1 Virtual Machine Abstract Syntax

$$
\begin{array}{llll}
S & ::= & d; S & \textit{statements} \\
  & | & e & \\
\\
d & ::= & \mathtt{val}\ f = e & \textit{declarations} \\
  & | & \mathtt{def}\ m(\overline{x : \tau}) : \tau = S & \\
  & | & \mathtt{type}\ L = \tau & \\
\\
e & ::= & x & \textit{expressions} \\
  & | & \mathtt{new}\ \tau\ \{\overline{d}\} & \\
  & | & e.m(\overline{e}) & \\
  & | & e.f & \\
  & | & \mathscr{L} & \\
\\
\mathscr{L} & ::= & n & \textit{literals}
\end{array}
$$

$$
\begin{array}{llll}
\beta & ::= & \mathtt{Unit} & \textit{base type} \\
  & | & L & \\
\\
\tau & ::= & \beta\{\overline{\sigma}\} & \textit{type} \\
\\
\sigma & ::= & \mathtt{val}\ f : \tau & \textit{decl type} \\
  & | & \mathtt{def}\ m(\overline{x : \tau}) : \tau & \\
  & | & \mathtt{type}\ L = \tau &
\end{array}
$$

Notation: overbar means a list of elements, as in Java

2 Standard prelude

```
type Int
    def +(i:Int):Int
    def −(i:Int):Int
    def ∗(i:Int):Int
    def /(i:Int):Int
```

3 Virtual Machine Typing Rules

$\boxed{\Gamma \vdash e : \tau}$

$$\frac{\Gamma \vdash d : \sigma \quad \Gamma, \sigma \vdash S : \tau}{\Gamma \vdash d; S : \tau}\ (\text{T-Stmt})$$

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\ (\text{T-Var})$$

$$\frac{\Gamma \vdash \overline{d} : \mathit{unfold}_\Gamma(\tau)}{\Gamma \vdash \mathtt{new}\ \ \tau\ \{\overline{d}\} : \tau}\ (\text{T-New})$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \mathtt{def}\ m(x : \tau_2) : \tau \in \mathit{unfold}_\Gamma(\tau_1) \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1.m(e_2) : \tau}\ (\text{T-Invk})$$

$$\frac{\Gamma \vdash e : \tau' \quad \mathtt{val}\ f : \tau \in \mathit{unfold}_\Gamma(\tau')}{\Gamma \vdash e.f : \tau}\ (\text{T-Field})$$

$$\frac{}{\Gamma \vdash n : \mathit{Int}}\ (\text{T-Int})$$

Technically Γ is a list of σ, but we often write $x : \tau$ for `val` $x : \tau$.

$\boxed{\Gamma \vdash d : \sigma}$

$$\frac{\Gamma \vdash \overline{\tau}\ \mathit{wf} \quad \Gamma,\ \overline{y : \tau} \vdash e : \tau_2}{\Gamma \vdash \mathtt{def}\ m(\overline{y : \tau}) : \tau_2 = e\ \ :\ \ \mathtt{def}\ m(\overline{y : \tau}) : \tau_2}\ \text{(T-Def)}$$

$$\frac{\Gamma \vdash e : \tau}{\Gamma \vdash \mathtt{val}\ f = e\ \ :\ \ \mathtt{val}\ f : \tau}\ \text{(T-Val)}$$

$$\frac{\Gamma \vdash \mathtt{type}\ L = \tau\ \mathit{wf}}{\Gamma \vdash \mathtt{type}\ L = \tau : \mathtt{type}\ L = \tau}\ \text{(T-Type)}$$

$\boxed{\mathit{unfold}_\Gamma(\tau) = \overline{\sigma}}$

$$\frac{}{\mathit{unfold}_\Gamma(\mathtt{Unit}) = \bullet}$$

$$\frac{\mathtt{type}\ L = \tau \in \Gamma \quad \mathit{unfold}_\Gamma(\tau) = \overline{\sigma}}{\mathit{unfold}_\Gamma(L) = \overline{\sigma}}$$

$$\frac{\mathit{unfold}_\Gamma(\beta) = \overline{\sigma}}{\mathit{unfold}_\Gamma(\beta\{\overline{\sigma'}\}) = \overline{\sigma} \leftarrow \overline{\sigma'}}$$

Note: $\overline{\sigma} \leftarrow \overline{\sigma'}$ means that we append the two lists, except that when the same symbol is defined in both $\overline{\sigma}$ and $\overline{\sigma'}$, we include only the (overriding) definition in $\overline{\sigma'}$.

Now, finally, type and declaration type well-formedness rules: $\boxed{\Gamma \vdash \tau\ \mathit{wf}}$

$$\frac{}{\Gamma \vdash \mathtt{Unit}\ \ \mathit{wf}}$$

$$\frac{\mathit{unfold}_\Gamma(L) = \overline{\sigma}}{\Gamma \vdash L\ \mathit{wf}}$$

$$\frac{\Gamma \vdash \beta\ \mathit{wf} \quad \Gamma \vdash \overline{\sigma'}\ \mathit{wf}}{\Gamma \vdash \beta\{\overline{\sigma'}\}\ \mathit{wf}}$$

$\boxed{\Gamma \vdash \sigma\ \mathit{wf}}$

$$\frac{\Gamma \vdash \overline{y : \tau}\ \mathit{wf} \quad \Gamma \vdash \tau_2\ \mathit{wf}}{\Gamma \vdash \mathtt{def}\ m(\overline{y : \tau}) : \tau_2\ \mathit{wf}}$$

$$\frac{\Gamma \vdash \tau\ \mathit{wf}}{\Gamma \vdash \mathtt{val}\ f : \tau\ \mathit{wf}}$$

$$\frac{\Gamma, \mathtt{type}\ L = \tau \vdash \tau\ \mathit{wf}}{\Gamma \vdash \mathtt{type}\ L = \tau\ \mathit{wf}}$$