Logical Verification Assignment 1

August 28, 2025

Recall the following simple language from the lecture:

$$
\begin{array}{lrcl}
\text{Types} & \tau & ::= & \mathsf{num} \mid \mathsf{str} \\
\text{Expressions} & e & ::= & x \mid \mathsf{num}[n] \mid \mathsf{str}[s] \mid \mathsf{plus}(e_1, e_2) \\
& & \mid & \mathsf{times}(e_1, e_2) \mid \mathsf{cat}(e_1, e_2) \mid \mathsf{len}(e) \mid \mathsf{let}\ x\ \mathsf{be}\ e_1\ \mathsf{in}\ e_2
\end{array}
$$

1 Problem 1

Prove by induction on the appropriate derivations that the order of variables in a typing context does not matter:

Lemma 1. *If* $\Gamma, x : \tau_1, y : \tau_2 \vdash e : \tau$ *then* $\Gamma, y : \tau_2, x : \tau_1 \vdash e : \tau$.

Feel free to skip cases that are analogous to cases that you have already considered.

2 Problem 2

Consider the following extension of our language:

$$
\begin{array}{lrcl}
\text{Types} & \tau & ::= & \mathsf{Bool} \mid \ldots \\
\text{Expressions} & e & ::= & \mathsf{true} \mid \mathsf{false} \mid \mathsf{if}\ e\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2
\end{array}
$$

Give the corresponding typing and operational semantics rules for the additional constructs.

3 Problem 3

Adapt the operational semantics rules of our language so that $\mathsf{let}\ x\ \mathsf{be}\ e_1\ \mathsf{in}\ e_2$ uses a *call-by-value* semantics, where the expression e_1 needs to be fully evaluated before being substituted into e_2. Show that the resulting language is type-safe by revisiting the progress and preservation theorems:

Lemma 2 (Preservation)**.** *If* $e : \tau$ *and* $e \mapsto e'$ *then* $e' : \tau$*.*

Lemma 3 (Progress)**.** *If* $e : \tau$ *then either* $e\ \mathsf{val}$ *or* $e \mapsto e'$*.*

You only need to prove cases involving the $\mathsf{let}\ x\ \mathsf{be}\ e_1\ \mathsf{in}\ e_2$ construct.