# TP numéro 2

**Sémantique des langages de programmation, ENSIIE**

Semestre 5, 2023–24

Le but de ce TP est d'implémenter en OCaml les sémantiques opérationnelle à petits pas et dénotationnelle de IMP, ce qui nous fournira un interpréteur pour ce langage.

Pour ceux qui n'aurait pas assez avancé au dernier TP, on trouvera à l'adresse `http://www.ensiie.fr/~guillaume.burel/download/imp.ml` la sémantique opérationnelle à petit pas sur les expressions arithmétiques.

Ce TP est à rendre pour le 25 octobre sur `exam.ensiie.fr` dans le dépôt `prru_isl_tp_2023`.

## Exercice 1 : Sémantique opérationnelle

### Exercice 1.1 : Expressions booléennes

On considère maintenant les expressions booléennes, représentées en OCaml par le type :

```
type exp_bool =
    Bool of bool
  | Inf of exp_arith * exp_arith
  | Egal of exp_arith * exp_arith
  | Not of exp_bool
  | Or of exp_bool * exp_bool
  | And of exp_bool * exp_bool
```

1. Définir et implémenter la sémantique opérationnelle à petits pas des expressions booléennes.
2. Tester cette sémantique sur l'expression not $(x = 0$ and $x \leq (7/z))$ pour la valuation $\{x \mapsto 1, y \mapsto 3, z \mapsto 2\}$ puis dans la valuation $\{x \mapsto 0, z \mapsto 0\}$.

### Exercice 1.2 : IMP

Pour définir la sémantique opérationnelle de IMP, on a besoin de pouvoir manipuler les valuations. Pour cela, on va maintenant utiliser des maps en OCaml. On redéfinit donc le type des valuation de la façon suivante :

```
module StringMap = Map.Make(String)

type valuation = int StringMap.t
```

1. Écrire une fonction `affiche_valuation` de type `valuation -> unit` qui affiche une valuation. On pourra utiliser la fonction `StringMap.iter`.
2. Modifier les fonctions écrites dans les parties précédentes pour prendre en compte le nouveau type des valuations. La fonction `StringMap.find` sera utile.

On représente en OCaml les programmes de IMP par le type

```
type imp =
    Skip
  | Aff of string * exp_arith
  | Seq of imp * imp
  | If of exp_bool * imp * imp
  | While of exp_bool * imp
```

3. Implémenter la sémantique opérationnelle à petit pas de IMP. La fonction `StringMap.add` permettra de faire l'opération $\sigma[x \leftarrow n]$.
4. Tester sur l'exemple
   $x := 24; y := 36;$ while not $x = y$ do if $x \leq y$ then $y := y - x$ else $x := x - y$

## Exercice 2 : Sémantique dénotationnelle

On cherche maintenant à donner une sémantique opérationnelle à IMP. On notera qu'on a généralisé le type des valeurs pour pouvoir l'utiliser pour les expressions booléennes et le programmes :

```
type 'a valeur = V of 'a | Err
```

### Exercice 2.1 : Expressions booléennes

5. Implémenter la sémantique dénotationnelle des expressions booléennes. On utilisera la version qui donne une sémantique coupe-circuit au `and` et au `or`.
6. Tester avec les mêmes expression et valuations que ci-dessus.

### Exercice 2.2 : IMP

7. Définir (sur papier) une sémantique dénotationnelle pour IMP.
   Indication : on pourra interpréter un programme IMP comme une fonction qui prend une valuation (celle avant l'exécution du programme) et qui retourne soit la valuation après exécution du programme, soit une erreur.
8. Implémenter cette sémantique.
9. Tester avec le même exemple que ci-dessus.