

Bidirectionalizing Programs with Duplication through Complement Function Derivation

Kazutaka Matsuda
The University of Tokyo

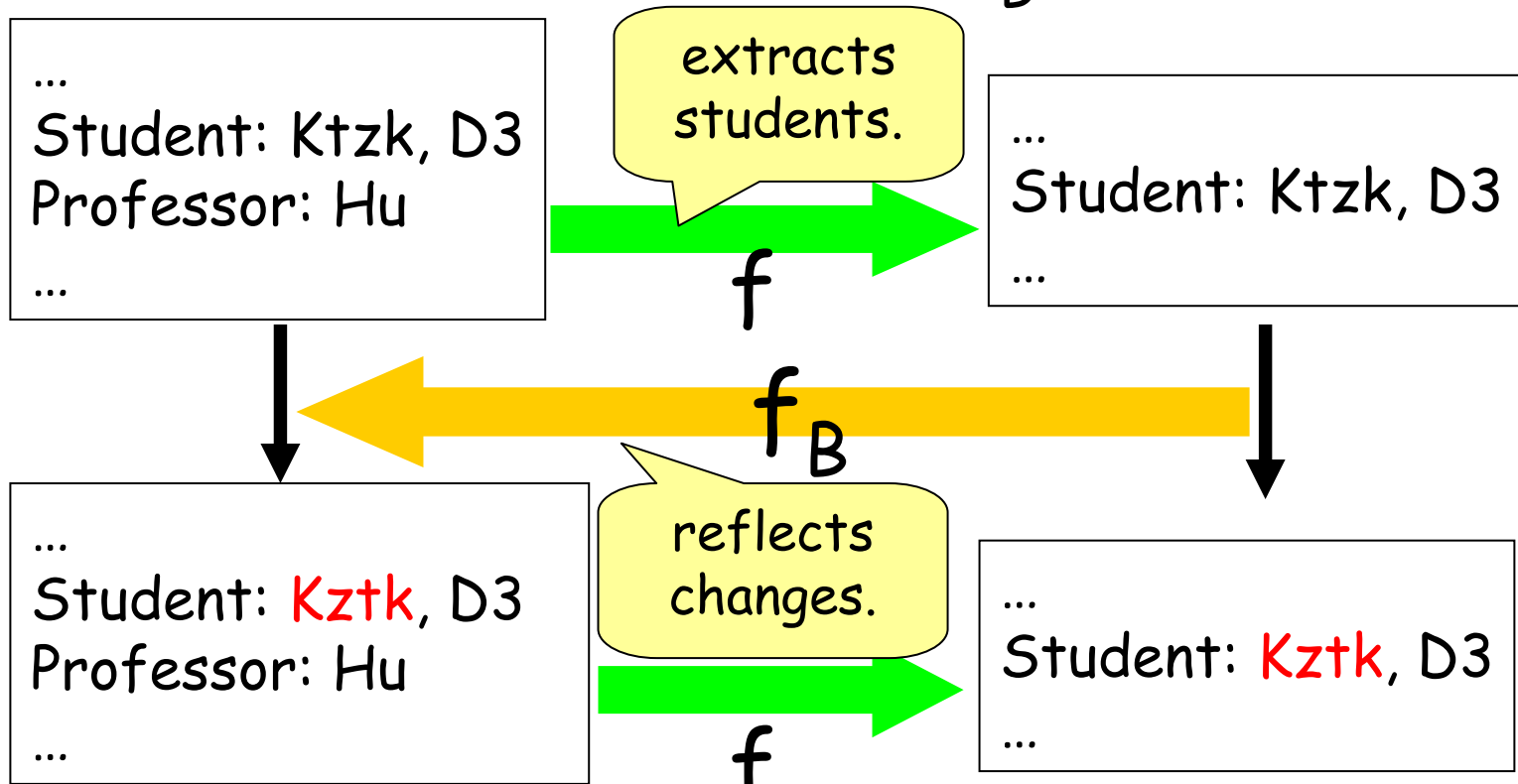
joint work with

Zhenjiang Hu, Keisuke Nakano, Makoto Hamana, and Masato Takeichi

2008-11-15 GRACE Bx Workshop

A Bidirectional Transformation

- Fwd. " $f :: S \rightarrow V$ " + Bwd. " $f_B :: S \times V \rightarrow S$ "



Bidirectionalization
How to derive " f_B " from " f "?

Divide the Problem

- Backward transformation

- "Side Effect"-free part

- Invisible information must be kept.

- Very well-behaved [Foster et al. 05]

$$\text{fst}(x,y) = x$$

$$\text{fst}_B((x,y),v) = (v,y)$$

- "Side Effect"full part

- Invisible information can be lost.

- "create" functions [Foster et al. 05]

- Reconciliation of view updates

- Canonizers [Foster et al. 08], Hu's dup [Hu et al 04]

Side-Effect:

Changes on data irrelevant to view construction³

Our Purpose

- Prev. bidirectionalization [Matsuda et al. 07]
 - fwd. trans. \rightarrow SE-free bwd. trans.
 - For **Affine & Treeless** [Wadler90] language
$$\boxed{\text{fst}(x,y) = x} \longrightarrow \boxed{\text{fst}_B((x,y),v) = (v,y)}$$
 - Successful results for
 - add, half, append, zip, snoc, map-like funcs, etc.

Our Purpose: Relax the Affine Restriction

non-affine features

- dups.: copying, multiple data traversals
- Naive treatment yields poor bwd. trans.

"Unzip" is NOT a Toy

- "Unzip" models transposition.

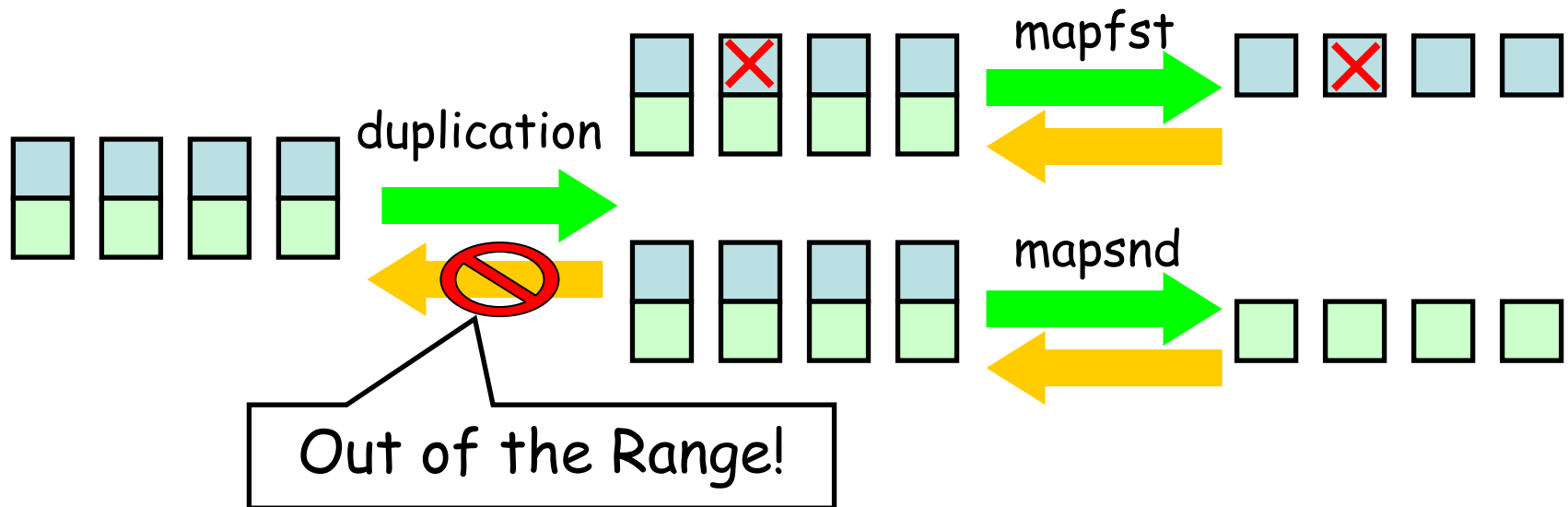
```
<persons>
  <person>
    <name>A</name>
    <math>8</math>
    <physics>2</physics>
  </person>
  <person>
    <name>B</name>
    <math>5</math>
    <physics>3</physics>
  </person>
</persons>
```



```
<scores>
  <math>
    <result>
      <name>A</name><score>8</score>
    </result>
    <result>
      <name>B</name><score>5</score>
    </result>
  </math>
  <physics>
    <result>
      <name>A</name><score>2</score>
    </result>
    <result>
      <name>B</name><score>3</score>
    </result>
  </physics>
</scores>
```

Naïve Bidirectionalization

- Duplication itself is injective.
 - $\text{dup}(x) = (x,x) \Leftrightarrow \text{dup}^{-1}(x,x) = x$
- However...



Observation

- Dup may yield poor bwd. trans.
 - "dup" is not problematic.
 - "unzip = (mapfst × mapsnd) ∘ dup" is problematic
- Requirements for a Language
 - Non-problematic dups are permitted.
 - Some use of problematic dups can be avoided.

My Plan in this Workshop

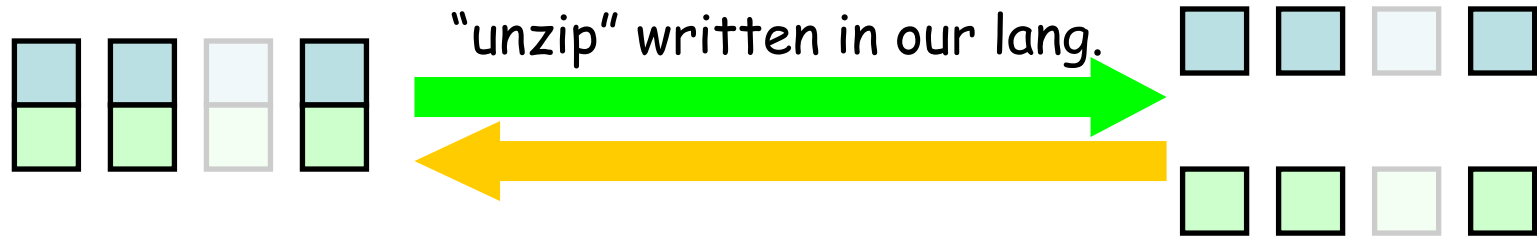
- I will show how to bidirectionalize programs containing duplications (as a technical presentation)
 - "Side Effect"-Free Bidirectionalization
 - A language to distinguish problematic/non-problematic dups.
 - Successful results
 - "unzip", "table of contents", dTdTTs [Rounds 70]
 - Implementation of the most important part.
<<http://www.ipl.t.u-tokyo.ac.jp/~kztk/b18n2/>>

Example of "unzip"

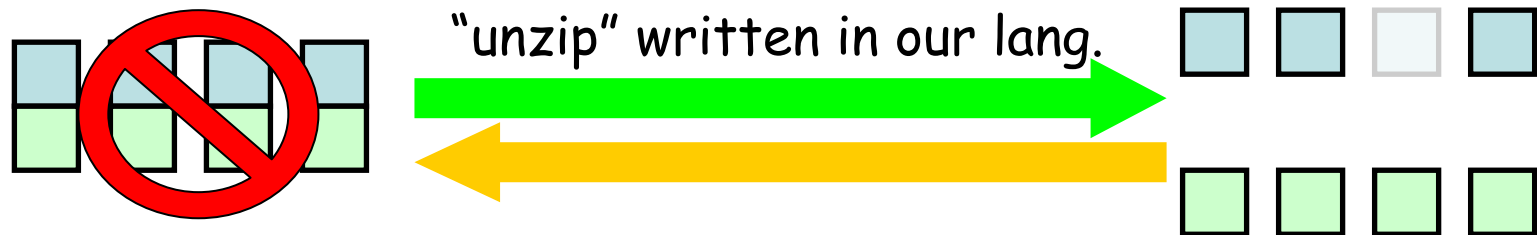
- OElement-Changes



- OInsertion/Deletion (Sync.)



- × Insertion/Deletion (ASync.)



Related Work

- Propagating Updates
[Hu et al 04, Mu et al 04, and so on]
 - Not Side-Effect Free
- Bidirectionalization using Polymorphism
[Voigtländer 09]
 - Side Effect Free
 - × Insertion/Deletion on view of “unzip”