8 Semantics of Programming Languages (nk480)

Languages like FORTH and POSTSCRIPT are stack-based languages; they store intermediate values on a stack rather than binding to variable names. In this question we will look at how to give a type system and operational semantics for a simple stack-based language. The syntax and informal meaning of our language is given by:

\[
e ::= \begin{align*}
    n & \quad \text{Push the numeral } n \text{ on the stack} \\
    b & \quad \text{Push the Boolean } b \text{ on the stack} \\
    \text{Add} & \quad \text{Replace the top two stack elements with their sum} \\
    \text{Eql} & \quad \text{Replace the top two stack elements with the result of comparing them for equality} \\
    \text{Cond}(e_1, e_2) & \quad \text{Delete the top stack element and execute } e_1 \text{ or } e_2, \\
    \text{Skip} & \quad \text{No-op} \\
    e_1; e_2 & \quad \text{Run } e_1 \text{ and then } e_2
\end{align*}
\]

\[
v ::= \begin{align*}
    b & \quad \text{Values} \\
    n & \quad \text{Values}
\end{align*}
\]

\[
s ::= \begin{align*}
    \cdot & \quad \text{Stacks} \\
    s, v & \quad \text{Stacks}
\end{align*}
\]

\[
\tau ::= \begin{align*}
    \text{bool} & \quad \text{Types} \\
    \text{num} & \quad \text{Types}
\end{align*}
\]

\[
\Gamma ::= \begin{align*}
    \cdot & \quad \text{Stack Types} \\
    \Gamma, \tau & \quad \text{Stack Types}
\end{align*}
\]

We take a value \( v \) to be a Boolean or numeral, and define a stack \( s \) to be a stack of values (growing at the right). Correspondingly, there are types \( \text{bool} \) and \( \text{num} \) for values, and stack types \( \Gamma \) for stacks \( s \).

The small-step operational semantics is then defined by a transition relation \( \langle e_1 \mid s_1 \rangle \rightarrow \langle e_2 \mid s_2 \rangle \). One rule for this relation is:

\[
\langle \text{Add} \mid s, n, m \rangle \rightarrow \langle \text{Skip} \mid s, n + m \rangle
\]

The typing relation is given as a relation \( \Gamma \vdash e \vdash \Gamma' \), which means that \( e \), when run with a stack of shape \( \Gamma \), yields a stack of shape \( \Gamma' \). One rule for this relation is:

\[
\Gamma, \text{num} \vdash \text{Add} \vdash \Gamma', \text{num}
\]

(a) Give the remaining rules for the operational semantics. [7 marks]

(b) Give the remaining rules for the typing judgement. [7 marks]

(c) Formulate and state the progress and preservation lemmas for this language. [6 marks]