$g \in [V_\theta \to_c V_*]$ $\quad (\theta \in \{int, bool, fun, tuple, alt\})$

$g_{\theta*} \in [V_* \to_c V_*]$

$$g_{\theta*}(a) = \begin{cases} g(b) & \text{if } a = \langle 0, b\rangle \text{ and } \exists i, b' \text{ s.t. } b = \phi\langle i, b'\rangle \text{ and } b' \in V_\theta \\ \langle 1, typeerror\rangle & \text{if } a = \langle 0, b\rangle \text{ and not } (\exists i, b' \text{ s.t. } b = \phi\langle i, b'\rangle \text{ and } b' \in V_\theta) \\ a & \text{otherwise.} \end{cases}$$

Intuitively, when $g_{\theta*}$ is applied to an element in V, i.e. $\langle 0, b\rangle$ (a value, not an error or $\perp$), it first does runtime typechecking and finds out whether b is a value of type θ. If not, $g_{\theta*}$ returns a typeerror $\langle 1, typeerror\rangle$. If yes, on the other hand, it casts b to a value b' of type θ, and runs g on b'.

Thus, $g_{\theta*}$ does type-testing and type-casting.

(4) Here is the definition of $[\![-]\!]$. We present only some cases. Look at the textbook for the complete definition.

$$[\![-]\!] \in [\langle exp\rangle \to \underbrace{V^{\langle var\rangle}}_{\text{often written as } E} \to V_*]$$

often written as E to emphasise that it is the set of environments.

$$[\![v]\!]\eta = \langle 0, \eta(v)\rangle$$

$$[\![e_0 e_1]\!]\eta = \begin{cases} [\![e_0]\!]\eta & \text{if } [\![e_0]\!]\eta \in \{\perp, \langle 1, error\rangle, \langle 1, typeerror\rangle\} \\ \langle 1, typeerror\rangle & \text{else if } [\![e_0]\!]\eta = \langle 0, b\rangle \text{ but not } (\exists b' \text{ s.t. } b = \phi(\langle 2, b'\rangle) \wedge b' \in V_{fun}) \\ [\![e_1]\!]\eta & \text{else if } [\![e_1]\!]\eta \in \{\perp, \langle 1, error\rangle, \langle 1, typeerror\rangle\} \\ b'(a) & \text{else if } [\![e_0]\!]\eta = \langle 0, b\rangle \text{ and } \exists b' \in V_{fun} \text{ s.t. } b = \phi(\langle 2, b'\rangle) \end{cases}$$

and $[\![e_1]\!]\eta = \langle 0, a\rangle$

More succinctly, we can write:

$$[\![e_0 e_1]\!]\eta = (\underbrace{\lambda f.}_{\text{function definition in math}} (\lambda z. \underbrace{f(z)}_{\text{function application in math}}))_* ([\![e_1]\!]\eta))_{\underbrace{fun*}_{\text{typechecking and typecasting}}} ([\![e_0]\!]\eta).$$

$$[\![\lambda v.e]\!]\eta = \langle 0, \phi(\langle 2, \lambda z. [\![e]\!][\eta \mid v:z]\rangle)\rangle$$

$$[\![\langle e_0, \cdots, e_{n-1}\rangle]\!]\eta = (\lambda z_0. \cdots (\lambda z_{n-1}. \langle 0, \phi(\langle 3, \langle z_0, \cdots, z_{n-1}\rangle\rangle)\rangle)_* ([\![e_{n-1}]\!]\eta) \cdots)_* ([\![e_0]\!]\eta)$$

$$[\![e.k]\!]\eta = (\lambda t. \text{ if } k \in dom(t) \text{ then } \langle 0, \overset{k+1\text{-th component of } t}{t_k}\rangle \text{ else } \langle 1, typeerror\rangle)_{\underset{tuple}{*}} ([\![e]\!]\eta)$$

t consists of at least $k+1$ components.

$$[\![@\, k\ e]\!]\eta = (\lambda z. \langle 0, \phi(\langle 4, \langle k, z\rangle\rangle)\rangle)_* ([\![e]\!]\eta)$$

$$[\![\text{sumcase } e \text{ of } (e_0, \cdots, e_{n-1})]\!]\eta = (\lambda\langle k, z\rangle. \text{ if } k < n \text{ then } (\lambda f. f(z))_{fun*} ([\![e_k]\!]\eta) \text{ else } \langle 1, typeerror\rangle)_{alt*} ([\![e]\!]\eta)$$

$$[\![k]\!]\eta = \langle 0, \phi(\langle 0, k\rangle)\rangle$$

$$[\![e_0 + e_1]\!]\eta = (\lambda i. (\lambda i'. \langle 0, \phi(\langle 0, i + i'\rangle)\rangle)_{int*} ([\![e_1]\!]\eta))_{int*} ([\![e_0]\!]\eta)$$

$[\![typeerror]\!]\eta = \langle 1, typeerror\rangle$ $\qquad$ $[\![true]\!]\eta = \langle 0, \phi(\langle 1, tt\rangle)\rangle$