ML 演習 第4回

おおいわ

June 20, 2000

パターンマッチ (補足)

- alias pattern
  - **パターンマッチの結果に別名を与える**

```
# match (1, (2, 3)) with (x, (y, z as a)) -> a
- : int * int = (2, 3)
```

  - **結合が弱いので注意。必要なら () を。**
    (**上の例では** y, (z as a) **ではなく**
    (y, z) as a **と結合している**)

今回の内容

- **モジュールシステム**
  - structure
  - signature
  - functor

structure (1)

- **変数や型などの定義の集合**
  - **例**: MultiSet (lecture4-1.ml)
  - **内部の変数には . 表記でアクセス**

```
# MultiSet.empty;;
- : 'a MultiSet.set = MultiSet.Leaf
# let a = MultiSet.add MultiSet.empty 5;;
val a : int MultiSet.set = MultiSet.Node
                           (5, MultiSet.Leaf, MultiSet.Leaf)
# MultiSet.member a 5;;
- : bool = true
```

structure (2)

- open: structure を「開く」
  - structure 内の定義を . 無しでアクセス

```
# open MultiSet;;
# add empty 5;;
- : int MultiSet.set = MultiSet.Node
                          (5, MultiSet.Leaf, MultiSet.Leaf)
# member (add empty 5) 10;;
- : bool = false
```

signature

- structure **に対する「型」**
  - **公開する/隠蔽する変数や型の指定**
  - **例**: MULTISET: **重複集合の抽象化**
    - type 'a set **は存在だけが示されている**
    - remove_top **は定義にない**

signature の適用 (1)

- signature を structure に適用

```
# module AbstractMultiSet = (MultiSet : MULTISET);;
module AbstractMultiSet : MULTISET
# let a = AbstractMultiSet.empty;;
val a : 'a AbstractMultiSet.set = <abstr>
# let b = AbstractMultiSet.add b 5;;
val b : int AbstractMultiSet.set = <abstr>
```

抽象データ型の内容は隠蔽される

signature の適用 (2)

```
# open AbstractMultiSet;;
# let a = add (add empty 5) 10;;
val a : int AbstractMultiSet = <abstr>
# AbstractMultiSet.remove_top;;
Unbound value AbstractMultiSet.remove_top;;
# MultiSet.remove_top a;;
This expression has type int AbstractMultiSet.set
  but it is used with type  à MultiSet.set
```

functor の定義

- structure から structure への「関数」
  - 例: lecture4-2.ml
    - signature ORDERED_TYPE
      - 一般の全順序・等値関係つきの型
    - functor MultiSet2
      - ORDERED_TYPE を持つ structure に対する集合の定義

functor と signature

- functor に対する signature の定義
  - SETFUNCTOR: MultiSet2 に対する functor signature
    - elem の型は concrete (Elt.t)
    - t の型は abstract
  - AbstractSet2: SETFUNCTOR で制限した functor MultiSet2

functor と signature (2)

```
# module AbstractStringSet =
                              AbstractSet2(OrderedString);;
module AbstractStringSet : sig ... end
# let sa = AbstractStringSet.add
   AbstractStringSet.empty "OCaml ";;
val sa : AbstractStringSet.t = <abstr>
# AbstractStringSet.member sa "ocaml ";;
- : bool = false
```

functor と signature (3)

```
# module NCStringSet = AbstractSet2(NCString);;
module NCStringSet : sig ... end
# let sa = NCStringSet.add NCStringSet.empty
    "OCaml";;
val sa : NCStringSet.t = <abstr>
# NCStringSet.member sa "ocaml";;
- : bool = true
# AbstractStringSet.add sa "ocaml";;
This expression has type NCStringSet.t =
   AbstractSet2(NCString.t) but is here used with type
   AbstractStringSet.t = AbstractSet2(OrderedString.t)
```

課題1

- リストなどの別のデータ構造を使って signature MULTISET に対する別の実装を与えよ。
  - structure の書き方の練習。そんなに難しくはないと思います。

課題2

- 前回の銀行口座の例で、structure と signature を用いて暗証と口座リストをきちんと隠蔽せよ。
  - signature の練習。前回関数を使って暗証を隠蔽してくれた人は口座リストだけでもいいし、できれば record を用いた実装を作って signature によるデータ型の abstraction を試してみてください。

課題3 (optional)

- ORDERED_TYPE で表現される型の key と、任意の型の値についての連想配列を作り出す functor を作れ。

課題3 (例1)

```
# module NCStringAssociation = Association(NCString);;
module NCStringAssociation :
  sig
    type key = NCString.t
    and 'a t = 'a Association(NCString).t
    val empty : 'a t
    val add : 'a t -> key -> 'a -> 'a t
    val remove : 'a t -> key -> 'a t
    val get : 'a t -> key -> 'a
    exception Not_Found
  end
```

課題3 (例2)

```
# open NCStringAssociation;;
# let sa = add empty "C" "/* */";;
val sa : string NCStringAssociation.t = <abstr>
# let sa = add sa "OCaml" "(* *)";;
val sa : string NCStringAssociation.t = <abstr>
# let sa = add sa "Perl" "# ";;
val sa : string NCStringAssociation.t = <abstr>
# get sa "ocaml";;
- : string = "(* *)"
```

提出方法

- **〆切**: 2000年7月3日 月曜日 24:00
- **提出先**: ml-report@yl.is.s.u-tokyo.ac.jp
- **題名**: Report 4 (**学生証番号**)