Trabalho 2015/1 - Semântica Formal
Interpretador de L1 baseado em semântica *big step* com ambientes

- O trabalho deve ser submetido pelo Moodle até as 23:59 minutos do dia **7 de JUNHO**.
- As apresentações serão nos dias 8 e 10 de junho. O calendário com dia e hora da apresentação de cada grupo está no Moodle.
- A apresentação é uma prova oral na qual serão feitas perguntas a todos os components do grupo.
- Cada integrante deve portanto estudar e estar preparado para
  - responder perguntas sobre a especificação (semântica operacional *big step* com ambientes) e perguntas sobre detalhes da implementação em OCaml
  - "operar"a implementação testando-a com programas L1 de teste e explicando o resultado obtido.

TRABALHO 1 (nota máxima 8,0) - implementar um avaliador para expressões da linguagem L1 sem recursão. A implementação deve seguir estritamente a semântica operacional *big step call by value* da linguagem e usar ambientes para guardar valores associados a identificadores.

Gramática abstrata da linguagem L1:

Como nesse trabalho não será implementado um verificador de tipos para a linguagem, e como a verificação de tipos de L1 é feita estaticamente, a implementação de uma avaliador pode assumir uma linguagem mais simples sem anotações de tipos em expressões.

$$\begin{array}{rcl} e & ::= & n \mid b \mid e_1 \ op \ e_2 \mid \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \\ & \mid & x \mid e_1\ e_2 \mid \texttt{fn}\ x\ \Rightarrow e \mid \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \end{array}$$

Valores produzidos pela avaliação

Note que os valores produzidos são números, booleamos e *closures* de funções. Um *closure* é uma tripla com o argumento, corpo da função, e o ambiente corrente no momento em que a função foi introduzida no programa:

$$v \quad ::= \quad n \mid b \mid \langle x, e, \mathsf{env} \rangle$$

Semântica Operacional *big step call by value* para L1 com ambientes

Um ambiente é um mapeamento de identificadores para os seus valores.

$$\mathsf{env} \vdash n \Downarrow n$$

$$\mathsf{env} \vdash b \Downarrow b$$

$$\frac{\mathsf{env}(x) = v}{\mathsf{env} \vdash x \Downarrow v}$$

$$\frac{\mathsf{env} \vdash e_1 \Downarrow \texttt{true} \qquad \mathsf{env} \vdash e_2 \Downarrow v}{\mathsf{env} \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Downarrow v}$$

$$\frac{\mathsf{env} \vdash e_1 \Downarrow \texttt{false} \qquad \mathsf{env} \vdash e_3 \Downarrow v}{\mathsf{env} \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \Downarrow v}$$

$$\frac{\mathsf{env} \vdash e_1 \Downarrow \langle x, e, \mathsf{env}' \rangle \qquad \mathsf{env} \vdash e_2 \Downarrow v' \qquad \{x \mapsto v'\} + \mathsf{env}' \vdash e \Downarrow v}{\mathsf{env} \vdash e_1 \ e_2 \Downarrow v}$$

$$\mathsf{env} \vdash \texttt{fn } x \Rightarrow e \Downarrow \langle x, e, \mathsf{env} \rangle$$

$$\frac{\mathsf{env} \vdash e_1 \Downarrow v' \qquad \{x \mapsto v'\} + \mathsf{env} \vdash e_2 \Downarrow v}{\mathsf{env} \vdash \texttt{let } x = e_1 \texttt{ in } e_2 \Downarrow v}$$

TRABALHO 2 (nota máxima 10,0) - este trabalho é semelhante ao trabalho 1, porém a linguagem L1 deve ser entendida para suportar a definição de funções recursivas:

$$\begin{array}{rcl} e & ::= & \ldots \\ & | & \texttt{let rec } f = \texttt{ fn } x \Rightarrow e_1 \texttt{ in } e_2 \end{array}$$

E a seguinte regra da semântica operacional deve ser adicionada:

$$\frac{\mathsf{env} \vdash \texttt{fn } x \Rightarrow e_1 \Downarrow \langle x, e_1, \mathsf{env} \rangle \qquad \{f \mapsto \langle x, e_1, \mathsf{env} \rangle\} + \{x \mapsto v'\} + \mathsf{env} \vdash e_2 \Downarrow v}{\mathsf{env} \vdash \texttt{let rec } f = \texttt{ fn } x \Rightarrow e_1 \texttt{ in } e_2 \Downarrow v}$$

TRABALHO 3 (nota máxima 12) - este trabalho é semelhante ao trabalho 2 porém, antes de serem avaliadas, as expressões da linguagem L1 devem ser transformadas para a representação usando *índices de de Bruijn*. Para esse trabalho devem ser feitos:

1. regras da semântica operacional *big step* com ambientes para a linguagem com índices de de Bruijn
2. transformação de expressões L1 com nomes de variáveis para expressões L1 com índices de de Bruijn
3. avaliador de expressões L1 com índices de de Bruijn seguindo a semântica dada

Abaixo segue a gramática para L1 usando índices de de Bruijn (note que não há nomes de identificadores na linguagem):

$$\begin{array}{rcl} e & ::= & n \\ & | & b \\ & | & e_1 \ op \ e_2 \\ & | & \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 \\ & | & \underline{n} \\ & | & e_1 \ e_2 \\ & | & \texttt{fn } \Rightarrow e \\ & | & \texttt{let } e_1 \texttt{ in } e_2 \\ & | & \texttt{let rec fn } \Rightarrow e_1 \texttt{ in } e_2 \end{array}$$

Estrutura do código

Abaixo segue a estrutura básica do código para os trabalhos 1 e 2. A estrutura para o avaliador trabalho 3 é semelhante mas a linguagem de entrada para o avaliador deve usar índices de de Bruijn e ambientes podem ser implementados como listas de índices de de Bruijn.

```
type variable = string

type operator =
   Sum | Diff | Mult | Div | Eq | Leq

type expr =
   Num of int | Bool of bool | Bop of expr * operator * expr
 | If of expr * expr * expr | Var of variable  | App of expr * expr
 | Lam of variable * expr | Let of variable * expr * expr
 | Lrec of variable * expr * expr

type value  =
  | Vnum of int | Vbool of bool | Vclos of variable * expr * env
and
     env = (variable * value) list

(* ambiente vazio  *)
let empty_env : env = []

(*  busca no ambiente:
lookup_env env x  == Some v
     (x,v) é o par contendo x adicionado mais recentemente a env
lookup_env env x == None, se env não possui par com x  *)

let rec lookup_env (env:env) (x:variable) : value option =
  failwith "unimplemented"

(* atualização do ambiente:
update env x v retorna um novo env contendo o par (x,v)  *)

let update_env (env:env) (x:variable) (v: value) : env =
  failwith "unimplemented"

let rec eval (env: env) (e: expr) : value option =
 failwith "unimplemented"
```