(Séquence 9.2

Évaluation d'expression arithmétique

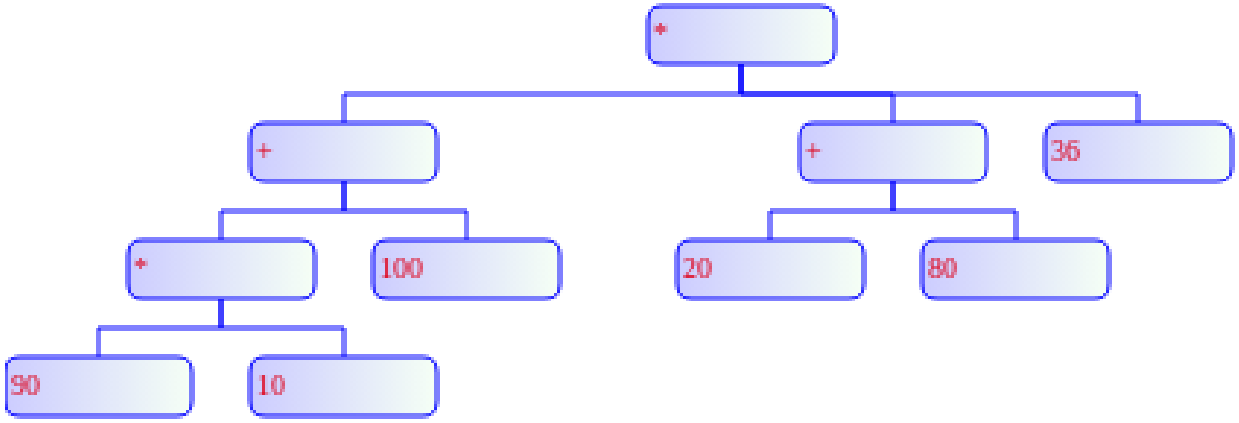
Expression arithmétique

Le but est d'évaluer des expressions arithmétiques, bien formées, dotées des seuls opérateurs `+` et `*`

- constante c'est-à-dire sans variable
  `(((90 * 10) + 100) * (20 + 80) * 36)`
- ou dans un **environnement** avec variables.
  `((90 * x) + y)`

Représentation expression arithmétique

- Visualisation et représentation par un arbre général (type `ExprArbre`)

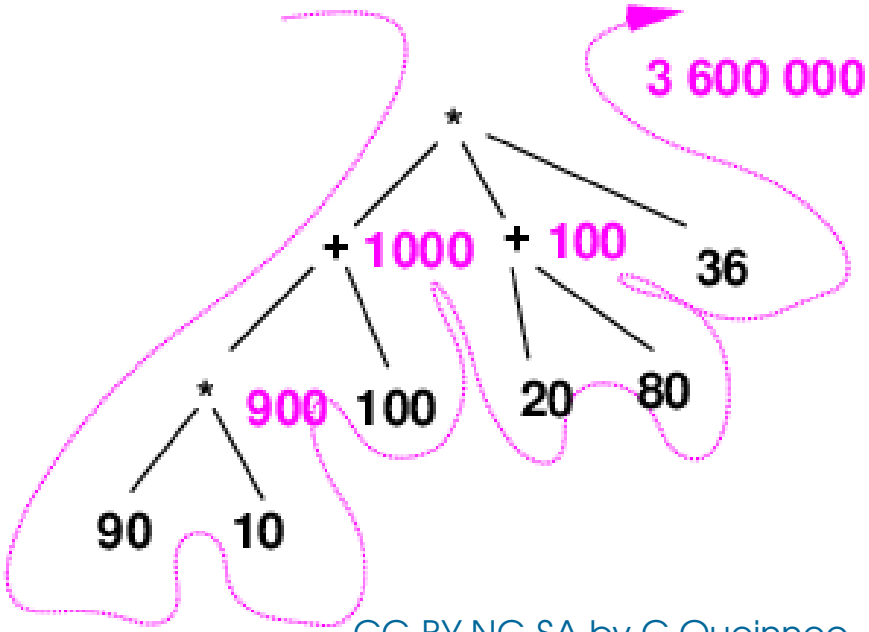


- Représentation par une S-expression (type `S-Expr`)

```
(* (+ (* 90 10) 100) (+ 20 80) 36)
(+ (* 90 x) y)
```

Définition de `evaluationExprArbre`

```
;;; evaluationExprArbre: ExprArbre-> Nombre
;;; (evaluationExprArbre G) rend la valeur de
;;; l'expression arithmetique représentée par G
(define (evaluationExprArbre G)
  (if (ag-feuille? G)
      (ag-etiquette G)
      (let ((args (map evaluationExprArbre
                       (ag-foret G))))
        (application (operation (ag-etiquette G))
                     args ))))
```



Définition de `operation`

```
;;; operation: Symbole -> Opération
;;; (operation operateur) rend la fonction
;;; correspondant au symbole operateur
(define (operation operateur)
  (cond ((equal? operateur '+) +)
        ((equal? operateur '*) *)
        (else (erreur 'operation
                      operateur
                      "pas défini"))))
```

Définition de `application`

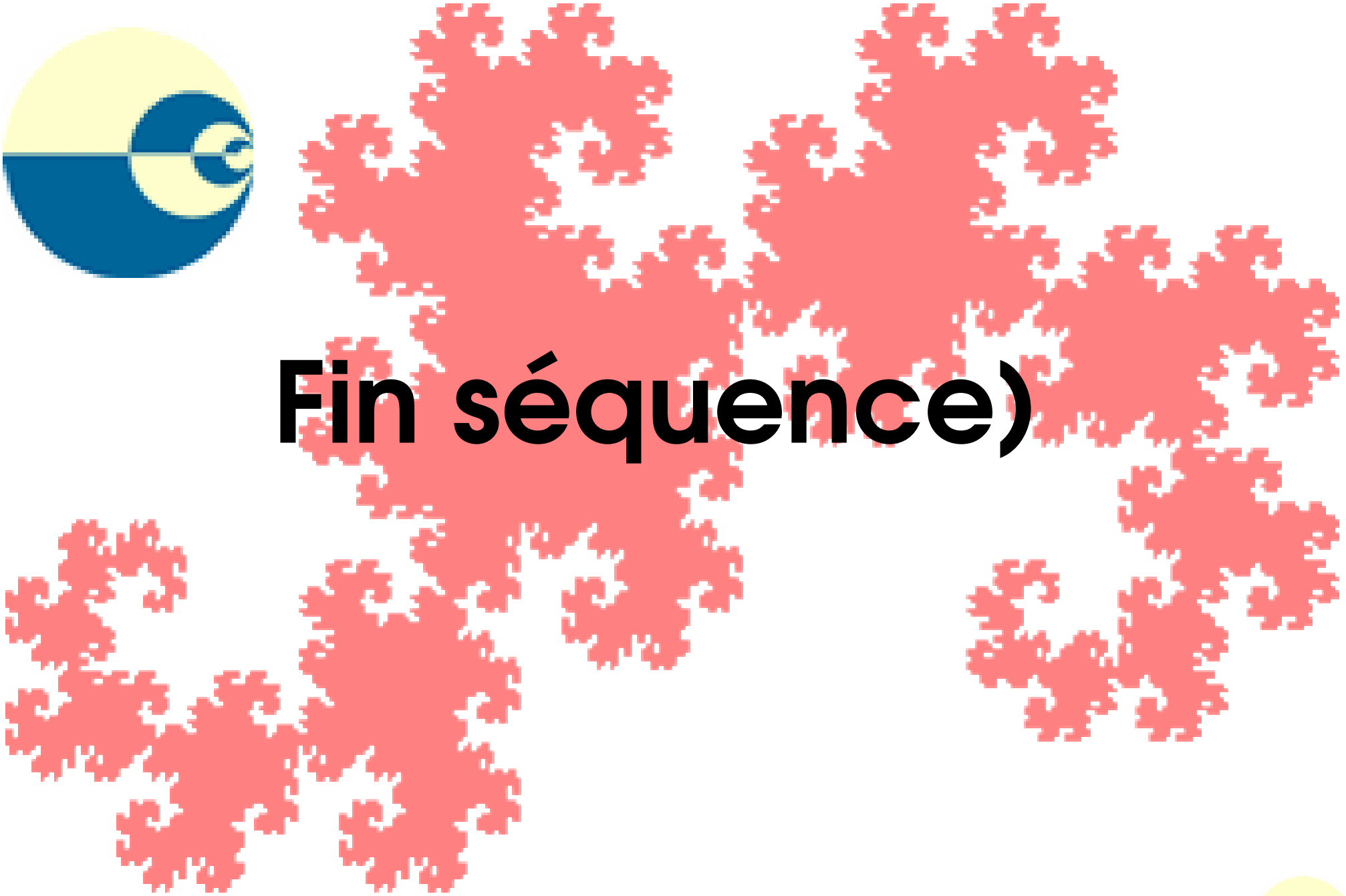
```
;;; application: Opération * LISTE[Nombre] -> Nombre
;;; (application op args) rend le résultat de
;;; l'application de l'opération op à la liste
;;; d'arguments args
;;; HYPOTHÈSE: args non vide
(define (application op args)
  (if (pair? (cdr args))
      (op (car args) (application op (cdr args)))
      (op (car args))))
```

Définition de `evaluation-sexpr`

Avec des Sexpressions plutôt que des arbres généraux, cela donne :

```
;;; evaluation-sexpr: S-Expr -> Nombre
;;; (evaluation-sexpr E) rend la valeur de
;;; l'expression arithmétique E
(define (evaluation-sexpr E)
  (if (pair? E)
      (let ((args (map evaluation-sexpr (cdr E))))
        (application (operation (car E)) args))
      E))
```

```
(evaluation-sexpr '(* (+ (* 90 10) 100) (+ 20 80) 36))
  → 3600000
```



Fin séquence)