Compilation
Deuxième partie, cahier des charges

À rendre le 27 novembre 2001

1 Compilation de votre projet

Votre projet doit se présenter sous forme d'une archive tar compressée (option "z" de tar), appelée *vos_noms*`.tgz` qui doit contenir un répertoire appelé *vos_noms* (exemple : `courant-filliatre.tgz`). Dans ce répertoire doivent se trouver les *sources* de votre programme (ne donnez pas les fichiers compilés). Quand on se place dans ce répertoire, la commande `make` doit créer votre compilateur, qui sera appelé `camlette`. La commande `make clean` doit effacer tous les fichiers que `make` a engendré et ne laisser dans le répertoire que les fichiers sources.

2 Typage des programmes Camlette

Nous donnons ici les règles sémantiques de bonne formation des programmes Camlette.

2.1 Types et environnements

Les expressions de types τ et les environnements de typage Γ sont donnés par la grammaire suivante :

$$\begin{array}{lcl} \tau & ::= & \texttt{unit} \mid \texttt{bool} \mid \texttt{int} \mid \texttt{float} \mid \texttt{string} \mid \tau \to \tau \mid x \\ \Gamma & ::= & \epsilon \mid \Gamma, x : \tau \mid \Gamma, \texttt{type}\ x \end{array}$$

où x désigne un identificateur.

On écrit $dom(\Gamma)$ pour l'ensemble des variables de Γ liées à des types, et $tdom(\Gamma)$ l'ensemble des types de Γ. On écrit $x : \tau \in \Gamma$ s'il existe Γ_1 et Γ_2 tels que $\Gamma = \Gamma_1, x : \tau, \Gamma_2$ avec $x \notin dom(\Gamma_2)$.

Le jugement $\Gamma \vdash \tau\ \texttt{bf}$ exprime qu'un type τ est bien formé dans un environnement Γ. Il se dérive par les règles suivantes :

$$\frac{\tau \in \{\texttt{unit}, \texttt{bool}, \texttt{int}, \texttt{float}, \texttt{string}\}}{\Gamma \vdash \tau\ \texttt{bf}} \qquad \frac{\Gamma \vdash \tau_1\ \texttt{bf} \quad \Gamma \vdash \tau_2\ \texttt{bf}}{\Gamma \vdash \tau_1 \to \tau_2\ \texttt{bf}} \qquad \frac{\texttt{type}\ x \in \Gamma}{\Gamma \vdash x\ \texttt{bf}}$$

2.2 Typage des expressions

Le jugement $\Gamma \vdash e : \tau$ exprime qu'une expression e a le type τ dans un environnement Γ. Il se dérive par les règles suivantes :

$$\frac{c \text{ constante de type } \tau}{\Gamma \vdash c : \tau} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\Gamma \vdash \tau_1\ \texttt{bf} \quad \Gamma, x : \tau_1 \vdash e : \tau_2}{\Gamma \vdash \texttt{fun}\ (x : \tau_1) \to e : \tau_1 \to \tau_2}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2} \qquad \frac{\Gamma \vdash \tau_1\ \texttt{bf} \quad \Gamma, x : \tau_1 \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let rec}\ x : \tau_1 = e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\frac{\Gamma \vdash e_2 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_1 : \tau_1}{\Gamma \vdash (e_2\ e_1) : \tau_2} \qquad \frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 : \tau}$$

$$\frac{\Gamma \vdash e : \texttt{int}}{\Gamma \vdash e : \texttt{float}} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau \quad op \in \{\texttt{+}, \texttt{*}, \texttt{-}, \texttt{/}\} \quad \tau \in \{\texttt{int}, \texttt{float}\}}{\Gamma \vdash e_1\ op\ e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau \quad op \in \{\texttt{=}, \texttt{<>}, \texttt{<}, \texttt{<=}, \texttt{>}, \texttt{>=}\} \quad \tau \in \{\texttt{int}, \texttt{float}, \texttt{string}\}}{\Gamma \vdash e_1\ op\ e_2 : \texttt{bool}}$$

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \texttt{bool} \quad op \in \{\texttt{\&\&}, \texttt{||}\}}{\Gamma \vdash e_1\ op\ e_2 : \texttt{bool}} \qquad \frac{\Gamma \vdash e : \texttt{bool}}{\Gamma \vdash \texttt{not}\ e : \texttt{bool}} \qquad \frac{\Gamma \vdash e : \tau \quad \tau \in \{\texttt{int}, \texttt{float}\}}{\Gamma \vdash \texttt{-}e : \tau}$$

2.3 Typage des déclarations

Le jugement $\vdash d \Rightarrow \Gamma$ exprime qu'un programme, i.e. une suite de déclarations d, est bien formé et que son évaluation donne un environnement Γ. Ce jugement se dérive par les règles suivantes :

$$\frac{}{\vdash \epsilon \Rightarrow \epsilon} \qquad \frac{\vdash d \Rightarrow \Gamma \quad x \notin tdom(\Gamma)}{\vdash d, \texttt{type}\ x \Rightarrow \Gamma, \texttt{type}\ x} \qquad \frac{\vdash d \Rightarrow \Gamma \quad x \notin dom(\Gamma) \quad \Gamma \vdash \tau\ \texttt{bf}}{\vdash d, \texttt{external}\ x : \tau \Rightarrow \Gamma, x : \tau}$$

$$\frac{\vdash d \Rightarrow \Gamma \quad \Gamma \vdash e : \tau}{\vdash d, \texttt{let}\ x = e \Rightarrow \Gamma, x : \tau} \qquad \frac{\vdash d \Rightarrow \Gamma \quad \Gamma \vdash \tau\ \texttt{bf} \quad \Gamma, x : \tau \vdash e : \tau}{\vdash d, \texttt{let rec}\ x : \tau = e \Rightarrow \Gamma, x : \tau}$$

3 Travail demandé

Vous devez étendre votre compilateur en lui ajoutant la phase d'analyse sémantique. Au delà des options déjà implantées dans la première partie, votre compilateur doit implanter également l'option suivante :

-no-code : s'arrête avant la production de code (effectue les analyses lexicale, syntaxique et sémantique).

En cas d'erreur de typage, `camlette` doit afficher un message sur la sortie standard d'erreur indiquant la position de l'erreur dans le fichier original et expliquant au mieux la nature de l'erreur (en français ou en anglais, peu importe). Exemples :

```
File "toto.ml", line 10, characters 24-29: Type error.
Unbound variable x.
```

```
File "toto.ml", line 10, characters 24-29: Type error.
This expression has type int but is used with type bool.
```