Exercise 1 (Progress Property)

Let \( t \) be a closed and well-typed term, i.e. \([\_] \vdash t : \tau\) for some \( \tau \). Show that \( t \) is either a value or there is a \( t' \) such that \( t \rightarrow_{cbv} t' \).

Exercise 2 (Normal Form)

Show that every type-correct \( \lambda\tau \)-term has a \( \beta \)-normal form.
Homework 3 (Typing)

a) Prove:

\[ \emptyset \vdash (\lambda x : \tau_2 \rightarrow \tau_3. \lambda y : \tau_1 \rightarrow \tau_2. \lambda z : \tau_1. x (y z)) : (\tau_2 \rightarrow \tau_3) \rightarrow (\tau_1 \rightarrow \tau_2) \rightarrow \tau_1 \rightarrow \tau_3 \]

b) Give suitable solutions for \(?\tau_1\), \(?\tau_2\), \(?\tau_3\) and \(?\tau_4\) and prove that the term is type-correct given your solution.

\[ \emptyset \vdash \lambda x : ?\tau_1. \lambda y : ?\tau_2. \lambda z : ?\tau_3. \ x \ y \ z : ?\tau_4 \]

Homework 4 (\(\beta\)-reduction preserves types)

A type system has the subject reduction property if evaluating an expression preserves its type. Prove that the simply typed \(\lambda\)-calculus (\(\lambda^\tau\)) has the subject reduction property:

\[ \Gamma \vdash t : \tau \wedge t \rightarrow_\beta t' \implies \Gamma \vdash t' : \tau \]

Hints: Use induction over the inductive definition of \(\rightarrow_\beta\) (Def. 1.2.2). State your inductive hypotheses precisely – it may help to introduce a binary predicate \(P(t, t')\) to express the property you are proving by induction. Also note that the proof will require rule inversion: Given \(\Gamma \vdash t : \tau\), the shape of \(t\) (variable, application, or \(\lambda\)-abstraction) may determine which typing rule must have been used to derive the typing judgment.

Within your proof, you are free to use the following lemma about substitution:

\[ \Gamma \vdash u : \tau_0 \wedge \Gamma[\ x : \tau_0] \vdash t : \tau \implies \Gamma \vdash t[u/x] : \tau \]  \hspace{1cm} (1)

Homework 5 (Implementation of multiset-ordering and reduction)

Implement the multiset ordering and the reduction strategy from the second tutorial exercise in your favorite programming language.