Sequence

佐原伸

２００５年3月２２日

1 Introduction

Sequence ライブラリ。

1.1 Sequence

1.1.1 責任

列を表す。

1.1.2 概要

Sequence 型で定義された機能以外の機能を定義する。

1.1.3 注意事項

歴史的経過のため、より関数型プログラミングに適した関数と、そうでないものがある。関数型プログラミングに適した関数は、英字名の場合大文字で始まる。大文字で始まる同一名の関数がある場合、小文字で始まる関数・操作は、古い定義で、互換性のために存在する。大文字で始まる同一名の関数が無い場合は、小文字で始まる関数・操作も関数型プログラミングに適している。

1.1.4 参照

多くの関数は、関数型プログラミング言語 Concurrent Clean のライブラリーから移植した。

class *Sequence*

values

$Rcsid = \text{"\$}Id: \ Sequence.vpp, v \ 1.1 \ 2005/11/25 \ 07:05:38 \ vdmtools \ Exp \ \$\text{"}$

functions

public static

$\Sigma\ [@型] : @型^* \to @型$

$\Sigma\ (列) \triangleq$

$\quad \Sigma$ 補助関数 $[@型]\ (列)\ (0)$

pre $\mathsf{is\_}(列, \mathbb{Z}^*) \lor \mathsf{is\_}(列, \mathbb{N}^*) \lor \mathsf{is\_}(列, {\mathbb{N}_1}^*) \lor$

$\quad \mathsf{is\_}(列, \mathbb{R}^*) \lor$

$\quad \mathsf{is\_}(列, \mathbb{Q}^*)$;

```
        Σ補助関数 [@型] : @型* → @型 → @型
        Σ補助関数 (列)(合計) ≙
          if 列 = []
          then 合計
          else Σ補助関数 [@型] (tl 列) (合計 + hd 列);
public static
        Π [@型] : @型* → @型
        Π (列) ≙
          Π補助関数 [@型] (1) (列)
        pre  is_(列, ℤ*) ∨ is_(列, ℕ*) ∨ is_(列, ℕ₁*) ∨
             is_(列, ℝ*) ∨
             is_(列, ℚ*) ;

        Π補助関数 [@型] : @型 → @型* → @型
        Π補助関数 (積)(列) ≙
          cases 列 :
            [先頭] ↷ 後続列 → Π補助関数 [@型] (積 × 先頭) (後続列),
            [] → 積
          end;
public static
        平均を得る [@型] : @型* → [ℝ]
        平均を得る (列) ≙
          if 列 = []
          then nil
          else 平均を得る補助関数 [@型] (0) (0) (列);

        平均を得る補助関数 [@型] : @型 → @型 → @型* → ℝ
        平均を得る補助関数 (合計)(要素数)(列) ≙
          cases 列 :
            [先頭] ↷ 後続列 → 平均を得る補助関数 [@型] (合計 + 先頭) (要素数 + 1) (
後続列),
            [] → 合計/要素数
          end;
public static
```

全順序昇順か？[@型] : (@型 × @型 → $\mathbb{B}$) → @型* → $\mathbb{B}$
全順序昇順か？(順序決定関数)(列) $\triangleq$
$\forall\, i, j \in$ inds 列 $\cdot\, i < j \;\Rightarrow$ 順序決定関数 (列 (i), 列 (j)) $\vee$ 列 $(i) =$ 列 (j);
public static
全順序降順か？[@型] : (@型 × @型 → $\mathbb{B}$) → @型* → $\mathbb{B}$
全順序降順か？(順序決定関数)(列) $\triangleq$
$\forall\, i, j \in$ inds 列 $\cdot\, i < j \;\Rightarrow$ 順序決定関数 (列 (j), 列 (i)) $\vee$ 列 $(i) =$ 列 (j);
public static
昇順か？[@型] : @型* → $\mathbb{B}$
昇順か？(列) $\triangleq$
全順序昇順か？[@型] ($\lambda\, x$: @型, y : @型 $\cdot\, x < y$) (列);
public static
降順か？[@型] : @型* → $\mathbb{B}$
降順か？(列) $\triangleq$
全順序降順か？[@型] ($\lambda\, x$: @型, y : @型 $\cdot\, x < y$) (列);
public static
$sort$[@型] : (@型 × @型 → $\mathbb{B}$) → @型* → @型*
$sort$ (順序決定関数)(列) $\triangleq$
cases 列 :
[] → [],
[要素] $\curvearrowright$ 部分列 →
$sort$[@型] (順序決定関数) ([部分列 (i) | $i \in$ inds 部分列 $\cdot$ 順序決定関数 (部分列 (i), 要素)]) $\curvearrowright$
[要素] $\curvearrowright$
$sort$[@型] (順序決定関数) ([部分列 (i) | $i \in$ inds 部分列 $\cdot \neg$ 順序決定関数 (部分列 (i), 要素)])
end;
public static
昇順 $Sort$[@型] : @型* → @型*
昇順 $Sort$ (列) $\triangleq$
$sort$[@型] ($\lambda\, x$: @型, y : @型 $\cdot\, x < y$) (列);
public static

```
        降順 Sort[@型] : @型* → @型*
        降順 Sort (列) ≜
          sort[@型] (λ x : @型, y : @型 · x > y) (列);
public static
        順序通りか？[@型] : (@型 × @型 → 𝔹)* → @型* → @型* → 𝔹
        順序通りか？(順序決定関数列)(列 1)(列 2) ≜
          cases mk- (列 1, 列 2) :
            mk- ([], []) → false,
            mk- ([], -) → true,
            mk- (-, []) → false,
            mk- ([先頭 1] ↷ 後続 1, [先頭 2] ↷ 後続 2) →
                if (hd 順序決定関数列) (先頭 1, 先頭 2)
                then true
                elseif (hd 順序決定関数列) (先頭 2, 先頭 1)
                then false
                else Sequence‘順序通りか？[@型] (tl 順序決定関数列) (後続 1) (
後続 2)
          end;
public static
        マージする [@型] : (@型 × @型 → 𝔹) → @型* → @型* → @型*
        マージする (順序決定関数)(s1)(s2) ≜
          cases mk- (s1, s2) :
            mk- ([], y) → y,
            mk- (x, []) → x,
            mk- ([先頭 1] ↷ 後続 1, [先頭 2] ↷ 後続 2) →
                if 順序決定関数 (先頭 1, 先頭 2)
                then [先頭1]↷Sequence‘マージする [@型] (順序決定関数) (後続 1) (s2)
                else [先頭2]↷Sequence‘マージする [@型] (順序決定関数) (s1) (
後続 2)
          end;
public static
```

```
        InsertAt[@型] : ℕ1 → @型 → @型* → @型*
        InsertAt (位置)(要素)(列) △
          cases mk- (位置, 列) :
            mk- (1, 列) → [要素] ⌢ 列,
            mk- (-, []) → [要素],
            mk- (位置, [先頭] ⌢ 後続列) → [先頭]⌢InsertAt[@型] (位置 − 1) (要素) (
後続列)
          end;
public static
        RemoveAt[@型] : ℕ1 → @型* → @型*
        RemoveAt (位置)(列) △
          cases mk- (位置, 列) :
            mk- (1, [-] ⌢ 後続列) → 後続列,
            mk- (位置, [先頭] ⌢ 後続列) → [先頭]⌢RemoveAt[@型] (位置 − 1) (
後続列),
            mk- (-, []) → []
          end;
public static
        RemoveDup[@型] : @型* → @型*
        RemoveDup (列) △
          cases 列 :
            [先頭] ⌢ 後続 → [先頭]⌢RemoveDup[@型] (filter[@型] (λ x : @型 · x ≠ 先頭) (
後続)),
            [] → []
          end;
public static
        RemoveMember[@型] : @型 → @型* → @型*
        RemoveMember (要素)(列) △
          cases 列 :
            [先頭] ⌢ 後続 → if 要素 = 先頭
                            then 後続
                            else [先頭]⌢RemoveMember[@型] (要素) (
後続),
            [] → []
          end;
```

```
public static
    RemoveMembers[@型] : @型* → @型* → @型*
    RemoveMembers (要素列)(列) ≜
      cases 要素列 :
        [] → 列,
        [先頭] ↷ 後続 → RemoveMembers[@型] (後続) (RemoveMember[@型] (先頭) (
列))
      end;
public static
    UpdateAt[@型] : ℕ1 → @型 → @型* → @型*
    UpdateAt (位置)(要素)(列) ≜
      cases mk- (位置, 列) :
        mk- (-, []) → [],
        mk- (1, [-] ↷ 後続列) → [要素] ↷ 後続列,
        mk- (位置, [先頭] ↷ 後続列) → [先頭]↷UpdateAt[@型] (位置 − 1) (要素) (
後続列)
      end;
public static
    take[@型] : ℤ → @型* → @型*
    take (i)(列) ≜
      列 (1, . . . , i);
public static
    TakeWhile[@型] : (@型 → 𝔹) → @型* → @型*
    TakeWhile (関数)(列) ≜
      cases 列 :
        [先頭] ↷ 後続列 →
            if 関数 (先頭)
            then [先頭] ↷ TakeWhile[@型] (関数) (後続列)
            else [],
        [] → []
      end;
public static
    drop[@型] : ℤ → @型* → @型*
    drop (i)(列) ≜
      列 (i + 1, . . . , len 列);
```

```
public static
        DropWhile[@型] : (@型 → 𝔹) → @型* → @型*
        DropWhile (関数)(列) ≜
          cases 列 :
            [先頭] ↷ 後続列 →
                if 関数 (先頭)
                then DropWhile[@型] (関数) (後続列)
                else 列,
            [] → []
          end;
public static
        Span[@型] : (@型 → 𝔹) → @型* → @型* × @型*
        Span (関数)(列) ≜
          cases 列 :
            [先頭] ↷ 後続列 →
                if 関数 (先頭)
                then let mk- (条件を満たす列, 条件を満たさない列) = Span[@型] (関数) (
後続列) in
                        mk- ([先頭] ↷ 条件を満たす列, 条件を満たさない
列)
                else mk- ([], 列),
            [] → mk- ([], [])
          end;
public static
        SubSeq[@型] : ℕ → ℕ → @型+ → @型*
        SubSeq (開始位置)(要素数)(列) ≜
          列 (開始位置, . . . , 開始位置 + 要素数 − 1);
public static
        last[@型] : @型* → @型
        last (列) ≜
          列 (len 列);
public static
        fmap[@型 1, @型 2] : (@型 1 → @型 2) → @型 1* → @型 2*
        fmap (関数)(列) ≜
          [関数 (列 (i)) | i ∈ inds 列];
```

public static

filter[@型] : (@型 → $\mathbb{B}$) → @型* → @型*

filter (関数)(列) $\triangleq$

[列 (i) | $i \in$ inds 列 · 関数 (列 (i))];

public static

Foldl[@型 1, @型 2] : (@型 1 → @型 2 → @型 1) → @型 1 → @型 2* → @型 1

Foldl (関数)(引数)(列) $\triangleq$

cases 列 :

[] → 引数,

[先頭] ↷ 後続列 → *Foldl*[@型 1, @型 2] (関数) (関数 (引数) (先頭)) (後続列)

end;

public static

Foldr[@型 1, @型 2] : (@型 1 → @型 2 → @型 2) → @型 2 → @型 1* → @型 2

Foldr (関数)(引数)(列) $\triangleq$

cases 列 :

[] → 引数,

[先頭] ↷ 後続列 → 関数 (先頭) (*Foldr*[@型 1, @型 2] (関数) (引数) (後続列))

end;

public static

isMember[@型] : @型 → @型* → $\mathbb{B}$

isMember (要素)(列) $\triangleq$

cases 列 :

[先頭] ↷ 後続 → 要素 = 先頭 ∨ *isMember*[@型] (要素) (後続),

[] → false

end;

public static

$isAnyMember[@型] : @型^* \to @型^* \to \mathbb{B}$
$isAnyMember\,(要素列)(列) \triangleq$
cases 要素列 :
$[先頭] \curvearrowright 後続 \to isMember[@型]\,(先頭)\,(列) \vee isAnyMember[@型]\,(後続)\,($列),
$[] \to$ false
end;
public static
$Index[@型] : @型 \to @型^* \to \mathbb{Z}$
$Index\,(要素)(列) \triangleq$
let $i = 0$ in
$Index$ 補助関数 $[@型]\,(要素)\,(列)\,(i)$;

$Index$ 補助関数 $[@型] : @型 \to @型^* \to \mathbb{Z} \to \mathbb{Z}$
$Index$ 補助関数 $(要素)(列)(索引) \triangleq$
cases 列 :
$[] \to 0$,
$[先頭] \curvearrowright 後続 \to$
if 先頭 $=$ 要素
then 索引 $+ 1$
else $Index$ 補助関数 $[@型]\,(要素)\,(後続)\,(索引 + 1)$
end;
public static
$IndexAll[@型] : @型 \to @型^* \to \mathbb{Z}$-set
$IndexAll\,(要素)(列) \triangleq$
$\{i \mid i \in$ inds 列 $\cdot$ 列 $(i) =$ 要素 $\}$;
public static
$flatten[@型] : @型^{**} \to @型^*$
$flatten\,(列) \triangleq$
conc 列;
public static
$compact[@型] : [@型]^* \to @型^*$
$compact\,(列) \triangleq$
$[列\,(i) \mid i \in$ inds 列 $\cdot$ 列 $(i) \neq$ nil $]$;
public static

$freverse[@型] : @型^* \to @型^*$
$freverse\,(列) \triangleq$
$[列\,(\mathsf{len}\ 列 + 1 - i) \mid i \in \mathsf{inds}\ 列];$
public static
$Permutations[@型] : @型^* \to @型^*\text{-set}$
$Permutations\,(列) \triangleq$
cases 列 :
$[], [\text{-}] \to \{列\},$
others $\to \bigcup\{\{[列\,(i)]^\frown j \mid j \in Permutations[@型]\,(RestSeq[@型]\,(列, i))\} \mid i \in \mathsf{inds}\ 列\}$
end;
public static
$RestSeq[@型] : @型^* \times \mathbb{N} \to @型^*$
$RestSeq\,(列, i) \triangleq$
$[列\,(j) \mid j \in (\mathsf{inds}\ 列 \setminus \{i\})];$
public static
$Unzip[@型\,1, @型\,2] : (@型\,1 \times @型\,2)^* \to @型\,1^* \times @型\,2^*$
$Unzip\,(列) \triangleq$
cases 列 :
$[] \to \mathsf{mk\text{-}}([], [])$,
$[\mathsf{mk\text{-}}(x, y)]^\frown 後続列 \to$
let $\mathsf{mk\text{-}}(xs, ys) = Unzip[@型\,1, @型\,2]\,(後続列)$ in
$\mathsf{mk\text{-}}([x]^\frown xs, [y]^\frown ys)$
end;
public static
$Zip[@型\,1, @型\,2] : @型\,1^* \times @型\,2^* \to (@型\,1 \times @型\,2)^*$
$Zip\,(列\,1, 列\,2) \triangleq$
$Zip2[@型\,1, @型\,2]\,(列\,1)\,(列\,2);$

列の組を、組の列に変換する（より関数型プログラミングに適した形式）

public static

$Zip2[@型\,1, @型\,2] : @型\,1^* \to @型\,2^* \to (@型\,1 \times @型\,2)^*$
$Zip2\,(列\,1)(列\,2) \triangleq$
cases mk- (列 1, 列 2) :
mk- ([先頭 1] ⌢ 後続列 1, [先頭 2] ⌢ 後続列 2) → [mk- (先頭1, 先頭 2)] ⌢ $Zip2[@型\,1, @型\,2]\,(後続列\,1)\,(後続列\,2)$,
mk- (-, -) → []
end
end *Sequence*

Test Suite : vdm.tc
Class : Sequence

Name	**#Calls**	**Coverage**
Sequence‘Π	5	55%
Sequence‘Σ	20	55%
Sequence‘Zip	4	√
Sequence‘Span	14	√
Sequence‘Zip2	16	√
Sequence‘drop	2	√
Sequence‘fmap	20	√
Sequence‘last	1	√
Sequence‘sort	35	√
Sequence‘take	11	√
Sequence‘昇順か？	2	√
Sequence‘降順か？	1	√
Sequence‘Foldl	12	√
Sequence‘Foldr	12	√
Sequence‘Index	338	√
Sequence‘Unzip	5	√
Sequence‘Π補助関数	14	√
Sequence‘Σ補助関数	3170	√
Sequence‘マージする	16	√
Sequence‘平均を得る	3	√
Sequence‘SubSeq	1	√
Sequence‘filter	9	√
Sequence‘昇順 Sort	1	√

Name	**#Calls**	**Coverage**
Sequence‘降順 Sort	1	√
Sequence‘順序通りか？	15	√
Sequence‘RestSeq	9	√
Sequence‘compact	3	√
Sequence‘flatten	1	√
Sequence‘全順序昇順か？	2	√
Sequence‘全順序降順か？	3	√
Sequence‘IndexAll	21	√
Sequence‘InsertAt	20	√
Sequence‘RemoveAt	29	√
Sequence‘UpdateAt	22	√
Sequence‘freverse	21	√
Sequence‘isMember	39	√
Sequence‘DropWhile	6	√
Sequence‘Index 補助関数	2071	√
Sequence‘RemoveDup	11	√
Sequence‘TakeWhile	6	√
Sequence‘平均を得る補助関数	9	√
Sequence‘isAnyMember	6	√
Sequence‘Permutations	11	√
Sequence‘RemoveMember	19	√
Sequence‘RemoveMembers	14	√
Total Coverage		**98%**