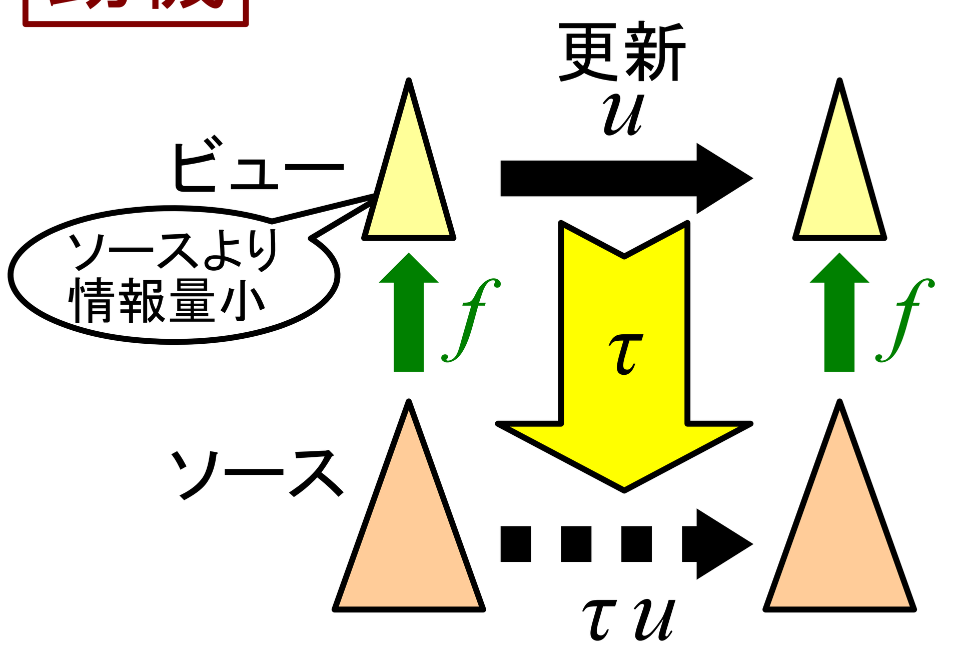
木構造における ビュー更新反映プログラムの 自動生成

松田一孝* 胡振江* 中野圭介* 浜名 誠** 武市 正人*
*東京大学大学院情報理工学系研究科 **群馬大学情報工学科

動機



f から τ を

自動生成したい

どんなτ?

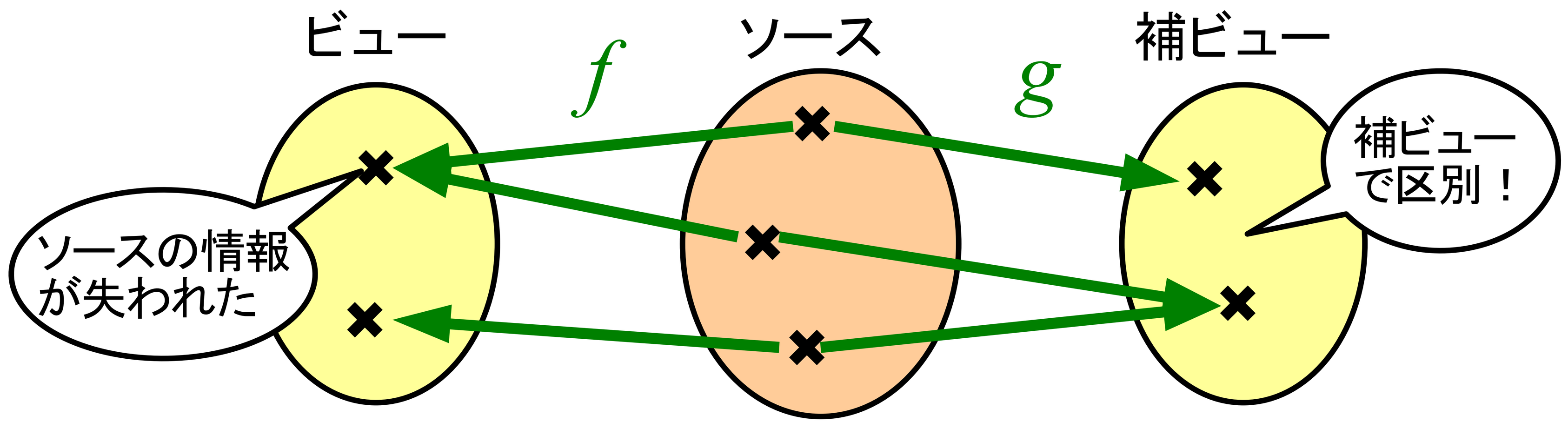
- ビューに変更のないときにはソースを変更しない
- 更新の反映が履歴に依存しない
- ビュー上で元に戻すとソースも元に戻る
- ビューの状態とソースの状態が対応づく

⇒ビューを透過的に扱いたい

補ビューによるアプローチ!!
[Bancilhon and Spyratos, TODS81]

補ビューって？

- ビューが含まない情報を補う
- ビュー定義関数 f と組にすることで単射になる関数 g で作成
- 一般的には一意ではない



補ビューを不変にすれば……透過的に扱える！

補ビューを不変にする更新反映

以下はビューを透過的にするために必要十分

$$\tau u = \langle f, g \rangle^{-1} \circ \langle u \circ f, g \rangle$$

逆計算が必要　但し $\langle f, g \rangle(x) = (f(x), g(x))$

上を計算しやすい,「よい」補ビュー定義関数 g は？

更新可能性大 = 補ビューの情報量小

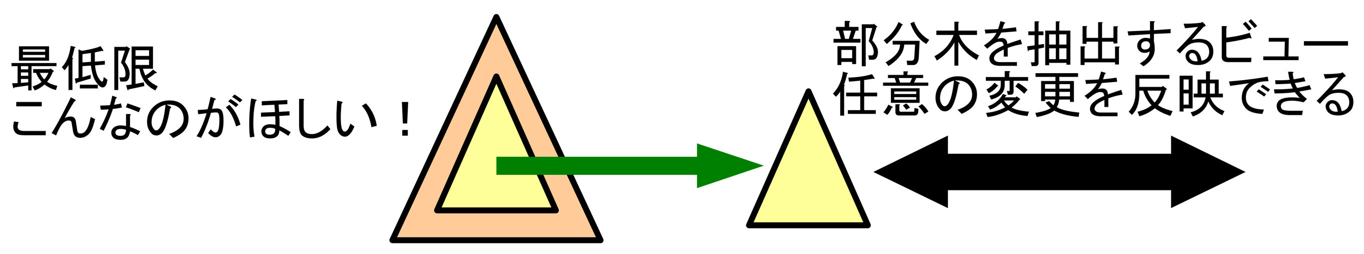
$$g \leqq h \Leftrightarrow$$
$$\forall x, y \in \text{ソース},\ \ h(x) = h(y) \Rightarrow g(x) = g(y)$$

≦で小さい ⇔ 補ビューとして更新可能性が高い

補ビュー不変なのだから，同じ値を返す場所の
多いもののほうが更新可能性が高い

木に対するアプローチ

これまでの補ビュー定義関数を求める研究：関係DBのみ
XML時代！ ⇒ 木も扱いたい！



本研究：一般的なデータ構造を！

- Treeless [Wadler 88] = 関数呼出しの引数は変数のみ
- 同じ変数を2度以上使用しない

例：Bの左の子であるBを取ってくる

```
subT(B(B(x,y),z)) = Just(B(x,y))
subT(B(A,z))      = subT(z)
subT(A)           = Nothing
```

☆補ビュー定義関数導出のアイデア☆

- <u>単射にするのに必要な情報を保存しておく</u>

ビュー定義関数　補ビュー定義関数

```
f(x,y) = x    →    g(x,y) = y
```

使っていない変数を補ビューに

```
f(A(x)) = x    →    g(A(x)) = C1
f(B(x)) = x         g(B(x)) = C2
```

<u>同じ値を生成しうる分岐はコンストラクタを用いて区別</u>
木オートマトンで判定できる！

- 単射な関数呼び出しに対しては補ビューを生成しない
- コンストラクタが必要のない場合はとりのぞく

例

デモもあります！

ビュー定義関数

```
add(S(x),y) = S(add(x,y))
add(   Z,y) = y
```

使用していない変数はなく，2つの分岐は同じ値を生成しうる

求まった補ビュー定義関数

```
add'(S(x),y) = C1(add'(x,y))
add'(   Z,y) = C2
```

第1引数の情報を保存 = 第2引数を変更できる

導出された補ビュー定義関数の性質

- Treeless形式で記述されたビュー定義について，
  補ビューの計算方法を与えた
  - 得られる補ビューの定義は，組化や逆化しやすい
- 部分木をとってくるビューを考えると，
  その部分木については自由に変更できる

現状の問題点

- 更新可能性が最大のものを返せない場合がある
  - 構造の複雑な変化への対応

ビュー定義関数　補ビュー定義関数

```
f(P(x))    = R                g(P(x))    = C1(x)
f(Q(x,y))  = Q(f(x),f(y))  →  g(Q(x,y))  = C2(g(x),g(y))
f(R)       = R                g(R)       = C3
```