Let (encoding)

Encoding - embedding a language feature in a language that doesn't have that feature.

Define Fbb to have the same grammar as Fb but without Let.

Want to define a function f to transform Fb ASTs into Fbb ASTs.

$$f(\text{True}) = \text{True}$$
$$f(e_1 + e_2) = f(e_1) + f(e_2)$$
$$f(\text{Let } x = e_1 \text{ In } e_2) = (\text{Function } x \to f(e_2))\ (f(e_1))$$

FbP (with pairs)

$v ::= \underbrace{\ldots}_{\text{all Fb values}} \mid (v, v)$

$e ::= \underbrace{\ldots}_{\text{all Fb expressions}} \mid (e, e) \mid \text{Fst } e \mid \text{Snd } e$

$e \Rightarrow_{FbP} v$ $\quad (1+1, 3) \Rightarrow (2, 3)$

Write operational semantics for FbP.

$$\frac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2}{(e_1, e_2) \Rightarrow (v_1, v_2)}$$

$$\frac{e \Rightarrow (v_1, v_2)}{\text{Fst } e \Rightarrow v_1}$$

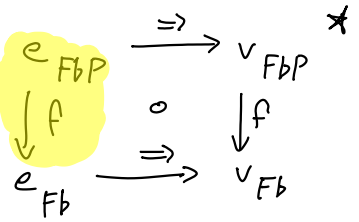
$$\frac{e \Rightarrow (v_1, v_2)}{\text{Snd } e \Rightarrow v_2}$$

Fst (1, 0 0)

Fst ((Function a → (a, a)) 5)

Encode Pairs: write a function that will transform FbP into Fb.

$$f((e_1, e_2)) = \text{Let } p = f(e_1) \text{ In Let } q = f(e_2) \text{ In Function } a \to \text{If } a \text{ Then } p \text{ Else } q$$



$$f(\text{Fst } e) = f(e)\ \text{True}$$
$$f(\text{Snd } e) = f(e)\ \text{False}$$

Fst (1, 0 0) $\not\Rightarrow_{FbP}$

(Function a → If a Then 1 Else 0 0) True $\Rightarrow_{Fb}$ 1

FbL

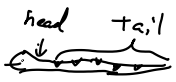
Fb with lists

Fb values `[]` `::`

$v ::= \ldots \mid Nil \mid Cons\ v\ v$

$e ::= \ldots \mid IsEmpty\ e \mid Hd\ e \mid Tl\ e \mid Cons\ e\ e$

Fb expressions



$$
\begin{aligned}
f(Nil) &= (True, 0) \\
f(Cons\ e_1\ e_2) &= (False, (f(e_1), f(e_2))) \\
f(IsEmpty\ e) &= Fst\ f(e) \\
f(Hd\ e) &= Fst(Snd\ f(e)) \\
f(Tl\ e) &= Snd(Snd\ f(e))
\end{aligned}
$$

Cons 3 (Cons 4 (Cons 5 Nil))