Problem Set 2: Simply Typed Lambda Calculus

Released: September 11, 2025
Due: September 25, 2025, 11:59pm

Submit your solutions to this homework on Canvas alone or in a group of 2 or 3. Your solutions must be submitted in pdf produced using LaTeX.

Problem 1 Laws of Exponentiation

We say that $x : A \vdash M : B$ and $y : B \vdash N : A$ form an *isomorphism* if $x : A \vdash N[M/y] = x$ and $y : B \vdash M[N/x] = y : B$. In this case we say A and B are isomorphic, written $A \cong B$.

Implement the lambda terms for the following isomorphisms. You do not need to include the typing derivation, just the terms themselves. Provide proofs that (1) and (4) are isomorphisms using the equational theory. You do not need to explicitly write the proof tree, but just explain what rules are used to justify the equalities.

1. $A \Rightarrow B \Rightarrow C \cong (A \times B) \Rightarrow C$
2. $A \Rightarrow (B \times C) \cong (A \Rightarrow B) \times (A \Rightarrow C)$
3. $(A \Rightarrow 1) \cong 1$
4. $(A + B) \Rightarrow C \cong (A \Rightarrow C) \times (B \Rightarrow C)$
5. $0 \Rightarrow C \cong 1$

This gives an idea of why $A \Rightarrow B$ is in category theory sometimes called the *exponential* and written B^A.

. . . . . .

Problem 2 Lawvere's Fixed Point Theorem

We saw that Boolean semantics of IPL was incomplete for the signature consisting of a single propositional variable, because the law of excluded middle is true in all Boolean models but not provable.

In this problem, we consider a similar question for Set-semantics of STLC. Define the signature Σ_{lfpt}:

1. Two base type X and D
2. Two function symbols $p : X \to (X \Rightarrow D)$ and $e : (X \Rightarrow D) \to X$
3. One equation (called "retraction") $f : X \Rightarrow D \vdash p(e(f)) = f : X \Rightarrow D$

Such a signature may seem strange at first glance, but for any type d in a functional programming language with recursive type definitions, we can define a corresponding type Xd and functions p, E that satisfying the retraction axiom. For example in Haskell:

```
data X d = E (X d -> d)

p :: X d -> (X d -> d)
p (E f) = f
```

For this problem, we will prove that this signature only has *trivial* models in Set, and show a famous application of this triviality. This triviality implies that functional programming languages that support unrestricted recursive type definitions cannot be given non-trivial Set-theoretic semantics. Later in the course we will see that there are non-trivial models in other categories besides Set.

1. In STLC generated by Σ_{lfpt}, construct a *fixed point combinator* for the type D, that is define
   - A term $\cdot \vdash \texttt{fix} : (D \Rightarrow D) \Rightarrow D$
   - That maps any function to a fixed point in that it satisfies the equation:

   $$f : D \Rightarrow D \vdash f(\texttt{fix}\, f) = \texttt{fix}\, f : D$$

2. Show that that the equation

   $$x : D, y : D \vdash x = y : D$$

   is true in any model of Σ_{lfpt} in Set.

3. As a corollary of part (2), prove *Cantor's theorem*: that there is no surjective function from a set S to its powerset $\mathcal{P}S$. Recall that the powerset of S is equivalent to the functions from S to the booleans 2^S. You will need to use the axiom of choice[1].

. . . . . .

[1] A more refined version of this argument can prove Cantor's theorem without the axiom of choice.

Problem 3 Simultaneous Substitution

When all variables are known to be distinct, substitution $M[N/x]$ can simply be defined as the replacement of x with N everywhere in the term M. This definition is the STLC version of the admissibility of the substitution principle of IPL:

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x : A \vdash N : B}{\Gamma \vdash N[M/x]}\ \textsc{Subst}(*)$$

The admissible principle of contraction can also be viewed as a textual substitution in the term:

$$\frac{\Gamma, x : A, y : A, \Delta \vdash M : C}{\Gamma, x : A, \Delta \vdash M[x/y] : C}\ \textsc{Contraction}(*)$$

On the other hand, the use of variables to stand for assumptions means that exchange and weakening have no effect on the proof term:

$$\frac{\Gamma, y : B, x : A, \Delta \vdash M : C}{\Gamma, x : A, y : B, \Delta \vdash M : C}\ \textsc{Exchange}(*) \qquad \frac{\Gamma, \Delta \vdash M : C}{\Gamma, x : A, \Delta \vdash M : C}\ \textsc{Weakening}(*)$$

In this exercise you will prove these principles are admissible and additionally prove some *equations* about substitution. Experience shows that the simplest way to prove these properties involves generalizing from "one-place" substitutions like $M[N/x]$ to "simultaneous" substitutions that simultaneously substitute for *all* free variables in a term.

We define a (simultaneous) substitution from Δ to Γ to be a function γ that for each variable $x : A \in \Gamma$ produces a term $\Delta \vdash \gamma(x) : A$. We write $\gamma : \Delta \to \Gamma$ to mean a substitution from Δ to Γ. Note that this definition is contravariant in that a substitution $\gamma : \Delta \to \Gamma$ maps variables in Γ to terms well-typed under Δ.

We can then define an admissible action of substitution:

$$\frac{\gamma : \Delta \to \Gamma \qquad \Gamma \vdash M : A}{\Delta \vdash M[\gamma] : A}\ \textsc{GenSubst}$$

Defined by induction on M as follows:

$$\begin{aligned}
x[\gamma] &= \gamma(x) \\
f(M_1, \ldots,)[\gamma] &= f(M_1[\gamma], \ldots) \\
(M, N)[\gamma] &= (M[\gamma], N[\gamma]) \\
(\pi_j M)[\gamma] &= \pi_j M[\gamma] \\
()[\gamma] &= () \\
(\sigma_j M)[\gamma] &= \sigma_j M[\gamma] \\
(\text{case}_+\, M\{\sigma_1 x_1 \to N_1 | \sigma_2 x_2 \to N_2\})[\gamma] &= (\text{case}_+\, M[\gamma]\{\sigma_1 x_1 \to N_1[\gamma, x_1/x_1] | \sigma_2 x_2 \to N_2[\gamma, x_2/x_2]\}) \\
(\text{case}_0\, M\{\})[\gamma] &= \text{case}_0\, M[\gamma]\{\} \\
(\lambda x.M)[\gamma] &= \lambda x.M[\gamma, x/x] \\
(MN)[\gamma] &= M[\gamma]\, N[\gamma]
\end{aligned}$$

Where the notation $\gamma, M/x$ is the extension of the the function to map x to M:

$$\begin{aligned}
(\gamma, M/x)(y) &= M && (\text{if } x = y) \\
(\gamma, M/x)(y) &= \gamma(y) && (\text{if } x \neq y)
\end{aligned}$$

Define the *identity* substitution $\text{id}_\Gamma : \Gamma \to \Gamma$ to map each variable in Γ to itself: $\text{id}(x) = x$.

Given $\gamma : \Delta \to \Gamma$ and $\delta : \Xi \to \Delta$, define the *composition* $\gamma \circ \delta : \Xi \to \Gamma$ as $(\gamma \circ \delta)(x) = (\gamma(x))[\delta]$.

Below assume $\gamma : \Delta \to \Gamma$, $\delta : \Xi \to \Delta$, $\xi : \Xi' \to \Xi$ and $\Gamma \vdash M : A$.

First, observe (no need to write the proof), that the following generalized weakening principle is admissible, if $\Gamma \subseteq \Gamma'$, then any term typeable in the smaller context Γ is also typable in the larger context Γ' with the same syntax:

$$\frac{\Gamma \vdash M : A}{\Gamma' \vdash M : A}\ \textsc{GenWeak}$$

The proof is by induction on the typing derivation of M.

1. Define simultaneous substitutions that correspond to the principles of one-place substitution, weakening, exchange and contraction.

2. Show (by induction on the derivation of $\Gamma \vdash M : A$) that $\Delta \vdash M[\gamma] : A$, i.e., that the GenSubst typing rule is admissible. You only need to show the following representative cases: $M = x$, $M = f(M_0, \ldots)$, $M = \lambda x.M'$ and $M = M'\, N$. Where is the GenWeak principle needed?

3. Show (by induction on M) that $M[\text{id}_\Gamma] = M$. Note that this and the following equalities are exact syntactic equalities, you will not need to use any $\beta\eta$ rules to prove it.

4. Show (by induction on M) that $M[\gamma \circ \delta] = (M[\gamma])[\delta]$
5. Show that $\gamma \circ \mathrm{id}_\Delta = \gamma$ and $\mathrm{id}_\Gamma \circ \gamma = \gamma$.
6. Show as a corollary that if $x_2 : A_2 \vdash N_1 : A_1$ and $x_3 : A_3 \vdash N_2 : A_2$ and $x_4 : A_4 \vdash N_3 : A_3$ then $(N_1[N_2/x_2])[N_3/x_3] = N_1[N_2[N_3/x_3]/x_2]$.
7. Show that $(\gamma \circ \delta) \circ \xi = \gamma \circ (\delta \circ \xi)$

. . . . . .