Part II Types: Exercise Sheet 1

Nathanael Alcock, Dylan McDermott, Shaun Steenkamp, Domagoj Stolfa, *et al.*

October 2018

Some questions due to Benjamin C. Pierce and Andrew Pitts.

Tips

- Draw proof trees in landscape (if by hand, it helps to start from the bottom). If your proof tree gets too large then break it up into named subtrees. In LaTeX, the `proof` and `ebproof` packages are helpful for drawing the proof trees.
- Collate all the typing rules you have come across in lectures so far into a "cheat sheet". Group by the type system (simply-typed λ-calculus, polymorphic λ-calculus, *etc.*). Pair up introduction and elimination rules and group related types (*e.g.* sums and products). Update it as the lecture course continues: you'll be glad you did when it comes to exam time!

Questions

1. Complete Exercises 1, 2, and 3 from Lecture 1.
   - Give cases of the operation semantics for $\leq$ and $+$.
   - Extend the progress proof to cover $e \wedge e'$.
   - Extend the preservation proof to cover $e \wedge e'$.
2. Complete Exercises 3, 4, and 5 from Lecture 2.
   - Prove substitution.
   - Prove progress.
   - Prove type preservation.
3. Show, by drawing proof trees showing the typing derivations, that the following terms have the indicated types:
   (a) `f : Bool→Bool ⊢ f (if false then true else false) : Bool`
   (b) `f : Bool→Bool ⊢ λx : Bool . f (if x then false else x ) : Bool → Bool`
4. The pure simply typed λ-calculus with no base types is *degenerate* in the sense that it has no well-typed terms at all. Why? (A base type is a type that is formed without using any other types.)

   Note: the "pure" simply-typed λ-calculus has the following rules:

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\ VAR$$

$$\frac{\Gamma, x : \tau \vdash E : \tau'}{\Gamma \vdash (\lambda x : \tau . E) : \tau \to \tau'}\ LAM$$

$$\frac{\Gamma \vdash E_1 : \tau' \to \tau \quad \Gamma \vdash E_2 : \tau'}{\Gamma \vdash E_1\ E_2 : \tau}\ APP$$

5. Under the Curry–Howard correspondence, a proposition P is represented by a type π. Proving P corresponds to providing a term p such that $\cdot \vdash p : \pi$.

(a) We represent the negation $\neg P$ by a type $\pi \to 0$. Why is this a reasonable representation (using your intuition from propositional logic)?

(b) Try to prove De Morgan's laws by converting each propositional formula to a type and providing a term of that type in the empty context. *(Hint: not every law holds.)*

i. $\neg P \wedge \neg Q \supset \neg (P \vee Q)$

ii. $\neg (P \vee Q) \supset \neg P \wedge \neg Q$

iii. $\neg P \vee \neg Q \supset \neg (P \wedge Q)$

iv. $\neg (P \wedge Q) \supset \neg P \vee \neg Q$

(c) De Morgan's laws all hold in propositional logic. What does (b) tell you about the simply-typed λ-calculus in relation to propositional logic?

Note: the simply-typed λ-calculus with `abort` $: \bot \to X$ *for any X actually corresponds to a form of logic called intuitionistic logic.*

6. For the simply-typed λ-calculus, show that if a term e is typeable in the empty context, $\cdot \vdash e : X$, then e must be *closed*, *i.e.* have no free variables. *(Hint: use rule induction to show that all provable typing judgements, $\Gamma \vdash e' : Y$, obey the property $fv(e') \subseteq dom(\Gamma)$, where $fv(e')$ is the set of free variables unbound in e' and $dom(\Gamma)$ is the set of typed variables assumed in the context Γ.)*

7. Complete Exercises 1 and 2 from Lecture 3.

- Extend the logical relation to support product types.
- Extend the logical relation to support sum types.

8. For each of the following PLC (System F) typing judgements, is there a PLC type A_i that make the judgement provable? (In each case, give a type A_i and typing derivation for, or explain why a typing derivation cannot exist.)

(a) $\cdot; \cdot \vdash \lambda x : (\forall \alpha \,.\, \alpha) \,.\, (\Lambda \beta \,.\, x\ \beta) : A_1$

(b) $\cdot; \cdot \vdash \Lambda \alpha \,.\, \lambda x : \alpha \,.\, \Lambda \beta \,.\, x\ \beta : A_2$

(c) $\cdot; \cdot \vdash \lambda x : A_3 \,.\, \Lambda \alpha \,.\, (x\ (\alpha \to \alpha)\ (x\ \alpha)) : A_3 \to \forall \beta \,.\, \beta$

(d) $\cdot; \cdot \vdash \lambda x : A_4 \,.\, \Lambda \alpha \,.\, (x\ (\alpha \to \alpha)\ (x\ \alpha)) : A_4 \to \forall \alpha \,.\, (\alpha \to \alpha)$

(e) $\cdot; \cdot \vdash \Lambda \alpha \,.\, \lambda x : A_5 \,.\, (x\ (\alpha \to \alpha)\ (x\ \alpha)) : \forall \alpha \,.\, (\alpha \to \alpha)$

Extension

9. In Lecture 2, it is claimed that the Curry–Howard correspondence is not an isomorphism.

(a) Show that, for the fragment of simply-typed λ-calculus on slide 9 of Lecture 2 *excluding* sum types, the correspondence is bijective when terms are considered up to β- and η-equivalence (that is, where terms with the same value are considered equivalent). *(Hint: in particular, all terms with type* 1 *are equivalent to* $\langle\rangle$*.)*

(b) What goes wrong when sum types are also considered?

10. (a) In the simply-typed λ-calculus, give a typing derivation for the Y combinator, $\lambda f.(\lambda x.f\ (x\ x))\ (\lambda x.f\ (x\ x))$, or explain why no such derivation exists.

(b) Show that for *some* type X there is no term Y such that $\cdot \vdash Y : ((X \to X) \to (X \to X)) \to (X \to X)$ and $Y\ f\ v \rightsquigarrow^* f\ (Y\ f)\ v$ for all values $\cdot \vdash f : (X \to X) \to (X \to X)$ and $\cdot \vdash v : X$. Is this the case for *every* type X?

11. (a) Find a context Γ under which the term $f\ x\ y$ has type `Bool`, *i.e.* $\Gamma \vdash f\ x\ y :$ `Bool`.

(b) Can you give a simple description of the set of all such contexts?

12. In the simply-typed λ-calculus, is there any context Γ and type τ for which $\Gamma \vdash x\ x : \tau$ (where x is a free variable defined in Γ)? If so, give Γ, τ and show a typing derivation, otherwise prove that no such context and type exists.

13. Is there a way of constructing a sequence of terms $t_1, t_2, \ldots$ in the simply-typed λ-calculus with only the base type 1, such that, for each n, the term t_n has size at most $O(n)$, but requires at least $O(2^n)$ steps of evaluation to reach a normal form?