Homework 5 (v1.0)

Due: Monday, April 16, 11:59pm
Extension: Wednesday, April 18, 11:59pm

Preface

Discussing high-level approaches to homework problems with your peers is encouraged. You must include at the top of all of your assignment documents a Collaboration Statement which declares any other people with whom you discussed homework problems. For example:

> *Collaboration Statement: I discussed problems 1 and 3 with Jamie Smith. I discussed problem 2 with one of the TAs. I discussed problem 4 with a personal tutor.*

If you did not discuss the assignment with anyone, you still must declare:

> *Collaboration Statement: I did not discuss homework problems with anyone.*

Copying answers or doing the work for another student is not allowed. Write your Collaboration Statement at the top of your hw5.ml file. Please do not write your Collaboration Statement in the text of an email; you must include it directly it in your assignment.

For the programming portion, you must write your name at the top *and the bottom* of your submitted *.ml file in an OCaml comment. *E.g.*:

```
<beginning of file>
(* Name: Jamie Smith *)
... your solutions to the assignment ...
(* Name: Jamie Smith *)
<end of the file>
```

You may do this assignment either individually or in groups of two. See submitting instructions for how to submit if you are working in a group.

Submitting

Programming Portion

Prepare the programming portion of your assignment in a text editor of your choice. You will be working from a provided file named `hw5.ml`. Do not change the file name when you submit your assignment.

Submit the programming portion of your assignment by emailing your `hw5.ml` file to me: David.Darais@uvm.edu with "CS 225 HW5" in the subject line.

If you are doing this assignment with a partner, only submit one file for both team members. In the email where you submit your assignment, let me know that it is a group submission, *e.g.*, by writing "This submission is for Jamie Smith and Alex Williams." Write the names of both team members on all submitted documents: writeups, code, *etc.*

Programming Portion (100 Points)

Download hw5.zip from the course webpage and extract it to a folder on your machine. Ensure that you are able to run the following commands from the directory without error:

```
> make
> make hw5
```

Running `make` should print out a log of compiling various OCaml files. Running `make hw5` should run a bunch of tests and report that 0 passed, 0 failed, and 27 are still "todo". In order for the `merlin` plugin to work, you must first run `make`.

Your assignment is to complete the definitions of `step`, `scope_ok`, and `infer` based on the small-step semantics, well-scoped relation and well-typed relation defined in the next section of the writeup. These are also described in the book (TAPL) Chapters 23 & 24. The primary difference between the book and this writeup is that the book uses Γ for both the type environment and scope environment. In this writeup and assignment, we split this into the usual type environment Γ, and a separate scope environment, which we notate S.

Look in the assignment file for `raise TODO` expressions and replace them with your solution.

At the beginning of the assignment are some useful hints for how to manipulate sets and maps, as well as a hint for the last type-checking case for "unpack" expressions.

At the end of `hw5.ml` is a test suite which will test your functions on a few test cases. Feel free to add your own tests. Just because you pass the provided tests doesn't guarantee your solution is correct.

1 Language Reference

This language contains booleans, natural numbers, products, functions, and universal and existential quantification. Only the definitions relevant to the assignment are shown. The specifications for relevant functions which are used in these definitions, but already implemented in the assignment, are shown in the next section.

Syntax for types:

$$\begin{aligned} X \in \text{tvar} &\coloneqq \ldots \textit{type variables} \ldots \\ \tau \in \text{type} &::= \texttt{bool} \mid \texttt{nat} \mid \tau \times \tau \mid \tau \rightarrow \tau \mid X \mid \forall X.\tau \mid \exists X.\tau \end{aligned}$$

Syntax for expressions:

$$\begin{array}{rcll} x \in \text{var} &\coloneqq& \ldots \textit{term variables} \ldots & \\ e \in \text{exp} &::=& \texttt{true} \mid \texttt{false} \mid \texttt{if}(e)\{e\}\{e\} & \textit{booleans} \\ &\mid& \texttt{zero} \mid \texttt{succ}(e) \mid \texttt{pred}(e) \mid \texttt{iszero}(e) & \textit{natural numbers} \\ &\mid& \langle e, e\rangle \mid \texttt{projl}(e) \mid \texttt{projr}(e) & \textit{products} \\ &\mid& x \mid \texttt{let}\ x \coloneqq e\ \texttt{in}\ e \mid \lambda(x:\tau).e \mid e(e) & \textit{variables, let and functions} \\ &\mid& \Lambda X.e \mid e[\tau] & \textit{universal quantification} \\ &\mid& \langle{}^*\tau, e\rangle\ \texttt{as}\ \exists X.\tau \mid \texttt{let}\ \langle{}^*X, x\rangle \coloneqq e\ \texttt{in}\ e & \textit{existential quantification} \end{array}$$

Syntax for values:

$$\begin{array}{rcll} nv \in \text{nval} &::=& \texttt{zero} \mid \texttt{succ}(nv) & \\ v \in \text{val} &::=& \texttt{true} \mid \texttt{false} & \textit{booleans} \\ &\mid& nv & \textit{natural numbers} \\ &\mid& \langle v, v\rangle & \textit{products} \\ &\mid& \lambda(x:\tau).e & \textit{functions} \\ &\mid& \Lambda X.e & \textit{universal quantification} \\ &\mid& \langle{}^*\tau, v\rangle\ \texttt{as}\ \exists X.\tau & \textit{existential quantification} \end{array}$$

Syntax for type system contexts:

$$\begin{array}{lll} \Gamma \in & \text{tenv} \Coloneqq \text{var} \rightharpoonup \text{type} & \textit{type environment} \\ S \in & \text{scope} \Coloneqq \wp(\text{tvar}) & \textit{type scope} \end{array}$$

The small-step transition relation:

$$\boxed{e \longrightarrow e}$$

If-True
$$\frac{}{\texttt{if}(\texttt{true})\{e_2\}\{e_3\} \longrightarrow e_2}$$

If-False
$$\frac{}{\texttt{if}(\texttt{false})\{e_2\}\{e_3\} \longrightarrow e_3}$$

If-Cong
$$\frac{e_1 \longrightarrow e_1'}{\texttt{if}(e_1)\{e_2\}\{e_3\} \longrightarrow \texttt{if}(e_1')\{e_2\}\{e_3\}}$$

Succ-Cong
$$\frac{e \longrightarrow e'}{\texttt{succ}(e_1) \longrightarrow \texttt{succ}(e_1')}$$

Pred-Zero
$$\frac{}{\texttt{pred}(\texttt{zero}) \longrightarrow \texttt{zero}}$$

Pred-Succ
$$\frac{}{\texttt{pred}(\texttt{succ}(nv)) \longrightarrow nv}$$

Pred-Cong
$$\frac{e \longrightarrow e'}{\texttt{pred}(e) \longrightarrow \texttt{pred}(e')}$$

IsZero-Zero
$$\frac{}{\texttt{iszero}(\texttt{zero}) \longrightarrow \texttt{true}}$$

IsZero-Succ
$$\frac{}{\texttt{iszero}(\texttt{succ}(nv)) \longrightarrow \texttt{false}}$$

IsZero-Cong
$$\frac{e \longrightarrow e'}{\texttt{iszero}(e) \longrightarrow \texttt{iszero}(e')}$$

Pair-Cong-1
$$\frac{e_1 \longrightarrow e_1'}{\langle e_1, e_2\rangle \longrightarrow \langle e_1', e_2\rangle}$$

Pair-Cong-2
$$\frac{e \longrightarrow e'}{\langle v, e\rangle \longrightarrow \langle v, e'\rangle}$$

ProjL-Pair
$$\frac{}{\texttt{projl}(\langle v_1, v_2\rangle) \longrightarrow v_1}$$

ProjL-Cong
$$\frac{e \longrightarrow e'}{\texttt{projl}(e) \longrightarrow \texttt{projl}(e')}$$

ProjR-Pair
$$\frac{}{\texttt{projr}(\langle v_1, v_2\rangle) \longrightarrow v_2}$$

ProjR-Cong
$$\frac{e \longrightarrow e'}{\texttt{projr}(e) \longrightarrow \texttt{projr}(e')}$$

Let-Val
$$\frac{}{\texttt{let}\ x \coloneqq v\ \texttt{in}\ e \longrightarrow [x \mapsto v]e}$$

Let-Cong
$$\frac{e_1 \longrightarrow e_1'}{\texttt{let}\ x \coloneqq e_1\ \texttt{in}\ e_2 \longrightarrow \texttt{let}\ x \coloneqq e_1'\ \texttt{in}\ e_2}$$

Apply-Lambda (β)
$$\frac{}{(\lambda(x:\tau).e)(v) \longrightarrow [x \mapsto v]e}$$

Apply-Cong-1
$$\frac{e_1 \longrightarrow e_1'}{e_1(e_2) \longrightarrow e_1'(e_2)}$$

Apply-Cong-2
$$\frac{e \longrightarrow e'}{v(e) \longrightarrow v(e')}$$

Type-Apply-Lambda
$$\frac{}{(\Lambda X.e)[\tau] \longrightarrow [X \mapsto \tau]e}$$

Type-Apply-Cong
$$\frac{e \longrightarrow e'}{e[\tau] \longrightarrow e'[\tau]}$$

Pack-Cong
$$\frac{e \longrightarrow e'}{\langle^{*}\tau, e\rangle\ \texttt{as}\ \exists X.\tau \longrightarrow \langle^{*}\tau, e'\rangle\ \texttt{as}\ \exists X.\tau}$$

Unpack-Pack
$$\frac{}{\texttt{let}\ \langle^{*}X, x\rangle \coloneqq (\langle^{*}\tau', v\rangle\ \texttt{as}\ \exists X.\tau)\ \texttt{in}\ e \longrightarrow [X \mapsto \tau'][x \mapsto v]e}$$

Unpack-Cong
$$\frac{e_1 \longrightarrow e_1'}{\texttt{let}\ \langle^{*}X, x\rangle \coloneqq e_1\ \texttt{in}\ e_2 \longrightarrow \texttt{let}\ \langle^{*}X, x\rangle \coloneqq e_1'\ \texttt{in}\ e_2}$$

The well-scoped relation:

$$\boxed{S \vdash \tau}$$

$$\textsc{Bool}\ \frac{}{S \vdash \texttt{bool}} \qquad \textsc{Nat}\ \frac{}{S \vdash \texttt{nat}} \qquad \textsc{Prod}\ \frac{S \vdash \tau_1 \qquad S \vdash \tau_2}{S \vdash \tau_1 \times \tau_2} \qquad \textsc{Fun}\ \frac{S \vdash \tau_1 \qquad S \vdash \tau_2}{S \vdash \tau_1 \rightarrow \tau_2} \qquad \textsc{TVar}\ \frac{X \in S}{S \vdash X} \qquad \textsc{Forall}\ \frac{S \cup \{X\} \vdash \tau}{S \vdash \forall X.\tau}$$

$$\textsc{Exists}\ \frac{S \cup \{X\} \vdash \tau}{S \vdash \exists X.\tau}$$

The well-typed relation:

$$\boxed{S, \Gamma \vdash e : \tau}$$

$$\textsc{True}\ \frac{}{S, \Gamma \vdash \texttt{true} : \texttt{bool}} \qquad \textsc{False}\ \frac{}{S, \Gamma \vdash \texttt{false} : \texttt{bool}} \qquad \textsc{If}\ \frac{S, \Gamma \vdash e_1 : \texttt{bool} \qquad S, \Gamma \vdash e_2 : \tau \qquad S, \Gamma \vdash e_3 : \tau}{S, \Gamma \vdash \texttt{if}(e_1)\{e_2\}\{e_3\} : \tau}$$

$$\textsc{Zero}\ \frac{}{S, \Gamma \vdash \texttt{zero} : \texttt{nat}} \qquad \textsc{Succ}\ \frac{S, \Gamma \vdash e : \texttt{nat}}{S, \Gamma \vdash \texttt{succ}(e) : \texttt{nat}} \qquad \textsc{Pred}\ \frac{S, \Gamma \vdash e : \texttt{nat}}{S, \Gamma \vdash \texttt{pred}(e) : \texttt{nat}} \qquad \textsc{IsZero}\ \frac{S, \Gamma \vdash e : \texttt{nat}}{S, \Gamma \vdash \texttt{iszero}(e) : \texttt{bool}}$$

$$\textsc{Pair}\ \frac{S, \Gamma \vdash e_1 : \tau_1 \qquad S, \Gamma \vdash e_2 : \tau_2}{S, \Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2} \qquad \textsc{ProjL}\ \frac{S, \Gamma \vdash e : \tau_1 \times \tau_2}{S, \Gamma \vdash \texttt{projl}(e) : \tau_1} \qquad \textsc{ProjR}\ \frac{S, \Gamma \vdash e : \tau_1 \times \tau_2}{S, \Gamma \vdash \texttt{projr}(e) : \tau_2} \qquad \textsc{Var}\ \frac{\Gamma(x) = \tau}{S, \Gamma \vdash x : \tau}$$

$$\textsc{Let}\ \frac{S, \Gamma \vdash e_1 : \tau_1 \qquad S, \Gamma[x \mapsto \tau_1] \vdash e_2 : \tau_2}{S, \Gamma \vdash \texttt{let}\ x \coloneqq e_1\ \texttt{in}\ e_2 : \tau_2} \qquad \textsc{Lambda}\ \frac{S \vdash \tau_1 \qquad S, \Gamma[x \mapsto \tau_1] \vdash e : \tau_2}{S, \Gamma \vdash \lambda(x : \tau_1).e : \tau_1 \rightarrow \tau_2}$$

$$\textsc{Apply}\ \frac{S, \Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \qquad S, \Gamma \vdash e_2 : \tau_1}{S, \Gamma \vdash e_1(e_2) : \tau_2} \qquad \textsc{TypeLambda}\ \frac{S \cup \{X\}, \Gamma \vdash e : \tau}{S, \Gamma \vdash \Lambda X.e : \forall X.\tau} \qquad \textsc{TypeApply}\ \frac{S \vdash \tau' \qquad S, \Gamma \vdash e : \forall X.\tau}{S, \Gamma \vdash e[\tau'] : [X \mapsto \tau']\tau}$$

$$\textsc{Pack}\ \frac{S, \Gamma \vdash e : [X \mapsto \tau']\tau}{S, \Gamma \vdash \langle {}^{*}\tau', e \rangle\ \texttt{as}\ \exists X.\tau : \exists X.\tau} \qquad \textsc{Unpack}\ \frac{S \vdash \tau_2 \qquad S, \Gamma \vdash e_1 : \exists X.\tau_1 \qquad S \cup \{X\}, \Gamma[x \mapsto \tau_1] \vdash e_2 : \tau_2}{S, \Gamma \vdash \texttt{let}\ \langle {}^{*}X, x \rangle \coloneqq e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\textsc{Unpack-Alt}\ \ (\textit{equivalent to}\ \textsc{Unpack})\ \frac{S \vdash \tau_2 \qquad S, \Gamma \vdash e_1 : \exists Y.\tau_1 \qquad S \cup \{X\}, \Gamma[x \mapsto [Y \mapsto X]\tau_1] \vdash e_2 : \tau_2}{S, \Gamma \vdash \texttt{let}\ \langle {}^{*}X, x \rangle \coloneqq e_1\ \texttt{in}\ e_2 : \tau_2}$$

2 Provided Helper Functions

These functions are fully implemented in the homework file `hw5.ml`, and not part of the assignment. However, we display the mathematical spec for these functions here, in case you want to try to understand that code and you find this spec helpful.

Free variables for types:

$$\begin{aligned}
\mathrm{FV} &\in \mathrm{type} \rightarrow \wp(\mathrm{tvar}) \\
\mathrm{FV}(\texttt{bool}) &\coloneqq \{\} \\
\mathrm{FV}(\texttt{nat}) &\coloneqq \{\} \\
\mathrm{FV}(\tau_1 \times \tau_2) &\coloneqq \mathrm{FV}(\tau_1) \cup \mathrm{FV}(\tau_2) \\
\mathrm{FV}(\tau_1 \rightarrow \tau_2) &\coloneqq \mathrm{FV}(\tau_1) \cup \mathrm{FV}(\tau_2) \\
\mathrm{FV}(X) &\coloneqq \{X\} \\
\mathrm{FV}(\forall X.\tau) &\coloneqq \mathrm{FV}(\tau) \setminus \{X\} \\
\mathrm{FV}(\exists X.\tau) &\coloneqq \mathrm{FV}(\tau) \setminus \{X\}
\end{aligned}$$

Substitution for types:

$$\begin{aligned}
[\_ \mapsto \_]\_ &\in \mathrm{tvar} \times \mathrm{type} \times \mathrm{type} \rightarrow \mathrm{type} \\
[X \mapsto \tau']\texttt{bool} &\coloneqq \texttt{bool} \\
[X \mapsto \tau']\texttt{nat} &\coloneqq \texttt{nat} \\
[X \mapsto \tau'](\tau_1 \times \tau_2) &\coloneqq [X \mapsto \tau']\tau_1 \times [X \mapsto \tau']\tau_2 \\
[X \mapsto \tau'](\tau_1 \rightarrow \tau_2) &\coloneqq [X \mapsto \tau']\tau_1 \rightarrow [X \mapsto \tau']\tau_2 \\
[X \mapsto \tau']Y &\coloneqq \begin{cases} \tau' & \textit{if} \quad X = Y \\ Y & \textit{if} \quad X \neq Y \end{cases} \\
[X \mapsto \tau'](\forall Y.\tau) &\coloneqq \begin{cases} \forall Y.\tau & \textit{if} \quad X = Y \\ \forall Y.[X \mapsto \tau']\tau & \textit{if} \quad X \neq Y \textit{ and } Y \notin \mathrm{FV}(\tau') \end{cases} \\
[X \mapsto \tau'](\exists Y.\tau) &\coloneqq \begin{cases} \exists Y.\tau & \textit{if} \quad X = Y \\ \exists Y.[X \mapsto \tau']\tau & \textit{if} \quad X \neq Y \textit{ and } Y \notin \mathrm{FV}(\tau') \end{cases}
\end{aligned}$$

Free variables for expressions:

$$\begin{aligned}
\mathrm{FV} &\in \mathrm{type} \rightarrow \wp(\mathrm{tvar}) \\
\mathrm{FV}(\texttt{true}) &\coloneqq \{\} \\
\mathrm{FV}(\texttt{false}) &\coloneqq \{\} \\
\mathrm{FV}(\texttt{if}(e_1)\{e_2\}\{e_3\}) &\coloneqq \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \cup \mathrm{FV}(e_3) \\
\mathrm{FV}(\texttt{zero}) &\coloneqq \{\} \\
\mathrm{FV}(\texttt{succ}(e)) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{pred}(e)) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{iszero}(e)) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\langle e_1, e_2 \rangle) &\coloneqq \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(\texttt{projl}(e)) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{projr}(e)) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(x) &\coloneqq \{x\} \\
\mathrm{FV}(\texttt{let}\ x \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \mathrm{FV}(e_1) \cup (\mathrm{FV}(e_2) \setminus \{x\}) \\
\mathrm{FV}(\lambda(x:\tau).e) &\coloneqq \mathrm{FV}(e) \setminus \{x\} \\
\mathrm{FV}(e_1(e_2)) &\coloneqq \mathrm{FV}(e_1) \cup \mathrm{FV}(e_2) \\
\mathrm{FV}(\Lambda X.e) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(e[\tau]) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\langle^* \tau, e \rangle\ \texttt{as}\ \exists X.\tau) &\coloneqq \mathrm{FV}(e) \\
\mathrm{FV}(\texttt{let}\ \langle^* X, x \rangle \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \mathrm{FV}(e_1) \cup (\mathrm{FV}(e_2) \setminus \{x\})
\end{aligned}$$

Substitution for expressions in expressions:

$$
\begin{aligned}
[\_ \mapsto \_]\_ &\in \text{var} \times \text{exp} \times \text{exp} \rightarrow \text{exp} \\
[x \mapsto e'](\texttt{true}) &\coloneqq \texttt{true} \\
[x \mapsto e'](\texttt{false}) &\coloneqq \texttt{false} \\
[x \mapsto e'](\texttt{if}(e_1)\{e_2\}\{e_3\}) &\coloneqq \texttt{if}([x \mapsto e']e_1)\{[x \mapsto e']e_2\}\{[x \mapsto e']e_3\} \\
[x \mapsto e'](\texttt{zero}) &\coloneqq \texttt{zero} \\
[x \mapsto e'](\texttt{succ}(e)) &\coloneqq \texttt{succ}([x \mapsto e']e) \\
[x \mapsto e'](\texttt{pred}(e)) &\coloneqq \texttt{pred}([x \mapsto e']e) \\
[x \mapsto e'](\texttt{iszero}(e)) &\coloneqq \texttt{iszero}([x \mapsto e']e) \\
[x \mapsto e'](\langle e_1, e_2\rangle) &\coloneqq \langle [x \mapsto e']e_1, [x \mapsto e']e_2\rangle \\
[x \mapsto e'](\texttt{projl}(e)) &\coloneqq \texttt{projl}([x \mapsto e']e) \\
[x \mapsto e'](\texttt{projr}(e)) &\coloneqq \texttt{projr}([x \mapsto e']e) \\
[x \mapsto e'](y) &\coloneqq \begin{cases} e' & \textit{if} \quad x = y \\ y & \textit{if} \quad x = y \end{cases} \\
[x \mapsto e'](\texttt{let}\ y \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \begin{cases} \texttt{let}\ y \coloneqq [x \mapsto e']e_1\ \texttt{in}\ e_2 & \textit{if} \quad x = y \\ \texttt{let}\ y \coloneqq [x \mapsto e']e_1\ \texttt{in}\ [x \mapsto e']e_2 & \textit{if} \quad x \neq y \textit{ and } y \notin \text{FV}(e') \end{cases} \\
[x \mapsto e'](\lambda(y:\tau).e) &\coloneqq \begin{cases} \lambda(y:\tau).e & \textit{if} \quad x = y \\ \lambda(y:\tau).[x \mapsto e']e & \textit{if} \quad x \neq y \textit{ and } y \notin \text{FV}(e') \end{cases} \\
[x \mapsto e'](e_1(e_2)) &\coloneqq [x \mapsto e']e_1([x \mapsto e']e_2) \\
[x \mapsto e'](\Lambda X.e) &\coloneqq \Lambda X.[x \mapsto e']e \\
[x \mapsto e'](e[\tau]) &\coloneqq [x \mapsto e']e[\tau] \\
[x \mapsto e'](\langle {}^*\tau, e\rangle\ \texttt{as}\ \exists X.\tau) &\coloneqq \langle {}^*\tau, [x \mapsto e']e\rangle\ \texttt{as}\ \exists X.\tau \\
[x \mapsto e'](\texttt{let}\ \langle {}^*X, y\rangle \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \begin{cases} \texttt{let}\ \langle {}^*X, y\rangle \coloneqq [x \mapsto e']e_1\ \texttt{in}\ e_2 & \textit{if} \quad x = y \\ \texttt{let}\ \langle {}^*X, y\rangle \coloneqq [x \mapsto e']e_1\ \texttt{in}\ [x \mapsto e']e_2 & \textit{if} \quad x \neq y \textit{ and } y \notin \text{FV}(e') \end{cases}
\end{aligned}
$$

Substitution for types in expressions:

$$
\begin{aligned}
[\_ \mapsto \_]\_ &\in \text{tvar} \times \text{type} \times \text{exp} \rightarrow \text{exp} \\
[X \mapsto \tau'](\texttt{true}) &\coloneqq \texttt{true} \\
[X \mapsto \tau'](\texttt{false}) &\coloneqq \texttt{false} \\
[X \mapsto \tau'](\texttt{if}(e_1)\{e_2\}\{e_3\}) &\coloneqq \texttt{if}([X \mapsto \tau']e_1)\{[X \mapsto \tau']e_2\}\{[X \mapsto \tau']e_3\} \\
[X \mapsto \tau'](\texttt{zero}) &\coloneqq \texttt{zero} \\
[X \mapsto \tau'](\texttt{succ}(e)) &\coloneqq \texttt{succ}([X \mapsto \tau']e) \\
[X \mapsto \tau'](\texttt{pred}(e)) &\coloneqq \texttt{pred}([X \mapsto \tau']e) \\
[X \mapsto \tau'](\texttt{iszero}(e)) &\coloneqq \texttt{iszero}([X \mapsto \tau']e) \\
[X \mapsto \tau'](\langle e_1, e_2\rangle) &\coloneqq \langle [X \mapsto \tau']e_1, [X \mapsto \tau']e_2\rangle \\
[X \mapsto \tau'](\texttt{projl}(e)) &\coloneqq \texttt{projl}([X \mapsto \tau']e) \\
[X \mapsto \tau'](\texttt{projr}(e)) &\coloneqq \texttt{projr}([X \mapsto \tau']e) \\
[X \mapsto \tau'](y) &\coloneqq y \\
[X \mapsto \tau'](\texttt{let}\ x \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \texttt{let}\ x \coloneqq [X \mapsto \tau']e_1\ \texttt{in}\ [X \mapsto \tau']e_2 \\
[X \mapsto \tau'](\lambda(x:\tau).e) &\coloneqq \lambda(x:[X \mapsto \tau']\tau).[X \mapsto \tau']e \\
[X \mapsto \tau'](e_1(e_2)) &\coloneqq [X \mapsto \tau']e_1([X \mapsto \tau']e_2) \\
[X \mapsto \tau'](\Lambda Y.e) &\coloneqq \begin{cases} \Lambda Y.e & \textit{if} \quad X = Y \\ \Lambda Y.[X \mapsto \tau']e & \textit{if} \quad X \neq Y \textit{ and } Y \notin \text{FV}(\tau') \end{cases} \\
[X \mapsto \tau'](e[\tau]) &\coloneqq [X \mapsto \tau']e[[X \mapsto \tau']\tau] \\
[X \mapsto \tau'](\langle {}^*\tau, e\rangle\ \texttt{as}\ \exists Y.\tau) &\coloneqq \begin{cases} \langle {}^*[X \mapsto \tau']\tau, [X \mapsto \tau']e\rangle\ \texttt{as}\ \exists Y.\tau & \textit{if} \quad X = Y \\ \langle {}^*[X \mapsto \tau']\tau, [X \mapsto \tau']e\rangle\ \texttt{as}\ \exists Y.[X \mapsto \tau']\tau & \textit{if} \quad X \neq Y \textit{ and } Y \notin \text{FV}(\tau') \end{cases} \\
[X \mapsto \tau'](\texttt{let}\ \langle {}^*Y, x\rangle \coloneqq e_1\ \texttt{in}\ e_2) &\coloneqq \begin{cases} \texttt{let}\ \langle {}^*Y, x\rangle \coloneqq [X \mapsto \tau']e_1\ \texttt{in}\ e_2 & \textit{if} \quad X = Y \\ \texttt{let}\ \langle {}^*Y, x\rangle \coloneqq [X \mapsto \tau']e_1\ \texttt{in}\ [X \mapsto \tau']e_2 & \textit{if} \quad X \neq Y \textit{ and } Y \notin \text{FV}(\tau') \end{cases}
\end{aligned}
$$