BiFluX: A Bidirectional Functional Update Language for XML

Hugo Pacheco
Joint work with Tao Zan and Zhenjiang Hu

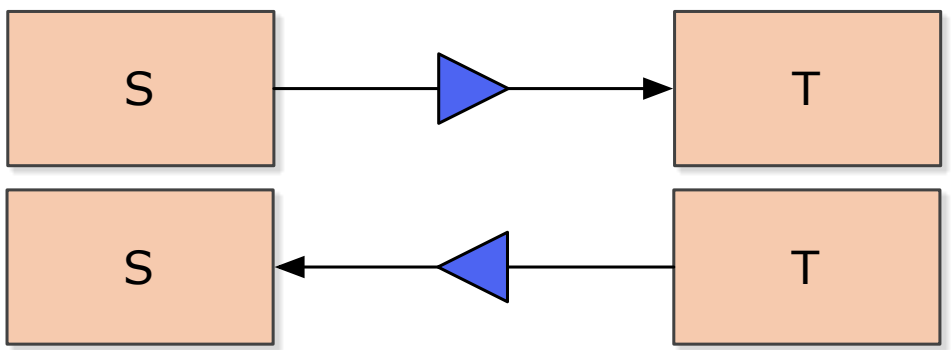
National Institute of Informatics, Tokyo, Japan

BIRS workshop
BX – Theory and Applications Across Disciplines

Banff, December 5th, 2013

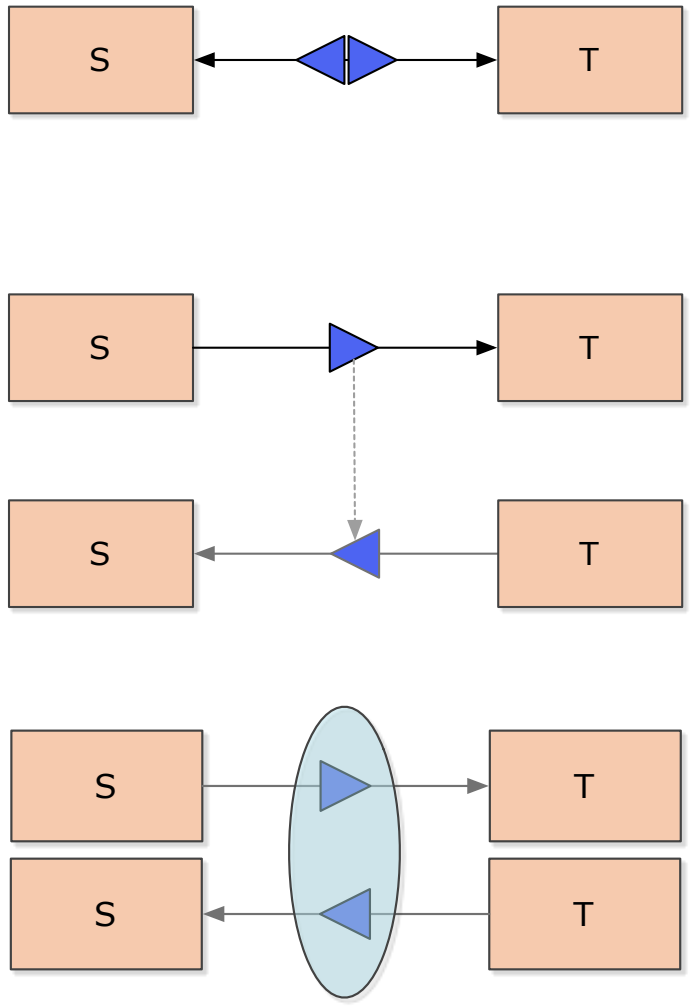
"A mechanism for maintaining the consistency of two (or more) related sources of information."

[Czarnecki et al., ICMT 2009]



BX approaches

- relational: derive both transformations from a relation between the two schemas
- bidirectionalization: write one transformation, derive the other
- combinatorial: write a single program that denotes both transformations

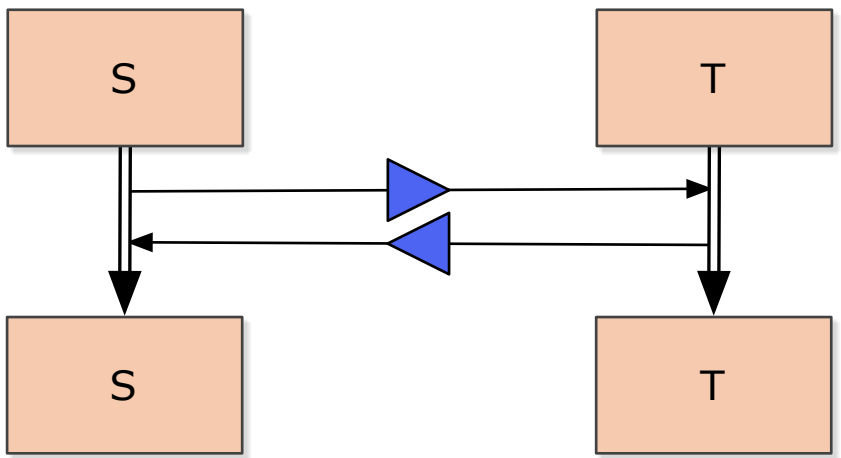


Current Picture

- due to the latent ambiguity of BXs
- existing approaches focus mainly on enforcing consistency
- from the programmer's perspective, they suffer either from:
  - supporting only "trivial" BXs
  - providing arbitrary bidirectional behavior
  - giving little control of what the BX does
  - being impractical to specify complex BXs

BXs = Updates

"Intuitively, a BX translates updates on a source model into updates on a target model, and vice-versa, so that the updated models are consistent."



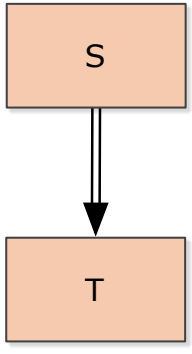
XML Update Languages

- XML transformation languages (XQuery, XSLT, XDuce) are bad for specifying small updates
- a few dedicated languages for in-place XML updates:
  - XQuery Update Facility [W3C]:
    - imperative language
    - ill-understood semantics semantics (aliasing, side-effects, depends on traversal order)
  - Flux (Functional Lightweight Updates for XML) [Cheney, ICFP 2008]:
    - functional language
    - clear semantics
    - straightforward type-checking
  - XUpdate, XQuery!, etc...

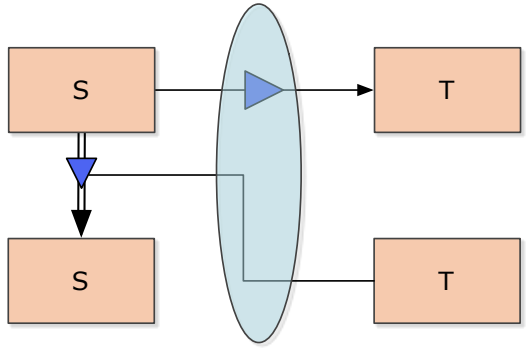
Our proposal: BiFluX

- we propose BiFluX, a bidirectional variant of Flux
- particular class of BXs: lenses, view updating
- modest syntactic extension
  - notion of view (feat. pattern matching, non-in-place updates)
  - static restrictions to ensure well-behavedness

- Flux: fixed input schema & new output schema
- unidirectional in-place semantics



- BiFluX: fixed source and view schemas
- bidirectional semantics as lenses



Is this a bidirectional *update*?

```
UPDATE $source/books/book BY
         INSERT BEFORE title
         VALUE <author>$view</author>
WHERE title = "Through the Looking-Glass"
```

$$S = \mathit{books}\,[\mathit{book}\,[\mathit{author}\,[\mathit{String}]+, \mathit{title}\,[\mathit{String}]]*]$$
$$V = \mathit{String}$$

Is this a bidirectional *update*?

```
UPDATE $source/books/book BY
         INSERT BEFORE title
         VALUE <author>$view</author>
WHERE title = "Through the Looking-Glass"
```

$S = \mathit{books}\ [\mathit{book}\ [\mathit{author}\ [\mathit{String}]+, \mathit{title}\ [\mathit{String}]]*]$
$V = \mathit{String}$

- adds the view as the last author to the source authors
- violates the GetPut law of lenses!

Is this a bidirectional *update*?

```
UPDATE $source/books/book BY
        REPLACE author[last()]
        WITH <author>$view</author>
WHERE title = "Through the Looking-Glass"
```

$S = books\ [book\ [author\ [String]+, title\ [String]]*]$

$V = String$

Is this a bidirectional *update*?

```
UPDATE $source/books/book BY
        REPLACE author[last()]
        WITH <author>$view</author>
WHERE title = "Through the Looking-Glass"
```

$S = \mathit{books}\ [\mathit{book}\ [\mathit{author}\ [\mathit{String}]+, \mathit{title}\ [\mathit{String}]]*]$
$V = \mathit{String}$

- replaces the last author in the source with the view author
- well-behaved *update*!

BiFluX Core Language

- BiFluX $\rightarrow$ core language $\rightarrow$ lenses
- we consider two different semantics
  - default bidirectional semantics as lenses

Hugo Pacheco and Zhenjiang Hu and Sebastian Fischer
Monadic Combinators for "Putback" Style Bidirectional Programming
PEPM 2014.

  - Flux "standard" in-place semantics (insert, delete, ...)

James Cheney
FLUX: FunctionaL Updates for XML
ICFP 2008.

- core BiFluX language:

e	::=	"core XQuery expressions"
p	::=	"simple XPath expressions"
pat	::=	"linear, sequence-based XDuce patterns"
u	::=	"Flux in-place updates"
s	::=	"BiFluX bidirectional updates"

BiFluX Syntax

- BiFluX high-level language (changes to Flux in red):

Stmt	::=	*Upd* [**WHERE** *Expr*] | **IF** *Expr* **THEN** *Stmt* **ELSE** *Stmt*
	|	| *Stmt* ; *Stmt* | { *Stmt* } | **LET** *Pat* = *Expr* **IN** *Stmt*
	|	**CASE** *Expr* **OF** { *Cases* }
Upd	::=	**INSERT** (**BEFORE** | **AFTER**) *PatPath* **VALUE** *Expr*
	|	**INSERT AS** (**FIRST** | **LAST**) **INTO** *PatPath* **VALUE** *Expr*
	|	**DELETE** [**FROM**] *PatPath* | **REPLACE** [**IN**] *PatPath* **WITH** *Expr*
	|	**UPDATE** *PatPath* **BY** *Stmt*
	|	**UPDATE** *PatPath* **BY** *VStmt* **FOR VIEW** *PatPath* [**Match**]
	|	**KEEP** *Path* **AS** (**FIRST** | LAST) | **CREATE VALUE** *Expr*
Cases	::=	*Pat* → *Stmt* | *Cases* ′|′ *Cases*
VStmt	::=	*VUpd* ′|′ *VUpd* | *VUpd*
VUpd	::=	**MATCH** → *Stmt*
	|	**UNMATCHS** → *Stmt*
	|	**UNMATCHV** → *Stmt*
Match	::=	**MATCHING BY** *Path*
	|	**MATCHING SOURCE BY** *Path* **VIEW BY** *Path*
PatPath	::=	[*Pat* **IN**] *Path*

A *bookstore* BiFluX Example

```
UPDATE $book IN $source/book BY
{
  MATCH -> REPLACE price WITH $price
| UNMATCHV -> CREATE VALUE <book category='undefined'>
                             <title/>
                             <author>??</author>
                             <year>??</year>
                             <price/>
                           </book>
}
FOR VIEW book[$title AS v:title, $price AS v:price] IN $view/*
MATCHING SOURCE BY $book/title VIEW BY $title
```

A *bookstore* BiFluX Example: Forward

- Source:

```
<bookstore>
 <book>
   <title >Harry Potter</title>
   <author>J K. Rowling</author>
   <year>2005</year>
   <price>29.99</price>
 </book>
 <book category='Programming'>
   <title >Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price>39.95</price>
 </book>
</bookstore>
```

- View:

```
<books>
 <book>
   <title>Harry Potter</title>
   <price>29.99</price>
 </book>
 <book>
   <title>Learning XML</title>
   <price>39.95</price>
 </book>
</books>
```

A *bookstore* BiFluX Example: Update

- Source:

```
<bookstore>
 <book>
   <title >Harry Potter</title>
   <author>J K. Rowling</author>
   <year>2005</year>
   <price>29.99</price>
 </book>
 <book category='Programming'>
   <title >Learning XML</title>
   <author>Erik T. Ray</author>
   <year>2003</year>
   <price>39.95</price>
 </book>
</bookstore>
```

- Updated View:

```
<books>
 <book>
  <title>XPath for Dummies</title>
  <price>19.99</price>
 </book>
 <book>
   <title>Harry Potter</title>
   <price>19.99</price>
 </book>
 <book>
   <title>Learning XML</title>
   <price>19.99</price>
 </book>
</books>
```

A *bookstore* BiFluX Example: Backward

- Updated Source:

```
<bookstore>
 <book category='undefined'>
   <title>XPath for Dummies</title>
   <author>??</author> <year>??</year>
   <price>19.99</price>
 </book>
 <book>
   <title>Harry Potter</title>
   <author>J K. Rowling</author> <year>2005</year>
   <price>19.99</price>
 </book>
 <book category='Programming'>
   <title>Learning XML</title>
   <author>Erik T. Ray</author> <year>2003</year>
   <price>19.99</price>
 </book>
</bookstore>
```

Summary

- proposed a novel *programming by update* bidirectional paradigm
- presented BiFluX, a bidirectional XML update language
- BiFluX is work in progress (much more under the hood)
- for demos and more info, see...

`http://www.prg.nii.ac.jp/projects/BiFluX`