Linguagem L0

Sintaxe:

Termos:

$$e \ \in \mathsf{L0}$$
$$e ::= \mathsf{true} \mid \mathsf{false} \mid \mathsf{0} \mid \mathsf{succ}\ e \mid \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \mid \mathsf{iszero}\ e \mid \mathsf{pred}\ e$$

Valores:

$$v \ \in \ \mathsf{Values}$$
$$v ::= \ \mathsf{true} \mid \mathsf{false} \mid nv$$
$$nv \ \in \ \mathsf{NV}$$
$$nv ::= \ \mathsf{0} \mid \mathsf{succ}\ nv$$

Tipos:

$$T \ \in \ \mathsf{Types}$$
$$T ::= \mathsf{nat} \mid \mathsf{bool}$$

Semântica operacional small-step:

A relação $\longrightarrow \subseteq (\mathsf{L0} \times \mathsf{L0})$ é a menor relação satisfazendo as seguintes regras:

$$\frac{}{\mathsf{if\ true\ then}\ e_2\ \mathsf{else}\ e_3 \longrightarrow e_2} \quad (\text{E-IfTrue})$$

$$\frac{}{\mathsf{if\ false\ then}\ e_2\ \mathsf{else}\ e_3 \longrightarrow e_3} \quad (\text{E-IfFalse})$$

$$\frac{e \longrightarrow e'}{\mathsf{if}\ e\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \longrightarrow \mathsf{if}\ e'\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3} \quad (\text{E-IF})$$

$$\frac{e \longrightarrow e'}{\mathsf{succ}\ e \longrightarrow \mathsf{succ}\ e'} \quad (\text{E-Succ})$$

$$\frac{}{\mathsf{pred}\ 0 \longrightarrow 0} \quad (\text{E-PredZero})$$

$$\frac{}{\mathsf{pred}\ (\mathsf{succ}\ nv) \longrightarrow nv} \quad (\text{E-PredSucc})$$

$$\frac{e \longrightarrow e'}{\mathsf{pred}\ e \longrightarrow \mathsf{pred}\ e'} \quad (\text{E-Pred})$$

$$\frac{}{\mathsf{iszero}\ 0 \longrightarrow \mathsf{true}} \quad (\text{E-IsZeroZero})$$

$$\frac{}{\mathsf{iszero}\ (\mathsf{succ}\ nv) \longrightarrow \mathsf{false}} \quad (\text{E-IsZeroSucc})$$

$$\frac{e \longrightarrow e'}{\mathsf{iszero}\ e \longrightarrow \mathsf{iszero}\ e'} \quad (\text{E-IsZero})$$

A relação $\longrightarrow^* \subseteq (\mathsf{L0} \times \mathsf{L0})$ é o fecho transitivo e reflexivo de $\longrightarrow$.

Semântica operacional big-step:

A relação $\Downarrow \subseteq (\mathsf{L0} \times \mathsf{Values})$ é a menor relação satisfazendo as seguintes regras:

$$\frac{}{v \Downarrow v} \quad (\text{B-Values})$$

$$\frac{e_1 \Downarrow \mathsf{true} \qquad e_2 \Downarrow v_2}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \Downarrow v_2} \quad (\text{B-IfTrue})$$

$$\frac{e_1 \Downarrow \mathsf{false} \qquad e_3 \Downarrow v_3}{\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 \Downarrow v_3} \quad (\text{B-IfFalse})$$

$$\frac{e \Downarrow nv}{\mathsf{succ}\ e \Downarrow \mathsf{succ}\ nv} \quad (\text{B-Succ})$$

$$\frac{e \Downarrow 0}{\mathsf{pred}\ e \Downarrow 0} \quad (\text{B-PredZero})$$

$$\frac{e \Downarrow \mathsf{succ}\ nv}{\mathsf{pred}\ e \Downarrow nv} \quad (\text{B-PredSucc})$$

$$\frac{e \Downarrow 0}{\mathsf{iszero}\ e \Downarrow \mathsf{true}} \quad (\text{B-IsZeroZero})$$

$$\frac{e \Downarrow \mathsf{succ\ nv}}{\mathsf{iszero}\ e \Downarrow \mathsf{false}} \quad (\text{B-IsZeroSucc})$$

Sistema de Tipos:

A relação $(\vdash \_ : \_) \subseteq (\mathsf{L0} \times \mathsf{Types})$ é a menor relação satisfazendo as seguintes regras:

$$\frac{}{\vdash \mathsf{0} : \mathsf{nat}} \quad (\text{T-Zero}) \qquad \frac{\vdash e_1 : \mathsf{bool} \quad \vdash e_2 : T \quad \vdash e_3 : T}{\vdash \mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3 : T} \quad (\text{T-If})$$

$$\frac{\vdash e : \mathsf{nat}}{\vdash \mathsf{succ}\ e : \mathsf{nat}} \quad (\text{T-Succ}) \qquad \frac{\vdash e : \mathsf{nat}}{\vdash \mathsf{iszero}\ e : \mathsf{bool}} \quad (\text{T-IsZero})$$

$$\frac{}{\vdash \mathsf{true} : \mathsf{bool}} \quad (\text{T-True})$$

$$\frac{}{\vdash \mathsf{false} : \mathsf{bool}} \quad (\text{T-False}) \qquad \frac{\vdash e : \mathsf{nat}}{\vdash \mathsf{pred}\ e : \mathsf{nat}} \quad (\text{T-Pred})$$

Algoritmo de inferência de tipos (Racket/Advanced Student)

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;             Sintaxe                    ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; um termo é
;;  'true,  ou
;;  'false, ou
;;  'zero,  ou
;;  (list 'succ   t),      onde t:termo, ou
;;  (list 'pred   t),      onde t:termo, ou
;;  (list 'iszero t),      onde t:termo, ou
;;  (list 'if t1 t2 t3),   onde t1,t2,t3:termo

;; um tipo é
;;   'bool ou 'nat

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;            Exemplos                    ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(define ex1
  (list 'if
        (list 'iszero
          (list 'pred
             (list 'succ 'zero)))
        'true
        'false))

(define ex2
  (list 'iszero
        (list 'if 'true 'false 'zero)))


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;          Inferência                   ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; typeInfer : termo -> tipo ou false

;; (typeInfer t) devolve o tipo associado
;; ao termo t pelo sistema de tipos de L0,
;; ou false se não houver esse tipo

(define (typeInfer t)
 (cond

  [(equal? t 'true)  'bool]

  [(equal? t 'false) 'bool]

  [(equal? t 'zero)   'nat]

  [(equal? (first t) 'succ)
    (cond
    [(equal? 'nat (typeInfer (second t))) 'nat]
    [else                              false])]

  [(equal? (first t) 'pred)
    (cond
    [(equal? 'nat (typeInfer (second t))) 'nat]
    [else                              false])]

  [(equal? (first t) 'iszero)
    (cond
    [(equal? 'nat (typeInfer (second t))) 'bool]
    [else                               false])]

  [(equal? (first t) 'if)
    (local
     (
     (define tp1 (typeInfer (second t)))
     (define tp2 (typeInfer (third  t)))
     (define tp3 (typeInfer (fourth t)))
     )
    (cond
    [(and (equal? 'bool tp1)
          (equal? tp2   tp3))        tp2]
    [else                        false]))]))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;             Testes               ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(check-expect (typeInfer ex1) 'bool)
(check-expect (typeInfer ex2) false)
```

Definições

- $\mathsf{FN}(e) = \neg\exists e'.\ (e \longrightarrow e')$
- $\mathsf{PossuiFN}(e) = \exists e'.\ (e \longrightarrow^* e' \wedge \mathsf{FN}(e'))$
- $\mathsf{Erro}(e) = \mathsf{FN}(e)\ \wedge\ e \notin \mathsf{Values}$
- $\mathsf{Leva\text{-}a\text{-}Erro}(e) = \exists e'.\ (e \longrightarrow^* e' \wedge \mathsf{Erro}(e'))$
- $\mathsf{Diverge}(e) = \forall e'.$ (se $e \longrightarrow^* e'$ então $\exists e''.(e' \longrightarrow e''))$.

Propriedades de linguagens

- valores não progridem: se $v \in \mathsf{Values}$ então $\mathsf{FN}(v)$
- normalização (fraca): se $e \in \mathsf{L0}$ então $\mathsf{PossuiFN}(e)$
- normalização (forte): se $e \in \mathsf{L0}$ então $\neg\mathsf{Diverge}(e)$
- determinismo: se $e \longrightarrow e_1$ e $e \longrightarrow e_2$ então $e_1 = e_2$
- compatibilidade entre semânticas: $e \longrightarrow^* v$ se, e somente se, $e \Downarrow v$
- unicidade de tipos: se $\vdash e : T_1$ e $\vdash e : T_2$, então $T_1 = T_2$
- preservação de tipos: se $e \longrightarrow e'$ então, para todo T, se $\vdash e : T$ então $\vdash e' : T$.
- progresso: se $\vdash e : T$, então $e \in \mathsf{Values}$ ou $\exists e'.e \longrightarrow e'$
- segurança: se $\vdash e : T$ então $\neg\mathsf{Leva\text{-}a\text{-}Erro}(e)$.

Propriedades de algoritmos (`typeInfer` como exemplo)

- **consistência**: se `(typeInfer` e`) =` T então $\vdash e : T$
- **completude**: se $\vdash e : T$ então `(typeInfer` e`) =` T

Simple Stack Machine

Sintaxe:

$$
\begin{array}{lcl}
n & \in & \mathbb{N} \\
z & \in & \mathbb{Z} \\
b & \in & \{\mathsf{true}, \mathsf{false}\} \\
i & \in & \mathsf{Inst} \\
i & ::= & \mathsf{INT}\ z \mid \mathsf{BOOL}\ b \mid \mathsf{POP} \mid \mathsf{COPY} \\
 & & \mathsf{INC} \mid \mathsf{DEC} \mid \mathsf{ADD} \mid \mathsf{INV} \mid \mathsf{EQ} \mid^{\mathsf{GT}} \\
 & & \mathsf{JUMP}\ n \mid \mathsf{JMPIFZERO}\ n \\
 & & \\
 & & \mathsf{VAR}\ x \\
 & & \mathsf{CLOS}(env, x, \bar{i}) \\
 & & \mathsf{APPLY} \\
\bar{i} & ::= & [] \mid i :: \bar{i}
\end{array}
$$

$$
\begin{array}{lcl}
code & \in & \mathsf{Code} \\
code & ::= & [] \mid i :: code \\
stack & \in & \mathsf{Stack} \\
stack & ::= & [] \mid z :: stack \\
\mathsf{State} & = & \mathsf{Code} \times \mathsf{Stack} \\
 & & \\
sv & \in & \mathsf{SValues} \\
sv & ::= & ([], z :: [])
\end{array}
$$

Semântica operacional small-step:

A relação $\rhd \subseteq \mathsf{State} \times \mathsf{State}$ é a menor relação tal que as seguinte regras valem:

$$\frac{}{(\mathsf{PUSH}\ z :: code,\ stack) \rhd (code,\ z :: stack)}$$

$$\frac{}{(\mathsf{POP} :: code,\ z :: stack) \rhd (code,\ stack)}$$

$$\frac{}{(\mathsf{COPY} :: code,\ z :: stack) \rhd (code,\ z :: z :: stack)}$$

$$\frac{}{(\mathsf{INC} :: code,\ z :: stack) \rhd (code,\ (z+1) :: stack)}$$

$$\frac{}{(\mathsf{DEC} :: code,\ z :: stack) \rhd (code,\ (z-1) :: stack)}$$

$$\frac{}{(\mathsf{JUMP}\ n :: i_1 :: \ldots :: i_n :: code,\ stack) \rhd (code,\ stack)}$$

$$\frac{}{(\mathsf{JMPIFZERO}\ n :: i_1 :: \ldots :: i_n :: code,\ 0 :: stack) \rhd (code,\ stack)}$$

$$\frac{z \neq 0}{(\mathsf{JMPIFZERO}\ n :: code,\ z :: stack) \rhd (code,\ stack)}$$

A relação $\rhd^* \subseteq \mathsf{State} \times \mathsf{State}$ é o fecho transitivo e reflexivo de $\rhd$.

Função de compilação L0/SSM

$\mathscr{C} : \mathsf{L0} \to \mathsf{Code}$

$$
\begin{array}{lcl}
\mathscr{C}(\mathsf{true}) & = & [\mathsf{PUSH\ 1}] \\
\mathscr{C}(\mathsf{false}) & = & [\mathsf{PUSH\ 0}] \\
\mathscr{C}(\mathsf{0}) & = & [\mathsf{PUSH\ 0}] \\
\mathscr{C}(\mathsf{succ}\ e_1) & = & \mathscr{C}(e_1) +\!+ [\mathsf{INC}] \\
\mathscr{C}(\mathsf{if}\ e_1\ \mathsf{then}\ e_2\ \mathsf{else}\ e_3) & = & \mathscr{C}(e_1) +\!+ [\mathsf{JMPIFZERO}\ (n_2{+}1)] +\!+ \mathscr{C}(e_2) +\!+ [\mathsf{JUMP}\ n_3] +\!+ \mathscr{C}(e_3) \\
 & & \textit{onde}\ \ n_2 = \textit{length}(\mathscr{C}(e_2))\ \ \textit{e}\ \ n_3 = \textit{length}(\mathscr{C}(e_3)) \\
\mathscr{C}(\mathsf{iszero}\ e_1) & = & \mathscr{C}(e_1) +\!+ [\mathsf{JMPIFZERO\ 2;\ PUSH\ 0;\ JUMP\ 1;\ PUSH\ 1}] \\
\mathscr{C}(\mathsf{pred}\ e_1) & = & \mathscr{C}(e_1) +\!+ [\mathsf{COPY;\ JMPIFZERO\ 1;\ DEC}]
\end{array}
$$

$\rho : \mathsf{Values} \to \mathbb{Z}$

$$
\begin{aligned}
\rho(\mathsf{true}) &= 1 \\
\rho(\mathsf{false}) &= 0 \\
\rho(\mathsf{0}) &= 0 \\
\rho(\mathsf{succ}\ nv) &= 1 + \rho(nv)
\end{aligned}
$$

Propriedades de compilação

- preservação de avaliações sem erros: se $e \longrightarrow^* v$ então $(\mathscr{C}(e), []) \ \rhd^*\ ([], \rho(v) :: [])$
- unicidade de representação: se $\rho(v_1) = \rho(v_2)$ então $v_1 = v_2$
  (inválida para a semântica apresentada)