Graual typing

Mid-term exam, MPRI 2-4

2021/12/01 — Duration: 2h45

Answers are judged by their correctness, but also by their clarity, conciseness, and accuracy. You don't have to justify answers unless explicitly required. Although the questions are in English, it is permitted to answer in either French or English—and recommended to answer in French if this is your mother language.

Let $\mathcal{S}$ stand for the simply-typed λ-calculus. In this problem, we study $\mathcal{G}$, an explicitly-typed extension of $\mathcal{S}$ with *gradual typing*. Gradual typing allows a combination of static and dynamic typing. For that purpose, we first introduce $\mathcal{D}$, an extension of $\mathcal{S}$ with dynamic typing. The static and dynamic semantics of $\mathcal{G}$ will then be given by elaboration into $\mathcal{D}$.

We write $\vdash_{\mathcal{L}} M : \tau$ to mean that M is well-typed and has type τ in the language $\mathcal{L}$, where $\mathcal{L}$ stands for $\mathcal{S}$, $\mathcal{D}$, or $\mathcal{G}$. Similarly, we write $\longrightarrow_{\mathcal{L}}$ for reduction in $\mathcal{L}$. (The indice may however be omitted when it is clear from context.)

Syntax. The syntax of types and expressions of $\mathcal{D}$ is as follows:

$$\begin{array}{rcl@{\qquad}rcl}
\tau & ::= & \mathsf{unit} \mid \tau \to \tau \mid ? & M & ::= & V \mid M\,M \mid \langle M : \tau \bowtie \tau \rangle \\
\zeta & ::= & \mathsf{unit} \mid ? \to ? & V & ::= & () \mid \lambda x : \tau.\, M \mid \langle V : \zeta \rhd ? \rangle \\
\bowtie & ::= & \lhd \mid \rhd & & \mid & \langle V : \tau_1 \to \tau_2 \bowtie \tau_1' \to \tau_2' \rangle \mid \langle V : \mathsf{unit} \bowtie \mathsf{unit} \rangle
\end{array}$$

Types. The letter τ ranges over types, which (recursively) extend simple types with a new primitive type ? that stands for an *unknown* type. The letter ζ ranges over a subset of types, called *shapes*, used to describe the topmost structure of types. Notice that the unkown type ? is *intentionally* excluded from ζ.

Materialization. The relation $\preceq$, called *materialization*, is inductively defined by

$$? \preceq \tau \qquad\qquad \mathsf{unit} \preceq \mathsf{unit} \qquad\qquad \frac{\tau_1 \preceq \tau_1' \qquad \tau_2 \preceq \tau_2'}{\tau_1 \to \tau_2 \preceq \tau_1' \to \tau_2'}$$

Materialization is covariant on arrow types: therefore, it is not a form of subtyping. Intuitively, materialization increases precision in types: it replaces some occurrences of the unknown type ? by some other type. For example, we have $(? \to \mathsf{unit}) \preceq (\mathsf{unit} \to \mathsf{unit})$. The reverse relation $\succeq$ decreases precision.

Expressions. The expressions of $\mathcal{D}$ extends those of $\mathcal{S}$ with cast expressions $\langle M : \tau \bowtie \tau' \rangle$. Notice that $\bowtie$ is a meta-variable that can take only two concrete values, namely $\lhd$ and $\rhd$. That is, there are two dual forms of cast expressions, $\langle M : \tau \lhd \tau' \rangle$ and $\langle M : \tau \rhd \tau' \rangle$. Both of them convert a term M of gradual type τ into one of gradual type τ', but they have opposite requirements about the relationship between τ and τ': the first form requires $\tau \preceq \tau'$, whereas the second form requires $\tau \succeq \tau'$. Thus, a positive cast $\langle M : \tau \triangleright \tau' \rangle$ loses precision, turning a term M of type τ into a term of a less precise type τ'. Conversely, a negative cast $\langle M : \tau \triangleleft \tau' \rangle$ turns a term M of type τ into one of a more precise type τ'.

Values. Values V include constants and functions, as usual, and *delayed casts*. A delayed cast is either:

- a superficial positive cast $\langle V : \zeta \triangleright ? \rangle$.
- a deep cast $\langle V : \tau_1 \to \tau_2 \bowtie \tau_1' \to \tau_2' \rangle$ or $\langle V : \mathsf{unit} \bowtie \mathsf{unit} \rangle$ between types of the same shape.

Notation. To exploit the symmetry between the two dual forms of casts, we introduce the following notation:

- We write $\bowtie^\perp$ for the dual of $\bowtie$: that is, $\triangleleft^\perp$ is $\triangleright$, and is $\triangleright^\perp$ is $\triangleleft$. Hence, $(\bowtie^\perp)^\perp$ equals $\bowtie$.
- We write $\preceq^\triangleleft$ for $\preceq$ and $\preceq^\triangleright$ for $\succeq$, so that $\preceq^\bowtie$ means either $\preceq$ or $\succeq$ according to the value of the meta-variable $\bowtie$.

Typing. The typing rules of $\mathcal{D}$ are as follows:

$$\textsc{Var}\ \frac{x : \tau}{\Gamma \vdash_{\mathcal{D}} x : \tau} \qquad \textsc{Const}\ \Gamma \vdash_{\mathcal{D}} () : \mathsf{unit} \qquad \textsc{App}\ \frac{\Gamma \vdash_{\mathcal{D}} M : \tau' \to \tau \qquad \Gamma \vdash_{\mathcal{D}} M' : \tau'}{\Gamma \vdash_{\mathcal{D}} M\ M' : \tau}$$

$$\textsc{Abs}\ \frac{\Gamma, x : \tau \vdash_{\mathcal{D}} M : \tau'}{\Gamma \vdash_{\mathcal{D}} \lambda x : \tau.\, M : \tau \to \tau'} \qquad \textsc{Cast}\ \frac{\Gamma \vdash_{\mathcal{D}} M : \tau \qquad \tau \preceq^{\bowtie} \tau'}{\Gamma \vdash_{\mathcal{D}} \langle M : \tau \bowtie \tau' \rangle : \tau'}$$

Shapes We define an idempotent function $\sharp$ that maps a type τ to either a shape ζ or the unknown type ?. (Remember that ? is not itself a shape and is treated specially.)

$$\sharp\mathsf{unit} = \mathsf{unit} \qquad \sharp(\tau \to \tau') = ? \to ? \qquad \sharp? = ?$$

Semantics The semantics of $\mathcal{D}$ is given as a small-step, call-by-value reduction relation $\longrightarrow_{\mathcal{D}}$ with a left-to-right evaluation order. The shallow evaluation contexts E are:

$$E \ ::= \ [\,]\ M \mid V\ [\,] \mid \langle [\,] : \tau \bowtie \tau' \rangle$$

The reduction rules are:

$$\begin{array}{rcllr}
(\lambda x : \tau.\, M)\ V & \longrightarrow_{\mathcal{D}} & M[x \leftarrow V] & & (\textsc{Beta}) \\
\langle V : \tau_1 \to \tau_2 \bowtie \tau_1' \to \tau_2' \rangle\ V' & \longrightarrow_{\mathcal{D}} & \langle (V\ \langle V' : \tau_1' \bowtie^\perp \tau_1 \rangle) : \tau_2 \bowtie \tau_2' \rangle & & (\textsc{Cast-App}) \\
\langle V : ? \bowtie ? \rangle & \longrightarrow_{\mathcal{D}} & V & & (\textsc{Cast-Id}) \\
\langle V : ? \triangleleft \tau \rangle & \longrightarrow_{\mathcal{D}} & \langle \langle V : ? \triangleleft \sharp\tau \rangle : \sharp\tau \triangleleft \tau \rangle & \tau \neq \sharp\tau & (\textsc{Cast-Neg}) \\
\langle V : \tau \triangleright ? \rangle & \longrightarrow_{\mathcal{D}} & \langle \langle V : \tau \triangleright \sharp\tau \rangle : \sharp\tau \triangleright ? \rangle & \tau \neq \sharp\tau & (\textsc{Cast-Pos}) \\
\langle \langle V : \zeta \triangleright ? \rangle : ? \triangleleft \zeta \rangle & \longrightarrow_{\mathcal{D}} & V & & (\textsc{Cast-Cancel})
\end{array}$$

$$\frac{M \longrightarrow_{\mathcal{D}} M'}{E[M] \longrightarrow_{\mathcal{D}} E[M']} \qquad (\textsc{Context})$$

Rules Beta and Context are as usual. Rule Cast-App breaks a cast that lies inside an application down into smaller casts that lie outside of the application, possibly allowing a Beta redex to appear. Cast-Id removes trivial casts. Rules Cast-Neg and Cast-Pos decompose casts into smaller ones. Rule Cast-Cancel removes a successful cast, where the expected shape of the negative cast operation agrees with the shape of the "delayed positive cast" value.

1 Warming up

Question 1 *Give the full reduction sequence for the term* $M_1 \overset{\mathsf{def}}{=} \langle(\lambda x{:}?.\,x){:}? \to ?{\lhd}\mathsf{unit} \to \mathsf{unit}\rangle\ ()$.

Answer: As usual, we omit the Context steps and mark the redex in red instead.

$$\begin{array}{rcll}
M_1 & = & \langle(\lambda x{:}?.\,x) : ? \to ? \lhd \mathsf{unit} \to \mathsf{unit}\rangle\ () & (\textsc{Cast-App}) \\
& \longrightarrow_{\mathcal{D}} & \langle\big((\lambda x{:}?.\,x)\ \langle() : \mathsf{unit} \rhd ?\rangle\big) : ? \lhd \mathsf{unit}\rangle & (\textsc{Beta}) \\
& \longrightarrow_{\mathcal{D}} & \langle\langle() : \mathsf{unit} \rhd ?\rangle : ? \lhd \mathsf{unit}\rangle & (\textsc{Cast-Cancel}) \\
& \longrightarrow_{\mathcal{D}} & () &
\end{array}$$

□

Question 2 *Give the full reduction sequence for the term*

$$M_2 \overset{\mathsf{def}}{=} \langle\big(\lambda x{:}?.\,\langle x : ? \lhd \mathsf{unit} \to \mathsf{unit}\rangle\ ()\big) : ? \to \mathsf{unit} \lhd \mathsf{unit} \to \mathsf{unit}\rangle\ ()$$

Is the result a value?

Answer: For conciseness, we write u for unit (and omit the Context steps).

$$\begin{array}{rcll}
M_2 & = & \langle\big(\lambda x{:}?.\,\langle x : ? \lhd \mathsf{u} \to \mathsf{u}\rangle\ ()\big) : ? \to \mathsf{u} \lhd \mathsf{u} \to \mathsf{u}\rangle\ () & (\textsc{Cast-App}) \\
& \longrightarrow_{\mathcal{D}} & \langle\big(\lambda x{:}?.\,\langle x : ? \lhd \mathsf{u} \to \mathsf{u}\rangle\ ()\big)\ \langle() : \mathsf{u} \rhd ?\rangle : \mathsf{u} \lhd \mathsf{u}\rangle & (\textsc{Beta}) \\
& \longrightarrow_{\mathcal{D}} & \langle\langle\langle() : \mathsf{u} \rhd ?\rangle : ? \lhd \mathsf{u} \to \mathsf{u}\rangle\ () : \mathsf{u} \lhd \mathsf{u}\rangle & (\textsc{Cast-Neg}) \\
& \longrightarrow_{\mathcal{D}} & \langle\langle\langle\langle() : \mathsf{u} \rhd ?\rangle : ? \lhd ? \to ?\rangle : ? \to ? \lhd \mathsf{u} \to \mathsf{u}\rangle\ () : \mathsf{u} \lhd \mathsf{u}\rangle & \\
& \not\longrightarrow_{\mathcal{D}} & &
\end{array}$$

The result is stuck. In particular, Rule Cast=App does not apply because application $\langle\langle() : \mathsf{unit} \rhd ?\rangle : ? \lhd ? \to ?\rangle$ is not a value (it a cast error). □

Question 3 *Show that* $\tau \preceq \tau'$ *implies* $\sharp\tau \preceq \sharp\tau'$.

Answer: We show $\sharp\tau \preceq \sharp\tau'$ (**1**) by cases on the last rule used in the proof of $\tau \preceq \tau'$.

Case $? \preceq \tau'$. Then $\sharp\tau$ is $?$, hence (1).

Case $\mathsf{unit} \preceq \mathsf{unit}$. Then both $\sharp\tau$ and $\sharp\tau'$ are unit, hence (1).

Case $\tau_1 \to \tau_2 \preceq \tau_1' \to \tau_2'$. Then, both $\sharp\tau$ and $\sharp\tau'$ are $? \to ?$, hence (1) by reflexivity of $\preceq$ (which easily follows from its definition). □

Question 4 *Is* $(\lambda x{:}?.\,x)\ ()$ *typable in* $\mathcal{D}$*? (Justify, briefly.)*

Answer: **It is not typable in $\mathcal{D}$,** since the application is ill-typed. The function $\lambda x{:}\tau.\,M$, which expects an argument of type $?$ is applied to $()$ of type unit.

□

We write $\tau^{\mathcal{S}}$ for a type of $\mathcal{S}$, *i.e.* a type τ where the unknown type ? does not occur. We write $M^{\mathcal{S}}$ for a term of $\mathcal{S}$, *i.e.* a term M without casts and where all type annotations are types of the form $\tau^{\mathcal{S}}$.

Question 5 *Show that $\mathcal{D}$ is a conservative extension of $\mathcal{S}$, that is:*

(1) $\vdash_{\mathcal{S}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$ *iff* $\vdash_{\mathcal{D}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$;

(2) $M_1^{\mathcal{S}} \longrightarrow_{\mathcal{S}} M_2^{\mathcal{S}}$ *iff* $M_1^{\mathcal{S}} \longrightarrow_{\mathcal{D}} M_2^{\mathcal{S}}$.

(A brief argument is expected, not detailed proofs by induction.)
Answer:

(1) Typing: A typing derivation in in $\mathcal{S}$ is clearly a typing derivation in $\mathcal{D}$. Conversely, a typing derivation of $\vdash_{\mathcal{D}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$ cannot mention rule Cast and cannot have any occurrence of ?. Hence, it is also a derivation in $\mathcal{S}$.

(2) Reduction: Since reduction in $\mathcal{D}$ never introduces new casts, a reduction sequence starting with a term $M^{\mathcal{S}}$ (hence without casts) cannot contain any Cast- reduction rule. Hence, the possible reduction sequences originating from $\mathcal{S}$ are exactly the same in $\mathcal{S}$ and $\mathcal{D}$. □

Question 6 *Is the definition of the type system syntax-directed? (A yes/no answer expected.)*
Answer: Yes. *(Besides, not only the shape of typing derivations, but typing derivations themselves, are unique.)* □

We write Ω for the set of untyped λ-terms. We define an erasure function $|\cdot|$ from $\mathcal{D}$ to Ω that removes type annotations on abstractions and casts. It is defined as follows:

$$|()| = () \qquad |x| = x \qquad |\lambda x : \tau .\, M| = \lambda x .\, |M| \qquad |M_1\ M_2| = |M_1|\ |M_2|$$

$$|\langle M : \tau \bowtie \tau' \rangle| = |M|$$

Question 7 *Show that every untyped λ-term is the erasure of some well-typed term in $\mathcal{D}$.*
Answer: We inductively define an encoding $[\![\cdot]\!]$ from Ω to $\mathcal{D}$:

$$[\![()]\!] = \langle () : \mathsf{unit} \triangleright ? \rangle \qquad [\![x]\!] = x \qquad [\![\lambda x.\, a]\!] = \langle \lambda x : ?.\, [\![a]\!] : ? \to ? \triangleright ? \rangle$$

$$[\![a_1\ a_2]\!] = \langle [\![a_1]\!] : ? \triangleleft ? \to ? \rangle\ [\![a_2]\!]$$

By construction, for every a in Ω, we have $|[\![a]\!]| = a$ and $\Gamma \vdash_{\mathcal{D}} [\![a]\!] : ?$ where Γ binds free variables of a to ?.

□

2 Soundness

We prove the soundness of $\mathcal{D}$ via subject reduction and progress.

Lemma 1 (Subject reduction) *If $\Gamma \vdash_{\mathcal{D}} M : \tau$ and $M \longrightarrow_{\mathcal{D}} M'$, then $\Gamma \vdash_{\mathcal{D}} M' : \tau$.*

Question 8 *Prove the subject reduction lemma. Please, clearly explain the structure of the proof, list all cases, and provide full detail in the cases of* Cast-App *and* Cast-Neg*; the proofs of the other cases can be omitted.*
Answer: We assume $\Gamma \vdash_{\mathcal{D}} M : \tau$ (**1**) and $M \longrightarrow_{\mathcal{D}} M'$ and show $\Gamma \vdash_{\mathcal{D}} M' : \tau$. As usual, the proof is by induction on $M \longrightarrow_{\mathcal{D}} M'$.

Case Context. Omitted. (This is the only case that relies on the induction hypothesis.)

Case Beta. Omitted.

Case Cast-App. The hypothesis is

$$\langle V_1 : \tau_2 \to \tau_1 \bowtie \tau_2' \to \tau_1' \rangle\ V_2 \longrightarrow_{\mathcal{D}} \langle (V_1\ \langle V_2 : \tau_2' \bowtie^{\perp} \tau_2 \rangle) : \tau_1 \bowtie \tau_1' \rangle$$

By inversion of typing (1), τ must be τ_1' and we have $\Gamma \vdash V_1 : \tau_2 \to \tau_1$ (**2**) and $\Gamma \vdash V_2 : \tau_2'$ (**3**) and $\tau_2 \to \tau_1 \preceq^{\bowtie} \tau_2' \to \tau_1'$, that is, $\tau_2 \preceq^{\bowtie} \tau_2'$ (**4**) and $\tau_1 \preceq^{\bowtie} \tau_1'$ (**5**). (4) can equivalently be written $\tau_2' \preceq^{\bowtie^{\perp}} \tau_2$ (**6**). From (3) and (6), we have $\Gamma \vdash \langle V_2 : \tau_2' \bowtie^{\perp} \tau_2 \rangle : \tau_2$ by Cast. Hence $\Gamma \vdash V_1\ \langle V_2 : \tau_2' \bowtie^{\perp} \tau_2 \rangle : \tau_1$ by App. Finally, by combination with (5), we have $\Gamma \vdash \langle V_1\ \langle V_2 : \tau_2' \bowtie^{\perp} \tau_2 \rangle : \tau_1 \bowtie \tau_1' \rangle : \tau_1'$, as expected.

Case Cast-Id. Trivial.

Case Cast-Neg. The hypothesis is

$$\langle V : ? \lhd \tau \rangle \longrightarrow_{\mathcal{D}} \langle \langle V : ? \lhd \sharp\tau \rangle : \sharp\tau \lhd \tau \rangle$$

By inversion of typing (1), we have $\Gamma \vdash V : ?$ and $? \preceq \tau$ (**7**). Since $? \preceq \sharp\tau$ (by definiition of $\preceq$) and $\sharp\tau \preceq \tau$ (easy check), we have $\Gamma \vdash \langle \langle V : ? \lhd \sharp\tau \rangle : \sharp\tau \lhd \tau \rangle : \tau$.

Case Cast-Pos. Similar to Cast-Neg.

Case Cast-Cancel. Trivial. □

As usual, the proof of progress relies on the following classification lemma:

Lemma 2 (Classification)

- *If* $\vdash_{\mathcal{D}} V : \mathsf{unit}$, *then* V *is either the value* $()$ *or a delayed cast* $\langle V : \mathsf{unit} \bowtie \mathsf{unit} \rangle$.
- *If* $\vdash_{\mathcal{D}} V : \tau \to \tau'$, *then* V *is* ...
- *If* $\vdash_{\mathcal{D}} V : \tau$ *is* $?$, *then* V *is* ...

Question 9 *Complete the last two cases in the statement of the classification lemma.*
Answer:

- If $\vdash_{\mathcal{D}} V : \tau \to \tau'$, then V is either a function $\lambda x : \tau.\, M$ or a delayed functional cast $\langle V' : \tau_1 \to \tau_1' \bowtie \tau \to \tau' \rangle$.
- If $\vdash_{\mathcal{D}} V : ?$, then V is a cast of the form $\langle V' : \zeta \rhd ? \rangle$. □

A *cast error* is a term of the form $\bar{E}[\langle \langle V : \zeta \rhd ? \rangle : ? \lhd \zeta' \rangle]$ with $\zeta \neq \zeta'$ where $\bar{E}$ stands for a deep evaluation context, defined as usual as: $\bar{E} ::= [\,] \mid E[\bar{E}]$.

Lemma 3 (Progress) *If* $\vdash_{\mathcal{D}} M : \tau$ *then* M *reduces or* M *is a value or a* cast error.

The proof is long and a bit involved, so we do not ask you to prove this lemma here.

3 Gradual typing

We write $\lceil\cdot\rceil$ for the *partial* erasure function that removes *negative* casts from $\mathcal{D}$ and is *undefined on positive casts*:

$$\lceil()\rceil = () \qquad \lceil\lambda x:\tau.\, M\rceil = \lambda x:\tau.\, \lceil M\rceil \qquad \lceil M_1\ M_2\rceil = \lceil M_1\rceil\ \lceil M_2\rceil \qquad \lceil\langle M:\tau\lhd\tau'\rangle\rceil = \lceil M\rceil$$

Notice that the last rule applies only to *negative* casts.

We now introduce $\mathcal{G}$ as a variant of $\mathcal{D}$, defined as:

$$\Gamma \vdash_{\mathcal{G}} M : \tau \iff \exists\tau, M' \in \mathcal{D},\ M = \lceil M'\rceil \ \wedge\ \Gamma \vdash_{\mathcal{D}} M' : \tau$$

In other words, a term of $\mathcal{G}$ corresponds to a term of $\mathcal{D}$ with only negative casts $\langle M:\tau\lhd\tau'\rangle$, no positive casts $\langle M:\tau\rhd\tau'\rangle$. The casts are implicit in $\mathcal{G}$, present in the type derivation but not in the term syntax. Note that the types τ in the function annotations $\lambda x:\tau.\, M$ are types of $\mathcal{D}$, including the unknown type $?$.

Below, to show that $\vdash_{\mathcal{G}} M : \tau$, you just need to present a term $M' \in \mathcal{D}$, called a witness, such that $\vdash_{\mathcal{D}} M' : \tau$ and $\lceil M'\rceil = M$ (no further justification being required).

Question 10 *Let M_3 be the term $\lambda x:?.\, x\ x$. Show that $\vdash_{\mathcal{G}} M_3 : ? \to \mathsf{unit}$ by giving a witness term M_3'. Is M_3' unique? If not, give another witness M_3''.*
Answer:

$$\begin{aligned} M_3' &\overset{\text{def}}{=} \lambda x:?.\, \langle x : ? \lhd ? \to \mathsf{unit}\rangle\ x \\ M_3'' &\overset{\text{def}}{=} \lambda x:?.\, \langle\langle x : ? \lhd ? \to ?\rangle\ x : ? \lhd \mathsf{unit}\rangle \\ M_3''' &\overset{\text{def}}{=} \langle\lambda x:?.\, \langle x : ? \lhd ? \to ?\rangle\ x : ? \to ? \lhd ? \to \mathsf{unit}\rangle \end{aligned}$$

(There are many more solutions, for instance inserting identity coercions...) □

Question 11 *Let M_4 be the term $()\ ()$. Is M_4 well-typed in $\mathcal{G}$? If so, give a witness term M_4'.*
Answer: M_4 is not typable in $\mathcal{G}$. *(We have an obvious type error; we can only gain information and not lose some to* hide *the type error within a dynamic check.)* □

Question 12 *Let M_5 be the term $(\lambda x:?.\, x\ ())\ ()$. Is M_5 well-typed in $\mathcal{G}$? If so, give a witness M_5'.*
Answer: We may take for M_5' the term M_2 of question 2, that is:

$$M_5' \overset{\text{def}}{=} \langle(\lambda x:?.\, \langle x : ? \lhd \mathsf{unit} \to \mathsf{unit}\rangle\ ()) : ? \to \mathsf{unit} \lhd \mathsf{unit} \to \mathsf{unit}\rangle\ ()$$

Another (simpler) choice, among many others, is, for any τ:

$$M_5'' \overset{\text{def}}{=} (\lambda x:?.\, \langle x : ? \lhd \mathsf{unit} \to \mathsf{unit} \to \tau\rangle\ ())\ ()$$ □

We may also express typing directly in $\mathcal{G}$. This is done by reusing the typing rules of $\mathcal{D}$, except rule Cast, which is replaced by the materialization rule:

$$\textsc{Mat}\quad \frac{\Gamma \vdash_{\mathcal{G}} M : \tau \qquad \tau \preceq \tau'}{\Gamma \vdash_{\mathcal{G}} M : \tau'}$$

Below, you may consider the two definitions equivalent and use either one.

Question 13 *Let M_6 be the identity $\lambda x{:}?.\,x$. Describe* all *the types of M_6,* i.e. *the types τ such that $\vdash_{\mathcal{G}} M_6 : \tau$? (Justify.)*
Answer: They are all arrow types, *i.e.* types of the form $\tau_1 \to \tau_2$.

Indeed, since $\vdash_{\mathcal{G}} M_6 : ? \to ?$ and $? \to ? \preceq \tau_1 \to \tau_2$ for any τ_1 and τ_2, we have $\vdash_{\mathcal{G}} M_6 : \tau$ by rule Mat.

Conversely, Assume $\vdash_{\mathcal{G}} M_6 : \tau$. We show that τ is an arrow type $\tau_1 \to \tau_2$ by induction on the derivation. By inversion of typing, the derivation may end either with rule Abs and the conclusion is an arrow type, or with rule Mat: in this case $\vdash_{\mathcal{G}} M_6 : \tau'$ and $\tau' \preceq \tau$. By induction hypothesis τ' is an arrow type $\tau'_1 \to \tau'_2$. By inversion of $\preceq$, τ must then be an arrow type $\tau_1 \to \tau_2$. □

Question 14 *Is it the case that $\vdash_{\mathcal{S}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$ iff $\vdash_{\mathcal{G}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$?*
Answer: Yes, the property holds. The direct implication is obvious, since if $\vdash_{\mathcal{S}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$ then $\vdash_{\mathcal{D}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$ holds and $\lceil M^{\mathcal{S}} \rceil = M^{\mathcal{S}}$.

Conversely, assume $\vdash_{\mathcal{G}} M^{\mathcal{S}} : \tau^{\mathcal{S}}$. We show by induction on the derivation that rule Mat cannot be used *except with identical types of the form* $\langle M : \tau^{\mathcal{S}} \lhd \tau^{\mathcal{S}} \rangle$. Indeed, the premise of Cast is then by induction hypothesis always of the form $\tau^{\mathcal{S}} \preceq \tau'$ which then implies that $\tau' = \tau^{\mathcal{S}}$. □

Precision We extend the materialization relation $\preceq$ to typing contexts, point-wise:

$$\Gamma \preceq \Gamma' \iff \mathit{dom}\,\Gamma = \mathit{dom}\,\Gamma' \;\wedge\; \forall x \in \mathit{dom}\,\Gamma,\ \Gamma(x) \preceq \Gamma'(x)$$

We also extend the materialization relation $\preceq$ from types to terms of $\mathcal{G}$ as follows:

$$\text{P-Var}\quad x \preceq x \qquad \text{P-Const}\quad () \preceq () \qquad \text{P-App}\quad \frac{M \preceq M' \qquad N \preceq N'}{M\ N \preceq M'\ N'} \qquad \text{P-Abs}\quad \frac{M \preceq M' \qquad \tau \preceq \tau'}{\lambda x{:}\tau.\,M \preceq \lambda x{:}\tau'.\,M'}$$

Lemma 4 (Monotonicity, admitted) *If $\Gamma \vdash_{\mathcal{G}} M : \tau$ and $\Gamma \succeq \Gamma'$ then $\Gamma' \vdash_{\mathcal{G}} M : \tau$.*

Lemma 5 (Static gradual guarantee) *A decrease in precision preserves well-typedness: If $\vdash_{\mathcal{G}} M : \tau$ and $M \succeq M'$ then $\vdash_{\mathcal{G}} M' : \tau$.*

Question 15 *Prove the static gradual guarantee.*
Answer: We show the stronger property that *if $\Gamma \vdash_{\mathcal{G}} M : \tau$ (**1**) and $M \succeq M'$ (**2**) then $\Gamma \vdash_{\mathcal{G}} M' : \tau$ (**3**)*, by simultaneous induction on the typing derivation in $\mathcal{G}$ and of $M \succeq M'$.

Case Var*: M is x or* $()$. Then M' is M, hence (3) is just (1).

Case App*: M is $M_1\ M_2$.* By inversion of typing (1), we have $\Gamma \vdash M_1 : \tau_2 \to \tau$ and $\Gamma \vdash M_2 : \tau_2$. By inversion of (2), we know that M' is of the form $M'_1\ M'_2$ with $M_1 \succeq M'_1$ and $M_2 \succeq M'_2$. Hence, by induction hypothesis, we have $\Gamma \vdash M'_1 : \tau_2 \to \tau$ and $\Gamma \vdash M'_2 : \tau_2$. We conclude $\Gamma \vdash M' : \tau$ by rule App.

Case Abs*: M is $\lambda x{:}\tau_1.\,N$.* By inversion of typing, we know that $\Gamma, x : \tau_1 \vdash M_1 : \tau_2$ (**4**) where τ is $\tau_1 \to \tau_2$. By (2), we know that M' is $\lambda x{:}\tau'_1.\,M'_1$ with $\tau_1 \succeq \tau'_1$ (**5**) and $M_1 \succeq M'_1$. By induction hypothesis applied to (4), we have $\Gamma, x : \tau_1 \vdash M'_1 : \tau_2$. By monotonicity and (5), we have $\Gamma, x : \tau'_1 \vdash M'_1 : \tau_2$. By rule Abs, we have $\Gamma \vdash \lambda x{:}\tau'_1.\,M' : \tau'_1 \to \tau_2$. By (5), we have $\Gamma \vdash \lambda x{:}\tau'_1.\,M' : \tau_1 \to \tau_2$, *i.e.* (3).

Case Mat. By inversion of typing, we have $\Gamma \vdash M : \tau_1$ and $\tau_1 \preceq \tau$ (**6**). By induction hypothesis, we have $\Gamma \vdash M' : \tau_1$ and we conclude $\Gamma \vdash M' : \tau$ by Rule Mat and (6). □

Question 16 *Show that subject reduction would not hold in $\mathcal{G}$, if we reduced terms with rules* Beta *and* Context*.*
Answer: Take the term $M_7 \overset{\mathsf{def}}{=} M_6\ ()\ ()$. We have $\vdash M_7 : \tau$ for any τ, since by question 13 $\vdash M_6 : \mathsf{unit} \to \mathsf{unit} \to \tau$. We would then have $M_7 \longrightarrow_{\mathcal{G}}^{*} ()\ () = M_4$, while $\not\vdash_{\mathcal{G}} M_4 : \tau$, as explained in question 11.

Another choice is to take M_5 from question 12 for M_7: it is well-typed by question 12, but also reduces to M_4, which is ill-typed by quuestion 11. □

We instead give the semantics of $\mathcal{G}$ by elaboration of well-typed terms into $\mathcal{D}$. We write $\Gamma \vdash M \rightsquigarrow M' : \tau$ to mean that $\Gamma \vdash_{\mathcal{D}} M' : \tau$ and $M = \lceil M' \rceil$.

Question 17 (Semantics) *Our proposal is that the semantics of a well-typed term $\vdash_{\mathcal{G}} M : \tau$ be given by the semantics of any term M' such that $\Gamma \vdash M \rightsquigarrow M' : \tau$. Why is this proposal problematic? (Explain the problem but do not solve it.)*
Answer: Since the elaboration of a term is not unique, the semantics of a term is ill-defined—unless all possible elaborations of the same term would always have the same semantics. □

4 Progress in $\mathcal{D}$

The following question is optional, as it is long and the case analysis is somewhat involved. Answer it only if you have sufficient time.

Question 18 (Optional) *Prove the progress lemma.*
Answer: The proof is by induction on M. We assume that $\vdash M : \tau$ (**1**). We assume that M does not reduce and is not a cast error and show that the only remaining case is that M is a value. Notice that cast errors are closed by evaluation contexts. Hence, if M is of the form $E[M']$ then M' does reduce and is not a cast error; since typing is compositional M' is also well-typed and therefore, it must be a value by induction hypothesis (**2**).

Case M is $()$ or a function $\lambda x : \tau.\, M'$. Then M is a value, as expected.

Case M is an application $M_1\ M_2$. Since $[\,]\ M_2$ is an evaluation context, M_1 must be a value by (2). Hence M_2 is also an evaluation context and M_2 must also be a value. By inversion of typing $\vdash M_1 : \tau_2 \to \tau$. By the classification lemma, either M_1 is a function, in which case M reduces by Beta, or M_1 is a functional coercion, in which case M reduces by Cast-App. Contradiction.

Case M is a coercion $\langle M' {:} \tau' \bowtie \tau'' \rangle$. By inversion of typing, τ'' must be τ and $\tau' \preceq^{\bowtie} \tau$ (**3**). Since $\langle [\,] : \tau' \bowtie \tau \rangle$ is an evaluation context, M' must be a value V. We reason by cases on $\sharp\tau' \bowtie \sharp\tau$:

- Case $? \bowtie ?$. This implies that both τ' and τ are $?$. Then M reduces by Cast-Id. Contradiction.
- Case $? \bowtie \zeta$. This implies τ' is $?$. By (3), $\bowtie$ must be $\triangleleft$. Moreover, $\sharp\sigma$ is ζ, since otherwise M would reduce by Cast-Neg. By inversion of typing, we have $\vdash V : ?$. By the classification lemma, V must be of the form $\langle V' : \zeta' \triangleright ? \rangle$. Moreover, $\zeta' = \zeta$, since otherwise, M would be the cast error $\langle \langle V' : \zeta' \triangleright ? \rangle : ? \triangleleft \zeta \rangle$. Then, M reduces by rule Cast-Cancel. Contradiction.

- Case $\zeta \bowtie ?$. This implies τ' is $?$. By (3), $\bowtie$ must be $\triangleright$. Moreover, $\sharp\sigma$ is ζ, since otherwise M would reduce by Cast-Pos. Then, M is a superficial positive cast, which is a value.
- Case $\zeta \bowtie \zeta'$. By (3), we have $\zeta = \zeta'$. Then, M is a deep-cast value. $\square$