Jacques Garrigue, 2011 年 11 月 2 日

再帰的データ構造

8　再帰的データ構造

型略称の定義

```
# type nat = unit list ;;                          (* 型の略称を定義する *)
type nat = unit list
# let succ (n : nat) : nat = () :: n;;
val succ : nat -> nat = <fun>
```

データ型の定義　略称とは別に，本当に新しい型も定義できる．

```
# type sort = Heart | Diamond | Spade | Club ;; (* 構成子は大文字で始まる
*)
# type figure = King | Queen | Jack | Number of int ;;
# type card = Joker | Card of sort * figure ;;      (* Card の引数は対 *)
# let joker = Joker ;;
val joker : card = Joker
# let heart_ace = Card (Heart, Number 1) ;;
val heart_ace : card = Card (Heart, Number 1)
# let strength card =
    match card with                              (* パターンマッチングが使える *)
      Joker -> 15
    | Card (_, Number 1) -> 14
    | Card (_, King)     -> 13
    | Card (_, Queen)    -> 12
    | Card (_, Jack)     -> 11
    | Card (_, Number n) -> n
  ;;
val strength : card -> int
# strength heart_ace;;
- : int = 14
# strength (Card (Club, Number(-5)));;
- : int = -5
```

再帰データ型　型定義は再帰的であってもいい．

```
# type nat = Zero | Succ of nat ;;
# let rec add_nat n1 n2 =
    match n1 with
      Zero   -> n2
    | Succ n -> Succ (add_nat n n2)
  ;;
val add_nat : nat -> nat -> nat
```

```
# add_nat (Succ (Succ Zero)) (Succ Zero) ;;
- : nat = Succ (Succ (Succ Zero))
```

練習問題 8.1 上記の自然数の定義を使って，引き算と掛け算を定義しなさい．

リスト

リストも自分で定義できる．

```
# type 'a mylist =                                    (* 'a は型引数 *)
    | Nil
    | Cons of 'a * 'a mylist ;;
# let nums = Cons (1, Cons (2, Nil)) ;;
val nums : int mylist = Cons (1, Cons (2, Nil))
# let rec length l =
    match l with
      Nil -> 0
    | Cons (_, l') -> 1 + length l'
  ;;
val length : 'a mylist -> int
# length nums ;;
- : int = 2
```

再帰データ型が将来のコードの変更を安全にしてくれる．

```
# type 'a mylist =                                  (* mylist の定義を変更 *)
    | Nil
    | Cons of 'a * 'a mylist
    | Concat of 'a mylist * 'a mylist                    (* リストの連結 *)
# let rec length l =                                 (* 元の定義をそのまま *)
    match l with
      Nil -> 0
    | Cons (_, l') -> 1 + length l'
  ;;
Warning P: this pattern-matching is not exhaustive.
Here is an example of a value that is not matched:
Concat (_, _)
val length : 'a mylist -> int = <fun>
```

間違ったプログラムがコンパイルできるものの，問題箇所を指摘したワーニングが出力される．(エラーにもできる)

練習問題 8.2 *1.* 変更された *mylist* に対する *length* の定義を完成させよ．

2. 変更された *mylist* を普通の *list* に変換する関数を定義せよ．