# Abstract machine $E$

The abstract machine $E$ is for a practical implementation of the environment semantics. There are two kinds of states in the abstract machine $E$.

- $\sigma \triangleright e @ \eta$ means that the machine is currently analyzing expression $e$ under environment $\eta$. In order to evaluate a variable in $e$, we look up the environment $\eta$.

- $\sigma \smashleft v$ means that the machine is currently returning value $v$ the stack $\sigma$. We do not need an environment for $v$ because the evaluation of $v$ has been finished.

## Question 1. [50 points] Complete the definition of the abstract machine $E$.

### expression

$$v ::= x | \lambda x : A . e | e e$$

### value

$$v ::= [\eta, \lambda x : A . e]$$

### environment

$$\eta ::= \emptyset | \eta, x \mapsto v$$

### frame

$$\phi ::= \Box_{\eta} e | [\eta, \lambda x : A . e] \Box$$

### stack

$$\sigma ::= \emptyset | \sigma ; \phi$$

### state

$$s ::= \sigma \triangleright e @ \eta | \sigma \smashleft v$$

$$x \mapsto v \in \eta$$

$$\sigma \triangleright x \mapsto \eta \mapsto_{E} \sigma \smashleft v$$

$$\sigma \triangleright \lambda x : A . e @ \eta \mapsto_{E} \sigma \smashleft [\eta, \lambda x : A . e]$$

$$\sigma \triangleright e_1 \circ e_2 @ \eta \mapsto_{E} \sigma ; \Box_{\eta} e_2 \triangleright e_1 @ \eta$$

$$\sigma ; \Box_{\eta} e_2 \smashleft [\eta', \lambda x : A . e] \mapsto_{E} \sigma ; [\eta', \lambda x : A . e] \Box \triangleright e_2 @ \eta$$

$$\sigma ; [\eta, \lambda x : A . e] \Box \smashleft v \mapsto_{E} \sigma \triangleright e @ \eta, x \mapsto v$$

## Question 2. [20 points] Suppose that we use an environment evaluation judgment $\eta \vdash e \mapsto v$. State the correctness of the abstract machine $E$.

$\eta \vdash e \mapsto v$ if and only if $\sigma \triangleright e @ \eta \mapsto_{E} \sigma \smashleft v$. 
2 Subtyping

**Question 3. [30 points]** The principle of subtyping is a principle specifying when a type is a subtype of another type. It states that \( A \) is a subtype of \( B \) if an expression of type \( A \) may be used wherever an expression of type \( B \) is expected. Formally we write \( A \leq B \) if \( A \) is a subtype of \( B \).

The principle of subtyping justifies two subtyping rules: reflexivity and transitivity. Fill in the blank to complete the two subtyping rules:

\[
\begin{align*}
A \leq A & \quad \text{Refl}_\leq \\
A \leq B \quad B \leq C & \quad \frac{}{A \leq C} \quad \text{Trans}_\leq
\end{align*}
\]

The rule of subsumption is a typing rule which enables us to change the type of an expression to its supertype. Assume that we use a typing judgment \( \Gamma \vdash e : A \) as in the course notes, and give the rule of subsumption:

\[
\begin{align*}
\Gamma \vdash e : A \quad A \leq B & \quad \frac{}{\Gamma \vdash e : B} \quad \text{Sub}
\end{align*}
\]