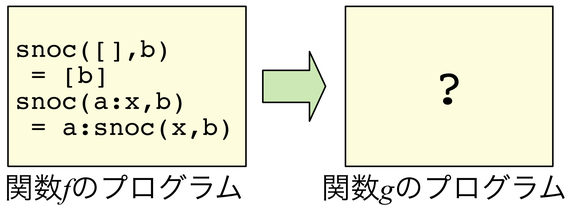
木文法の構文解析を利用したプログラム逆計算

○松田一孝[1] 胡振江[2] 武市正人[1] 穆信成[3] ([1]東京大学 [2]国立情報学研究所 [3]Academia Sinica)

目的

プログラム逆計算



- gがfの左逆 $\Leftrightarrow g(f(x)) = x$
- gがfの右逆 $\Leftrightarrow f(g(y)) = y$

応用例

- エンコードとデコード
- 双方向化[Matsudaら07]

問題点

正しさ

導出した関数は本当に
左逆／右逆?

効率性

導出した関数は効率的?
単純に導出
⇒非決定的プログラム

パターンの重なり

```
isnoc(b:[]) = ([],b)
isnoc(a:y)
 = let (x,b) = isnoc(y)
   in (a:x,b)
```

プログラムと文法

プログラムの出力の推定

プログラム

```
snoc([],b)   = b:[]
snoc(a:x,b) = a:snoc(x,b)
```

木文法

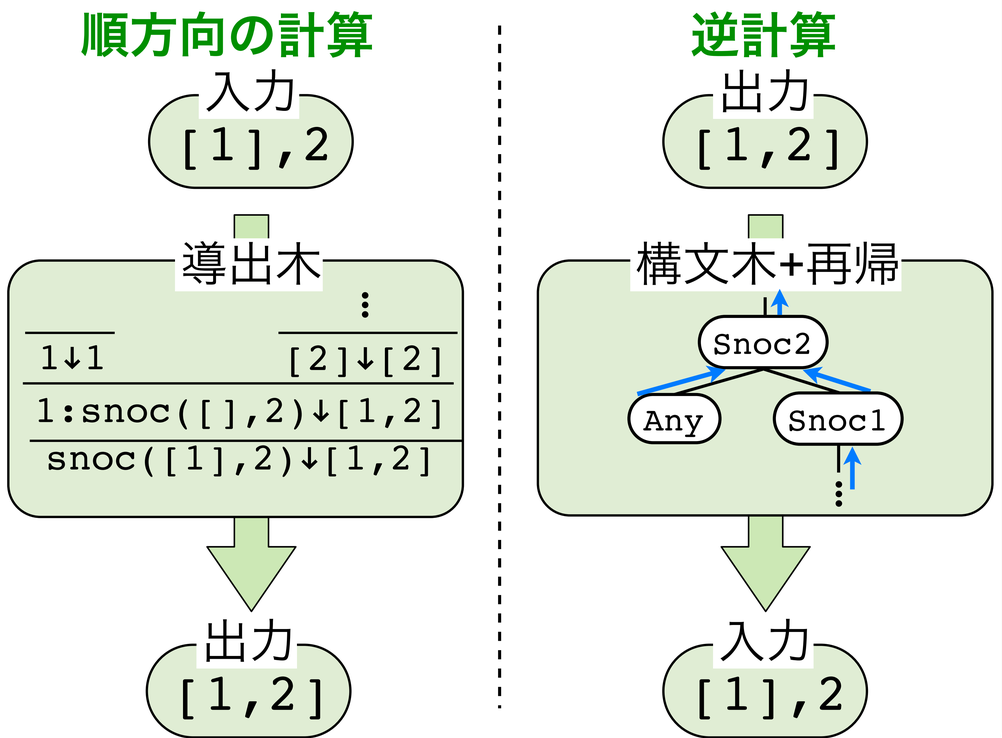
```
Snoc → Any:E1
Snoc → Any:Snoc
E1   → []
```

長さ1以上のリスト

推定の正しさの証明

対応付け
- 文法の構文木
- プログラムの評価の導出木

逆計算のアイデア



例

```
snoc([],b)   = b:[]
snoc(a:x,b) = a:snoc(x,b)
```

```
Snoc → Any:E1    {λb ().([],b)}
Snoc → Any:Snoc  {λa (x,b).(a:x,b)}
E1   → []        {()}
```

正規木文法
⇒**効率的**に構文解析可
⇒非決定性の対処法

特長

- **正しさ**の保証
  - 導出木と構文木の対応付け
- 非決定性の対処による**効率化**
  - 文法の決定化，ガイド[Biehlら96]

議論

- 使用する文法
  - 多相[Hosoyaら06]，文脈自由木
- 他のプログラム逆計算手法
  - 属性文法[Yellin 88]
  - LR構文解析の利用[Glück&Kawabe 05]