$$\frac{(x : v) \in \rho}{\rho \vdash x \Downarrow v} \quad (\textsc{var})$$

$$\frac{\lambda \notin dom(\rho)}{\rho \vdash \lambda x.e \Downarrow < \lambda x.e \ \mathbf{in}\ \rho >} \quad (\textsc{abs})$$

$$\frac{\begin{array}{c}\rho \vdash e_1 \Downarrow < \lambda x.e \ \mathbf{in}\ \rho_1 > \\ \rho \vdash e_2 \Downarrow v_2 \\ \rho_1, (x : v_2) \vdash e \Downarrow v\end{array}}{\rho \vdash (e_1\ e_2) \Downarrow v} \quad (\textsc{app})$$

Figure 1: Environment-passing Interpreter

$$\frac{\textsf{quote} \notin dom(\rho)}{\rho \vdash (\textsf{quote}\ d) \Downarrow d} \quad \text{(QUOTE)}$$

$$\frac{\begin{array}{c}\rho \vdash e^* \Downarrow v^* \\ \ldots \textsf{list} \notin dom(\rho)\end{array}}{\rho \vdash (\textsf{list}\ e^*\ \ldots) \Downarrow v^*\ \ldots} \quad \text{(LIST)}$$

$$\frac{(x : v) \in \rho}{\rho \vdash x \Downarrow v} \quad \text{(VAR)}$$

$$\frac{\lambda \notin dom(\rho)}{\rho \vdash \lambda x.e \Downarrow < \lambda x.e\ \mathbf{in}\ \rho >} \quad \text{(ABS)}$$

$$\frac{\begin{array}{c}\rho \vdash e_1 \Downarrow < \lambda x.e\ \mathbf{in}\ \rho_1 > \\ \rho \vdash e_2 \Downarrow v_2 \\ \rho_1, (x : v_2) \vdash e \Downarrow v\end{array}}{\rho \vdash (e_1\ e_2) \Downarrow v} \quad \text{(APP)}$$

Figure 2: Environment-passing Interpreter (shadowing allowed)

0.1 A Challenge from McCarthy

McCarthy (1978) posed this problem in his description of *value*, a minimal LISP interpreter:

> Difficult mathematical type exercise: Find a list e such that $value\ e = e$.

$$\mathsf{I}x \rhd x \qquad (\rhd\text{-}\mathsf{I})$$

$$\mathsf{K}xy \rhd x \qquad (\rhd\text{-}\mathsf{K})$$

$$\mathsf{S}xyz \rhd xz(yz) \qquad (\rhd\text{-}\mathsf{S})$$

Figure 3: Contraction

$$\frac{M \rhd M'}{M \rhd_{1w} M'} \; (\rhd_{1w}\text{-}\textsc{contraction})$$

$$\frac{M \rhd_{1w} M'}{MN \rhd_{1w} M'N} \qquad (\rhd_{1w}\text{-}\textsc{left})$$

$$\frac{N \rhd_{1w} N'}{MN \rhd_{1w} MN'} \qquad (\rhd_{1w}\text{-}\textsc{right})$$

Figure 4: One-step Reduction

$$M \rhd_w M \qquad (\rhd_w\text{-}\textsc{reflexive})$$

$$\frac{M \rhd_{1w} N \quad N \rhd_w P}{M \rhd_w P} \; (\rhd_w\text{-}\textsc{transitive})$$

Figure 5: Weak Reduction

$$\mathsf{I}\ L_\eta\ \lambda x.x \qquad (L_\eta\text{-}\mathsf{I})$$

$$\mathsf{K}\ L_\eta\ \lambda xy.x \qquad (L_\eta\text{-}\mathsf{K})$$

$$\mathsf{S}\ L_\eta\ \lambda xyzw.(\lambda v.xzv\ \lambda v.yzv)w \quad (L_\eta\text{-}\mathsf{S})$$

$$\frac{M\ L_\eta\ M' \quad N\ L_\eta\ N'}{MN\ L_\eta\ M'N'} \; (L_\eta\text{-}\textsc{compound})$$

Figure 6: **SKI**-to-Call-by-Value λ-Calculus

0.2 (Some) Homework for Free

Let us consider an exercise from a standard textbook on combinatory logic (Hindley and Seldin 2008). This exercise[1], which is marked "Tricky" in the text, asks the reader to construct combinatory logic terms $\mathsf{B'}$ and W that satisfy $\mathsf{B'}xyz \triangleright_w y(xz)$ and $\mathsf{W}xy \triangleright_w xyy$.

[1]Exercise 2.17b on p. 26.

Barendregt (1984) gives the following definition for a fixpoint combinator, F:

$$\exists F. \forall X. FX = X(FX)$$

$$\frac{(x : T) \in \Gamma}{\Gamma \vdash x : T} \qquad (\textsc{var})$$

$$\frac{\Gamma, (x : T_1) \vdash e : T_2}{\Gamma \vdash \lambda x.e : T_1 \rightarrow T_2} \qquad (\textsc{abs})$$

$$\frac{\begin{array}{c}\Gamma \vdash e_1 : T_1 \rightarrow T \\ \Gamma \vdash e_2 : T_1\end{array}}{\Gamma \vdash (e_1\ e_2) : T} \qquad (\textsc{app})$$

Figure 7: Simply-typed λ-calculus

$$\frac{(x : v) \in \rho}{\rho \vdash x \Downarrow v} \quad (\textsc{var})$$

$$\frac{\lambda \notin dom(\rho)}{\rho \vdash \lambda x.e \Downarrow < \lambda x.e \textbf{ in } \rho >} \quad (\textsc{abs})$$

$$\frac{\rho \vdash e_1 \Downarrow < \lambda x.e \textbf{ in } \rho_1 > \quad \rho \vdash e_2 \Downarrow v_2 \quad \rho_1, (x : v_2) \vdash e \Downarrow v}{\rho \vdash (e_1\ e_2) \Downarrow v} \quad (\textsc{app})$$

Figure 8: Environment-passing Interpreter

```
(define (eval-expᵒ ρ expr v)
  (matchᵉ expr
    (,x (symbolᵒ x)
     (lookupᵒ `(,x : ,v) ρ))
    ((λ (,x) ,e)
     (≡ `((λ (,x) ,e) in ,ρ) v))
    ((,e₁ ,e₂)
     (fresh (x e ρ₁ v₂)
       (eval-expᵒ ρ e₁ `((λ (,x) ,e) in ,ρ₁))
       (eval-expᵒ ρ e₂ v₂)
       (eval-expᵒ `((,x : ,v₂) . ,ρ₁) e v)))))
```

Figure 9: Environment-passing Interpreter

```
(define (evalᵒ expr v) (eval-expᵒ '() expr v))

(define (lookupᵒ binding ρ)
  (matchᵉ (binding ρ)
    (((,x : ,v) ((,x : ,v) . ,_)))
    (((,x : ,v) ((,y : ,_) . ,ρ₁))
     (≢ x y) (lookupᵒ binding ρ₁))))
```

Figure 10: Interpreter Helper Relations (miniKanren)

$$\frac{\textsf{quote} \notin dom(\rho)}{\rho \vdash (\textsf{quote}\ d) \Downarrow d} \quad \text{(QUOTE)}$$

$$\frac{\rho \vdash e^* \Downarrow v^* \quad \ldots \textsf{list} \notin dom(\rho)}{\rho \vdash (\textsf{list}\ e^* \ \ldots) \Downarrow v^* \ \ldots} \quad \text{(LIST)}$$

$$\frac{(x : v) \in \rho}{\rho \vdash x \Downarrow v} \quad \text{(VAR)}$$

$$\frac{\lambda \notin dom(\rho)}{\rho \vdash \lambda x.e \Downarrow < \lambda x.e \ \mathbf{in}\ \rho >} \quad \text{(ABS)}$$

$$\frac{\rho \vdash e_1 \Downarrow < \lambda x.e \ \mathbf{in}\ \rho_1 > \quad \rho \vdash e_2 \Downarrow v_2 \quad \rho_1, (x : v_2) \vdash e \Downarrow v}{\rho \vdash (e_1\ e_2) \Downarrow v} \quad \text{(APP)}$$

Figure 11: Environment-passing Interpreter (shadowing allowed)

```
(define (eval-expᵒ ρ expr v)
  (fresh ()
    (absentᵒ 'in expr)
    (matchᵉ expr
      ((quote ,datum)
       (not-in-domᵒ 'quote ρ) (≡ datum v))
      ((list . ,e*)
       (not-in-domᵒ 'list ρ) (eval-listᵒ ρ e* v))
      (,x (symbolᵒ x) (lookupᵒ `(,x : ,v) ρ))
      ((λ (,x) ,e)
       (not-in-domᵒ 'λ ρ) (≡ `((λ (,x) ,e) in ,ρ) v))
      ((,e₁ ,e₂)
       (fresh (x e ρ₁ v₂)
         (eval-expᵒ ρ e₁ `((λ (,x) ,e) in ,ρ₁))
         (eval-expᵒ ρ e₂ v₂)
         (eval-expᵒ `((,x : ,v₂) . ,ρ₁) e v))))))
```

Figure 12: Environment-passing Interpreter (shadowing allowed, miniKanren)

```
(define (evalᵒ expr v) (eval-expᵒ '() expr v))

(define (eval-listᵒ ρ expr v)
  (matchᵉ (expr v)
    ((() ()))
    (((,e . ,e*) (,ve . ,ve*))
     (eval-expᵒ ρ e ve) (eval-listᵒ ρ e* ve*))))

(define (lookupᵒ binding ρ)
  (matchᵉ (binding ρ)
    (((,x : ,v) ((,x : ,v) . ,_)))
    (((,x : ,v) ((,y : ,_) . ,ρ₁))
     (≢ x y) (lookupᵒ binding ρ₁))))

(define (not-in-domᵒ x ρ)
  (matchᵉ ρ
    (())
    (((,y . ,v) . ,ρ₁) (≢ y x) (not-in-domᵒ x ρ₁))))
```

Figure 13: Interpreter Helper Relations (miniKanren)

$$\mathsf{I}x \rhd x \qquad (\rhd\text{-}\mathsf{I})$$

$$\mathsf{K}xy \rhd x \qquad (\rhd\text{-}\mathsf{K})$$

$$\mathsf{S}xyz \rhd xz(yz) \qquad (\rhd\text{-}\mathsf{S})$$

Figure 14: Contraction

$$
\begin{array}{l}
(\textbf{defmatch}^e\ (\rhd^o\ T\ \hat{T})\\
\quad (((\mathsf{I}\ ,x)\ ,x))\\
\quad ((((\mathsf{K}\ ,x)\ ,y)\ ,x))\\
\quad (((((\mathsf{S}\ ,x)\ ,y)\ ,z)\ ((,x\ ,z)\ (,y\ ,z)))))
\end{array}
$$

Figure 18: Contraction (miniKanren)

$$\frac{M \rhd M'}{M \rhd_{1w} M'}\ (\rhd_{1w}\text{-CONTRACTION})$$

$$\frac{M \rhd_{1w} M'}{MN \rhd_{1w} M'N}\ (\rhd_{1w}\text{-LEFT})$$

$$\frac{N \rhd_{1w} N'}{MN \rhd_{1w} MN'}\ (\rhd_{1w}\text{-RIGHT})$$

Figure 15: One-step Reduction

$$
\begin{array}{l}
(\textbf{defmatch}^e\ (\rhd^o_{1w}\ T\ \hat{T})\\
\quad ((,M\ ,\hat{M})\ (\rhd^o\ M\ \hat{M}))\\
\quad (((,M\ ,N)\ (,\hat{M}\ ,N))\ (\rhd^o_{1w}\ M\ \hat{M}))\\
\quad (((,M\ ,N)\ (,M\ ,\hat{N}))\ (\rhd^o_{1w}\ N\ \hat{N})))
\end{array}
$$

Figure 19: One-step Reduction (miniKanren)

$$M \rhd_w M \qquad (\rhd_w\text{-REFLEXIVE})$$

$$\frac{M \rhd_{1w} N \quad N \rhd_w P}{M \rhd_w P}\ (\rhd_w\text{-TRANSITIVE})$$

Figure 16: Weak Reduction

$$
\begin{array}{l}
(\textbf{defmatch}^e\ (\rhd^o_w\ T\ \hat{T})\\
\quad ((,M\ ,M))\\
\quad ((,M\ ,P)\ (\textbf{fresh}\ (N)\ (\rhd^o_{1w}\ M\ N)\ (\rhd^o_w\ N\ P))))
\end{array}
$$

Figure 20: Weak Reduction (miniKanren)

$$\mathsf{I}\ L_\eta\ \lambda x.x \qquad (L_\eta\text{-}\mathsf{I})$$

$$\mathsf{K}\ L_\eta\ \lambda xy.x \qquad (L_\eta\text{-}\mathsf{K})$$

$$\mathsf{S}\ L_\eta\ \lambda xyzw.(\lambda v.xzv\ \lambda v.yzv)w \qquad (L_\eta\text{-}\mathsf{S})$$

$$\frac{M\ L_\eta\ M' \quad N\ L_\eta\ N'}{MN\ L_\eta\ M'N'}\ (L_\eta\text{-COMPOUND})$$

Figure 17: **SKI**-to-Call-by-Value λ-Calculus

$$
\begin{array}{l}
(\textbf{defmatch}^e\ (L^o_\eta\ T\ \hat{T})\\
\quad ((\mathsf{I}\ (\lambda\ (x)\ x)))\\
\quad ((\mathsf{K}\ (\lambda\ (x)\ (\lambda\ (y)\ x))))\\
\quad ((\mathsf{S}\ (\lambda\ (x)\\
\qquad\quad (\lambda\ (y)\\
\qquad\qquad (\lambda\ (z)\\
\qquad\qquad\quad (\lambda\ (w)\\
\qquad\qquad\qquad (((\lambda\ (v)\ ((x\ z)\ v))\\
\qquad\qquad\qquad\quad (\lambda\ (v)\ ((y\ z)\ v)))\\
\qquad\qquad\qquad\ \ w)))))))\\
\quad (((,M\ ,N)\ (,\hat{M}\ ,\hat{N}))\ (L^o_\eta\ M\ \hat{M})\ (L^o_\eta\ N\ \hat{N})))
\end{array}
$$

Figure 21: **SKI**-to-CBV λ-Calculus (miniKanren)

$$\frac{(x:T) \in \Gamma}{\Gamma \vdash x : T} \quad (\textsc{var})$$

$$\frac{\Gamma, (x:T_1) \vdash e : T_2}{\Gamma \vdash \lambda x.e : T_1 \to T_2} \quad (\textsc{abs})$$

$$\frac{\Gamma \vdash e_1 : T_1 \to T \qquad \Gamma \vdash e_2 : T_1}{\Gamma \vdash (e_1\ e_2) : T} \quad (\textsc{app})$$

Figure 22: Simply-typed λ-calculus

```
(define (⊢ᵒ Γ expr type)
  (matchᵉ (expr type)
    ((,x ,T) (symbolᵒ x) (lookupᵒ `(,x : ,T) Γ))
    (((λ (,x) ,e) (,T₁ → ,T₂))
     (⊢ᵒ `((,x : ,T₁) . ,Γ) e T₂))
    (((,e₁ ,e₂) ,T)
     (fresh (T₁)
       (⊢ᵒ Γ e₁ `(,T₁ → ,T))
       (⊢ᵒ Γ e₂ T₁)))))
```

Figure 23: Simply-typed λ-calculus

$$
\begin{array}{l}
(\textbf{let}\ ((F_V\ (eval\ (car\ (\textbf{run}\ 1\ (F_V) \\
\qquad\qquad\qquad\qquad\qquad\qquad (\textbf{fresh}\ (F) \\
\qquad\qquad\qquad\qquad\qquad\qquad\quad (\textbf{eigen}\ (X) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad (\triangleright_w^o\ `(,F\ ,X)\ `(,X\ (,F\ ,X))) \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad (L_\eta^o\ F\ F_V))))) \\
\qquad\qquad\qquad\qquad (environment\ '(\mathsf{rnrs})))))) \\
\quad ((F_V\ (\lambda\ (f) \\
\qquad\qquad (\lambda\ (n) \\
\qquad\qquad\quad (\textbf{if}\ (=\ n\ \mathsf{0}) \\
\qquad\qquad\qquad \mathsf{1} \\
\qquad\qquad\qquad (*\ n\ (f\ (-\ n\ \mathsf{1}))))))) \\
\quad\ \mathsf{5})) \Rightarrow \mathsf{120}
\end{array}
$$

References

Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.

J. Roger Hindley and Jonathan P. Seldin. *Lambda-Calculus and Combinators: An Introduction*. Cambridge University Press, New York, NY, USA, second edition, 2008.

John McCarthy. A micro-manual for lisp - not the whole truth. *SIGPLAN Not.*, 13(8):215–216, August 1978.