

$$\begin{array}{ll}
x & \in \text{Variables} \\
eff & \in \text{Effects} \\
\\
v & ::= x \mid h \mid \lambda x. e \mid eff \\
e & ::= v \mid v \ v \mid \text{let } x = e \text{ in } e \\
& \quad \mid \text{inst } () \mid \text{with } v \text{ handle } e \\
& \quad \mid \text{perform } e \ e \\
h & ::= \text{handler } v \ (\text{val } x \rightarrow e) \ ((x, k) \rightarrow e) \\
\\
F & ::= \Box v \mid \text{let } x = \Box \text{ in } e \\
& \quad \mid \text{with } v \text{ handle } \Box \mid \text{perform } \Box \ e \mid \text{perform } v \Box \\
s & ::= \Box \mid F :: s \mid s @ s
\end{array}$$

Figure 1: the syntax of λ_{eff}

$$\begin{aligned}
flatfn(\Box) &= \lambda x. x \\
flatfn(F :: s) &= \lambda x. (flatfn(s)) \ F[x]
\end{aligned}$$

Figure 2: utils for the semantics

$$\begin{aligned}
& \langle e; s; es \rangle \mapsto \langle e'; s'; es' \rangle \\
& \langle F[e]; s; es \rangle \mapsto \langle e; F :: s; es \rangle \quad (\text{PUSH}) \\
& \langle v; F :: s; es \rangle \mapsto \langle F[v]; s; es \rangle \quad (\text{POP}) \\
& \langle v; []; es \rangle \mapsto \langle v; []; es \rangle \quad (\text{RESULT}) \\
& \langle \lambda x.e; (\Box v) :: s; es \rangle \mapsto \langle e[x = v]; s; es \rangle \quad (\text{APPLY}) \\
& \langle \text{inst } (); s; es \rangle \mapsto \langle \text{eff}; s; es \rangle \quad (\text{INSTANCIATE}) \\
& \langle \text{perform eff } v; F :: s; es \rangle \mapsto \langle \text{perform eff } v; s; F :: es \rangle \quad (\text{RETHROW}) \\
& \left\langle \begin{array}{l} \text{perform eff } v; \\ (\text{with } h \text{ handle } \Box) :: s; \\ \text{es} \end{array} \right\rangle \mapsto \langle e_{\text{eff}}[x = v, k = \text{flatfn}(es)]; s; [] \rangle \quad (\text{HANDLE}) \\
& \text{where } h = \text{handler eff } (\text{val } x \rightarrow e_v) ((x, k) \rightarrow e_{\text{eff}}) \\
& \langle \text{perform eff } v; []; es \rangle \mapsto \text{abort} \quad (\text{LEAK})
\end{aligned}$$

Figure 3: the semantics of λ_{eff}