Structural ML type inference

Final exam, MPRI 2-4

March 11th, 2025 — Duration: 2h45

Answers are judged by their correctness, clarity, conciseness, and accuracy.

Although the questions are in English, it is permitted to answer in French or English. It is recommended to answer in French if this is your mother tongue.

1 Type assignment

Throughout this exam, we are interested in pure λ-terms extended with a polymorphic let-expression. The syntax of terms is recalled in Figure 1a while the syntax of monotypes is given in Figure 1b. Flexible type variables correspond to the notion of weak polymorphic type in OCaml (which are pretty-printed as `'_weak1`, `'_weak2`, *etc.* by the OCaml type-checker): they act as a placeholder for a *single* type that is currently unknown.

x, y, z	$\ldots$		(*term variables*)				
t, s	$::=$		(*terms*)				
	$\mid$	x	(*variable*)	X, Y	$\ldots$		(*flexible type var.*)
	$\mid$	$\lambda x.\, t$	(*abstraction*)	τ, ν	$::=$		(*types*)
	$\mid$	$t\ s$	(*application*)		$\mid$	X	(*flexible type var.*)
	$\mid$	$\text{let}\, x = t \,\text{in}\, s$	(*polymorphic let*)		$\mid$	$\tau \to \nu$	(*function type*)

(a) Terms

(b) Types

Γ	$::=$		(*context*)	Δ	$::=$		(*context suffix*)
	$\mid$	ϵ	(*empty*)		$\mid$	$\cdot$	
	$\mid$	Γ, X	(*type variable*)		$\mid$	X, Δ	
	$\mid$	$\Gamma, X := \tau$	(*type var. instantiation*)		$\mid$	$X := \tau, \Delta$	
	$\mid$	$\Gamma, x : \sigma$	(*term variable*)				
	$\mid$	$\Gamma, \Diamond$	(*rank separator*)	σ	$::=$	$\forall \Delta.\, \tau$	(*type scheme*)

(c) Contexts

Figure 1: Syntactic categories

To account for type variable instantiation and let-polymorphism, we adopt a structured presentation of the typing context (Figure 1c). Aside from the usual extension of a context Γ with type and term variable declarations (respectively written Γ, X and $\Gamma, x : \sigma$), it can also be extended to register a type variable instantiation (written $\Gamma, X := \tau$). Moreover, it can be decorated with an explicit rank separator (written $\Gamma, \Diamond$), which will play a key role to

support an algorithmic treatment of type inference (Section 3). Building upon contexts, we define type schemes σ as pairs of a monotype quantified over a particular context suffix Δ. Once quantified over, these type variables become *rigid*. For instance, the type scheme

$$\forall X_0, X_1, X_2 := X_0 \to X_1, X_3.\, X_3 \to X_2$$

denotes the System F type

$$\forall\, \alpha, \beta, \gamma.\, \gamma \to (\alpha \to \beta)$$

As usual, monotypes are a special case of type schemes: we shall write τ to stand for the empty type scheme "$\forall \cdot . \tau$".

We specify a type system with let-polymorphism through a declarative type assignment (Figure 2, page 7), building upon valid structured contexts, well-kinded types, well-kinded schemes and a contextualized notion of type equality (Figure 4, page 8).

Question 1. *Exhibit a type τ such that the type assignment*

$$\epsilon, X \vdash \mathit{let}\, f = \lambda x.\, x\ \mathit{in}\ f\ f : \tau$$

holds, including a proof that the type assignment is indeed satisfied.

□

Question 2. *Asked whether or not there exists a context Γ such that the equality judgment*

$$\Gamma \vdash X \to X \to Y \equiv (Y \to Z) \to Z$$

holds, a student claims that the contexts

$$\epsilon, Z := X \to Y, X := Y \to Z$$

and

$$\epsilon, X := Y \to Z, Z := X \to Y$$

are two valid solutions. Explain why both answers are incorrect.

□

2 Type substitution

A type substitution consists of a sequence of type variable instantiations. Its syntax is

$$\begin{array}{rcll} \gamma & ::= & & (\mathit{environment}) \\ & \mid & \emptyset & (\mathit{empty}) \\ & \mid & \gamma[X \mapsto \tau] & (\mathit{instantiation}) \end{array}$$

Type substitutions act on types as a parallel substitution of type variables, *i.e.*

$$\begin{array}{rcll} \gamma(X) & = & \tau & \text{if } [X \mapsto \tau] \in \gamma \\ \gamma(X) & = & X & \text{otherwise} \\ \gamma(\tau \to \nu) & = & \gamma(\tau) \to \gamma(\nu) & \end{array}$$

Question 3. *Extend this definition so that it operates over type schemes: given any well-kinded type scheme σ, define $\gamma(\sigma)$.*

□

We say that a type substitution γ *witnesses a context extension* from Γ_0 to Γ_1 when the judgment $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$ holds (Figure 3, page 7). Intuitively, the context Γ_1 corresponds to a "more concrete" instance of Γ_0, wherein some type variables of Γ_0 may have been instantiated. Note, however, that both the rank separator and term variable declarations induce a critical restriction on type substitutions. The following two questions illustrate key aspects of this design choice.

Question 4. *Prove that there exists a type substitution γ_1 that witnesses a context extension from $\epsilon, W, \Diamond, X, Y, Z$ to $\epsilon, X, Y, W := X \to Y, \Diamond, Z$.*

□

Question 5. *Argue whether or not the type substitution $\emptyset[X \mapsto Y][Y \mapsto Y]$ witnesses a context extension from $\epsilon, X, \Diamond, Y$ to $\epsilon, \Diamond, Y$. If it is the case, provide a proof. Otherwise, identify and explain the root cause of the failure.*

□

Question 6. *Let $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$ be a type substitution.*
Prove that, for any context suffix Δ such that $\Gamma_1, \Delta \vdash \textbf{\textit{ctx}}$, we also have $\gamma : \Gamma_0 \sqsubseteq \Gamma_1, \Delta$.

□

Question 7. *Let Γ be any valid context. Define a type substitution ι_Γ such that*

- *it amounts to the identity substitution, i.e. for any type τ well-kinded in Γ, we have $\Gamma \vdash \iota_\Gamma(\tau) \equiv \tau$, and*
- *it witnesses a context extension from Γ to itself, i.e. we have $\iota_\Gamma : \Gamma \sqsubseteq \Gamma$.*

Aside from the definition of ι_Γ itself, your answer should also prove the validity of ι_Γ as a suitable witness. It is not necessary to prove that it is an identity.

□

Question 8. *Let $\gamma_1 : \Gamma_0 \sqsubseteq \Gamma_1$ and $\gamma_2 : \Gamma_1 \sqsubseteq \Gamma_2$ be two type substitutions.*
Define a type substitution $\gamma_2 \circ \gamma_1$ such that

1. *for any type τ well-kinded in Γ_0, we have*

$$\Gamma_2 \vdash (\gamma_2 \circ \gamma_1)(\tau) \equiv \gamma_2(\gamma_1(\tau))$$

2. *$\gamma_2 \circ \gamma_1$ witnesses a context extension from Γ_0 to Γ_2, i.e. we have $\gamma_2 \circ \gamma_1 : \Gamma_0 \sqsubseteq \Gamma_2$*

To answer this question, no proof is needed: the definition of the composition operator is enough.

□

As hinted before, the rank separator (and, similarly, term variable declarations) play a key rôle in structuring contexts. This gets reflected in type substitutions through a factorization property. In effect, we have that if $\gamma : \Gamma_0, \Diamond, \Gamma_2 \sqsubseteq \Gamma$, then there exists Γ_1 and Γ_3 such that

1. the rank separator is carried over by the type substitution, *i.e.* $\Gamma = \Gamma_1, \Diamond, \Gamma_3$

2. we can extract a restriction, denoted $\gamma|_{\Gamma_0}$, of the subtitution γ that witnesses an extension from the separated prefix Γ_0 to the separated prefix Γ_1, *i.e.* such that we have

$$\gamma|_{\Gamma_0} : \Gamma_0 \sqsubseteq \Gamma_1$$

which is related to γ (in a manner we shall leave unspecified here, for brevity).

In the remainder of the exam, we assume that the validity, type equality and type assignment judgments are *stable* by context extension: for any $\gamma : \Gamma_1 \sqsubseteq \Gamma_2$, we have

(Stable-Ctxt) if $\Gamma_1 \vdash \mathbf{ctx}$ then $\Gamma_2 \vdash \mathbf{ctx}$

(Stable-Kind) if $\Gamma_1 \vdash \tau$ then $\Gamma_2 \vdash \gamma(\tau)$

(Stable-Scheme) if $\Gamma_1 \vdash \sigma$ then $\Gamma_2 \vdash \gamma(\sigma)$

(Stable-Eq) if $\Gamma_1 \vdash \tau \equiv \nu$ then $\Gamma_2 \vdash \gamma(\tau) \equiv \gamma(\nu)$

(Stable-Type) if $\Gamma_1 \vdash t : \tau$ then $\Gamma_2 \vdash t : \gamma(\tau)$

Type substitutions let us internalize the instantiation of type schemes, as witnessed by the following property.

Property 1. *Let Γ be a valid context. Let ν be a well-kinded type in context Γ and $\forall\Delta.\,\tau$ be a well-kinded type scheme in the same context.*

If $\Gamma \vdash \forall\Delta.\,\tau \succ \nu$ then there exists a type substitution $\zeta : \Gamma, \Delta \sqsubseteq \Gamma$ such that $\Gamma \vdash \zeta(\tau) \equiv \nu$.

Two type substitutions $\gamma_0 : \Gamma_0 \sqsubseteq \Gamma_1$ and $\gamma_1 : \Gamma_0 \sqsubseteq \Gamma_1$ are *equivalent*, written $\gamma_0 \equiv \gamma_1$, if their action on types is identical, *i.e.* for any τ such that $\Gamma_0 \vdash \tau$, we have $\Gamma_1 \vdash \gamma_0(\tau) \equiv \gamma_1(\tau)$.

A type substitution $\gamma_2 : \Gamma_0 \sqsubseteq \Gamma_2$ *factors through* another type substitution $\gamma_1 : \Gamma_0 \sqsubseteq \Gamma_1$ if there exists a *co-factor* $\zeta : \Gamma_1 \sqsubseteq \Gamma_2$ such that $\gamma_2 \equiv \zeta \circ \gamma_1$.

Question 9. *Exhibit a co-factor ζ of*

$$\gamma_2 \triangleq \emptyset[W \mapsto X \to X][X \mapsto X][Y \mapsto X][Z \mapsto Z]$$

from $\epsilon, W, \Diamond, X, Y, Z$ to $\epsilon, X, W := X \to X, Z, \Diamond$ trough the context extension γ_1 defined in Question 4.

□

3 Structural type inference

Definition 1. *We postulate the existence of a unification judgment*

$$\boxed{\Gamma_0 \vdash \tau \equiv \nu \dashv \Gamma_1}$$

Unifying type τ with ν in context Γ_0 results in context Γ_1

satisfying the following properties:

Soundness: *If $\Gamma_0 \vdash \tau \equiv \nu \dashv \Gamma_1$ then there exists a type substitution $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$ such that*

1. *γ unifies τ and ν, i.e. $\Gamma_1 \vdash \gamma(\tau) \equiv \gamma(\nu)$*
2. *γ is minimal: any other type substitution γ_1 unifying τ and ν factors through γ.*

Completeness: *If*

1. $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$,
2. $\Gamma_0 \vdash \tau$,
3. $\Gamma_0 \vdash \nu$, *and*
4. $\Gamma_1 \vdash \gamma(\tau) \equiv \gamma(\nu)$

then there exists Γ_2 *such that* $\Gamma_0 \vdash \tau \equiv \nu \dashv \Gamma_2$.

Definition 2. *Let* Γ_0 *be a context and* t *be a term.*
A solution to the type inference *of* t *in context* Γ_0 *is a triple of*

1. *an output typing context* Γ_1,
2. *a type* τ *such that* $\Gamma_1 \vdash t : \tau$, *and*
3. *a type substitution* $\gamma_1 : \Gamma_0 \sqsubseteq \Gamma_1$.

A solution is minimal *if any other solution* $(\Gamma_2, \nu, \gamma_2 : \Gamma_0 \sqsubseteq \Gamma_2)$ *factors through it with cofactor* ζ *such that* $\Gamma_2 \vdash \zeta(\tau) \equiv \nu$.

Question 10. *We define*

- $\Gamma \triangleq \epsilon, U, \Diamond, Z, g : Z, Y, f : Y, X, x : X$
- $t \triangleq g\ (f\ x)$

Exhibit a minimal solution (including the target context of the type substitution) to the type inference of t *in context* Γ. *It is not required to prove minimality.*

□

In Figure 5, we sketch an algorithmic type inference system. The rules Infer-Let and Infer-App are to be completed in Questions 11 and 12. We expect the following two properties to hold:

Soundness of type inference: If $\Gamma_0 \vdash t : \tau \dashv \Gamma_1$ then there exists a type substitution $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$ such that (Γ_1, τ, γ) is a minimal solution to the type inference of t in context Γ_0 ;

Completeness of type inference: If the type substitution $\gamma : \Gamma_0 \sqsubseteq \Gamma_1$ is a solution to the type inference of a term t in context Γ_0, then there exists a context Γ_2 and a type ν such that

$$\Gamma_0 \vdash t : \nu \dashv \Gamma_2$$

Question 11. *Propose a completed definition of the* Infer-Let *rule (without proof).*

□

Question 12. *Propose a completed definition of the* Infer-App *rule (without proof).*

□

The following questions tackle more challenging, ***research-level*** *problems. You may not be able to address them in full during the remaining time of the exam. However, feel free to* ***sketch*** *how you would attack them, clearly identifying the roadblocks you are stumbling upon, as if you were writing notes to another researcher or a PhD supervisor.*

Question 13. *Let*

1. $\gamma_1 : \Gamma_0 \sqsubseteq \Gamma_1$ *be a minimal solution to the type inference of* t *with type* τ*, and*
2. $\gamma_2 : \Gamma_1 \sqsubseteq \Gamma_2$ *be a minimal solution to the type inference of* s *with type* ν*.*

Prove that $\gamma_2 \circ \gamma_1 : \Gamma_0 \sqsubseteq \Gamma_2$ *is also a minimal solution to the type inference of both* t *and* s simultaneously*, i.e. any other solution* $\gamma' : \Gamma_0 \sqsubseteq \Gamma'_2$ *such that* $\Gamma'_2 \vdash t : \tau'$ *and* $\Gamma'_2 \vdash s : \nu'$ *factors through* $\gamma_2 \circ \gamma_1$*.*

□

This last property justifies the validity of a greedy implementation of type inference: it shows that locally minimal solutions of type inference (as well as unification solutions) compose to provide a globally minimal solution.

Question 14. *Prove that the soundness property holds in the* Infer-Var *case.*

If you find that some auxiliary lemmas are needed, give the statement of each necessary lemma, without proof.

□

Question 15. *Prove that if* $\gamma_1 : \Gamma_0, \Diamond \sqsubseteq \Gamma_1, \Diamond, \Delta$ *is a minimal solution to the type inference of a term* t *with inferred type* τ*, then for any type substitution* $\gamma_2 : \Gamma_0 \sqsubseteq \Gamma_2$*, suffix context* Δ' *and type* ν *such that* $\Gamma_2, \Diamond, \Delta' \vdash t : \nu$*, we have that* γ_2 *factors through* γ_1 *with cofactor* ζ *verifying* $\Gamma_2 \vdash \zeta(\forall \Delta.\, \tau) \succ \forall \Delta'.\, \nu$*.*

If you find that some auxiliary lemmas are needed, give the statement of each necessary lemma, without proof.

□

Question 16. *Propose a declarative specification (by means of a system of inference rules) for the unification judgment* $\Gamma_0 \vdash \tau \equiv \nu \dashv \Gamma_1$ *axiomatized in Definition 1. The soundness and completeness proofs are not necessary.*

This is an open ended, bonus question.

□

$$\boxed{\Gamma \vdash t : \tau}$$

Term t has type τ in context Γ

$$\frac{x : \forall \Delta.\, \nu \in \Gamma \qquad \Gamma \vdash \forall \Delta.\, \nu \succ \tau}{\Gamma \vdash x : \tau}\ \textsc{Chk-Var} \qquad \frac{\Gamma, x : \tau \vdash t : \nu}{\Gamma \vdash \lambda x.\, t : \tau \to \nu}\ \textsc{Chk-Lam}$$

$$\frac{\Gamma \vdash t : \tau \to \nu \qquad \Gamma \vdash s : \tau}{\Gamma \vdash t\ s : \nu}\ \textsc{Chk-App} \qquad \frac{\Gamma, \Diamond, \Delta \vdash s : \nu \qquad \Gamma, x : \forall \Delta.\, \nu \vdash t : \tau}{\Gamma \vdash \mathrm{let}\, x = s\, \mathrm{in}\, t : \tau}\ \textsc{Chk-Let}$$

$$\boxed{\Gamma \vdash \sigma_0 \succ \sigma_1}$$

Scheme σ_0 is more general than scheme σ_1 in context Γ

$$\frac{\Gamma \vdash \tau \equiv \nu}{\Gamma \vdash \tau \succ \nu}\ \textsc{Eq} \qquad \frac{\Gamma, \Delta \vdash \sigma_0 \succ \nu}{\Gamma \vdash \sigma_0 \succ \forall \Delta.\, \nu}\ \textsc{Gen} \qquad \frac{\Gamma, X := \tau_1 \vdash \forall \Delta.\, \tau \succ \nu}{\Gamma \vdash \forall X := \tau_1, \Delta.\, \tau \succ \nu}\ \textsc{Def}$$

$$\frac{\Gamma \vdash \tau_1 \qquad \Gamma, X := \tau_1 \vdash \forall \Delta.\, \tau \succ \nu}{\Gamma \vdash \forall X, \Delta.\, \tau \succ \nu}\ \textsc{Inst}$$

Figure 2: Type assignment system for ML

$$\boxed{\gamma : \Gamma_0 \sqsubseteq \Gamma_1}$$

γ witnesses a context extension from context Γ_0 to context Γ_1

$$\frac{}{\emptyset : \epsilon \sqsubseteq \epsilon, \Delta}\ \textsc{Subst-Emp} \qquad \frac{\gamma : \Gamma_0 \sqsubseteq \Gamma_1 \qquad \Gamma_1 \vdash \tau}{\gamma[X \mapsto \tau] : \Gamma_0, X \sqsubseteq \Gamma_1}\ \textsc{Subst-FlexVar}$$

$$\frac{\gamma : \Gamma_0 \sqsubseteq \Gamma_1 \qquad \Gamma_1 \vdash \tau \equiv \gamma(\nu)}{\gamma[X \mapsto \tau] : \Gamma_0, X := \nu \sqsubseteq \Gamma_1}\ \textsc{Subst-Def} \qquad \frac{\gamma : \Gamma_0 \sqsubseteq \Gamma_1}{\gamma : \Gamma_0, x : \sigma \sqsubseteq \Gamma_1, x : \gamma(\sigma), \Delta}\ \textsc{Subst-TeVar}$$

$$\frac{\gamma : \Gamma_0 \sqsubseteq \Gamma_1}{\gamma : \Gamma_0, \Diamond \sqsubseteq \Gamma_1, \Diamond, \Delta}\ \textsc{Subst-Locality}$$

Figure 3: Type substitution structure

$$\boxed{\Gamma \vdash \mathbf{ctx}}$$

Γ is a valid context

$$\frac{}{\epsilon \vdash \mathbf{ctx}}\ \textsc{Valid-End} \qquad \frac{\Gamma \vdash \mathbf{ctx} \qquad X \# \Gamma}{\Gamma, X \vdash \mathbf{ctx}}\ \textsc{Valid-Flex} \qquad \frac{\Gamma \vdash \tau \qquad X \# \Gamma}{\Gamma, X := \tau \vdash \mathbf{ctx}}\ \textsc{Valid-Def}$$

$$\frac{\Gamma \vdash \sigma \qquad x \# \Gamma}{\Gamma, x : \sigma \vdash \mathbf{ctx}}\ \textsc{Valid-TeVar} \qquad \frac{\Gamma \vdash \mathbf{ctx}}{\Gamma, \Diamond \vdash \mathbf{ctx}}\ \textsc{Valid-Locality}$$

$$\boxed{\Gamma \vdash \sigma}$$

Type scheme σ is well-kinded in context Γ

$$\frac{\Gamma, \Delta \vdash \nu}{\Gamma \vdash \forall \Delta.\, \nu}\ \textsc{Valid-Forall}$$

$$\boxed{\Gamma \vdash \tau}$$

Type τ is well-kinded in context Γ

$$\frac{\Gamma \vdash \mathbf{ctx} \qquad X \in \Gamma}{\Gamma \vdash X}\ \textsc{Valid-FlexVar} \qquad \frac{\Gamma \vdash \tau \qquad \Gamma \vdash \nu}{\Gamma \vdash \tau \to \nu}\ \textsc{Valid-Arr}$$

$$\boxed{\Gamma \vdash \tau \equiv \nu}$$

Types τ and ν are equal in context Γ

$$\frac{\Gamma \vdash \tau}{\Gamma \vdash \tau \equiv \tau}\ \textsc{Refl} \qquad \frac{\Gamma \vdash \nu \equiv \nu}{\Gamma \vdash \nu \equiv \nu}\ \textsc{Sym} \qquad \frac{\Gamma \vdash \tau_0 \equiv \tau_1 \qquad \Gamma \vdash \tau_1 \equiv \tau_2}{\Gamma \vdash \tau_0 \equiv \tau_2}\ \textsc{Trans}$$

$$\frac{\Gamma \vdash \mathbf{ctx} \qquad X := \tau \in \Gamma}{\Gamma \vdash X \equiv \tau}\ \textsc{Unfold} \qquad \frac{\Gamma \vdash \tau_0 \equiv \tau_1 \qquad \Gamma \vdash \tau_2 \equiv \tau_3}{\Gamma \vdash \tau_0 \to \tau_2 \equiv \tau_1 \to \tau_3}\ \textsc{Congr-Arr}$$

Figure 4: Context and type validity

$$\boxed{\Gamma_0 \vdash t : \sigma \dashv \Gamma_1}$$

Term t in context Γ_0 has inferred scheme σ in context Γ_1

$$\frac{\Gamma_0, \Diamond \vdash t : \tau \dashv \Gamma_1, \Diamond, \Delta}{\Gamma_0 \vdash t : \forall \Delta.\, \tau \dashv \Gamma_1}\ \textsc{Infer-Gen}$$

$$\boxed{\Gamma_0 \vdash t : \tau \dashv \Gamma_1}$$

Term t in context Γ_0 has inferred type τ in context Γ_1

$$\frac{x : \forall \Delta.\, \nu \in \Gamma_0}{\Gamma_0 \vdash x : \nu \dashv \Gamma_0, \Delta}\ \textsc{Infer-Var} \qquad \frac{\Gamma_0, X, x : X \vdash t : \tau \dashv \Gamma_1, x : X, \Delta}{\Gamma_0 \vdash \lambda x.\, t : X \to \tau \dashv \Gamma_1, \Delta}\ \textsc{Infer-Lam}$$

$$\frac{?}{\Gamma_0 \vdash t\ s : X \dashv ?}\ \textsc{Infer-App} \qquad \frac{?}{\Gamma_0 \vdash \mathrm{let}\, x = s\ \mathrm{in}\ t : \tau \dashv ?}\ \textsc{Infer-Let}$$

Figure 5: Algorithmic type inference