1 Grammar

$$t ::= \text{int} | t \to t | t \star t \tag{1}$$

$$e^1 ::= i | x | \lambda x.e | \text{let } x = e_1 \text{ in } e_2 | e_1 e_2 | \sim (e^0) \tag{2}$$

$$e^0 ::= i | x | \lambda x.e | \text{let } x = e_1 \text{ in } e_2 | e_1 e_2 | \text{let rec } x = e_1 \text{ in } e_2 | \langle e^1 \rangle \tag{3}$$

2 Typing Judgements

$$\frac{}{\Gamma \overset{n}{\vdash} i\text{: int}}(INT) \qquad \frac{\Gamma^n(x)=t}{\Gamma^n \overset{n}{\vdash} x{:}t}(VAR) \qquad \frac{\Gamma^0(x)=\{t^\star ::= \text{int } | t^\star \star t^\star\}}{\Gamma^1 \overset{1}{\vdash} x{:}t}(CSP)$$

$$\frac{\Gamma^n(x)=t_0 \qquad \Gamma^n, x{:}t_0 \overset{n}{\vdash} e{:}t_1}{\Gamma^n \overset{n}{\vdash} \lambda x.e{:}t_0 \to t_1}(LAM) \qquad \frac{\Gamma^n \overset{n}{\vdash} e_0{:}t_1 \to t_2 \qquad \Gamma^n \overset{n}{\vdash} e_1{:}t_1}{\Gamma^n \overset{n}{\vdash} (e_0 e_1){:}t_2}(APP)$$

$$\frac{\Gamma^n \overset{n}{\vdash} e_0{:}t_0 \qquad \Gamma^n; x{:}t_0 \overset{n}{\vdash} e_1{:}t_1}{\Gamma^n \overset{n}{\vdash} (\text{ let } x=e_0 \text{ in } e_1){:}t_1}(LET) \quad \frac{\Gamma^0(f)=t_0 \to t_1; x{:}t_0 \overset{0\,0}{\vdash}\ e_0{:}t_1 \qquad \Gamma^0, f{:}(t_1 \to t_2) \overset{0}{\vdash} e_1{:}t_2}{\Gamma \overset{0}{\vdash} \text{ let rec } f(x)=e_0 \text{ in } e_1{:}t_2}(LETREC)$$

$$\frac{\Gamma^1 \overset{1}{\vdash} e{:}t}{\Gamma^0 \overset{0}{\vdash} \langle e \rangle{:}\langle t \rangle}(BRAC) \quad \frac{\Gamma^0 \overset{0}{\vdash} e{:}\langle t \rangle}{\Gamma^1 \overset{1}{\vdash} \sim e{:}t}(ESC)$$

3 Notes

- LETREC is only valid in stage zero, which prevents the user from constructing a recursive function within a set of braces
- CSP is the only way to bring values from stage zero into stage one. CSP cannot be applied to values whose type involves a lambda abstraction (ie functions which could be recursive). Although an expression whose type does not involve arrows can contain a recursive function, that expression will be fully evaluated in stage zero and will persist across as a ground value; this works for the same reason that `lift` is sometimes necessary. I need help proving this.
- I left out the typing rules for projection and tupling since they're standard.