(Séquence 2.4

Erreur et hypothèse

Détection d'erreur

```
;;; aire-couronne : Nombre * Nombre -> Nombre
;;; ERREUR lorsque r1 < r2
;;; (aire-couronne r1 r2) rend l'aire de la couronne
;;; de rayon extérieur r1 et de rayon intérieur r2
;;; HYPOTHÈSE: r1 et r2 positifs
(define (aire-couronne r1 r2)
  (if (< r1 r2)
      (erreur 'aire-couronne
              "rayon extérieur (" r1 ") <"
              "rayon intérieur (" r2 ")")
      (- (aire-disque r1) (aire-disque r2))))
```

La détection d'erreur a donc un coût.

La notion d'hypothèse

```
;;; aire-couronne-sans : Nombre * Nombre -> Nombre
;;; (aire-couronne-sans r1 r2) rend l'aire de la
;;; couronne de rayon extérieur r1 et de rayon
;;; intérieur r2
;;; HYPOTHÈSES: r1 et r2 positifs et r1 >= r2
(define (aire-couronne-sans r1 r2)
  (- (aire-disque r1) (aire-disque r2)))
```

La preuve que les hypothèses sont respectées est maintenant à la charge de l'appelant.

Vocabulaire pour une application

Dans l'application `(* (+ 1 2) 3)` :

- l'expression en ***position fonctionnelle*** est `*`
- la fonction est la multiplication (la multiplication est la valeur de la variable `*`)
- les ***paramètres d'appel*** sont les expressions `(+ 1 2)` et `3`
- les ***arguments*** de la multiplication sont `3` et `3`

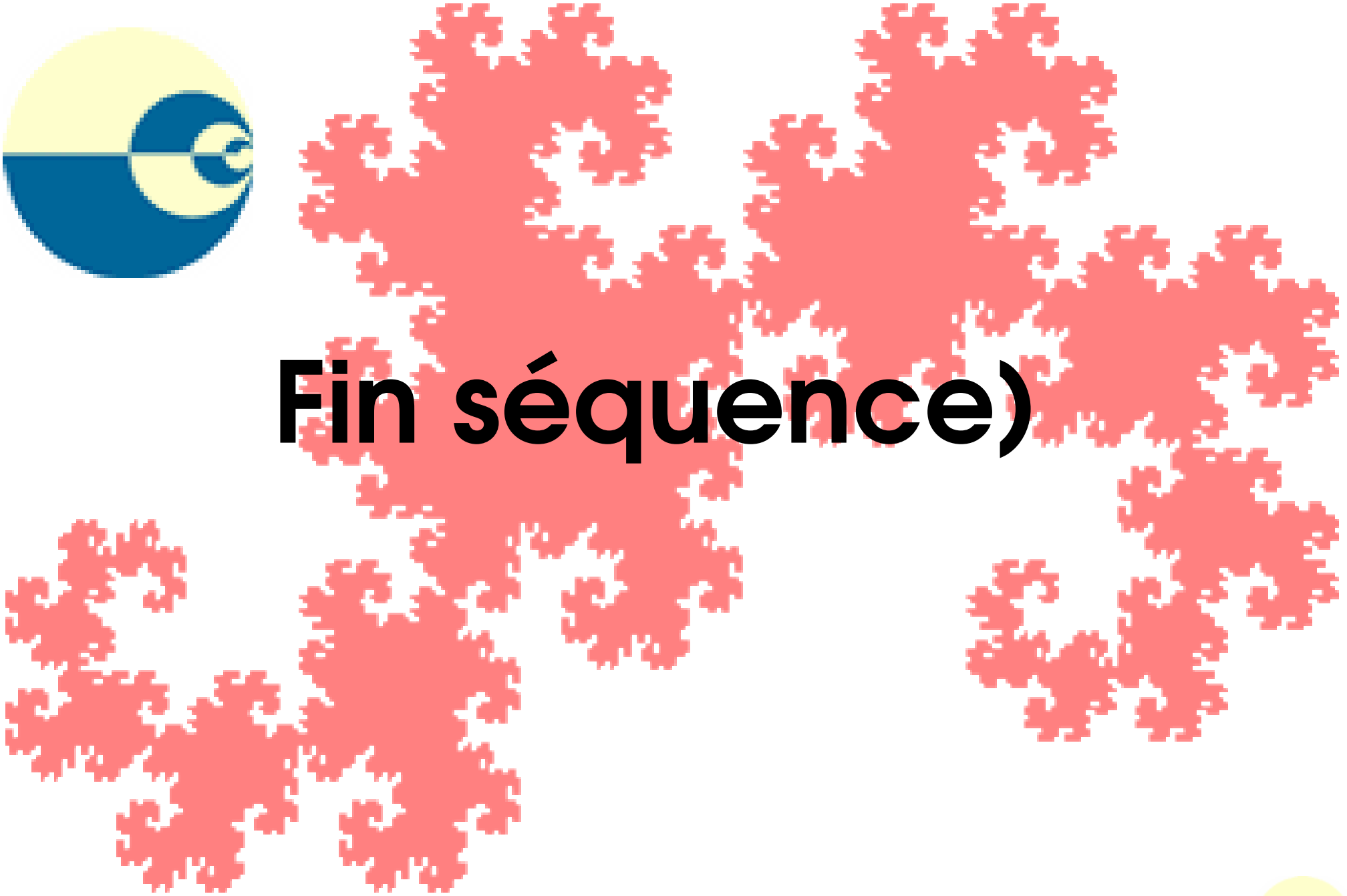
Vocabulaire pour une définition

Dans la définition

```
(define (produit x y)
   (* x y) )
```

- `x` et `y` sont les ***variables***
- les ***arguments*** de l'appel à `produit` sont les valeurs de ces variables.

De nombreux abus de langage sont toutefois faits.



Fin séquence)