TD Types

Dans la première partie de cette feuille, on reprend comme base le mini-langage d'expressions avec arithmétique et tableaux utilisé dans le cours. Rappel des règles de typage :

$$\frac{}{\Gamma \vdash n : \texttt{int}} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \texttt{ - } e_2 : \texttt{int}} \qquad \frac{}{\Gamma \vdash x : \Gamma(x)}$$

$$\frac{\Gamma \vdash e_1 : \tau \qquad \dots \qquad \Gamma \vdash e_k : \tau}{\Gamma \vdash \texttt{[}e_1\texttt{, } \dots\texttt{, } e_k\texttt{]} : \tau\texttt{[]}} \qquad \frac{\Gamma \vdash e_1 : \tau\texttt{[]} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1\texttt{[}e_2\texttt{]} : \tau}$$

Exercice 1 (Expressions typables) Peut-on trouver un type pour la variable `t` tel que les expressions suivantes soient typables ? Si oui, préciser les types possibles et donner une dérivation, et si non expliquer l'incohérence.

1. `t[1] + 2`
2. `t[t[3]]`
3. `t[3][4]`
4. `t[3][t[4]]`
5. `[t[1], t[3]]`
6. `[t, t[0]]`

□

Exercice 2 (Paires) On veut étendre le langage avec une notion de paire. On ajoute à la syntaxe des expressions les trois constructions suivantes :

— $\texttt{(}e_1\texttt{, } e_2\texttt{)}$ pour la construction d'une paire avec les valeurs des expressions e_1 et e_2,
— $\texttt{fst(}e\texttt{)}$ pour l'extraction de la première composante de la paire obtenue en évaluant e,
— $\texttt{snd(}e\texttt{)}$ pour l'extraction de la deuxième composante,

On étend également la syntaxe des types avec la construction :

— $\tau_1 \times \tau_2$ pour le type des paires dont la première composante a le type τ_1 et la deuxième composante a le type τ_2.

Donner des règles de typage pour ce langage étendu. □

Exercice 3 (Booléens) On veut étendre le langage avec des booléens. On ajoute :

— les constantes `true` et `false`,
— les opérations binaires $e_1 \texttt{ < } e_2$, $e_1 \texttt{ = } e_2$ et $e_1 \texttt{ \&\& } e_2$,
— une expression conditionnelle $e_0 \texttt{?} e_1 \texttt{:} e_2$.

Donner des règles de typage pour ce langage étendu. *Attention à ce que chaque opération soit aussi permissive que possible, mais pas plus que cela !* Que penser d'une expression de la forme $e_0\texttt{?}e_1$ sans résultat alternatif ? (vous pouvez réfléchir à ce qui se passe en caml avec une expression similaire) □

Exercice 4 (Opérateurs paresseux) Dans le cadre de l'extension du langage avec les booléens, définir par des équations récursives une fonction F qui transforme une expression e en éliminant l'opérateur `&&` : chaque opération $e_1 \texttt{ \&\& } e_2$ doit être remplacée par une expression conditionnelle.

Démontrer que si $\Gamma \vdash e : \tau$, alors $\Gamma \vdash F(e) : \tau$. □

Exercice 5 (Préservation du typage par substitution) On définit l'opération de substitution $e^{\{x \leftarrow e'\}}$ de la variable x par l'expression e' dans l'expression e par les équations suivantes.

$$\begin{array}{rcl}
n^{\{x\leftarrow e'\}} & = & n \\
(e_1 \texttt{ - } e_2)^{\{x\leftarrow e'\}} & = & e_1^{\{x\leftarrow e'\}} \texttt{ - } e_2^{\{x\leftarrow e'\}} \\
y^{\{x\leftarrow e'\}} & = & \begin{cases} e' & \text{si } x = y \\ y & \text{si } x \neq y \end{cases} \\
(e_1\texttt{[}e_2\texttt{]})^{\{x\leftarrow e'\}} & = & e_1^{\{x\leftarrow e'\}}\texttt{[}e_2^{\{x\leftarrow e'\}}\texttt{]} \\
\texttt{[}e_1\texttt{, } \dots\texttt{, } e_k\texttt{]}^{\{x\leftarrow e'\}} & = & \texttt{[}e_1^{\{x\leftarrow e'\}}\texttt{, } \dots\texttt{, } e_k^{\{x\leftarrow e'\}}\texttt{]}
\end{array}$$

Montrer que si on a les jugements $\Gamma \vdash e' : \tau'$ et $\Gamma, x : \tau' \vdash e : \tau$, alors on a également $\Gamma \vdash e^{\{x\leftarrow e'\}} : \tau$. □

Dans cette seconde partie, on regarde le typage d'un mini-langage fonctionnel, comportant de l'arithmétique, des variables locales et des fonctions. Voici les grammaires des expressions et des types de ce langage :

e	$::=$	n	constante entière
	$\mid$	e `-` e	opération binaire
	$\mid$	x	variable
	$\mid$	`let` x `=` e `in` e	définition locale
	$\mid$	`fun` x `->` e	fonction
	$\mid$	$e\ e$	application
τ	$::=$	`int`	entiers
	$\mid$	$\tau \to \tau$	fonctions

Et voici ses règles de typage :

$$\frac{}{\Gamma \vdash n : \texttt{int}} \qquad \frac{\Gamma \vdash e_1 : \texttt{int} \qquad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1 \texttt{ - } e_2 : \texttt{int}}$$

$$\frac{}{\Gamma \vdash x : \Gamma(x)} \qquad \frac{\Gamma \vdash e_1 : \sigma \qquad \Gamma, x : \sigma \vdash e_2 : \tau}{\Gamma \vdash \texttt{let } x \texttt{ = } e_1 \texttt{ in } e_2 : \tau}$$

$$\frac{\Gamma, x : \sigma \vdash e : \tau}{\Gamma \vdash \texttt{fun } x \texttt{ -> } e : \sigma \to \tau} \qquad \frac{\Gamma \vdash e_1 : \sigma \to \tau \qquad \Gamma \vdash e_2 : \sigma}{\Gamma \vdash e_1\ e_2 : \tau}$$

Exercice 6 (Expressions typables) Les expressions suivantes sont-elles typables ? Si oui donner une dérivation, et si non expliquer l'incohérence.

1. `let f = fun x -> x+1 in f (f 1)`
2. `let f = fun x -> x+1 in f f`
3. `let f = fun x -> fun y -> x in f 1`
4. `let f = fun x -> fun y -> x in f 1 2 3`
5. `let f = fun x -> fun y -> x in f (fun z -> z) 2 3`
6. `fun x -> fun y -> fun z -> x z (y z)`

□

Exercice 7 (Point fixe) On veut étendre le langage avec une définition récursive de variable locale :

— let rec $x = e_1$ in e_2

Donner une règle de typage pour cette nouvelle construction, et une dérivation de typage pour l'expression suivante :

```
let rec f =
  fun x -> 1 + f x
in
  f 0
```

□

Exercice 8 (Monotonie) Pour deux environnements Γ et Δ, on note $\Gamma \subseteq \Delta$ si pour tout $x \in \mathrm{dom}(\Gamma)$ on a $x \in \mathrm{dom}(\Delta)$ et $\Gamma(x) = \Delta(x)$.

Montrer que si $\Gamma \vdash e : \tau$ et $\Gamma \subseteq \Delta$, alors $\Delta \vdash e : \tau$. □