# TD1 : Systèmes de types

**Description du langage**

Expressions $e$ ::= $x$ | fun $x \to e$ | $(e_1\ \ e_2)$ | $i$ | $(e_1 + e_2)$ | $(e_1 < e_2)$ | not $e$
Instructions $c$ ::= while $e$ do $c$ | print $e$ | if $e$ then $c_1$ else $c_2$ | $\{c_1; c_2\}$

**Typage** Types : `T ::= int | bool | T->T | instr`

$$\frac{}{\Gamma,\ \texttt{x:T} \vdash \texttt{x:T}}\ (\text{Var}) \qquad \frac{\Gamma,\ \texttt{x:T1} \vdash \texttt{e:T2}}{\Gamma \vdash \texttt{fun x -> e : T1->T2}}\ (\text{Fun}) \qquad \frac{\Gamma \vdash \texttt{e1:T1 -> T2} \qquad \Gamma \vdash \texttt{e2:T1}}{\Gamma \vdash \texttt{(e1 e2): T2}}\ (\text{Call})$$

$$\frac{}{\Gamma \vdash \texttt{i : int}}\ (\text{Int}) \qquad \frac{\Gamma \vdash \texttt{e1 : int} \qquad \Gamma \vdash \texttt{e2 : int}}{\Gamma \vdash \texttt{(e1 + e2) : int}}\ (\text{Add})$$

$$\frac{\Gamma \vdash \texttt{e1 : T} \qquad \Gamma \vdash \texttt{e2 : T}}{\Gamma \vdash \texttt{(e1 < e2) : bool}}\ (\text{Lt}) \qquad \frac{\Gamma \vdash \texttt{e : bool}}{\Gamma \vdash \texttt{not e : bool}}\ (\text{Not})$$

**Instructions**

$$\frac{\Gamma \vdash \texttt{e : bool} \qquad \Gamma \vdash \texttt{c : instr}}{\Gamma \vdash \texttt{while e do c : instr}}\ (\text{While}) \qquad \frac{\Gamma \vdash \texttt{e : int}}{\Gamma \vdash \texttt{print e : instr}}\ (\text{Print})$$

$$\frac{\Gamma \vdash \texttt{e : bool} \qquad \Gamma \vdash \texttt{c1 : T} \qquad \Gamma \vdash \texttt{c2 : T}}{\Gamma \vdash \texttt{if e then c1 else c2 : T}}\ (\text{If})$$

$$\frac{\Gamma \vdash \texttt{c1 : instr} \qquad \Gamma \vdash \texttt{c2 : instr}}{\Gamma \vdash \texttt{(c1 ; c2) : instr}}\ (\text{Seq}) \qquad \frac{\Gamma,\ \texttt{x:T} \vdash \texttt{e : T}}{\Gamma \vdash \texttt{x:= e : instr}}\ (\text{Aff})$$

**Exercice 1** **Typage d'expressions simples**

Déterminez le type des expressions suivantes en utilisant les règles de typage fournies. Si le type dépend de l'environnement, précisez-le.

a) (1+(2+3))
b) (x+y)
c) `fun x -> (x + 1)`
d) `fun x -> if (x < 5) then (x + 1) else (x + 2)`
e) `fun x -> not (x < 5)`
f) `(fun x -> x+x) (y+2)`
g) `(f x) + 1`

**Questions :**

1. Quel est le type de chaque expression ?
2. Décrivez l'arbre de dérivation de typage pour chaque expression.

**Exercice 2** **Typage d'instructions composées**

Considérez les instructions suivantes :

a) `while (x < 10) do print x; x := (x + 1)`

b) `if (x < 5) then print x else print (x + 1)`

**Questions :**

1. Vérifiez que ces instructions sont bien typées en utilisant les règles de typage.
2. Expliquez comment les règles de typage s'appliquent à chaque composant de l'instruction.

**Exercice 3** **Typage avec des fonctions d'ordre supérieur**

Considérez la fonction suivante :

```
fun f -> fun x -> f (x + 1)
```

**Questions :**

1. Déterminez le type de cette fonction.
2. Expliquez comment les règles de typage s'appliquent à cette fonction.

**Exercice 4** **Extension du langage**

Étendez le langage en ajoutant les constructions suivantes :

a) Boucle `for` avec la syntaxe :`for x from e1 to e2 do c`

b) Listes et opérations de base avec les syntaxes :

```
nil
cons e1 e2
head e
tail e
```

**Questions :**

1. Définissez la syntaxe de chaque nouvelle construction.
2. Proposez des règles de typage pour ces constructions.
3. Donnez un exemple d'utilisation de chaque construction et vérifiez leur typage.