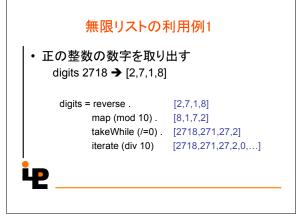
```
無限リスト
胡振江
```

```
無限リストの例

- [1..] → [1,2,3,4,5,...]
- take n [1..] → [1..n]
- [m..] !! n → m+n
- map factorial [0..] → scan (*) 1 [1..]
- [x^2 | x <- [1..], odd x] → [1,9,25,...]
- [[m^n | m <-[1..]] | n<-[2..]]
→ [[1,4,9,16,...],
[1,8,27,64,...],...]
```

```
    「^nの定義
        power f 0 = id
        power f (n+1) = f. power f n
    関数iterate
        iterate f x = [x, f x, f^2 x, ...]
        iterate f x = x : iterate f (f x)
        例:
        iterate (+1) 1 = [1,2,3,4,...]
        iterate (*2) 1 = [1,2,4,8,...]
        [m...] = iterate (+1) m
        [m..n] = takewhile (<=n) (iterate (+1) m)</li>
```



```
無限リスト利用例2

• リストを長さnの部分に分割する
group 2 [1,2,3,4,5,6] → [[1,2],[3,4],[5,6]]

group n = map (take n) .
takeWhile (/=[]) .
iterate (drop n)
```



### 素数の生成

ギリシャの数学者Eratosthenesの手法

- 1 数の並び2,3,...を書き下ろす
- 2 この並びの最初の要素pを素数として登録する
- 3 この並びからpの倍数を消去する
- 4 2へ戻る

Ł

```
primes = map head (iterate sieve [2..])

sieve (p:xs) = [x | x<-xs, x 'mod' p /= 0]

2 3 4 5 6 7 8 9 10 11 12 13 14 15 ...
3 5 7 9 11 13 15 ...
5 7 11 13 ...
```

### 極限としての無限リスト

- 極限(limit)
  - 数学において無限の対象を扱うひとつの方法
  - 例:π = 3.14159265358979323846...は
    - 3
    - 3.1
    - 3.14
    - 3.141 3.1415
- \_\_\_.
- Щ
- の極限値であると考えられる。

### 無限リスト

- 「近似リスト」の列の極限とみなす
- 例:[1..]は次の列の一つの極限である
  - 1:⊥
  - 1:2:⊥ 1:2:3:⊥
- 擬リスト:値⊥で終るリスト
  - x1 : x2 : ... : xn : ⊥

# **P**

### 連続性

- リストの列
  - xs1, xs2, xs3 , ...
- が無限リストで、その極限が xs である
- f が計算可能関数(computable function)



- 無限リスト
  - f xs1, f xs2, f xs3, ...
- 極限は f xs である。

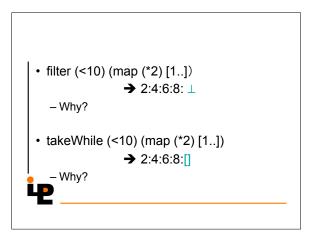
- map (\*2) [1..]の計算
  - map (\*2)  $\perp$  =  $\perp$
  - map (\*2) (1 :  $\bot$ ) = 2 :  $\bot$
  - map (\*2)  $(1:2:\bot) = 2:4:\bot$
  - ...
    - **→** [2,4,6,8,10,...]



• filter even [1..] の計算

filter even ⊥ = ⊥
filter even (1: ⊥) = ⊥
filter even (1:2:⊥) = 2: ⊥
filter even (1:2:3: ⊥) = 2: ⊥
filter even (1:2:3:4: ⊥) = 2:4: ⊥
...

→ [2,4,6,...]



### 擬リストに関する推論

擬リストxsに対して、p(xs)が成立する



- p(⊥)が成立する
- p(xs)が成立する→ p(x:xs)が成立する



### xs ++ ys = xs (xs:擬リスト)

証明:xsに関する帰納法

- ⊥の場合:

±++ys = ± → OK <++.0>

– x:xsの場合:

Ł

### takeの補題

- リスト上の帰納法で証明できない場合もある iterate f x = x: map f (iterate f x)
- takeの補題

```
xs == ys
```



すべての自然数nに対して、 take n xs == take n ys

が成立する。

### iterate f(f x) = map f(iterate f x)

take n (iterate f (f x))

- = take n (map f (iterate f x)
- 0の場合: take 0 xs = [] → 自明
- n+1の場合:

take (n+1) (iterate f (f x))

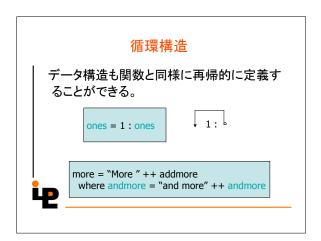
= f x : take n (map f (iterate f (f x)))= take (n+1) (f x : map f (iterate f (f x)))

= take (n+1) (map f ( $\underline{x}$ : iterate f ( $\underline{f}$   $\underline{x}$ )) <map.2>

= take (n+1) (map f (iterate f x))

<iterate.1>

# nats = [0...] 注:定義 nats = 0: map (1+) nats take n nats = [0..n-1]の証明 take (n+1) nats = take (n+1) (0: map (1+) nats) = 0: take n (map (1+) nats)) = 0: map (1+) (take n nats) -- 要証明 = 0: map (1+) [0..n-1] = 0: [1..n] = [0..n]



# 例:forever

次のようなことを計算する関数の定義を考える。 forever x = [x,x,...]

・ 循環構造のない定義:

forever x = x: forever x

循環構造のある定義: forever x = xs

ĻР

where xs = x : xs

### 循環構造による効率化

iterative の二つの定義:

iterate1 f x = x: map f (iterate1 f x)

iterate2 f x = zs

where zs = x : map f zs

最初のn項の計算では

iterate1: O(n^2)

iterate2: O(n)

**Ł** 



## Hammingの問題

次のような無限リストを生成する プログラムを設計せよ。

- 1. リストは重複のない上昇列である。
- 2. リストは1から始まる。
- 3. リストに数 x が含まれているならば、数 2\*x, 3\*x, 5\*xもまたリストに含まれている。
- 4. リストにはそれ以外の数がふくまれていない。

