

Sintaxe abstrata:

$e \in \text{Expr}$
 $e ::= n \mid b \mid e_1 \text{ op } e_2 \mid \text{if } e_1 \text{ then } e_2 \text{ else } e_3$
 $\quad \mid x \mid e_1 e_2 \mid \text{fn } x:T \Rightarrow e \mid \text{let } x:T = e_1 \text{ in } e_2 \mid \text{let rec } f:T_1 \rightarrow T_2 = (\text{fn } x:T_1 \Rightarrow e_1) \text{ in } e_2$
 $\quad \mid e_1 := e_2 \mid !e \mid \text{new } e \mid \text{skip} \mid \text{while } e_1 \text{ do } e_2 \mid e_1; e_2 \mid \boxed{1}$

 $T \in \text{Types}$
 $T ::= \text{int} \mid \text{bool} \mid T_1 \rightarrow T_2 \mid T \text{ ref} \mid \text{unit}$

Semântica operacional *small-step*:

$v \in \text{Values}$
 $v ::= n \mid b \mid \text{fn } x:T \Rightarrow e \mid \boxed{1}$

$$\frac{||[n]|| = ||[n_1+n_2]||}{n_1+n_2, \sigma \longrightarrow n, \sigma} \quad (\text{OP}+) \quad \frac{}{(\text{fn } x:T \Rightarrow e) v, \sigma \longrightarrow \{v/x\}e, \sigma} \quad (\text{E-}\beta)$$

$$\frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{e_1 \text{ op } e_2, \sigma \longrightarrow e'_1 \text{ op } e_2, \sigma'} \quad (\text{OP1}) \quad \frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{e_1 e_2, \sigma \longrightarrow e'_1 e_2, \sigma'} \quad (\text{E-APP1})$$

$$\frac{e_2, \sigma \longrightarrow e'_2, \sigma'}{v \text{ op } e_2, \sigma \longrightarrow v \text{ op } e'_2, \sigma'} \quad (\text{OP2}) \quad \frac{e_2, \sigma \longrightarrow e'_2, \sigma'}{v e_2, \sigma \longrightarrow v e'_2, \sigma'} \quad (\text{E-APP2})$$

$$\text{if true then } e_2 \text{ else } e_3, \sigma \longrightarrow e_2, \sigma \quad (\text{IF1}) \quad \text{if false then } e_2 \text{ else } e_3, \sigma \longrightarrow e_3, \sigma \quad (\text{IF2})$$

$$\frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{\text{if } e_1 \text{ then } e_2 \text{ else } e_3, \sigma \longrightarrow \text{if } e'_1 \text{ then } e_2 \text{ else } e_3, \sigma'} \quad (\text{IF3})$$

$$\frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{\text{let } x:T = e_1 \text{ in } e_2, \sigma \longrightarrow \text{let } x:T = e'_1 \text{ in } e_2, \sigma'} \quad (\text{E-LET1})$$

$$\frac{}{\text{let } x:T = v \text{ in } e_2, \sigma \longrightarrow \{v/x\} e_2, \sigma} \quad (\text{E-LET2})$$

$$\frac{}{\text{let rec } f:T_1 \rightarrow T_2 = \text{fn } y:T_1 \Rightarrow e_1 \text{ in } e_2, \sigma \longrightarrow \{\alpha/f\} e_2, \sigma} \quad (\text{E-LETREC})$$

$$\alpha = \text{fn } y:T_1 \Rightarrow \text{let rec } f:T_1 \rightarrow T_2 = \text{fn } y:T_1 \Rightarrow e_1 \text{ in } e_1$$

$$\frac{l \in \text{Dom}(\sigma)}{l := v, \sigma \longrightarrow \text{skip}, \sigma[l \mapsto v]} \quad (\text{ATR1}) \quad \frac{e, \sigma \longrightarrow e', \sigma'}{l := e, \sigma \longrightarrow l := e', \sigma'} \quad (\text{ATR2}) \quad \frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{e_1 := e_2, \sigma \longrightarrow e'_1 := e_2, \sigma'} \quad (\text{ATR})$$

$$\frac{l \in \text{Dom}(\sigma) \quad \sigma(l) = v}{!l, \sigma \longrightarrow v, \sigma} \quad (\text{DEREF}) \quad \frac{e, \sigma \longrightarrow e', \sigma'}{\text{new } e, \sigma \longrightarrow \text{new } e', \sigma'} \quad (\text{NEW})$$

$$\frac{l \notin \text{Dom}(\sigma)}{\text{new } v, \sigma \longrightarrow l, \sigma[l \mapsto v]} \quad (\text{NEW1})$$

$$\frac{}{\text{skip}; e_2, \sigma \longrightarrow e_2, \sigma}$$

$$\frac{e, \sigma \longrightarrow e', \sigma'}{!e, \sigma \longrightarrow !e', \sigma'} \quad (\text{DEREF1})$$

$$\frac{e_1, \sigma \longrightarrow e'_1, \sigma'}{e_1; e_2, \sigma \longrightarrow e'_1; e_2, \sigma'} \quad (\text{SEQ2})$$

$$\text{while } e_1 \text{ do } e_2, \sigma \longrightarrow \text{if } e_1 \text{ then } (e_2; \text{while } e_1 \text{ do } e_2) \text{ else skip}, \sigma \quad (\text{E-WHILE})$$

Sistema de Tipos:

| | | | |
|---|-----------|---|-----------|
| $\frac{}{\Gamma \vdash n : \text{int}}$ | (T-INT) | $\frac{\Gamma(x) = T}{\Gamma \vdash x : T}$ | (T-VAR) |
| $\frac{}{\Gamma \vdash b : \text{bool}}$ | (T-BOOL) | $\frac{\Gamma, x \mapsto T \vdash e : T'}{\Gamma \vdash (\text{fn } x:T \Rightarrow e) : T \rightarrow T'}$ | (T-FUN) |
| $\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 + e_2 : \text{int}}$ | (T-OP+) | $\frac{\Gamma \vdash e_1 : T \rightarrow T' \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 e_2 : T'}$ | (T-APP) |
| $\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : T \quad \Gamma \vdash e_3 : T}{\Gamma \vdash \text{if } e_1 \text{ then } e_2 \text{ else } e_3 : T}$ | (T-IF) | $\frac{\Gamma \vdash e_1 : T \quad \Gamma, x \mapsto T \vdash e_2 : T'}{\Gamma \vdash \text{let } x:T = e_1 \text{ in } e_2 : T'}$ | (T-LET) |
| $\frac{\Gamma, x \mapsto T_1, f \mapsto (T_1 \rightarrow T_2) \vdash e_1 : T_2 \quad \Gamma, f \mapsto (T_1 \rightarrow T_2) \vdash e_2 : T}{\Gamma \vdash \text{let rec } f:T_1 \rightarrow T_2 = (\text{fn } x:T_1 \Rightarrow e_1) \text{ in } e_2 : T}$ | | | |
| | | | |
| $\frac{\Gamma \vdash e_1 : T \text{ ref} \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1 := e_2 : \text{unit}}$ | (T-ATR) | $\frac{}{\Gamma \vdash \text{skip} : \text{unit}}$ | (T-SKIP) |
| $\frac{\Gamma \vdash e : T \text{ ref}}{\Gamma \vdash ! e : T}$ | (T-DEREF) | $\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{unit}}{\Gamma \vdash \text{while } e_1 \text{ do } e_2 : \text{unit}}$ | (T-WHILE) |
| $\frac{\Gamma \vdash e : T}{\Gamma \vdash \text{new } e : T \text{ ref}}$ | (T-NEW) | $\frac{\Gamma \vdash e_1 : \text{unit} \quad \Gamma \vdash e_2 : T}{\Gamma \vdash e_1; e_2 : T}$ | (T-SEQ) |

Trabalho

O trabalho consiste em implementar em OCaml um interpretador para a linguagem L1 com as construções imperativas definidas acima. O avaliador do interpretador deve ser implementado seguindo o estilo *big step* com ambientes. Os *data types* **expr** e **tipo** devem se estendidos da seguinte forma (o *data types* **valor** também deve ser modificado para representação dos novos valores da linguagem):

```
type expr =  
  ...  
| Ase of expr * expr  
| Dref of expr  
| New of expr  
| Seq of expr * expr  
| Whl of expr * expr  
| Skip  
  
type tipo =  
  ...  
| TyRef of tipo  
| TyUnit
```

O trabalho deve ser realizado em **grupos com 3 componentes**.

Observação: note que endereços representados por l, l_1, \dots **não podem aparecer em programas**. Endereços podem aparecer apenas em passos intermediários da avaliação de programas no estilo *small step*.