The syntax for the core language is as follows:

\[
C ::= L := E \mid C_1 ; C_2 \mid \text{if } E \text{ then } C_1 \text{ else } C_2 \text{ fi} \mid \text{while } E \text{ do } C \text{ od} \mid \text{skip}
\]

\[
E ::= N \mid @L \mid E_1 + E_2 \mid \neg E \mid E_1 = E_2
\]

\[
L ::= \text{loc}_i \text{ if } i > 0 \]

\[
N ::= n \text{ if } n \in \mathbb{Z}
\]

The typing rules are given as follows:

For Location:

\[
\text{loc}_i : \text{intloc} \quad \text{if } i > 0
\]

For Numeral

\[
n : \text{int} \quad \text{if } n \in \mathbb{Z}
\]

For Expression:

\[
N : \text{int} \quad L : \text{intloc} \quad E_1 : \text{intexp} \quad E_2 : \text{intexp}
\]

\[
@L : \text{intexp} \quad E_1 + E_2 : \text{intexp}
\]

\[
\neg E : \text{boolexp} \quad E_1 : \tau \text{exp} \quad E_2 : \tau \text{exp}
\]

\[
\text{if } \tau \in \{\text{int, bool}\}
\]

For command:

\[
L : \text{intloc} \quad E : \text{intexp} \quad C_1 : \text{comm} \quad C_2 : \text{comm}
\]

\[
L := E : \text{comm} \quad C_1 ; C_2 : \text{comm}
\]

\[
\text{if } E \text{ then } C_1 \text{ else } C_2 \text{ fi} : \text{comm}
\]

\[
\text{while } E \text{ do } C \text{ od} : \text{comm}
\]

\[
\text{skip} : \text{comm}
\]

Discussion:

My style of writing the proofs differs a bit from Schmidt’s. Here is a step-by-step description of how to build the derivation that the syntactically well-formed command

\[
\text{loc}_1 ::= 5 + @\text{loc}_2
\]

is a well-typed command. The derivation is built by following the structure of the syntax tree for the assignment \( L := E \) where, in this case, \( L = \text{loc}_1 \) and \( E = 5 + @\text{loc}_2 \). The rule for assignment is the only rule that applies because it is at the root of the syntax tree. Applying it yields the following partial derivation.
On the left branch of the proof we must show \( \text{loc}_1 \) is a well-typed \( \text{intloc} \) and on the right branch that \( 5 + \text{loc}_2 \) is an \( \text{intexp} \). But since \( 1 > 0 \), \( \text{loc}_1 \) : \( \text{intloc} \) holds and the left branch of the proof is finished. We indicate that by drawing a line over it in the derivation below. To show that \( 5 + \text{loc}_2 \) is well typed we apply the rule for addition \( E_1 + E_2 : \text{intexp} \) where \( E_1 \) is 5 and \( E_2 \) is \( \text{loc}_2 \). Working up in the rule we must show that both \( E_1 \) and \( E_2 \) have type \( \text{intexp} \). Having applied these rules, the derivation appears as follows.

\[
\begin{array}{c}
\text{loc}_1 : \text{intloc} \quad 5 + \text{loc}_2 : \text{intexp} \\
\hline
\text{loc}_1 := 5 + \text{loc}_2 : \text{comm}
\end{array}
\]

Now, there are still two open branches left. We must show that \( 5 : \text{intexp} \) and the \( \text{loc}_2 \) is an \( \text{intexp} \). On the left size we apply the rule to shown \( N : \text{intexp} \), i.e. in this case that \( 5 : \text{intexp} \) which simply leaves the branch open with the goal of showing that \( 5 : \text{int} \) which is simply true since \( 5 \in \mathbb{Z} \) and we close that branch having applied two rules. To show that \( \text{loc}_2 : \text{intexp} \) we apply the rule whose base matches \( \text{loc}_1 : \text{intexp} \) and let \( i = 2 \). Working up the rule this leaves the goal to show that \( \text{loc}_2 \) is an \( \text{intloc} \) - which it is because \( 2 > 0 \). This yields the following completed derivation tree.

\[
\begin{array}{c}
5 : \text{int} \\
\hline
\text{loc}_1 : \text{intloc} \quad \text{loc}_2 : \text{intloc} \quad \text{loc}_2 : \text{intexp} \\
\hline
\text{loc}_1 := 5 + \text{loc}_2 : \text{comm}
\end{array}
\]

Here’s a proof that the command \( \text{if } \text{loc}_1 = 0 \text{ then } \text{loc}_1 := \text{loc}_2 \text{ else skip fi} \) is well-typed.

\[
\begin{array}{c}
\text{loc}_1 : \text{intloc} \\
\hline
\text{lo}c_1 : \text{intexp} \quad 0 : \text{int} \\
\hline
\text{loc}_1 := \text{loc}_2 : \text{comm}
\end{array}
\]

Assignment:

1. Read pages 1 – 14 of Schmidt.
2. Using the style of proof presented here, prove that the example in figure 1.5 (pp. 10) is well-typed.
3. Prove the following command is well-typed.

\[
\begin{align*}
loc_1 := 0; & \textbf{while } \neg (\@loc_1 = 0) \textbf{ do } \loc_1 := \@loc_2 + 1 \textbf{ od } : \text{comm}
\end{align*}
\]