{(new\ C(\overline{v})).m(\overline{u})} \longrightarrow [\overline{\mathtt{x}} \mapsto \overline{\mathtt{u}}, \mathtt{this} \mapsto \mathtt{new\ C(\overline{v})}]\mathtt{t}_0} \quad \text{(E-InvkNew)}$$

$$\frac{\mathtt{C} <: \mathtt{D}}{\mathtt{(D)(new\ C(\overline{v}))} \longrightarrow \mathtt{new\ C(\overline{v})}} \quad \text{(E-CastNew)}$$

$$\frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{\mathtt{t}_0\mathtt{.f} \longrightarrow \mathtt{t}_0'\mathtt{.f}} \quad \text{(E-Field)}$$

$$\frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{\mathtt{t}_0\mathtt{.m(\overline{t})} \longrightarrow \mathtt{t}_0'\mathtt{.m(\overline{t})}} \quad \text{(E-Invk-Recv)}$$

$$\frac{\mathtt{t}_i \longrightarrow \mathtt{t}_i'}{\mathtt{v}_0\mathtt{.m(\overline{v},\ t}_i\mathtt{,\ \overline{t})} \longrightarrow \mathtt{v}_0\mathtt{.m(\overline{v},\ t}_i'\mathtt{,\ \overline{t})}} \quad \text{(E-Invk-Arg)}$$

$$\frac{\mathtt{t}_i \longrightarrow \mathtt{t}_i'}{\mathtt{new\ C(\overline{v},\ t}_i\mathtt{,\ \overline{t})} \longrightarrow \mathtt{new\ C(\overline{v},\ t}_i'\mathtt{,\ \overline{t})}} \quad \text{(E-New-Arg)}$$

$$\frac{\mathtt{t}_0 \longrightarrow \mathtt{t}_0'}{\mathtt{(C)t}_0 \longrightarrow \mathtt{(C)t}_0'} \quad \text{(E-Cast)}$$

Term typing $\boxed{\Gamma \vdash \mathtt{t} : \mathtt{C}}$

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

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \qquad fields(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\Gamma \vdash \mathtt{t}_0\mathtt{.f}_i : \mathtt{C}_i} \quad \text{(T-Field)}$$

$$\frac{\Gamma \vdash \mathtt{t}_0 : \mathtt{C}_0 \quad mtype(\mathtt{m}, \mathtt{C}_0) = \overline{\mathtt{D}} \to \mathtt{C} \quad \Gamma \vdash \overline{\mathtt{t}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{t}_0\mathtt{.m(\overline{t})} : \mathtt{C}} \quad \text{(T-Invk)}$$

$$\dfrac{\begin{array}{c}\textit{fields}(\mathtt{C}) = \overline{\mathtt{D}}\;\overline{\mathtt{f}} \\ \Gamma \vdash \overline{\mathtt{t}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} <: \overline{\mathtt{D}}\end{array}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\overline{\mathtt{t}}) : \mathtt{C}} \tag{T-New}$$

$$\dfrac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{D} <: \mathtt{C}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \tag{T-UCast}$$

$$\dfrac{\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} <: \mathtt{D} \qquad \mathtt{C} \neq \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \tag{T-DCast}$$

$$\dfrac{\begin{array}{c}\Gamma \vdash \mathtt{t}_0 : \mathtt{D} \qquad \mathtt{C} \not<: \mathtt{D} \qquad \mathtt{D} \not<: \mathtt{C} \\ \textit{stupid warning}\end{array}}{\Gamma \vdash (\mathtt{C})\mathtt{t}_0 : \mathtt{C}} \tag{T-SCast}$$

Method typing $\boxed{\mathtt{M}\ \mathtt{OK}\ \mathtt{in}\ \mathtt{C}}$

$$\dfrac{\begin{array}{c}\overline{\mathtt{x}} : \overline{\mathtt{C}}, \mathtt{this} : \mathtt{C} \vdash \mathtt{t}_0 : \mathtt{E}_0 \qquad \mathtt{E}_0 <: \mathtt{C}_0 \\ \textit{CT}(\mathtt{C}) = \mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{\ldots\} \\ \textit{override}(\mathtt{m}, \mathtt{D}, \overline{\mathtt{C}} \rightarrow \mathtt{C}_0)\end{array}}{\mathtt{C}_0\ \mathtt{m}\ (\overline{\mathtt{C}}\ \overline{\mathtt{x}})\ \{\mathtt{return}\ \mathtt{t}_0;\}\ \mathtt{OK}\ \mathtt{in}\ \mathtt{C}}$$

Class typing $\boxed{\mathtt{C}\ \mathtt{OK}}$

$$\dfrac{\begin{array}{c}\mathtt{K} = \mathtt{C}(\overline{\mathtt{D}}\ \overline{\mathtt{g}}, \overline{\mathtt{C}}\ \overline{\mathtt{f}}) \qquad \{\mathtt{super}(\overline{\mathtt{g}});\ \mathtt{this}.\overline{\mathtt{f}} = \overline{\mathtt{f}};\} \\ \textit{fields}(\mathtt{D}) = \overline{\mathtt{D}}\ \overline{\mathtt{g}} \qquad \overline{\mathtt{M}}\ \mathtt{OK}\ \mathtt{in}\ \mathtt{C}\end{array}}{\mathtt{class}\ \mathtt{C}\ \mathtt{extends}\ \mathtt{D}\ \{\overline{\mathtt{C}}\ \overline{\mathtt{f}};\ \mathtt{K}\ \overline{\mathtt{M}}\}\ \mathtt{OK}}$$