Notations

We use $\overline{::}_2$ to push a continuation on the continuation stack of the topmost fiber, without modifyin the effect handler:

$$\kappa \overline{::}_2 (\langle \kappa s, \eta \rangle \overline{::} \mu s) = \langle \kappa \overline{::} \kappa s, \ \eta \rangle \overline{::} \mu s$$

Computations

$$[\![V\,W]\!] = \overline{\lambda}\kappa s.\ [\![V]\!] \underline{@} [\![W]\!] \underline{@} \downarrow \kappa s$$

$$[\![\mathbf{absurd}\ V]\!] = \overline{\lambda}\kappa s.\ \mathbf{absurd}\ [\![V]\!]$$

$$[\![\mathbf{return}\ V]\!] = \overline{\lambda}\langle \kappa \overline{::} \kappa s, \ \eta \rangle \overline{::} \mu s.\ \kappa \underline{@} [\![V]\!] \underline{@} \downarrow(\langle \kappa s, \eta \rangle \overline{::} \mu s)$$

$$[\![\mathbf{let}\ x \leftarrow M\ \mathbf{in}\ N]\!] = \overline{\lambda}\langle \kappa \overline{::} \kappa s, \ \eta \rangle \overline{::} \mu s.$$
$$[\![M]\!] \overline{@} \left(\left(\underline{\lambda}x\ ks.\ [\![N]\!] \overline{@} (\kappa \overline{::}_2 \uparrow ks) \right) \overline{::}_2 (\langle \kappa s, \eta \rangle \overline{::} \mu s) \right)$$

$$[\![\mathbf{perform}\ V]\!] = \overline{\lambda}\langle \kappa s, \ \eta \rangle \overline{::} \mu s.\ \eta \underline{@} [\![V]\!] \underline{@} [\langle \downarrow \kappa s, \eta \rangle] \underline{@} \downarrow \mu s$$

$$[\![\mathbf{handle}\ M\ \mathbf{with}\ \{H_v, H_x, H_f\}]\!] = \overline{\lambda}\mu s.\ [\![M]\!] \overline{@} (\langle [\![H_v]\!] \overline{::} [\![H_x]\!] \overline{::} [], \ [\![H_f]\!] \rangle \overline{::} \mu s)$$

where

$$
\begin{aligned}
H_v : \quad [\![\mathbf{return}\ x \mapsto N]\!] \quad &= \quad \underline{\lambda}x\ ks.\ \underline{\mathbf{let}}\ \langle ks', \ \eta \rangle \underline{::} \langle ks'', \iota \rangle \underline{::} ms = ks\ \underline{\mathbf{in}} \\
&\qquad\qquad [\![N]\!] \overline{@} \uparrow(\langle ks'\ \underline{\mathbf{concat}}\ ks'', \ ms \rangle) \\
H_f : \quad [\![p\,r \mapsto N]\!] \quad &= \quad \underline{\lambda}p\ s\ ks.\ \underline{\mathbf{let}}\ r = \mathbf{fun}\ s\ \underline{\mathbf{in}} \\
&\qquad\qquad [\![N_\ell]\!] \overline{@} \uparrow ks \\
H_x : \quad [\![e \mapsto N]\!] \quad &= \quad \underline{\lambda}e\ ks.\ [\![N]\!] \overline{@} \uparrow ks
\end{aligned}
$$

$$[\![\mathbf{raise}\ V]\!] = \overline{\lambda}\langle \kappa \overline{::} \chi \overline{::} \kappa s \rangle \overline{::} \mu s.\ \chi \underline{@} [\![V]\!] \underline{@} (\langle \downarrow \kappa s, \eta \rangle \underline{::} \mu s)$$

$$[\![\mathbf{try}\ M\ \mathbf{with}\ H]\!] = \overline{\lambda}\langle \kappa s, \eta \rangle \overline{::} \mu s.\ [\![M]\!] \overline{@} (\langle K_{\mathrm{ret}} \overline{::} [\![H]\!] \overline{::} \kappa s, \ \eta \rangle \overline{::} \mu s)$$

where

$$K_{\mathrm{ret}} = \underline{\lambda}x\ ks.\ \underline{\mathbf{let}}\ \langle kx \underline{::} k \underline{::} ks', \ \eta \rangle \underline{::} ms = ks\ \underline{\mathbf{in}}$$
$$k \underline{@} x \underline{@} (\langle ks', \eta \rangle \underline{::} ms)$$

$$[\![e \mapsto N]\!] = \underline{\lambda}e\ ks.\ [\![N]\!] \overline{@} \uparrow ks$$

Note: In the setting of js_of_ocaml, K_{ret} is not strictly necessary, as the source language has explicit POPTRAP statements: we could either push a dummy continuation and have POPTRAP pop it (but we need to push something for the stack to look as expected to the rest

of the translation), or push K_{ret} and have `POPTRAP` simply return through it. Currently, I'm doing both, but it is superfluous.

Top level program

$$\top[\![M]\!] = [\![M]\!] \; \overline{@} \; (\langle (\underline{\lambda}x \; ks. \; x) \; \overline{::} \; (\underline{\lambda}z \; ks. \; \textbf{absurd} \; z) \; \overline{::} \; \uparrow[]\rangle, \; (\underline{\lambda}z \; ks. \; \textbf{absurd} \; z) \; \overline{::} \; \uparrow[])$$