Sebastian Funk
sebastian.funk@gmail.com

1 Warmup questions

Do these without looking at the notes and in at most three sentences per question:

1. In the calculus of truth and falsehood, define Γ, Δ, proofs, refutations, contradictions and continuations.
2. What makes dependent types so powerful? Give an example of its usefulness in every day programming.
3. What are the downsides of dependent types?
4. Write down and carefully explain (in English) the subst rule in the dependent types part of the course.

2 Regular questions

1. Complete Exercises 1 and 2 from Lecture 9:
   (a) Show that $\neg A \vee B, A; \cdot \vdash B\ \mathsf{true}$ is derivable.
   (b) Show that $\neg(\neg A \wedge \neg B); \cdot \vdash A \vee B\ \mathsf{true}$ is derivable.
2. Give the proof (and refutation) terms corresponding to the derivations in the previous question.
3. Let $\mathsf{upc}(p) \triangleq \mu u : A.\ \langle p \mid_A u \rangle$ be a proof (and refutation) term from the calculus presented in Lectures 9 and 10.
   (a) Show that $p : A; \cdot \vdash \mathsf{upc}(p) : A\ \mathsf{true}$.
   (b) Show that for all $k : \neg A$ we have $\langle \mathsf{upc}(p) \mid_A k \rangle \mapsto \langle p \mid_A k \rangle$.
   (c) Terms $p : A\ \mathsf{true}$ correspond to proofs of A. Describe, in English, the proof that corresponds to $\mathsf{upc}(p)$ with respect to the proof corresponding to p.
4. Complete Exercises 1 and 2 from Lecture 10:
   (a) Give the embedding (i.e., the e° and k° translations) of classical into intuitionistic logic for the Gödel-Gentzen translation. You just need to give the embeddings for sums, since that is the only case different from the lectures.
   (b) Using the intuitionistic (λ-) calculus extended with continuations, give a typed term proving Peirce's law:
   $$((X \to Y) \to X) \to X$$
5. 2021 Paper 8 Question 15 part a and c (you can skip part b).
6. 2019 Paper 8 Question 13
7. Using the `amb` primitive from Lecture 11 implement a function:

   ```
   eq-at :  α list -> α list -> int * int
   ```

   such that `eq-at xs ys` returns `(i,j)` if `nth(xs,i) = nth(ys,j)` and fails otherwise. You may assume the existence of any helper functions without definition.

8. Using the dependent type theory introduced in Lecture 12 show that if $\Gamma \vdash A\ \mathsf{type}$ then the following typing judgement holds:

$$\Gamma \vdash \mathsf{sym}_A : \Pi x : A.\, \Pi y : A.\, ((x = y : A) \to (y = x : A))$$

where

$$\mathsf{sym}_A \triangleq \lambda x : A.\, \lambda y : A.\, \lambda p : (x = y : A).\, \mathsf{subst}[z : A.\, (z = x : A)](p, \mathsf{refl}\, x)$$

and $X \to Y$ is shorthand for $\Pi x : X.Y$ if x does not appear in Y.

9. Define terms with the following types:
   (a) $\Gamma \vdash \mathsf{trans}_A : \Pi x : A.\, \Pi y : A.\, \Pi z : A.\, ((x = y : A) \to (y = z : A) \to (x = z : A))$
   (b) $\Gamma \vdash \mathsf{cong}_{A,B} : \Pi x : A.\, \Pi y : A.\, \Pi f : (A \to B).\, ((x = y : A) \to (fx = fy : B))$

   assuming that $\Gamma \vdash A\ \mathsf{type}$ and $\Gamma \vdash B\ \mathsf{type}$.

3 Extension questions (optional)

1. Download and install Agda and try out some of the examples from the lectures:

   `https://agda.readthedocs.io/en/latest/getting-started/index.html`

2. Consider types $\Gamma \vdash A\ \mathsf{type}$ and $\Gamma, x : A \vdash B\ \mathsf{type}$. If we have terms a_1 and a_2 and a proof that they are equal, $\Gamma \vdash p : (a_1 = a_2 : A)$, then the types $[a_1/x]B$ and $[a_2/x]B$ should also be "equal" in some sense. And so, given $\Gamma \vdash b_1 : [a_1/x]B$ and $\Gamma \vdash b_2 : [a_2/x]B$ we might want to consider the type of equalities between b_1 and b_2.
   (a) Show that the following rule is not (in general) derivable:

$$\frac{\Gamma \vdash A\ \mathsf{type} \quad \Gamma, x : A \vdash B\ \mathsf{type} \quad \Gamma \vdash p : (a_1 = a_2 : A) \quad \Gamma \vdash b_1 : [a_1/x]B \quad \Gamma \vdash b_2 : [a_2/x]B}{\Gamma \vdash (b_1 = b_2 : [a_1/x]B)\ \mathsf{type}}$$

   (b) Define the type of *heterogeneous equalities* like so:

$$(b_1 \approx b_2 : B\ \mathsf{over}\ p) \triangleq (\mathsf{subst}[x : A.\, B](p, b_1) = b_2 : [a_2/x]B)$$

   Show that the following rule is derivable:

$$\frac{\Gamma \vdash A\ \mathsf{type} \quad \Gamma, x : A \vdash B\ \mathsf{type} \quad \Gamma \vdash p : (a_1 = a_2 : A) \quad \Gamma \vdash b_1 : [a_1/x]B \quad \Gamma \vdash b_2 : [a_2/x]B}{\Gamma \vdash (b_1 \approx b_2 : B\ \mathsf{over}\ p)\ \mathsf{type}}$$

   (c) Define a term $\mathsf{hrefl}\, b$ such that the following rule is derivable:

$$\frac{\Gamma \vdash A\ \mathsf{type} \quad \Gamma, x : A \vdash B\ \mathsf{type} \quad \Gamma \vdash a : A \quad \Gamma \vdash b : [a/x]B}{\Gamma \vdash \mathsf{hrefl}\, b : (b \approx b : B\ \mathsf{over}\ (\mathsf{refl}\, a))}$$

3. (a) What logical operator do Π-types (or *dependent products*) correspond to? Justify your answer.
   (b) Given your answer to (a), what logical operator are we still missing?
   (c) Extend the syntax of the dependently typed language introduced in the lectures with this dual of Π-types (also called Σ-*types* or *dependent sums*) and give suitable typing rules for them. (Search engines are your friend.)
   (d) In first-order logic, the axiom of choice can be stated as:

$$(\forall x \in A.\, \exists y \in B.\, P(x, y)) \implies (\exists f : A \to B.\, \forall x \in A.\, P(x, f(x)))$$

   Given $\Gamma \vdash A\ \mathsf{type}$ and $\Gamma \vdash B\ \mathsf{type}$, give a type $\Gamma \vdash \mathsf{AC}\ \mathsf{type}$ corresponding to the axiom of choice.
   (e) Define a term $\Gamma \vdash \mathsf{ac} : \mathsf{AC}$.

4 Practise material (optional)

These are some extra questions on everything in the course. We won't go through them in the supervision but I'll mark them for you. Do as many as you can.

1. 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?

2. Show that if a term e is typeable in the empty context, 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 of e' and $dom(\Gamma)$ is the set of typed variables assumed in the context Γ.)*

3. (a) Write a function `isnil` that takes a Church-encoded list and returns a Church boolean.

   (b) Write a function $\texttt{head} : X \to \texttt{CList}_X \to X$ which returns the first argument if the list is empty, and the head of the list otherwise.

   (c) Write a function $\texttt{tail} : \texttt{CList}_X \to \texttt{CList}_X$ which returns the empty list when applied to an empty list, and the tail of the list otherwise.

4. Suppose that $\Theta, \beta \vdash F\ \mathsf{type}$, and fmap is a term such that:

$$\Theta; \cdot \vdash \mathsf{fmap} : \forall\beta_1.\, \forall\beta_2.\, (\beta_1 \to \beta_2) \to ([\beta_1/\beta]F \to [\beta_2/\beta]F)$$

   The type μF is defined to be $\forall\beta.\, (F \to \beta) \to \beta$.

   (a) Show that $\Theta \vdash \mu F\ \mathsf{type}$ and $\Theta \vdash [\mu F/\beta]F\ \mathsf{type}$.

   (b) Define terms fold and intro such that:

$$\Theta; \cdot \vdash \mathsf{fold} : \forall\beta.\, (F \to \beta) \to \mu F \to \beta$$
$$\Theta; \cdot \vdash \mathsf{intro} : [\mu F/\beta]F \to \mu F$$

   (Hint: you will need to use fold in the definition of intro.)

   (Category-theoretic background: altogether these properties say that $(\mu F, \mathsf{intro})$ is a weak initial F-algebra: given any F-algebra $(\nu F, f : [\nu F/\beta]F \to \nu F)$, we get fold)

5. Exam question 2022 Paper 9 Question 13