Projet Compilation : PreScript

Deuxième partie, cahier des charges
À rendre pour le lundi 2/12 à 10h00

Maîtrise d'Informatique 2002-2003

1 Compilation de votre projet

Votre projet sera rendu sous forme d'une archive tar compressée (option "z"), appelée *vos_noms*`.tgz` qui contiendra un répertoire appelé *vos_noms* (exemple : `paulin-miquel.tgz`). Dans ce répertoire se trouveront 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 `prescript`. 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 PreScript

Dans cette section, x, f et l représentent des identificateurs de variables, de fonctions et de labels respectivement.

2.1 Types et environnements

Les types de base (*base*) de PreScript sont `unit`, `bool`, `int`, `float`, `char`, `string` et `path`. La grammaire des types est :

$$\tau ::= \mathit{base} \mid x \mid \tau\ \texttt{array} \mid \{l_1 : \tau_1; \cdots; l_n : \tau_n\}$$

Dans l'environnement de PreScript, il y a deux espaces de noms : un pour les noms de variables et de fonctions, et un pour les noms de types. On définit l'environnement par la grammaire suivante :

$$\Gamma ::= \epsilon \mid \Gamma, x : \tau \mid \Gamma, \texttt{type}\ x = \tau \mid \Gamma, \texttt{fun}\ f(\tau_1, \cdots, \tau_n) : \tau$$

Étant donné un environnement Γ, $\mathit{dom}(\Gamma)$ est l'ensemble de ses noms de variables ou de fonctions, $\mathit{tdom}(\Gamma)$ est l'ensemble des ses noms de types.

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

$$\frac{\tau \in \{\texttt{bool}, \texttt{int}, \texttt{float}, \texttt{char}, \texttt{string}, \texttt{path}\}}{\Gamma \vdash \tau\ \texttt{bf}} \qquad \frac{(\texttt{type}\ x = \tau) \in \Gamma}{\Gamma \vdash x\ \texttt{bf}} \qquad \frac{\Gamma \vdash \tau\ \texttt{bf}}{\Gamma \vdash \tau\ \texttt{array bf}}$$

$$\frac{\Gamma \vdash \tau_1\ \texttt{bf} \quad \cdots \quad \Gamma \vdash \tau_n\ \texttt{bf}}{\Gamma \vdash \{l_1 : \tau_1, \cdots, l_n : \tau_n\}\ \texttt{bf}}\ \text{pour } 1 \leq i, j \leq n,\ i \neq j \Rightarrow l_i \neq l_j$$

Le jugement $\Gamma \vdash \tau \preccurlyeq \tau'$ signifie que τ est un sous-type de τ' dans l'environnement Γ. Il se dérive par les règles suivantes :

$$\frac{}{\Gamma \vdash \texttt{int} \preccurlyeq \texttt{float}} \qquad \frac{}{\Gamma \vdash \tau \preccurlyeq \tau} \qquad \frac{\Gamma \vdash \tau_1 \preccurlyeq \tau_2 \quad \Gamma \vdash \tau_2 \preccurlyeq \tau_3}{\Gamma \vdash \tau_1 \preccurlyeq \tau_3}$$

$$\frac{(\texttt{type}\ x = \tau_1) \in \Gamma \quad \Gamma \vdash \tau_1 \preccurlyeq \tau_2}{\Gamma \vdash x \preccurlyeq \tau_2} \qquad \frac{(\texttt{type}\ x = \tau_2) \in \Gamma \quad \Gamma \vdash \tau_1 \preccurlyeq \tau_2}{\Gamma \vdash \tau_1 \preccurlyeq x}$$

$$\frac{\Gamma \vdash \tau_1 \preccurlyeq \tau_2}{\Gamma \vdash \tau_1\ \texttt{array} \preccurlyeq \tau_2\ \texttt{array}} \qquad \frac{\Gamma \vdash \tau_1 \preccurlyeq \sigma_1 \quad \cdots \quad \Gamma \vdash \tau_n \preccurlyeq \sigma_n}{\Gamma \vdash \{l_1 : \tau_1; \cdots; l_n : \tau_n\} \preccurlyeq \{l_1 : \sigma_1; \cdots; l_n : \sigma_n\}}$$

On dira que τ est un type *numérique* lorsque $\tau \preccurlyeq$ `float` (dans le contexte courant). De même, on dira que $\tau \equiv \tau'$ lorsque $\tau \preccurlyeq \tau'$ et $\tau' \preccurlyeq \tau$ (dans le contexte courant).

Nous aurons également besoin de la fonction $\sup_\Gamma$ qui étant donné deux types τ_1 et τ_2 calcule le sup de ces deux types (lorsqu'il existe) pour la relation de sous-typage $\preccurlyeq$ définie précédemment. Cette fonction peut-être définie de la façon suivante :

- $\sup_\Gamma(\tau,\tau) := \tau$
- $\sup_\Gamma(\texttt{int},\texttt{float}) = \sup_\Gamma(\texttt{float},\texttt{int}) := \texttt{float}$
- Si $\sup_\Gamma(\tau,\tau')$ est défini alors $\sup_\Gamma(\tau\ \texttt{array},\tau'\ \texttt{array}) := (\sup_\Gamma(\tau,\tau'))\ \texttt{array}$
- Si $\forall i\ \sup_\Gamma(\tau_i,\tau_i')$ est défini, alors :
  $\sup_\Gamma(\{l_1 : \tau_1; \cdots; l_n : \tau_n\}, \{l_1 : \tau_1'; \cdots; l_n : \tau_n'\}) := \{l_1 : \sup_\Gamma(\tau_1,\tau_1'); \cdots; l_n : \sup_\Gamma(\tau_n,\tau_n')$
- Si $(\texttt{type}\ x = \tau) \in \Gamma$ alors $\sup_\Gamma(x,\tau') = \sup_\Gamma(\tau',x) := \sup_\Gamma(\tau,\tau')$.
- La fonction $\sup_\Gamma$ est indéfinie dans tous les autres cas.

2.2 Typage des expressions

Le typage des expressions est assuré par deux jugements :

1. Un jugement d'inférence noté $\Gamma \vdash e : \tau$, qui infère le type τ d'une expression e dans le contexte Γ ;
2. Un jugement de contrainte noté $\Gamma \vdash e :> \tau$, qui vérifie que τ est un type correct pour l'expression e dans le contexte Γ (mais pas forcément le plus petit type).

La distinction entre ces deux jugements est rendue nécessaire par le fait que le type d'une expression n'est pas nécessairement unique à cause du sous-typage et des définitions de type. Le jugement de contrainte s'exprime en fonction du jugement d'inférence par la règle suivante :

$$\frac{\Gamma \vdash e : \tau \quad \Gamma \vdash \tau \preccurlyeq \tau'}{\Gamma \vdash e :> \tau'}$$

Le jugement $\Gamma \vdash e : \tau$ est défini à l'aide des règles suivantes :

$$\frac{c \text{ constante de type } \tau}{\Gamma \vdash c : \tau} \qquad \frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau} \qquad \frac{\texttt{fun}\ f(\tau_1,\cdots,\tau_n) : \tau \in \Gamma \quad \Gamma \vdash e_1 :> \tau_1 \quad \cdots \quad \Gamma \vdash e_n :> \tau_n}{\Gamma \vdash f(e_1,\cdots,e_n) : \tau}$$

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

$$\frac{op \in \{\texttt{+},\ \texttt{-},\ \texttt{*}\} \quad \Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ op\ e_2 : \sup_\Gamma(\tau_1,\tau_2)}\ \tau_1,\tau_2 \text{ numériques} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ \texttt{/}\ e_2 : \texttt{float}}\ \tau_1,\tau_2 \text{ numériques}$$

$$\frac{op \in \{\texttt{mod},\ \texttt{div}\} \quad \Gamma \vdash e_1 :> \texttt{int} \quad \Gamma \vdash e_2 :> \texttt{int}}{\Gamma \vdash e_1\ op\ e_2 : \texttt{int}}$$

$$\frac{op \in \{\texttt{<},\ \texttt{<=},\ \texttt{>},\ \texttt{>=}\} \quad \Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ op\ e_2 : \texttt{bool}}\ \tau_1,\tau_2 \text{ numériques ou } \tau_1 \equiv \tau_2 \equiv \texttt{char} \text{ ou } \tau_1 \equiv \tau_2 \equiv \texttt{string}$$

$$\frac{op \in \{\texttt{=},\ \texttt{<>}\} \quad \Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash e_1\ op\ e_2 : \texttt{bool}}\ \tau_1,\tau_2 \text{ types de base et } \sup_\Gamma(\tau_1,\tau_2) \text{ défini}$$

$$\frac{}{\Gamma \vdash \texttt{[]} : \texttt{unit array}} \qquad \frac{\Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash \texttt{[}e_1\ \texttt{;}\ \cdots\ \texttt{;}e_n\texttt{]} : \sup_\Gamma(\tau_1,\ldots,\tau_n)\ \texttt{array}}\ (n \geq 1)$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash \texttt{\{}l_1\texttt{=}e_1\ \texttt{;}\ \cdots\ \texttt{;}\ l_n\texttt{=}e_n\texttt{\}} : \texttt{\{}l_1 \texttt{:} \tau_1\ \texttt{;}\ \cdots\ \texttt{;}l_n \texttt{:} \tau_n\texttt{\}}}\ (\forall i \neq j \quad l_i \neq l_j)$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{var}\ x = e_1\ \texttt{in}\ e_2 : \tau_2} \qquad \frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 :> \tau}{\Gamma \vdash e_1 \texttt{:=} e_2 : \texttt{unit}} \qquad \frac{\Gamma \vdash e_1 :> \texttt{unit} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\texttt{;}e_2 : \tau}$$

$$\frac{\Gamma \vdash e_1 :> \texttt{bool} \quad \Gamma \vdash e_2 :> \texttt{unit}}{\Gamma \vdash \texttt{if } e_1 \texttt{ then } e_2 : \texttt{unit}} \qquad \frac{\Gamma \vdash e_1 :> \texttt{bool} \quad \Gamma \vdash e_2 : \tau_2 \quad \Gamma \vdash e_3 : \tau_3}{\Gamma \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : \sup_\Gamma(\tau_2, \tau_3)}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \Gamma \vdash e_3 : \tau_3 \quad \Gamma, x : \sup_\Gamma(\tau_1, \tau_2, \tau_3) \vdash e_4 :> \texttt{unit}}{\Gamma \vdash \texttt{for } x \texttt{ = } e_1 \texttt{ to } e_2 \texttt{ step } e_3 \texttt{ do } e_4 \texttt{ done} : \texttt{unit}} \; \tau_1, \tau_2, \tau_3 \text{ numériques}$$

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma \vdash e_2 : \tau_2 \quad \Gamma, x : \sup_\Gamma(\tau_1, \tau_2) \vdash e_3 :> \texttt{unit}}{\Gamma \vdash \texttt{for } x \texttt{ = } e_1 \texttt{ to } e_2 \texttt{ do } e_3 \texttt{ done} : \texttt{unit}} \; \tau_1, \tau_2 \text{ numériques ou } \tau_1 \equiv \tau_2 \equiv \texttt{char}$$

$$\frac{\Gamma \vdash e_1 :> \texttt{bool} \quad \Gamma \vdash e_2 :> \texttt{unit}}{\Gamma \vdash \texttt{while } e_1 \texttt{ do } e_2 \texttt{ done} : \texttt{unit}} \qquad \frac{\Gamma \vdash e : \tau \quad \tau \equiv \{l_1 : \tau_1 \texttt{ ; } \cdots \texttt{ ; } l_n : \tau_n\}}{\Gamma \vdash e.l_i : \tau_i} \; (1 \leq i \leq n)$$

$$\frac{\Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 :> \texttt{int} \quad \tau \equiv \tau' \texttt{ array}}{\Gamma \vdash e_1\texttt{.(}e_2\texttt{)} : \tau'} \qquad \frac{\Gamma \vdash e_1 :> \texttt{string} \quad e_2 :> \texttt{int}}{\Gamma \vdash e_1\texttt{.[}e_2\texttt{]} : \texttt{char}}$$

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 :

$$\frac{}{\epsilon \Rightarrow \epsilon} \qquad \frac{\vdash d \Rightarrow \Gamma \quad \Gamma \vdash e : \tau}{\vdash d, \texttt{var } x = e \Rightarrow \Gamma, x : \tau} \qquad \frac{\vdash d \Rightarrow \Gamma \quad \Gamma \vdash \tau \texttt{ bf} \quad x \notin \mathit{tdom}(\Gamma)}{\vdash d, \texttt{type } x = \tau \Rightarrow \Gamma, \texttt{type } x = \tau}$$

$$\frac{\vdash d \Rightarrow \Gamma \quad \forall i \; \Gamma \vdash \tau_i \texttt{ bf} \quad \Gamma \vdash \tau \texttt{ bf} \quad \Gamma, x_1 : \tau_1, \cdots, x_n : \tau_n, \texttt{fun } f(\tau_1, \cdots, \tau_n) : \tau \vdash e :> \tau}{\vdash d, \texttt{fun } f(x_1 : \tau_1, \cdots, x_n : \tau_n) : \tau = e \Rightarrow \Gamma, \texttt{fun } f(\tau_1, \cdots, \tau_n) : \tau}$$

2.4 Primitives polymorphes

Le langage PreScript comporte deux primitives polymorphes `array_make` et `array_length` dont les règles de typage sont les suivantes :

$$\frac{\Gamma \vdash e_1 :> \texttt{int} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \texttt{array\_make(}e_1\texttt{, } e_2\texttt{)} : \tau \texttt{ array}}$$

$$\frac{\Gamma \vdash e : \tau \quad \tau \equiv \tau' \texttt{ array}}{\Gamma \vdash \texttt{array\_length(}e\texttt{)} : \texttt{int}}$$

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, `prescript` 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.pre", line 10, characters 24-29: Type error.
Unbound variable x.

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