

Associations

```
type ('a,'b) pmap (* 'a type des clefs, 'b type des E_x *)
comp : 'a -> 'a -> int (* comparaison sur clefs *)
empty : ('a,'b) pmap
add : 'a -> 'b -> ('a,'b) pmap -> ('a,'b) pmap
mem : 'a -> ('a,'b) pmap -> bool
find : 'a -> ('a,'b) pmap -> 'b (* évt Not_found*)
remove : 'a -> ('a,'b) pmap -> ('a,'b) pmap
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 97

Associations

Lecture de l'association : find

Données : clef x et association m

Valeur de find $x m$:

- E_x si associé à x dans m
- Sinon exception levée : **Not_found**

```
find : 'a -> ('a,'b) pmap -> 'b (* évt Not_found*)
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 99

Associations

Informations nécessaires : **clef et élément lié** (binding)

AVL construit sur clefs \rightsquigarrow **information supplémentaire** : **élément lié**

```
type ('a,'b) pmap =
| EM
| NM of (('a,'b) pmap * 'a * 'b * ('a,'b) pmap * int);;

comp : 'a -> 'a -> int
empty : ('a,'b) pmap
height : ('a,'b) pmap -> int

add : 'a -> 'b -> ('a,'b) pmap (* overriding *)
mem : 'a -> ('a,'b) pmap -> bool
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 98

Associations

Application $N \rightarrow \mathcal{P}(N)$

Recherche d'un nœud, de successeurs

type 'a graph

```
add_vertex : 'a -> 'a graph -> 'a graph
```

```
add_arc : 'a -> 'a -> 'a graph -> 'a graph
```

Graphe comme association :

```
type 'a graph = ('a, 'a avl) pmap
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 100

graphes

Associations

graphes, parcours

Trouver un chemin. Fonction récursive : **bonne fondation** ? (car circuits !)

On peut :

- Parcourir le graphe en **profondeur**
- Remplir les **déjà vus**
- Interrompre en cas de succès (ou d'échec)

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 101

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 102

Arbres

Arité variable

Exemple : termes

- Variables $\in T$
- Symbole f d'arité $n \geq 0$ et t_1, \dots, t_n des termes $\rightarrow f(t_1, \dots, t_n) \in T$

Quelle structure ?

Ensembles

Données **composites** : listes, etc.

Comparaison : **coûteux**

Mémoire : **coûteux**

\leadsto **Recherche** : horreur

Données chaînées

(pas atomique)

(redondance)

Ensembles

Données **composites** : listes, etc.

Comparaison : pas à pas jusqu'à **différence**

Mémoire : débuts **communs** \rightarrow **partage**

\leadsto **Recherche** : bornée par longueur du plus grand élément

Partage des préfixes : arbres de préfixes

trie

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 103

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 104

Ensembles

- Nœuds : `bool`
- Branches : étiquetées par composants d'éléments

Élem. contenu si `bool final = true`

Invariant : **bien formé**

Invariant \Rightarrow recherche **bornée par longueur** du plus grand
QUEL QUE SOIT le nombre d'éléments !

type 'a trie

```
mem : 'a list -> 'a trie -> bool
add : 'a list -> 'a trie -> 'a trie
remove : 'a list -> 'a trie -> 'a trie
inter : 'a trie -> 'a trie -> 'a trie
```

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 105

Ensembles

- Nœuds : `bool`
- Branches : étiquetées par composants d'éléments

Élem. contenu si `bool final = true`

Invariant : **bien formé**

Invariant \Rightarrow recherche **bornée par longueur** du plus grand
QUEL QUE SOIT le nombre d'éléments !

```
type 'a trie = T of (bool * ('a, 'a trie) pmap)
mem : 'a trie -> bool
add : 'a -> 'a trie -> 'a trie
remove : 'a -> 'a trie -> 'a trie
inter : 'a trie -> 'a trie -> 'a trie
map, fold, etc.
```

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 106

Arbres de préfixes

Ensembles

Toujours plus loin...

`bool` ou autre

\leadsto Associations

Si **false**, que faire ?

\leadsto Type option

type 'a option = None | Some of 'a

Constatation

type 'a trie =

```
empty : 'a trie
mem : 'a trie -> bool
add : 'a -> 'a trie -> 'a trie
remove : 'a -> 'a trie -> 'a trie
inter : 'a trie -> 'a trie -> 'a trie
map, fold, etc.
```

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 107

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 108

Constatation

type ('a, 'b) pmap

```
empty : ('a, 'b) pmap  
mem : 'a → ('a, 'b) pmap → bool  
add : 'a → 'b → ('a, 'b) pmap → ('a, 'b) pmap  
remove : 'a → ('a, 'b) pmap → ('a, 'b) pmap  
find : 'a → ('a, 'b) pmap → 'b (* évt Not_found*)  
fold : ('a → 'b → 'c → 'c) → ('a, 'b) pmap → 'c → 'c
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 109

Constatation

type 'a avl

```
empty : 'a avl  
mem : 'a → 'a avl → bool  
add : 'a → 'a avl → 'a avl  
remove : 'a → 'a avl → 'a avl  
fold : ('a → 'b → 'b) → 'a avl → 'b → 'b
```

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 110

Constatation

Conflits de noms

Représentation concrète polluante

Duplication

→ **Modules**

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 111

Modules

Dans **.mli** interface

Dans **.ml** implantation

Compilation **séparée** avec `ocamlc -c`

D'abord **.mli** → **.cmi**

Ensuite **.ml** → **.cmo**

Fichiers

Informations publiques

Cambouis

*

Vérification de conformité

Utilisation : dénomination par **majuscule**

* **Un seul** fichier à compiler : le modifié (sauf si interface)

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 112

Modules

Définir un module **comme un type**, etc.

Déclaration : **module**

Encadrement : **struct ... end**

```
module A = struct
  let a = 42
  let f x = a + x
end
```

Code arbitraire

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 113

Langage

Modules

Définir un module **comme un type**, etc.

Déclaration : **module**

Encadrement : **struct ... end**

```
module A = struct
  let a = 42
  module B = struct
    let b = 666
    let f x = a * b * x
  end
  let f x = B.f (73 + x)
end
```

→ Organisation, nommage

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 114

Modules

Modules : **typés**

Type de module : **signature**

```
module type S = sig
  val f : int -> int
end
```

```
module A : S = struct
  let a = 42
  let f x = 42 + x
end
```

Valeur **a invisible** hors module

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 115

Langage

(.mli = interface d'un module fichier)

Modules

Paramétrage par modules

Exemple de arbres AVL : comparaison ?

Ordre sup ? add comp e t = ... **Bad** karma (diff. comp utilisables)
et puis lourd !

XU - ensie - Prog. fonctionnelle 2011/2012

[UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE](#) 116

Modules

Paramétrage par modules

Exemple de arbres AVL : comparaison ?

comp dans déf. AVL ?

Bad karma (inefficace, complexe)

Foncteurs

Modules

Paramétrage par modules

Exemple de arbres AVL : comparaison ?

Foncteurs

```
module F (X : S) = struct ... end
```

Good karma

Foncteurs

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 117

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 118

Modules

Paramétrage par modules

Exemple de arbres AVL : comparaison ?

Foncteurs

```
module type OrderedSet = sig
```

```
  type t
```

```
  val comp : t -> t -> int
```

```
end
```

```
module Avl(X : OrderedSet) = struct
```

```
  type elt = X.t
```

```
  type avl = E | N of (avl * elt * avl * int)
```

```
  let ...
```

Foncteurs

Modules

Paramétrage par modules

Exemple de arbres AVL : comparaison ?

Foncteurs

Instanciation :

```
module Entiers = struct
```

```
  type t = int
```

```
  let comp x y = x - y
```

```
end
```

```
module EnsEntiers = Avl(Entiers)
```

Good karma

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 119

XU - ensie - Prog. fonctionnelle 2011/2012

UN JEU DE SLIDES N'EST PAS UN POLY DE RÉFÉRENCE 120

Conclusion

On ne vous a pas menti

Ce n'était pas un cours d'Ocaml !

Vu :

- Fonctions objets **comme les autres** Puissant
- Structures (ensembles, séquences, etc.) **dynamiques**
 - Listes : séquences, first in last out, **+** **+** ajout C^te - - recherche n
 - Arbres AVL : ensembles, **+** recherche $\log n$ - ajout $\log n$
 - * Associations
 - * Graphes
 - Tries : ensembles de « mots », **++** recherche (indep de n)

Conclusion

On ne vous a pas menti

Ce n'était pas un cours d'Ocaml !

Vu :

- Fonctions objets **comme les autres** Puissant
- Structures (ensembles, séquences, etc.) **dynamiques**
- Structures **persistantes** Efficace, simple
- Moyens confortables de manipuler ces structures Haut niveau
- Moyens de **prouver** le code correct Modèle formel

Plus loin : mélange impératif / fonctionnel