Termination proof & evaluation

Final exam, MPRI 2-4

2021/03/10 — 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 native language.

This exam studies the computational content of termination proofs for the simply-typed λ-calculus under a call-by-name evaluation strategy (Section 1) as well as an unspecified one (Section 2). We recall its core syntax and static semantics in Figure 1. We then go on to add control operators to the calculus (Section 3).

$$
\begin{array}{rcll}
t, t_0, t_1 & ::= & & \text{(terms)} \\
 & \mid & x & \text{(variable)} \\
 & \mid & \lambda x.\, t & \text{(abstraction)} \\
 & \mid & t_0\ t_1 & \text{(application)} \\
v & ::= & & \text{(values)} \\
 & \mid & \lambda x.\, t & \\
\\
A, B, C & ::= & & \text{(types)} \\
 & \mid & \mathsf{unit} & \text{(base)} \\
 & \mid & A \to B & \text{(function)} \\
\Gamma & ::= & & \text{(contexts)} \\
 & \mid & \epsilon & \text{(empty)} \\
 & \mid & \Gamma, x : A & \text{(variable decl.)}
\end{array}
$$

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

$$
\frac{\Gamma(x) = A}{\Gamma \vdash x : A}
\qquad
\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.\, t : A \to B}
\qquad
\frac{\Gamma \vdash t_0 : A \to B \qquad \Gamma \vdash t_1 : A}{\Gamma \vdash t_0\ t_1 : B}
$$

Figure 1: Simply-typed λ-calculus

1 Call-by-name semantics

We define the call-by-name reduction contexts as follows:

$$\begin{array}{rcll} E & ::= & & \text{(contexts)} \\ & | & \star & \text{(empty)} \\ & | & E \cdot t & \text{(cons)} \\[1ex] p & ::= & & \text{(program)} \\ & | & \langle t, E \rangle & \end{array}$$

Contexts are represented inside-out: $\star$ represents the empty context while $E \cdot t$ represents the "term with a hole" $E[[]\ t]$ (in an informal notation).

Question 1 *Define the function* plug *that plugs a term t into an evaluation context E, thus producing a term. (Feel free to use pattern-matching.)*

□

For conciseness and following common practice, we write $E[t]$ for plug E t, the result of plugging the term t in the context E. In particular, a program $\langle t, E\rangle$ can be understood as representing the term $E[t]$.

We define a typing relation on contexts through the inference system of Figure 2. The judgment $\Gamma \,|\, E : A \vdash B$ states that the context E awaits a term of type A to produce a result of type B at the top-level.

Question 2 *Show that* plug *preserves well-typedness, i.e. if $\Gamma \vdash \langle t, E\rangle : C$ then we have $\Gamma \vdash E[t] : C$.*

□

$$\boxed{\Gamma \,|\, E : A \vdash B}$$

$$\frac{}{\Gamma \,|\, \star : A \vdash A} \qquad \frac{\Gamma \vdash t : A \qquad \Gamma \,|\, E : B \vdash C}{\Gamma \,|\, E \cdot t : A \to B \vdash C}$$

$$\boxed{\Gamma \vdash p : A}$$

$$\frac{\Gamma \vdash t : A \qquad \Gamma \,|\, E : A \vdash C}{\Gamma \vdash \langle t, E\rangle : C}$$

Figure 2: Type system for call-by-name evaluation contexts

We define a one-step call-by-name reduction relation on programs as follows:

$$\langle \lambda x.\, t_0, E \cdot t_1 \rangle \to_N \langle t_0\,\{x \mapsto t_1\}, E \rangle$$

$$\langle t_0\; t_1, E \rangle \to_N \langle t_0, E \cdot t_1 \rangle$$

where the notation $t_0\,\{x \mapsto t_1\}$ stands for the usual capture-avoiding substitution of t_1 for the variable x in t_0. More generally, given an environment γ mapping variables to terms, we write $t\,\{\gamma\}$ to denote the corresponding substitution. We define the evaluation relation $\to_N^*$ as the reflexive-transitive closure of the one-step reduction relation $\to_N$. The result of the evaluation is a program value of the form

$$w ::= \langle v, \star \rangle \qquad v ::= \lambda x.\, t$$

To prove termination for call-by-name evaluation, we introduce the following mutually inductive logical predicates:

$$\begin{aligned} |\mathsf{unit}|_V(v) &= \mathit{True} \\ |A \to B|_V(v) &= \forall t, |A|(t) \Rightarrow \forall E, \|B\|(E) \Rightarrow \mathcal{N}(\langle v, E \cdot t \rangle) \end{aligned}$$

$$\|A\|(E) = \forall v, |A|_V(v) \Rightarrow \mathcal{N}(\langle v, E \rangle)$$

$$\begin{aligned} |A|(t) &= \forall E, \|A\|(E) \Rightarrow \mathcal{N}(\langle t, E \rangle) \\ |\Gamma|(\gamma) &= \forall x, |\Gamma(x)|(\gamma(x)) \end{aligned}$$

where

$$\mathcal{N}(p) = \exists w, p \to_N^* w$$

Seen as a predicate in a dependently-typed theory, $\mathcal{N}(p)$ can be understood as a subset type built from a pair of a computationally-relevant witness w (the value obtained after normalization) and a computationally-irrelevant proof of $p \to_N^* w$ (the justification that the value was indeed obtained from the original program).

In the following, we consider an hypothetical erasure mechanism, denoted $\mathcal{E}(-)$, that strips off the computationally-irrelevant component of a proof witness in the dependently-typed theory, producing an untyped λ-term as a result. For instance, given a witness (w, q) of $\mathcal{N}(p)$, the erasure $\mathcal{E}((w, q))$ should yield w, removing the proof witnessing the purely logical fact that w indeed derives from p. In this exam, we only ask that you suggest the possible output of the erasure on a handful of proof witnesses. Note that we are *not* concerned with *how* such an erasure mechanism could work in general: you only have to make sure that, on the specific proofs we consider, the erasure produces valid programs.

Question 3 *Prove the following propositions:*

$$\begin{aligned} &\forall t_0\, t_1\, E, \mathcal{N}(\langle t_0, E \cdot t_1 \rangle) \Rightarrow \mathcal{N}(\langle t_0\; t_1, E \rangle) \\ &\forall t_0\, t_1\, E, \mathcal{N}(\langle t_0\,\{x \mapsto t_1\}, E \rangle) \Rightarrow \mathcal{N}(\langle \lambda x.\, t_0, E \cdot t_1 \rangle) \end{aligned}$$

We call APP *a proof witness of the first proposition (i.e.,* $\mathsf{APP} \in \forall t_0\, t_1\, E, \mathcal{N}(\langle t_0, E \cdot t_1\rangle) \Rightarrow \mathcal{N}(\langle t_0\, t_1, E\rangle)$*) and* LAM *a proof witness of the second one (i.e.,* $\mathsf{LAM} \in \forall t_0\, t_1\, E, \mathcal{N}(\langle t_0\, \{x \mapsto t_1\}, E\rangle) \Rightarrow \mathcal{N}(\langle \lambda x.\, t_0, E \cdot t_1\rangle)$*). What is the computational content of these proof witnesses? Propose a* λ*-term corresponding to their erasure* $\mathcal{E}(\mathsf{APP})$ *and* $\mathcal{E}(\mathsf{LAM})$*.*

□

We conventionally write $\overline{v}$ for a witness of the predicate $|A|_V(v)$, $\overline{E}$ for a witness of the predicate $\|A\|(E)$, $\overline{t}$ for a witness of the predicate $|A|(t)$, and $\overline{\gamma}$ for a witness of $|\Gamma|(\gamma)$.

To document the proposition witnessed by an expression, we use the type ascription $\lfloor p \rfloor^{\in P}$, which asserts that p is a witness of the proposition P. Because type ascriptions can be long-running and nested, this document uses a (harmless) typographical gadget: the pairs of open and closing parentheses are unambiguously indexed by an integer so that the reader can quickly match them up. We would for example have ${}_1\lfloor\, {}_2\lfloor f\, {}_2\rfloor^{\in P \Rightarrow Q}\, {}_2\lfloor p\, {}_2\rfloor^{\in P}\, {}_1\rfloor^{\in Q}$ to document the fact that f witnesses an implication $P \Rightarrow Q$, p witnesses a proposition P, while $f\ p$ witnesses Q.

We will use the notation $\lfloor \_ \rfloor^{\in P}$ to denote the computationally-irrelevant witness of a proposition P , where the symbol "_" stands for "any valid proof satisfying the proposition P".

The adequacy lemma relates well-typed terms and evaluation contexts with the corresponding logical predicates:

Lemma 1 (Adequacy for the call-by-name calculus) *Let* t *be a well-typed term, of type* A *in a context* Γ*. Let* γ *be an environment such that* $|\Gamma|(\gamma)$*. Then for every evaluation context* E *such that* $\|A\|(E)$*, the program* $\langle t\, \{\gamma\}, E\rangle$ *normalizes, i.e. we have* $\mathcal{N}(\langle t\, \{\gamma\}, E\rangle)$*.*

Question 4 *We consider the following proof-as-program rendition of the adequacy lemma:*

$$\begin{array}{l}
\mathsf{adequacy}_N \in \forall \Gamma\ t\ A, \Gamma \vdash t : A \Rightarrow \forall \gamma, |\Gamma|(\gamma) \Rightarrow |A|(t\, \{\gamma\}) \\
\mathsf{adequacy}_N\ \Gamma\ x\ A\ \lfloor \_ \rfloor^{\in \Gamma \vdash x : A}\ \gamma\ \overline{\gamma} = \boxed{?_1} \\
\mathsf{adequacy}_N\ \Gamma\ (t_0\ t_1)\ B\ \lfloor \_ \rfloor^{\in \Gamma \vdash t_0\ t_1 : B}\ \gamma\ \overline{\gamma} = \\
\quad\quad \lambda E\ \lfloor \overline{E} \rfloor^{\in \|B\|(E)}. \\
\quad\quad\ {}_1\lfloor\ \mathsf{APP}\ (t_0\, \{\gamma\})\ (t_1\, \{\gamma\})\ E \\
\quad\quad\quad {}_2\lfloor\ \lfloor \mathsf{adequacy}_N\ \Gamma\ t_0\ (A \to B)\ \lfloor \_ \rfloor^{\in \Gamma \vdash t_0 : A \to B}\ \gamma\ \overline{\gamma}\ \rfloor^{\in \forall E, \|A \to B\|(E) \Rightarrow \mathcal{N}(\langle t_0\, \{\gamma\}, E\rangle)} \\
\quad\quad\quad\quad (E \cdot (t_1\, \{\gamma\})) \\
\quad\quad\quad\quad (\lambda v\ \lfloor \overline{v} \rfloor^{\in \forall t, |A|(t) \Rightarrow \forall E, \|B\|(E) \Rightarrow \mathcal{N}(\langle v, E \cdot t\rangle)}.\ \overline{v}\ (t_1\, \{\gamma\})\ (\mathsf{adequacy}_N\ \Gamma\ t_1\ \lfloor \_ \rfloor^{\in \Gamma \vdash t_1 : A})\ \gamma\ \overline{\gamma}\ E\ \overline{E}) \\
\quad\quad\ {}_2\rfloor^{\in \mathcal{N}(\langle (t_0\, \{\gamma\}), E \cdot (t_1\, \{\gamma\})\rangle)}\ \ {}_1\rfloor^{\in \mathcal{N}(\langle (t_0\ t_1)\, \{\gamma\}, E\rangle)} \\
\mathsf{adequacy}_N\ \Gamma\ (\lambda x.\, t)\ (A \to B)\ \lfloor \_ \rfloor^{\in \Gamma \vdash \lambda x.\, t : A \to B}\ \gamma\ \overline{\gamma} = \\
\quad\quad \lambda E\ \lfloor \overline{E} \rfloor^{\in \boxed{?_2}}.\ \overline{E}\ (\lambda x.\, t\, \{\gamma\})\ (\lambda t_1\ \overline{t_1}\ E'\ \overline{E'}. \\
\quad\quad\quad {}_1\lfloor\ \boxed{?_3}\ \ {}_2\lfloor\ \mathsf{adequacy}_N\ \boxed{?_4}\ \boxed{?_5}\ \boxed{?_6}\ \boxed{?_7\ \ ?_8}\ \boxed{?_9}\ \boxed{?_{10}}\ \boxed{?_{11}}\ \ {}_2\rfloor^{\in \boxed{?_{12}}}\ \ {}_1\rfloor^{\in \boxed{?_{13}}}\ )
\end{array}$$

Either complete the remaining holes with a proposition (in type ascriptions) or a proof witness so as to finish this proof-as-program. Or, alternatively, write a standard mathematical proof from scratch. Either way, your answer will be judged for accuracy: beware, the devil is in the details.

□

Question 5 *Prove the following proposition:*

$$\forall A, \|A\|(\star)$$

We call HOLE *a proof witness of this proposition (i.e.,* HOLE $\in \forall A, \|A\|(\star)$*). What is its computational content? Propose a λ-term corresponding to its erasure* $\mathcal{E}(\mathsf{HOLE})$*.*

□

Theorem 1 (Termination of call-by-name evaluation) *If t is a well-typed term in the empty context, then t normalizes, i.e. we have $\mathcal{N}(\langle t, \star\rangle)$.*

Question 6 *Prove Theorem 1, that is, propose a dependently-typed version of this theorem by filling the two following holes:*

$$\begin{aligned}&\mathsf{termination}_N \in\ ?_{14}\\ &\mathsf{termination}_N\ ?_{15}\end{aligned}$$

□

Question 7 *Propose a λ-term corresponding to* $\mathcal{E}(\mathsf{adequacy}_N)$*.*

□

Question 8 *Propose a λ-term corresponding to* $\mathcal{E}(\mathsf{termination}_N)$*. What is this object?*

□

2 Alternative model

We extend call-by-name reduction contexts with another constructor:

$$\begin{array}{rcll} E & ::= & & \text{(contexts)} \\ & | & (\ldots) & \\ & | & v \cdot E & \text{(snoc)} \end{array}$$

where $v \cdot E$ represents the "term with a hole" $E[v[]]$ (in an informal notation).

Question 9 *Extend the* plug *function of Exercise 1 and the type system of Figure 2 to account for this new context former.*

□

Using this constructor, we define an alternative one-step reduction relation on programs:

$$\langle v, E \cdot t_1\rangle \to_M \langle t_1, v \cdot E\rangle$$

$$\langle t_0\ t_1, E\rangle \to_M \langle t_0, E \cdot t_1\rangle$$

$$\langle v, (\lambda x.\, t) \cdot E\rangle \to_M \langle t\,\{x \mapsto v\}, E\rangle$$

from which we define the evaluation relation $\to_M^*$ as the reflexive-transitive closure of the one-step reduction relation $\to_M$.

To prove termination, we introduce the following mutually inductive logical predicates:

$$\begin{aligned}
|\mathsf{unit}|_V(v) &= \mathit{True} \\
|A \to B|_V(v_0) &= \forall v_1, |A|_V(v_1) \Rightarrow \forall E, \|B\|(E) \Rightarrow \mathcal{N}(\langle v_1, v_0 \cdot E\rangle) \\
|\Gamma|_V(\gamma) &= \forall x, |\Gamma(x)|_V(\gamma(x)) \\
\\
\|A\|(E) &= \forall v, |A|_V(v) \Rightarrow \mathcal{N}(\langle v, E\rangle) \\
\\
|A|(t) &= \forall E, \|A\|(E) \Rightarrow \mathcal{N}(\langle t, E\rangle)
\end{aligned}$$

where

$$\mathcal{N}(p) = \exists w, p \to_M^* w$$

Question 10 *Prove the following proposition:*

$$\forall A\ v, |A|_V(v) \Rightarrow |A|(v)$$

We call LIFT *a proof witness of this proposition (i.e.,* LIFT $\in \forall A\ v, |A|_V(v) \Rightarrow |A|(v)$*). Propose a* λ*-term corresponding to* $\mathcal{E}(\mathsf{LIFT})$*.*

□

Question 11 *State and prove an adequacy lemma relating the alternative reduction relation on programs and the above logical predicates, on the model of Lemma 1.*

You can write a standard mathematical proposition and its proof, or a type-as-proposation and its program-as-proof. Either way, your answer will be judged for accuracy: beware, the devil is in the details.

□

Question 12 *What can you tell about the reduction strategy of this calculus?*
Hint: the erasure of the adequacy lemma may guide you to an answer.

□

3 Adding control operators

We extend the syntax of terms with two constructors:

$$\begin{array}{rcll}
t, t_0, t_1 & ::= & & \text{(terms)} \\
 & | & (\ldots) & \\
 & | & \kappa\alpha.\, t & \text{(context capture)} \\
 & | & E \hookleftarrow t & \text{(context application)}
\end{array}$$

$$\begin{array}{rcll} \Delta & ::= & & \text{(co-contexts)} \\ & | & \epsilon & \text{(empty)} \\ & | & \Delta, \alpha : A & \text{(covariable decl.)} \end{array}$$

$$\boxed{\Gamma \vdash_{\mathcal{K}} t : A \mid \Delta}$$

$$\frac{\Gamma(x) = A}{\Gamma \vdash_{\mathcal{K}} x : A \mid \Delta}$$

$$\frac{\Gamma, x : A \vdash_{\mathcal{K}} t : B \mid \Delta}{\Gamma \vdash_{\mathcal{K}} \lambda x.\, t : A \to B \mid \Delta}$$

$$\frac{\Gamma \vdash_{\mathcal{K}} t_0 : A \to B \mid \Delta \qquad \Gamma \vdash_{\mathcal{K}} t_1 : A \mid \Delta}{\Gamma \vdash_{\mathcal{K}} t_0\ t_1 : B \mid \Delta}$$

$$\frac{\Gamma \vdash_{\mathcal{K}} t : A \mid \Delta, \alpha : A}{\Gamma \vdash_{\mathcal{K}} \kappa\alpha.\, t : A \mid \Delta}$$

$$\frac{\Gamma \mid E : A \vdash_{\mathcal{K}} \Delta \qquad \Gamma \vdash_{\mathcal{K}} t : A \mid \Delta}{\Gamma \vdash_{\mathcal{K}} E \hookleftarrow t : B \mid \Delta}$$

$$\boxed{\Gamma \mid E : A \vdash_{\mathcal{K}} \Delta}$$

$$\frac{\Delta(\alpha) = A}{\Gamma \mid \alpha : A \vdash_{\mathcal{K}} \Delta}$$

$$\frac{\Gamma \vdash_{\mathcal{K}} t : A \mid \Delta \qquad \Gamma \mid E : B \vdash_{\mathcal{K}} \Delta}{\Gamma \mid E \cdot t : A \to B \vdash_{\mathcal{K}} \Delta}$$

Figure 3: Calculus with control operators

where the context variables ($\alpha, \beta, \alpha_1, \ldots$) are drawn from a separate set from term variables ($x, y, z, x_1, \ldots$). A term is said to be *plain* if all occurrences of $E \hookleftarrow t$ (if any) are of the form $\alpha \hookleftarrow t$ for some context variable α.

We take the following definition of evaluation contexts:

$$\begin{array}{rcll} E & ::= & & \text{(contexts)} \\ & | & \alpha & \text{(covariable)} \\ & | & E \cdot t & \text{(cons)} \end{array}$$

Both terms and evaluation contexts can be equipped with a type system (Figure 3) that we assume to be sound with respect to the reduction relation $\to_K^*$ obtained from the reflexive-transitive closure of the following one-step reduction:

$$\langle \lambda x.\, t_0, E \cdot t_1 \rangle \to_K \langle t_0 \{x \mapsto t_1\}, E \rangle$$

$$\langle \kappa\alpha.\, t, E \rangle \to_K \langle t \{\alpha \mapsto E\}, E \rangle$$

$$\langle t_0\ t_1, E \rangle \to_K \langle t_0, E \cdot t_1 \rangle$$

$$\langle E' \hookleftarrow t, E \rangle \to_K \langle t, E' \rangle$$

Question 13 *Let A and B be any type. Exhibit a plain term* peirce *of type $((A \to B) \to A) \to A$, i.e. we must have*

$$\epsilon \vdash_{\mathcal{K}} \mathsf{peirce} : ((A \to B) \to A) \to A \mid \epsilon$$

□

Question 14 *Show that function application $t_0\ t_1$ is superfluous as it can be obtained through macro-expansion of a term which, itself, does not contain any application. To do so, give a term APP_{t_0,t_1} paremetrized by two terms t_0 and t_1 that behaves as the application $t_0\ t_1$, i.e. it is such that $\langle \mathsf{APP}_{t_0,t_1}, E\rangle \to_K^* \langle t_0, E \cdot t_1\rangle$.*

□

Question 15 *Propose a type-preserving translation from the call-by-name calculus of Section 1 (Figure 1 and Figure 2) to the calculus above (Figure 3), covering both terms and evaluation contexts. (Crucially, explain how typing judgements evolve during the translation from the former to the latter).*

□

Question 16 *Define the necessary logical predicates to carry out the termination proof for this calculus. State and prove an adequacy lemma relating reduction of programs with these logical predicates.*

You can write a standard mathematical proposition and its proof, or a type-as-proposation and its program-as-proof. Either way, your answer will be judged for accuracy: beware, the devil is in the details.

□

Question 17 *While computing the erasure of this adequacy lemma, can we dispense with the syntactic representations of evaluation contexts E (i.e., the first parameter in $|A|(t)$)?*

□