5 ストリームパーザの追加資料

ストリームの基礎

```
# #load"camlp4o.cma";;
Camlp4 Parsing version 3.10+dev14 (2006-10-27)

# let s = Stream.of_string "abcde";;       (* 文字列から文字のストリームを作る *)
val s : char Stream.t = <abstr>
# Stream.next s;;                                    (* ストリームの先頭 *)
- : char = 'a'
# match s with parser [< ''a' >] -> true;;
Exception: Stream.Failure.
# match s with parser [< ''b' >] -> true;;              (* 見たら除かれる *)
- : bool = true
# match s with parser [< ''b' >] -> 1 | [< >] -> 2;;
- : int = 2                                    (* Failure が起こると次に行く *)
# match s with parser [< ''b' >] -> 1 | [< ''c' >] -> 2;;
- : int = 2
# match s with parser [< ''d'; ''x' >] -> true;;
Exception: Stream.Error "".             (* 2 項目以降の Failure は Error になる *)
# match s with parser [< 'x >] -> x;;
- : char = 'e'
# Stream.next s;;
Exception: Stream.Failure.
# let s = [< ''a'; ''b' >];;                        (* 手でストリームを作る *)
val s : char Stream.t = <abstr>
# let s' = [< ''1'; s >];;                        (* ストリームを組み合わせる *)
val s' : char Stream.t = <abstr>
# match s' with parser [< 'x; 'y; 'z >] -> (x,y,z);;
- : char * char * char = ('1', 'a', 'b')
```

練習問題 5.1 *1.* ストリームをリストに変える関数を定義せよ.

```
val list_of_stream :  'a stream -> 'a list
```

2. リストをストリームに変える関数を定義せよ.

```
val stream_of_list :  'a list -> 'a stream
```

ストリームを使った字句解析

```
let rec decimal s =
  match s with parser
    [< ''0'..'9' as c >] -> f (digit c) s
and f n s =
  match s with parser
    [< ''0'..'9' as c >] -> f (n*10 + digit c) s
  | [< ''.' >] -> f' n 1 s
  | [< >] -> float n
and f' n d s =
  match s with parser
    [< ''0'..'9' as c >] -> f' (n*10 + digit c) (d*10) s
  | [< >] -> float n /. float d ;;
```

```
val decimal : char Stream.t -> float = <fun>
val f : int -> char Stream.t -> float = <fun>
val f' : int -> int -> char Stream.t -> float = <fun>
# let s = Stream.of_string "23.5 4.3" ;;
val s : char Stream.t = <abstr>
# decimal s ;;
- : float = 23.5
# Stream.next s ;;
- : char = ' '
# decimal s ;;
- : float = 4.3
```

複雑なストリームパターン

```
let rec decimal = parser
    [< ''0'..'9' as c; r = f (digit c) >] -> r
and f n = parser
    [< ''0'..'9' as c; r = f (n*10 + digit c) >] -> r
  | [< ''.'; r = f' n 1 >] -> r
  | [< >] -> float n
and f' n d = parser
    [< ''0'..'9' as c; r = f' (n*10 + digit c) (d*10) >] -> r
  | [< >] -> float n /. float d
```

繰り返し

```
let rec decimal_list = parser
    [< n = decimal; l = decimal_list >] -> n :: l
  | [< '' ''; l = decimal_list >] -> l
  | [< >] -> [] ;;
val decimal_list : char Stream.t -> float list
# let s = Stream.of_string "23.5 4.3" ;;
val s : char Stream.t = ¡abstr¿
# decimal_list s ;;
- : float list = [23.5; 4.3]
```

汎関数

```
let rec star pa = parser
    [< h = pa; t = star pa >] -> h :: t
  | [< >] -> [] ;;
val star : ('a Stream.t -> 'b) -> 'a Stream.t -> 'b list
let skip_spaces s =
  ignore (star (parser [< '' ''>] -> ()) s) ;;
val skip_spaces : char Stream.t -> unit
let decimal_list = star (fun s -> skip_spaces s; decimal s) ;;
val decimal_list : char Stream.t -> float list
```

練習問題 5.2 decimal と ident の定義の中で star や accumulate を使うように変更せよ．let rec なしに定義できますか.