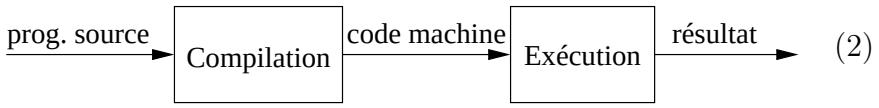
Programmation applicative – L2
TD 1 : Premiers concepts d'évaluation

Guillaume Artignan {artignan@lirmm.fr}
Annie Chateau {chateau@lirmm.fr}
Hervé Dicky {dicky@lirmm.fr}
Christophe Dony {dony@lirmm.fr}
Bruno Paiva Lima da Silva {bplsilva@gmail.com}

1 Introduction

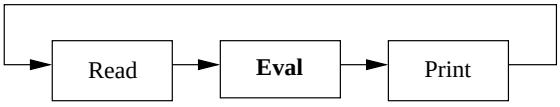
Scheme est un langage de la famille Lisp crée en 1975 par des enseignants (Sussman et Steele entre autres) du célèbre MIT. C'est un langage de programmation *fonctionnel*[1] dont le but est plus universitaire qu'industriel.

Il existe 2 grandes familles de langages de programmation, langage interprété et langage compilé :



- Langage dit *interprété* (1) (ex : Scheme, Lisp). C'est à dire qu'un programme, l'interpréteur, renvoie directement la valeur d'une expression écrite dans le langage.
- Langage dit *compilé* (2) (ex : C++, Java). La compilation sert à transformer le programme source en code machine spécifique et directement exécutable par une machine.

Programmer en Scheme consiste à donner des expressions (respectant la syntaxe de Scheme) à évaluer à l'interpréteur Scheme : la boucle d'interaction REP.



Il existe plusieurs interpréteurs Scheme disponibles en ligne. Nous utiliserons en TP l'interpréteur MzScheme fourni par le logiciel PLT DrScheme.

1. Se dit d'un style de programmation dans lequel on n'utilise que des fonctions comme en mathématique sans utiliser le mécanisme d'affectation (modification de la valeur d'une variable). En opposition à programmation *impérative*, qui se dit d'un style de programmation dans lequel on utilise des procédures dont les instructions modifient, par l'affectation en mémoire, des variables.

2 Préambule : Algèbre de Boole

Exercice 1 *Donner les valeurs des expressions suivantes (on note V pour Vrai et F pour Faux) :*

```
V et ( non F )
( V ou F ) et (( non V ) et F )
V ou X
( V ou X ) ou ( F et Y)
( non ( V et X ))
(( non V ) ou ( non X ))
```

Construire une table de vérité pour les quatre dernières expressions.

3 Syntaxe de Scheme (notation parenthésée préfixée et S-expressions)

Toute expression syntaxiquement correcte de Scheme est une *S-expression*. Une S-expression est :

- soit un nombre : `1, 2, 3`
- soit un symbole : `x, square, +, #t, #f`
- soit une liste (éventuellement vide) de S-expressions encadrées par 2 parenthèses `(S-expression ... S-expression)`

Scheme utilise la notation parenthésée préfixée : `(opérateur opérande`$_1$ `...` `opérande`$_k$`)`.

- Le parenthésage permet de ne pas avoir à se soucier des priorités entre les opérateurs.
- La notation préfixée à l'avantage de ne pas répéter l'opérateur lorsque il y a plusieurs opérandes.

`36 + 14`	$\rightsquigarrow$	`(+ 36 14)`
`3 × (2 + 4)`	$\rightsquigarrow$	`(* 3 (+ 2 4))`
`1 + 2 + 3 + 4 + 5`	$\rightsquigarrow$	`(+ 1 2 3 4 5)`

Exercice 2 *Transformer les expressions suivantes en notation préfixée :*

- $5 - 4$
- $2 - 6 + 10$
- $2 + 5 \times 3$
- $3 \times 5 - 7/2$
- $5 + 3 \times (4 - 2) + 5/(2 + 1)$

Exercice 3 *En utilisant les noms de fonctions Scheme :* `+, -, *, /, <, =, sqrt` ($\sqrt{\ldots}$)`, and, or, not`*, transformer les expressions algébriques suivantes en expressions Scheme :*

$1 - (2 - 3)$
$3 + 7 \times (2 + 1)$
$(a + 1) \neq b$
$\frac{a \times x + b}{c \times x + d}$

$(1 - 2) - 3$
$a \times x^2 + b \times x + c$
$(a < b)$ *ou* $(a = b^2)$
$\sqrt{b^2 - 4 \times a \times c}$

Exercice 4 *Les expressions Scheme suivantes sont-elles syntaxiquement correctes ? Si ce n'est pas le cas, préciser ce qui ne va pas.*

```
(+ 1 (* 2 3))
+
(+ 2 *)
(+ (/ 2 3) 1))(+ a b)
(= 1)
```

Exercice 5 *Donner la forme préfixée des expressions suivantes :*

a) $2 + 4x^2 - 3x^3$ b) $\frac{sin(x+y)\times cos(x-y)}{1+cos(x+y)}$

4 Les formes spéciales `define` et `lambda`

Une expression Scheme peut être de 3 types :

Expression atomique	Nombre, symbole
Forme spéciale	Définition (`define`)
	Lambda expression (`lambda`)
	Condition (`if, cond`)
	Quotation (`quote`)
	Sequence (`begin`)
	Affectation (`set!`)
Application de fonction	`(opérateur opérande`$_1$ `...` `opérande`$_k$`)`

La forme spéciale `define` permet de faire des abstractions. C'est à dire nommer par des symboles des données ou des fonctions. Ces noms sont stockés dans une mémoire de façon à être réutilisés.

- Définition de données : `(define` $\langle symbole \rangle$ $\langle expression \rangle$`)`
- Définition de fonctions : `(define` $\langle symbole \rangle$ `(lambda (`$\langle param_1 \dots param_k \rangle$`))` $\langle corps \rangle$`))`

La forme spéciale `lambda` est fondamentale en Scheme, c'est elle qui permet de **construire des fonctions**. Remarque – Notez que la forme spéciale `define` ne renvoie pas de valeur.

Exercice 6 *Définir en Scheme les fonctions suivantes à l'aide de lambda-expressions : :*

- $f : x \mapsto x^3$
- $g : y \mapsto 3y + 7$
- $h : z \mapsto 3f(z) + g(z) + 1$
- *La fonction constante* $k2 : x \mapsto 2$.
- *La fonction identité* $id : x \mapsto x$.
- *La fonction proj2 :* $(x, y) \mapsto y$.
- *La fonction einstein :* $(u, v) \to \frac{u+v}{1+\frac{uv}{c^2}}$ *avec* $c = 300000$

Exercice 7 *Indiquer l'effet des fonctions suivantes :*

```
(define (f1) 0)
(define (f2 x) (* x x))
(define (f3 x y) (+ (f2 x) (f2 y)))
(define (distc x1 y1 x2 y2) (f3 (- x1 x2) (- y1 y2)))
```

Exercice 8 *Suivre le cheminement de Scheme sur l'exemple suivant. Que se passe-t-il à chacune de ces commandes ?*

```
> (define pi 3.14)
> pi
> (define (carre x) (* x x))
> (carre 5)
> (define (cercle r) (* pi (carre r)))
> (cercle 12)
```

5 La forme spéciale `if`

La forme spéciale `if` est une structure de contrôle fondamentale. Sa syntaxe est : `(if exp-test exp-alors exp-sinon)`. Lors de l'évaluation d'un `if`, il y a d'abord évaluation de `exp-test` et si son résultat est `#t` alors il y a évaluation de `exp-alors` sinon il y a évaluation de `exp-sinon`.

Exercice 9 *Définissez la fonction* `abs` *qui renvoie la valeur absolue d'un nombre. Écrivez ensuite la fonction* `care-div` *qui fait la division de 2 nombres en vérifiant au préalable que le diviseur n'est pas nul.*

6 Expressions et fonctions non récursives

Exercice 10 *Donner la fonction Scheme* `fahrenheit->celsius` *qui permet de convertir une température donnée en degré Fahrenheit en température Celsius en utilisant la formule suivante :*

$$F(C) = \frac{9}{5}(C + 40) - 40$$

Écrire une fonction `liquide-cel?` *pour savoir si une température donnée en degrés celsius correspond à un état de l'eau qui n'est ni glace ni vapeur.*

Donner une fonction `liquide-fah?` *pour savoir si une température donnée en degrés fahrenheit correspond à un état de l'eau qui n'est ni glace ni vapeur.*

Exercice 11 *Donner les fonctions* `puissance-2, puissance-4, puissance-6` *et* `puissance-9`*, qui, étant donné un argument x, calculent respectivement x^2, x^4, x^6 et x^9.*

Exercice 12 *Écrire les fonctions* `unité, dizaine` *et* `centaine` *dont les évaluations sont respectivement égales au chiffre des unités, au chiffre des dizaines et au chiffre des centaines du nombre donné comme paramètre.*

Exercice 13 *Écrire les fonctions* `min-2` *et* `max-2` *qui, étant donnés deux nombres, donnent respectivement le plus petit et le plus grand des deux nombres.*

Écrire les fonctions `min-3` *et* `max-3` *qui, étant donnés deux nombres, donnent respectivement le plus petit et le plus grand des trois nombres.*