CIS 705 - Language Design Project
Assignment 2

Élodie-Jane Sims

The language we study for the project is a simple while-language with Lisp-like expressions for accessing and creating `cons` cells and functions definitions and call.

1 Syntax of the langage

The commands we consider are as follows.

$$
\begin{array}{rl}
C ::= & x := E \\
| & x := y.i \\
| & x.i := E \\
| & x := \texttt{cons}(E_1, E_2) \\
| & \texttt{dispose}(x) \\
| & C_1; C_2 \\
| & if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2 \\
| & \texttt{skip} \\
| & while\ \ E\ \ do\ \ C_1 \\
| & \texttt{fun}\ f\ (v1, ..., v_n)\{\ C\ \} \\
| & \texttt{call}\ f\ (a1, .., a_n)
\end{array}
$$

$$i\ ::= 1|2$$

Here, x, v_i, a_i range over variables. Here, f range over function names.
An expression can denote an integer, an atom, or a `cons` cell.

$$
\begin{array}{rll}
E ::= & x & \text{Variable} \\
| & \texttt{nil} & \text{Atom} \\
| & n & \text{Integer} \\
| & E1\ \ +\ \ E_2 & \text{Integers addition} \\
| & E1\ \ -\ \ E_2 & \text{Integers substraction} \\
| & E1\ \ \times\ \ E_2 & \text{Integers multiplication} \\
| & E1\ \ \div\ \ E_2 & \text{Integers division} \\
| & True & \text{Boolean true} \\
| & False & \text{Boolean false} \\
| & E1\ \ ||\ \ E_2 & \text{Boolean disjunction} \\
| & E1\ \ \&\&\ \ E_2 & \text{Boolean conjunction} \\
| & E1\ \ \Rightarrow\ \ E_2 & \text{Boolean implication} \\
| & \neg\ E_2 & \text{Boolean negation} \\
| & E1\ \ =\ \ E_2 & \text{Polymorphique equality}
\end{array}
$$

The second and third assignment statements read and update the heap, respectively. The fourth creates a new `cons` cell in the heap, and places a pointer to it in x.

Notice that in our language we do not handle two dereferencing in a simple statement (no $x.i.j$, no $x.i := y.j$), we assign exactly two cells at each assignment statements; those restrictions are just for simplicity and do not limit the expressivity of the language, we might change this in future or may be just at the implementation step.

2 Typing relation

2.1 Types Syntax

The types that an expression can take :

$$\begin{array}{rll} T := & INT & \\ & \mid BOOL & \\ & \mid NIL & \\ & \mid T\ LIST & \\ & \mid PAIR(T,T) & \\ & \mid BOTTOM & \text{not typable} \\ & \mid TOP & \text{some type we don't know} \end{array}$$

The set of errors that we can have :

$$\begin{array}{rll} ERROR := & DIV & \text{error of division by 0} \\ & \mid DEREF & \text{error of dereferencing a nil or something else} \\ & \mid TYPES & \text{error of types compatibility} \\ & \mid OUTOFSTACK & \text{error when a variable is not in the heap} \\ & \mid UNDECLAREDFUN & \text{error call of an undeclared function} \end{array}$$

The types that a programm can take

$$\begin{array}{rll} CT := & UNIT & \text{the program does not hold any error} \\ & \mid ERROR & \text{the program stop running because of error ERROR} \\ & \mid CT \times CT & \text{the program can have both behaviours} \end{array}$$

2.2 Subtyping relation

$$T \ <: \ T$$

$$T \ <: \ TOP$$

$$\frac{T_1 \ <: \ T_1'}{PAIR(T_1, T_2) \ <: \ PAIR(T_1', T_2)}$$

$$\frac{T_2 \ <: \ T_2'}{PAIR(T_1, T_2) \ <: \ PAIR(T_1, T_2')}$$

$$\frac{T_1 \ <: \ T_2}{T_1\ LIST \ <: \ T_2\ LIST}$$

$$PAIR(T, NIL) \ <: T\ LIST$$

$$NIL <: T\ LIST$$

$$\frac{T_1 \ <: \ T_2 \ \ T_2 \ <: \ T_3}{T_1 \ <: \ T_3}$$

For sets of types :

$$T_1 \subseteq: T_2 \text{ iff } \forall t1 \in T_1 . \exists t2 \in T_2 . t1 <: t2$$

$$T_1 \subset: T_2 \text{ iff } T_1 \subseteq: T_2 \wedge \exists t2 \in T_2 . \forall t1 \in T_1 . \neg t1 <: t2$$

2.3 Typing relation for expressions

Our typing relation is of the form $\Gamma \vdash_E E, \Omega$ with Γ maping variables to set of types and Ω set of errors.

$$\frac{x \in dom(\Gamma),\ \Gamma(x) = G \cup \{BOTTOM\}}{\Gamma \vdash_E x \ : \ G, \{OUTOFSTACK\}}$$

$$\frac{x \in dom(\Gamma),\ BOTTOM \notin \Gamma(x)}{\Gamma \vdash_E x \ : \ \Gamma(x), \emptyset}$$

$$\frac{x \notin dom(\Gamma)}{\Gamma \vdash_E x \ : \emptyset, \{OUTOFSTACK\}}$$

$$\frac{}{\Gamma \vdash_E \texttt{nil} \ : \ \{NIL\}, \emptyset}$$

$$\frac{}{\Gamma \vdash_E n \ : \ \{INT\}, \emptyset}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ \{INT\}, err1 \quad \Gamma \vdash_E E2 \ : \ \{INT\}, err2}{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ \{INT\}, err1 \cup err2}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ \{INT\} \subset: t1 \ \ \{INT\} \subseteq: t2}{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ \{INT\}, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ \{INT\} \subseteq: t1 \ \ \{INT\} \subset: t2}{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ \{INT\}, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ (INT \notin t1 \text{ or } INT \notin t2}{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ \emptyset, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ t, err}{\Gamma \vdash_E E_1 \ - \ E_2 \ : \ t, err}$$

$$\frac{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ t, err}{\Gamma \vdash_E E_1 \ \times \ E_2 \ : \ t, err}$$

$$\frac{\Gamma \vdash_E E_1 \ + \ E_2 \ : \ t, err}{\Gamma \vdash_E E_1 \ \div \ E_2 \ : \ t, err \cup \{DIV\}}$$

$$\frac{}{\Gamma \vdash_E \texttt{true} \ : \ \{BOOL\}, \emptyset}$$

$$\frac{}{\Gamma \vdash_E \texttt{false} \ : \ \{BOOL\}, \emptyset}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ \{BOOL\}, err1 \quad \Gamma \vdash_E E2 \ : \ \{BOOL\}, err2}{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ \{BOOL\}, err1 \cup err2}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ \{BOOL\} \subset: t1 \ \ \{BOOL\} \subseteq: t2}{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ \{BOOL\}, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ \{BOOL\} \subseteq: t1 \ \ \{BOOL\} \subset: t2}{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ \{BOOL\}, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ t1, err1 \quad \Gamma \vdash_E E2 \ : \ t2, err2 \ \ (BOOL \notin t1 \text{ or } BOOL \notin t2}{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ \emptyset, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ t, err}{\Gamma \vdash_E E_1 \ \&\& \ E_2 \ : \ t, err}$$

$$\frac{\Gamma \vdash_E E_1 \ || \ E_2 \ : \ t, err}{\Gamma \vdash_E E_1 \ \Rightarrow \ E_2 \ : \ t, err}$$

$$\frac{\Gamma \vdash_E E_2 \ \Rightarrow \ \texttt{false} \ : \ t, err}{\Gamma \vdash_E \neg E_2 \ : \ t, err}$$

The = case is a bit more complicate because we want to be able to check the equality between nil and a pair or a list.

$$\frac{\Gamma \vdash_E E_1 \ : \ \{INT\}, err1 \ \ \Gamma \vdash_E E_2 \ : \ \{INT\}, err2}{\Gamma \vdash_E E_1 \ = \ E_2 \ : \ \{BOOL\}, err1 \cup err2}$$

$$\frac{\Gamma \vdash_E E_1 \ : \ \{BOOL\}, err1 \ \ \Gamma \vdash_E E_2 \ : \ \{BOOL\}, err2}{\Gamma \vdash_E E_1 \ = \ E_2 \ : \ \{BOOL\}, err1 \cup err2}$$

Egality of pointers :

$$\frac{\begin{array}{c} \Gamma \vdash_E E_1 \ : \ t1, err1 \\ \Gamma \vdash_E E_2 \ : \ t2, err2 \\ \{INT\} \not\subseteq: t1, t2 \\ \{BOOL\} \not\subseteq: t1, t2 \\ t1 \neq \emptyset \\ t2 \neq \emptyset \end{array}}{\Gamma \vdash_E E_1 \ = \ E_2 \ : \ \{BOOL\}, err1 \cup err2}$$

$$\frac{\begin{array}{c} \Gamma \vdash_E E_1 \ : \ T_1, err1 \\ \Gamma \vdash_E E_2 \ : \ T_2, err2 \\ \exists t1 \subseteq: T_1 . t2 \subseteq: T_2 . \frac{\Gamma \vdash_E E_1 \, : \, t1, err1 \ \ \Gamma \vdash_E E_2 \, : \, t2, err2}{\Gamma \vdash_E E_1 \ = \ E_2 \, : \, \{BOOL\}, err} \\ \exists t1 \subseteq: T_1 . t2 \subseteq: T_2 . \frac{\Gamma \vdash_E E_1 \, : \, t1, err1 \ \ \Gamma \vdash_E E_2 \, : \, t2, err2}{\Gamma \vdash_E E_1 \ = \ E_2 \, : \, \emptyset, err'} \end{array}}{\Gamma \vdash_E E_1 \ = \ E_2 \ : \ \{BOOL\}, err1 \cup err2 \cup \{TYPES\}}$$

$$\frac{\begin{array}{c}\Gamma \vdash_E E_1 \ : \ T_1, err1 \\ \Gamma \vdash_E E_2 \ : \ T_2, err2 \\ \neg\exists t1 \subseteq: T_1.t2 \subseteq: T_2. \frac{\Gamma \vdash_E E_1 \,:\, t1, err1 \quad \Gamma \vdash_E E_2 \,:\, t2, err2}{\Gamma \vdash_E E_1 \ = \ E_2 : \{BOOL\}, err}\end{array}}{\Gamma \vdash_E E_1 \ = \ E_2 \ : \ \emptyset, err1 \cup err2 \cup \{TYPES\}}$$

2.4 Typing relation for commands

Our typing relation is of the form $\Gamma, fn \vdash C : CT, \Gamma', fn'$ with $\Gamma \ : \ Var \rightharpoonup_{fin} Types$, and $fn \ : \ Fun \rightharpoonup_{fin} (\mathcal{P}(Var), Commands)$.

Case there is for sure an error in the expression E.

$$\frac{\Gamma \vdash_E E : \emptyset, err}{\Gamma, fn \vdash x := E : err, \Gamma, fn}$$

Here we write err instead of changing the set into $err1 \times err2 \times ...$, in the future we will also write $\{UNIT\} \cup err$ for $UNIT \times err$.

Case there may be no error in E (the case for sure no error is included, it is when $err = \emptyset$).

$$\frac{\Gamma \vdash_E E : T, err \quad T \neq \emptyset}{\Gamma, fn \vdash x := E : \{UNIT\} \cup err, [\Gamma \mid x \mapsto T], fn}$$

Case there are no error of dereferencing.

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \\ \forall t \in T.\exists t1, t2.t = PAIR(t1, t2) \\ Ti = \{ti \mid \exists tj.PAIR(tij) \in T\}\end{array}}{\Gamma, fn \vdash x := y.i : \{UNIT\} \cup err, [\Gamma \mid x \mapsto Ti], fn}$$

Case there is for sure dererencing error :

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \forall t1, t2.\{PAIR(t1, t2)\} \not\subseteq: T \\ \forall t1.\{t1 \ LIST\} \not\subseteq: T\end{array}}{\Gamma, fn \vdash x := y.i : \{DEREF\} \cup err, \Gamma, fn}$$

Case there may be some errors of dereferencing :

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \ \vee \ \exists t1.t1 \ LIST \in T \vee TOP \in T \\ Ti = \{ti \mid \exists tj.\{PAIR(tij)\} \subseteq: T\} \text{(this include the list case)}\end{array}}{\Gamma, fn \vdash x := y.i : \{UNIT, DEREF\} \cup err, [\Gamma \mid x \mapsto Ti], fn}$$

Case there is for sure an error in the expression E.

$$\frac{\Gamma \vdash_E E : \emptyset, err}{\Gamma, fn \vdash x.i := E : err, \Gamma, fn}$$

Case there are no error of dereferencing and no error of matching types.

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \\ \forall t \in T.\exists t1, t2.t = PAIR(t1, t2) \\ \forall te \in Te.\exists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{UNIT\} \cup err, \Gamma, fn}$$

Case there are no error of dereferencing but for sure error of matching types.

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \\ \forall t \in T.\exists t1, t2.t = PAIR(t1, t2) \\ \forall te \in Te.\ \nexists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{TYPES\} \cup err, \Gamma, fn}$$

Case there are no error of dereferencing but may be some error of matching types.

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \\ \forall t \in T.\exists t1, t2.t = PAIR(t1, t2) \\ \exists te \in Te.\ \nexists tj.PAIR(tej) \in T \\ \exists te \in Te.\exists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{UNIT, TYPES\} \cup err, \Gamma, fn}$$

Case there are for sure error of dereferencing.

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \forall t1, t2.\{PAIR(t1, t2)\} \not\subseteq: T \\ \forall t1.\{t1\ LIST\} \not\subseteq: T\end{array}}{\Gamma, fn \vdash x.i := E : \{DEREF\} \cup err, \Gamma, fn}$$

Case there may be some error of dereferencing but no error of types

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T\ \vee\ \exists t1.t1\ LIST \in T \vee TOP \in T \\ \forall te \in Te.\exists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{UNIT, DEREF\} \cup err, \Gamma, fn}$$

Case there may be error of derferencing and for sure error of types :

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T\ \vee\ \exists t1.t1\ LIST \in T \vee TOP \in T \\ \forall te \in Te.\ \nexists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{TYPES, DEREF\} \cup err, \Gamma, fn}$$

Case there may be error of derferencing and may be error of types :

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, err\ Te \neq \emptyset \\ \Gamma \vdash_E x : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T\ \vee\ \exists t1.t1\ LIST \in T \vee TOP \in T \\ \exists te \in Te.\ \nexists tj.PAIR(tej) \in T\end{array}}{\Gamma, fn \vdash x.i := E : \{UNIT, DEREF, TYPES\} \cup err, \Gamma, fn}$$

$$\frac{\Gamma \vdash_E E_1 : \emptyset, err1}{\Gamma, fn \vdash x := \mathtt{cons}(E_1, E_2) : err1, \Gamma, fn}$$

$$\frac{\Gamma \vdash_E E_2 : \emptyset, err2}{\Gamma, fn \vdash x := \mathtt{cons}(E_1, E_2) : err1, \Gamma, fn}$$

$$\frac{\begin{array}{c}\Gamma \vdash_E E_1 : T_1, err1\ T_1 \neq \emptyset \\ \Gamma \vdash_E E_1 : T_2, err2\ T_2 \neq \emptyset \\ T' = \{PAIR(t1, t2) \mid t1 \in T_1\ t2 \in T_2\}\end{array}}{\Gamma, fn \vdash x := \mathtt{cons}(E_1, E_2) : \{UNIT\} \cup err1 \cup err2, [\Gamma \mid x \mapsto T'], fn}$$

Case there are no error of dereferencing.

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T \\ \forall t \in T.\exists t1, t2.t = PAIR(t1, t2) \\ Ti = \{ti \mid \exists tj.PAIR(tij) \in T\}\end{array}}{\Gamma, fn \vdash \mathtt{dispose}(y) : \{UNIT\} \cup err, \Gamma - y, fn}$$

Case there is for sure dererencing error :

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \forall t1, t2.\{PAIR(t1, t2)\} \not\subseteq : T \\ \forall t1.\{t1\ LIST\} \not\subseteq : T\end{array}}{\Gamma, fn \vdash \mathtt{dispose}(y) : \{DEREF\} \cup err, \Gamma, fn}$$

Case there may be some errors of dereferencing :

$$\frac{\begin{array}{c}\Gamma \vdash_E y : T, err \\ \exists t1, t2.PAIR(t1, t2) \in T\ \vee\ \exists t1.t1\ LIST \in T \vee TOP \in T \\ Ti = \{ti \mid \exists tj.\{PAIR(tij)\} \subseteq : T\}\text{(this include the list case)}\end{array}}{\Gamma, fn \vdash \mathtt{dispose}(y) : \{UNIT, DEREF\} \cup err, \Gamma - y, fn}$$

Case error for sure in C_1.

$$\frac{\begin{array}{c}\Gamma, fn \vdash C_1 : T1, \Gamma_1, fn_1 \\ UNIT \notin T1\end{array}}{\Gamma, fn \vdash C_1; C_2 : T1, \Gamma_1, fn_1}$$

Case error for sure in C_2 and not in C_1.

$$\frac{\begin{array}{c}\Gamma, fn \vdash C_1 : T1, \Gamma_1, fn_1 \\ UNIT \in T1 \\ \Gamma, fn \vdash C_2 : T2, \Gamma_2, fn_2 \\ UNIT \notin T2 \\ T = T1 \setminus \{UNIT\}\end{array}}{\Gamma, fn \vdash C_1; C_2 : T \cup T2, \Gamma_2, fn_2}$$

Case may be not error in C_1 and C_2

$$\frac{\begin{array}{c}\Gamma, fn \vdash C_1 : T1, \Gamma_1, fn_1 \\ UNIT \in T1 \\ \Gamma, fn \vdash C_2 : T2, \Gamma_2, fn_2 \\ UNIT \in T2\end{array}}{\Gamma, fn \vdash C_1; C_2 : T1 \cup T2, \Gamma_2, fn_2}$$

We define $\Gamma \subseteq: \Gamma'$ iff

- $dom(\Gamma) \subseteq dom(\Gamma')$
- if $x \in dom(\Gamma)$ then $\Gamma(x) \subseteq: \Gamma'(x)$
- if $x \in dom(\Gamma') \setminus dom(\Gamma)$ then $BOTTOM \in \Gamma'(x)$

For sure error in the evaluation of E:

$$\frac{\Gamma \vdash_E E : \emptyset, err}{\Gamma, fn \vdash if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2 : err, \Gamma, fn}$$

For sure E is a boolean :

$$\frac{\begin{array}{c}\Gamma \vdash_E E : \{BOOL\}, erre \\ \Gamma, fn \vdash C_1\ :\ T1, \Gamma_1, fn_1 \\ \Gamma, fn \vdash C_2\ :\ T2, \Gamma_2, fn_2 \\ \Gamma_1 <: \Gamma' \\ \Gamma_2 <: \Gamma'\end{array}}{\Gamma, fn \vdash if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2 : T1 \cup T2 \cup erre, \Gamma', fn_1 \cap_E fn_2}$$

For sure E is not a boolean:

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, erre \\ \{BOOL\} \not\subseteq: Te\ \wedge\ Te \neq \emptyset\end{array}}{\Gamma, fn \vdash if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2 : \{TYPES\} \cup erre, \Gamma, fn}$$

E may be a boolean :

$$\frac{\begin{array}{c}\Gamma \vdash_E E : Te, erre \\ \{BOOL\} \subseteq: Te \\ \Gamma, fn \vdash C_1\ :\ T1, \Gamma_1, fn_1 \\ \Gamma, fn \vdash C_2\ :\ T2, \Gamma_2, fn_2 \\ \Gamma_1 <: \Gamma' \\ \Gamma_2 <: \Gamma'\end{array}}{\Gamma, fn \vdash if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2 : T1 \cup T2 \cup erre \cup TYPES, \Gamma', fn_1 \cap_E fn_2}$$

We define $fn_1 \cap_E fn_2$ as :

- $dom(fn_1 \cap_E fn_2) \subseteq dom(fn_1) \cap dom(fn_2)$
- if $\begin{pmatrix} fn_1(f) = \langle (v_1, ..., v_n), Cf_1 \rangle \\ fn_2(f) = \langle (w_1, ..., w_m), Cf_2 \rangle \\ n = m \end{pmatrix}$
  then $\begin{pmatrix} (fn_1 \cap fn_2)(f) = \langle (u_1, ..., u_n), \\ if \ \ E \ \ then \ \ Cf_1\{u_i/v_i\} \ \ else \ \ Cf_2\{u_i/w_i\} \rangle \end{pmatrix}$
  with u_i variables which do not occur in Cf_1 and Cf_2.

$$\frac{}{\Gamma, fn \vdash \texttt{skip} : UNIT, \Gamma, fn}$$

Case E can't be evaluate in the first path :

$$\frac{\Gamma \vdash_E E : \emptyset, erre}{\Gamma, fn \vdash while \ \ E \ \ do \ \ C : erre, \Gamma, fn}$$

Usual rule :

$$\frac{\begin{array}{c} \Gamma, fn \vdash if \ \ E \ \ then\, C\ else\ \texttt{skip} : T, \Gamma', fn' \\ UNIT \notin T \end{array}}{\Gamma, fn \vdash while \ \ E \ \ do \ \ C : T, \Gamma', fn'}$$

$$\frac{\begin{array}{c} \Gamma, fn \vdash if \ \ E \ \ then\, C\ else\ \texttt{skip} : T, \Gamma', fn' \\ UNIT \in T \\ T_err = T \setminus \{UNIT\} \\ \Gamma', fn' \vdash while \ \ E \ \ do\, C : T'', \Gamma'', fn'' \end{array}}{\Gamma, fn \vdash while \ \ E \ \ do \ \ C : T'' \cup T_err, \Gamma'', fn''}$$

Case there can't be a type error coming from E.

$$\frac{\begin{array}{c} \Gamma \vdash_E E : \{BOOL\}, erre \\ \Gamma' \vdash_E E : \{BOOL\}, erre' \\ \exists \Gamma_n \begin{pmatrix} \Gamma \subseteq: \Gamma' \\ \Gamma_n \subseteq: \Gamma' \\ \Gamma', fn \vdash C : T, \Gamma_n, fn' \end{pmatrix} \end{array}}{\Gamma, fn \vdash while \ \ E \ \ do \ \ C : \{UNIT\} \cup T, \Gamma', fn'}$$

Case there can be a type error from E.

$$\frac{\begin{array}{c} \Gamma \vdash_E E : Te, erre \\ \{BOOL\} \subset: Te \\ \Gamma' \vdash_E E : Te', erre' \\ \{BOOL\} \subseteq: Te' \\ \exists \Gamma_n \begin{pmatrix} \Gamma \subseteq: \Gamma' \\ \Gamma_n \subseteq: \Gamma' \\ \Gamma', fn \vdash C : T, \Gamma_n, fn' \end{pmatrix} \end{array}}{\Gamma, fn \vdash while \ \ E \ \ do \ \ C : \{UNIT, TYPES\} \cup T, \Gamma', fn'}$$

$$\frac{\begin{array}{c} \Gamma \vdash_E E : Te, erre \\ \{BOOL\} \subseteq: Te \\ \Gamma' \vdash_E E : Te', erre' \\ \{BOOL\} \subset: Te' \\ \exists \Gamma_n \begin{pmatrix} \Gamma \subseteq: \Gamma' \\ \Gamma_n \subseteq: \Gamma' \\ \Gamma', fn \vdash C : T, \Gamma_n, fn' \end{pmatrix} \end{array}}{\Gamma, fn \vdash while \;\; E \;\; do \;\; C : \{UNIT, TYPES\} \cup T, \Gamma', fn'}$$

here we had to add $UNIT$ because the E can be directly false.

$$\frac{}{\Gamma, fn \vdash \texttt{fun}\; f\; (v1, ..., v_n)\{\; C\; \} : \{UNIT\}, \Gamma, [fn | f \mapsto ((v1, ..., v_n), C)]}$$

$$\frac{f \notin dom(fn)}{\Gamma, fn \vdash \texttt{call}\; f\; (a1, ..., a_n) : \{UNDECLARED\}, \Gamma, fn}$$

$$\frac{fn(f) = ((v1, .., v_n), C) \quad \Gamma, fn \vdash C\{a_i/v_i\} : T, \Gamma', fn'}{\Gamma, fn \vdash \texttt{call}\; f\; (a1, ..., a_n) : T, \Gamma', fn'}$$

3 Evaluation relation

$$\begin{aligned} Val &= Int \cup \{nil, true, false\} \cup Loc \\ S &= Var \rightharpoonup_{fin} Val \\ H &= Loc \rightharpoonup_{fin} Val \times Val \\ F &= Fun \rightharpoonup_{fin} (\mathcal{P} \qquad\qquad (Var), Commands)) \end{aligned}$$

Here, $Loc = \{l, ...\}$ is an infinite set of locations, $Var = \{x, y, ...\}$ is a set of variables, $Atoms = \{\texttt{nil}, \texttt{true}, \texttt{false}, a, ...\}$ is a set of atoms, $Fun = \{f, g, ...\}$ is a set of function's variables, and $\rightharpoonup_{fin}$ is for finite partial functions. We call an element $s \in S$ a stack, and $h \in H$ a heap.

We use $dom(h)$ to denote the domain of definition of a heap $h \in H$, and $dom(s)$ to denote the domain of a stack $s \in S$.

An expression is interpreted as a heap-independent value

$$[\![E]\!]s \in Val$$

where $dom(s)$ includes the free variables of E.

$$
\begin{aligned}
[\![x]\!]s &= s(x)\\
[\![\mathtt{nil}]\!]s &= nil\\
[\![n]\!]s &= n\\
[\![E_1 + E_2]\!]s &= [\![E_1]\!]s + [\![E_2]\!]s\\
[\![E_1 - E_2]\!]s &= [\![E_1]\!]s - [\![E_2]\!]s\\
[\![E_1 \times E_2]\!]s &= [\![E_1]\!]s \times [\![E_2]\!]s\\
[\![E_1 \div E_2]\!]s &= [\![E_1]\!]s \div [\![E_2]\!]s\\
[\![\mathtt{true}]\!]s &= true\\
[\![\mathtt{false}]\!]s &= false\\
[\![E_1 \mid\mid E_2]\!]s &= [\![E_1]\!]s \vee [\![E_2]\!]s\\
[\![E_1 \&\& E_2]\!]s &= [\![E_1]\!]s \wedge [\![E_2]\!]s\\
[\![E_1 \Rightarrow E_2]\!]s &= [\![E_1]\!]s \Rightarrow [\![E_2]\!]s\\
[\![\neg E_2]\!]s &= \neg[\![E_1]\!]s\\
[\![E_1 = E_2]\!]s &= [\![E_1]\!]s = [\![E_2]\!]s
\end{aligned}
$$

The functions $+, -, \times, \div$ are the usual functions on integers and $\vee, \wedge, \Rightarrow, \neg$ the usual functions on booleans and $=$ is the usual polymorphique equality function. We not add an error value so $[\![E]\!]$ is not always defined.

The semantics of our statements are given in a small-step semantic defined by the relation $\rightsquigarrow$ on configurations. The configurations include quadruples C, fn, s, h and terminal configurations fn, s, h for $s \in S$ and $h \in H$ and $fn \in F$.

In the following rules we use r to range over elements of $Val \times Val$, $\pi_i r$ for the first or second projection, and $(r|i \mapsto v)$ to indicate the pair like r except that the i'th component is replaced with v.

If we write a condition for example $h(s(y)) = r$ it means that $y \in dom(s)$ and $s(y) \in dom(h)$.

$$\frac{[\![E]\!]s = v}{x := E, fn, s, h \rightsquigarrow fn, [s|x \mapsto v], h}$$

$$\frac{h(s(y)) = r}{x := y.i, fn, s, h \rightsquigarrow fn, [s|x \mapsto \pi_i r], h}$$

$$\frac{h(s(x)) = r \quad [\![E]\!]s = v}{x.i = E, fn, s, h \rightsquigarrow fn, s, [h|s(x) \mapsto (r|i \mapsto v)]}$$

$$\frac{s(x) \in dom(h)}{\mathtt{dispose}(x), fn, s, h \rightsquigarrow fn, s, (h - s(x))}$$

$$\frac{l \in Loc \quad l \notin dom(h) \quad [\![E_1]\!]s = v_1, [\![E_2]\!]s = v_2}{x := \mathtt{cons}(E_1, E_2), fn, s, h \rightsquigarrow fn, [s|x \mapsto l], [h|l \mapsto \langle v_1, v_2\rangle]}$$

$$\frac{C_1, fn, s, h \rightsquigarrow C', fn', s', h'}{C_1; C_2, fn, s, h \rightsquigarrow C'; C_2, fn', s', h'}$$

$$\frac{C_1, fn, s, h \rightsquigarrow fn', s', h'}{C_1; C_2, fn, s, h \rightsquigarrow C_2, fn', s', h'}$$

$$\frac{[\![E]\!]s = True}{if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2, fn, s, h \rightsquigarrow C_1, fn, s, h}$$

$$\frac{[\![E]\!]s = False}{if\ \ E\ \ then\ \ C_1\ \ else\ \ C_2, fn, s, h \rightsquigarrow C_2, fn, s, h}$$

$$\frac{}{\texttt{skip}, fn, s, h \rightsquigarrow fn, s, h}$$

$$\frac{[\![E]\!]s = True}{while\ \ E\ \ do\ \ C, fn, s, h \rightsquigarrow C; while\ \ E\ \ do\ \ C, fn, s, h}$$

$$\frac{[\![E]\!]s = False}{while\ \ E\ \ do\ \ C, fn, s, h \rightsquigarrow fn, s, h}$$

$$\frac{}{\texttt{fun}\ f\ (v1, ..., v_n)\{\ C\ \}, fn, s, h \rightsquigarrow [fn|f \mapsto ((v1, ..., v_n), C)], s, h}$$

$$\frac{fn(f) = ((v1, .., v_n), C)}{\texttt{call}\ f\ (a1, ..., a_n), fn, s, h \rightsquigarrow C\{a_i/v_i\}, fn, s, h}$$

The location l in the fifth case is not specified uniquely, so a new location is chosen non-deterministically.

4 Theorems

We would like to prove some theorem like :

Theorem 1. *If $\emptyset, fn \vdash C : T, \Gamma', fn'$ then in case T contains $UNIT$ and some set of errors then the execution of C can either run without error or can get stuck by some error of the kind in the set of error in T. In case $T = \{UNIT\}$ then we know that C will be executed without any errors. In case T is only a set of error, we can say that if there is an error occuring it is one of the kind in T (but we also can have no error since we have restricted our typing to some special programs which do not change the types of data in the heap).*

In case the execution run without error we have that the variable values at the end can be types with the types in Γ'.

We could not say that if $UNIT \notin T$ there is for sure an error because we did ask that the types in the heap do not change while there can be some safe execution of the command which change the types in the heap.

Proof. We should first prove that for each simple statement the result is safe which some types in the set of types in Γ can run to an error, then we give this error in the result.

For the if then else case, we make the union of both set of types that can be in Γ.