Mini-Lua variable-binding semantics

This handout formalizes the rules for variable binding in Mini-Lua programs. We use a stripped-down abstract syntax for Mini-Lua programs, which only includes variable and function definitions, blocks, and variable uses (we use \( s \) for statements, \( e \) for expressions, and \( f \) and \( x \) for variables).

\[
\begin{align*}
  s & ::= \ s_1 ; \ s_2 \\
       & | \ x = \ e \\
       & | \ \text{local} \ x = \ e \\
       & | \ \text{local function} \ f \ (x_1, \ldots, x_n) \ s \\
       & | \ \text{if} \ e \ \text{then} \ e_1 \ \text{else} \ e_2 \\
\end{align*}
\]

\[
\begin{align*}
  e & ::= \ x 
\end{align*}
\]

Our “typing” judgements work on sets of globals (\( G \)) and environments (\( E \)), which are defined as follows:

\[
\begin{align*}
  G & \in 2^\text{Var} \\
  E & \in \text{Env} = \text{Var}^{\text{fin}} \to \{\text{local, glob}\}
\end{align*}
\]

The judgement forms are \( E \vdash s : (E', G) \), which means that under environment \( E \), the statement \( s \) defines the environment \( E' \) and the set of globals \( G \), and \( E \vdash e \ \text{Ok} \), which means that the variables used in \( e \) are defined in \( E \).

For statement sequencing, we use the environment from the first statement to check the second and union the set of globals.

\[
\frac{E_0 \vdash s_1 : (E_1, G_1) \quad E_1 \vdash s_2 : (E_2, G_2)}{E_0 \vdash s_1 ; s_2 : (E_2, G_1 \cup G_2)}
\]

A definition of a global variable extends the environment, assuming that the right-hand side is okay, as well as adding to the set of globals.

\[
\frac{E \vdash e \ \text{Ok}}{E \vdash x = e : (E \pm \{x \mapsto \text{glob}\}, \{x\})}
\]

A definition of a local variable also extends the environment, assuming that the right-hand side is okay.

\[
\frac{E \vdash e \ \text{Ok}}{E \vdash \text{local} \ x = e : (E \pm \{x \mapsto \text{local}\}, \{\})}
\]
A block localizes the environment (i.e., definitions do not escape), but note that the set of defined globals does escape.

\[
E \vdash s : \langle E', G \rangle \\
E \vdash \text{do s end} : \langle E, G \rangle
\]

Like a block, a function definition localizes the environment generated by its body. Note that the body is checked in an environment that includes the function name itself.

\[
E' = E \pm \{ f \mapsto \text{glob} \} \\
E'' = E' \pm \{ x_1 \mapsto \text{local}, \ldots, x_n \mapsto \text{local} \} \\
E'' \vdash s : \langle E''', G \rangle
\]

\[
E \vdash \text{function } f (x_1, \ldots, x_n) s : \langle E', G \rangle
\]

Local functions are similar to global functions.

\[
E' = E \pm \{ f \mapsto \text{local} \} \\
E'' = E' \pm \{ x_1 \mapsto \text{local}, \ldots, x_n \mapsto \text{local} \} \\
E'' \vdash s : \langle E''', G \rangle
\]

\[
E \vdash \text{local function } f (x_1, \ldots, x_n) s : \langle E', G \rangle
\]

The conditional statement also localizes any definitions in its arms.

\[
E \vdash e \text{ Ok} \\
E \vdash e_1 : \langle E_1, G_1 \rangle \\
E \vdash e_2 : \langle E_2, G_2 \rangle \\
E \vdash \text{if } e \text{ then } e_1 \text{ else } e_2 : \langle E, G_1 \cup G_2 \rangle
\]

Lastly, an expression is okay if its variables have been defined.

\[
x \in \text{dom}(E) \\
E \vdash x \text{ Ok}
\]