Bidirectional Transformation and Its Application to Dependable Document Construction

Zhenjiang Hu and Masato Takeichi

Department of Mathematical Informatics
The University of Tokyo

December 7, 2007

(Joint Work with Y. Hayashi, D. Liu, S.C. Mu and K. Nakano)

Bidirectional Transformation and Its Application to Dependable Document Construction

Zhenjiang Hu and Masato Takeichi

Department of Mathematical Informatics
The University of Tokyo

December 7, 2007

(Joint Work with Y. Hayashi, D. Liu, S.C. Mu and K. Nakano)

Bidirectional Transformation and Its Application to Dependable Software Construction

Zhenjiang Hu and Masato Takeichi

Department of Mathematical Informatics
The University of Tokyo

December 7, 2007

(Joint Work with Y. Hayashi, D. Liu, S.C. Mu and K. Nakano)

Bidirectional Transformation and Its Application to Dependable Document Construction

Zhenjiang Hu and Masato Takeichi

Department of Mathematical Informatics
The University of Tokyo

December 7, 2007

(Joint Work with Y. Hayashi, D. Liu, S.C. Mu and K. Nakano)

Outline

Bidirectional Transformation

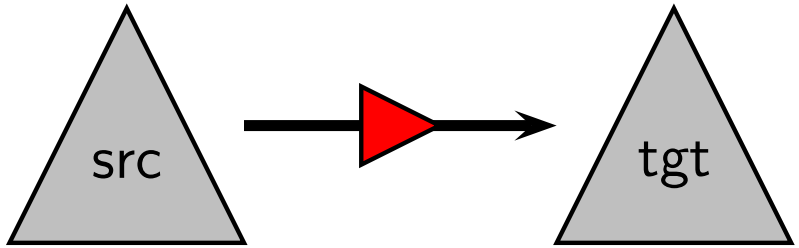
An idea originated from the view-updating technique in the DB community.

Bidirectional Transformation

An idea originated from the view-updating technique in the DB community.

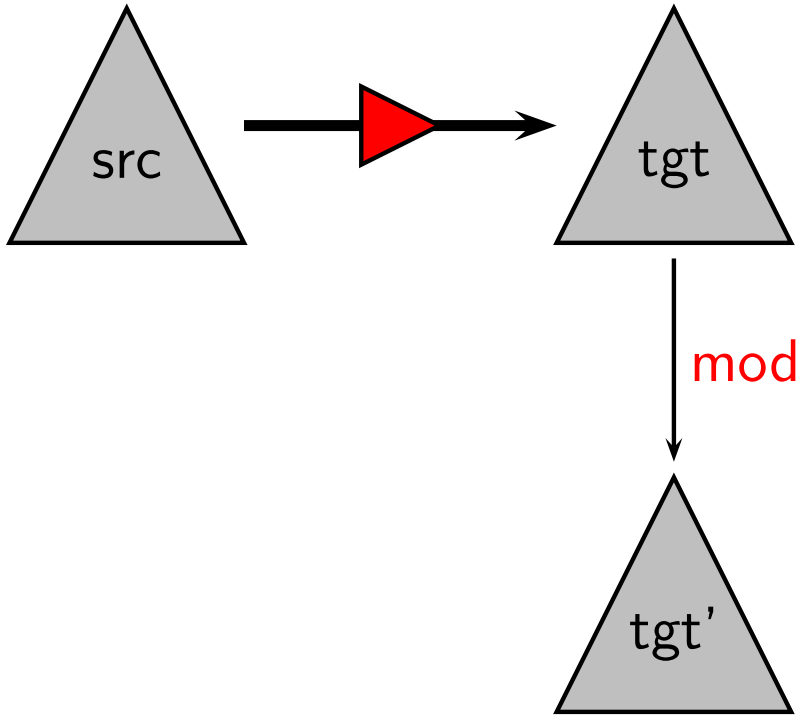
Bidirectional Transformation

An idea originated from the view-updating technique in the DB community.



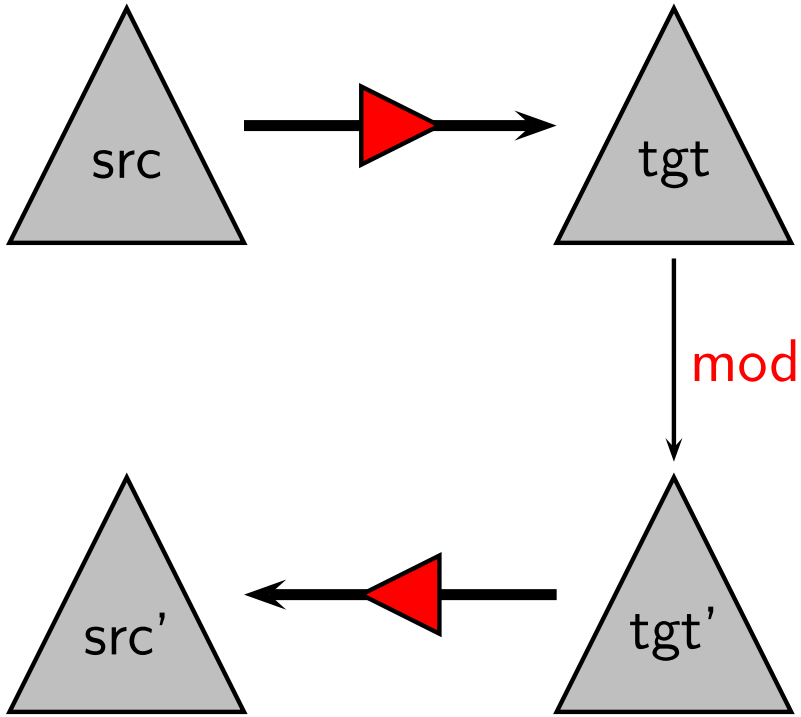
Bidirectional Transformation

An idea originated from the view-updating technique in the DB community.



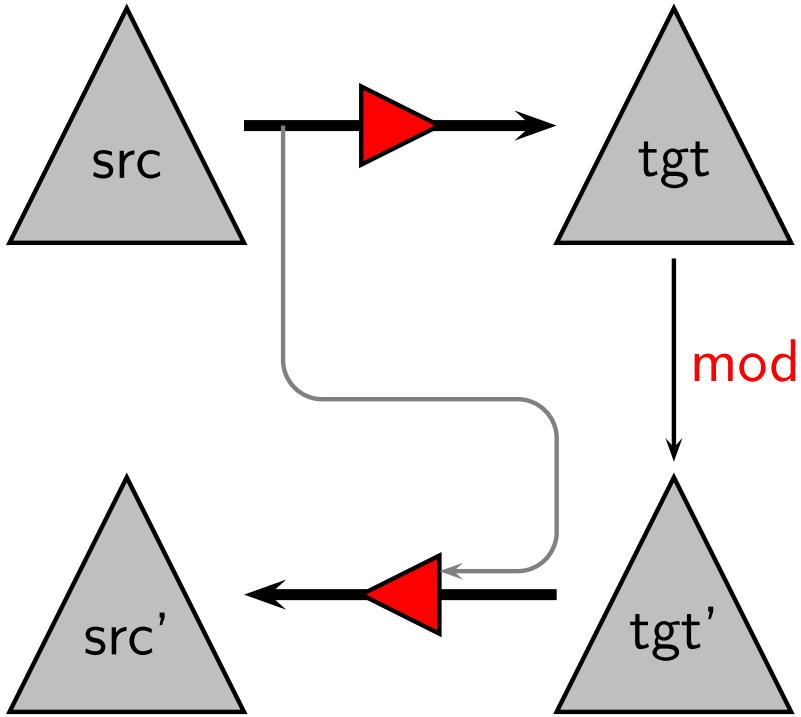
Bidirectional Transformation

An idea originated from the view-updating technique in the DB community.



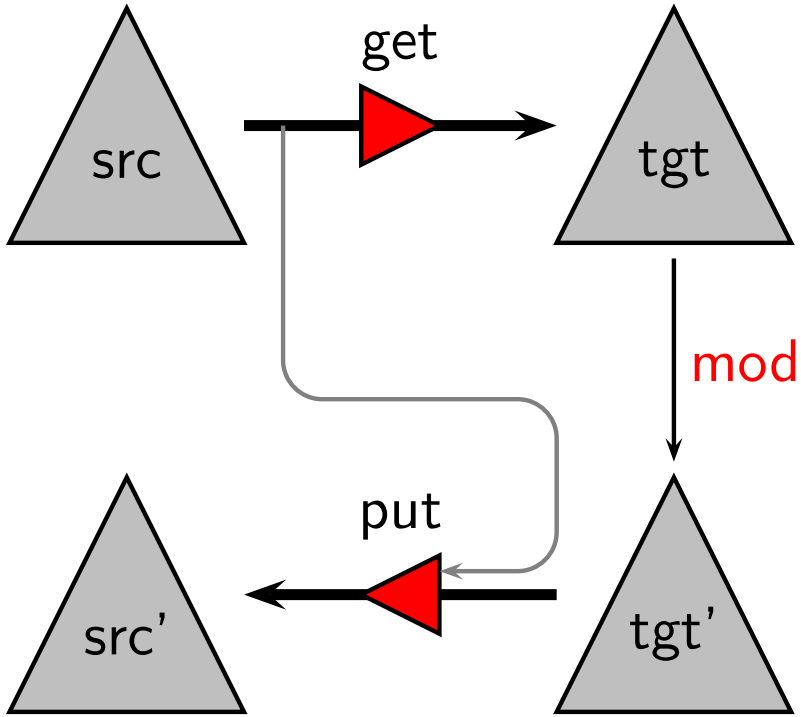
Bidirectional Transformation

An idea originated from the view-updating technique in the DB community.



Bidirectional Transformation

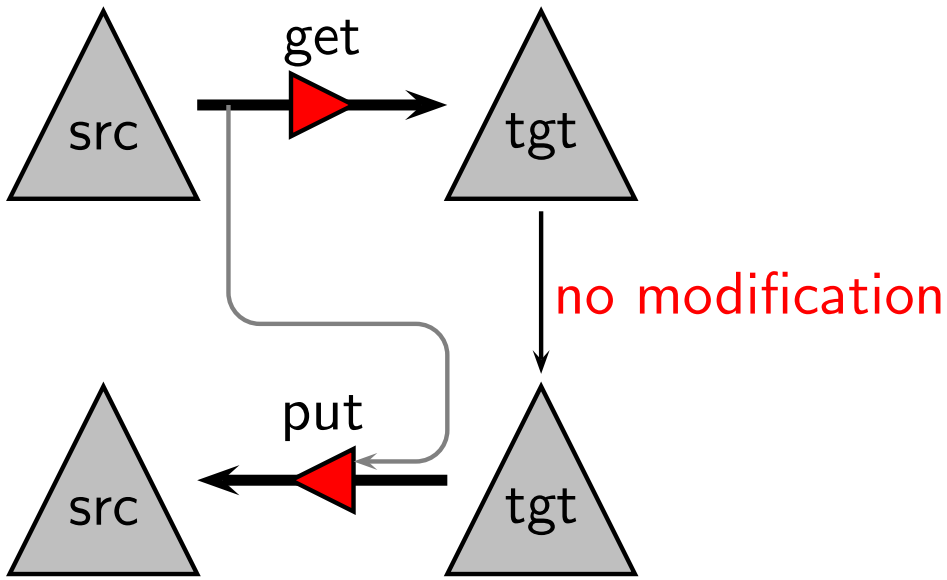
An idea originated from the view-updating technique in the DB community.



It consists of a pair of transformations: forward and backward.

Stability

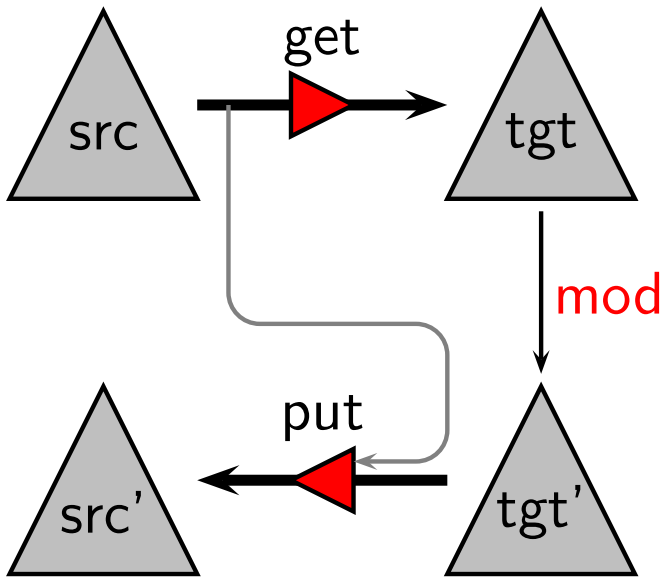
No change on the target implies no change on the source.



$$\mathrm{put}(\mathrm{get}(s), s) = s$$

Reflectivity

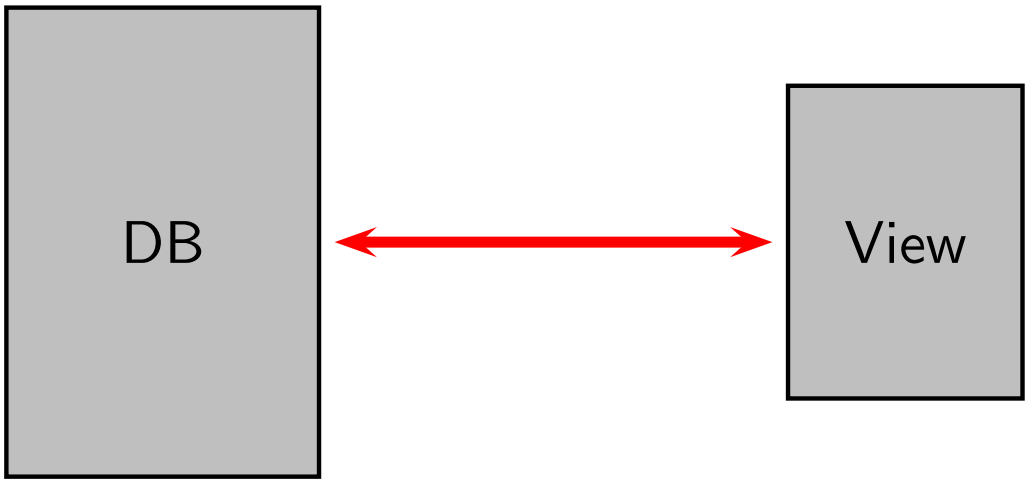
Permitted changes on the target should be reflected to the source.



$$\mathrm{get}(\mathrm{put}(t', s)) = t'$$

View Updating

Reflect changes on the view to the original relational database.



Ref: many studies in the database community
[Bancilhon&Spyratos:81, Dayal&Bernstein, Gottlob et. al.:88]

Replicated Data Synchronization

Synchronization of data in different formats.

Ref: the Homony project in Univ. of Pennsylvania
[Pierce et. al: POPL'05, PODS'06, POPL'08].

Outline

Document Engineering

Document engineering is concerned with principles, tools and processes that improve our ability to create, manage, and maintain documents.

Document Engineering

Document engineering is concerned with principles, tools and processes that improve our ability to create, manage, and maintain documents.

Documents:

- Document Source
- Document View
- View Generator

Document Engineering

Document engineering is concerned with principles, tools and processes that improve our ability to create, manage, and maintain documents.

Documents:

- Document Source — XML source
- Document View — HTML
- View Generator — XSLT

Document Engineering

Document engineering is concerned with principles, tools and processes that improve our ability to create, manage, and maintain documents.

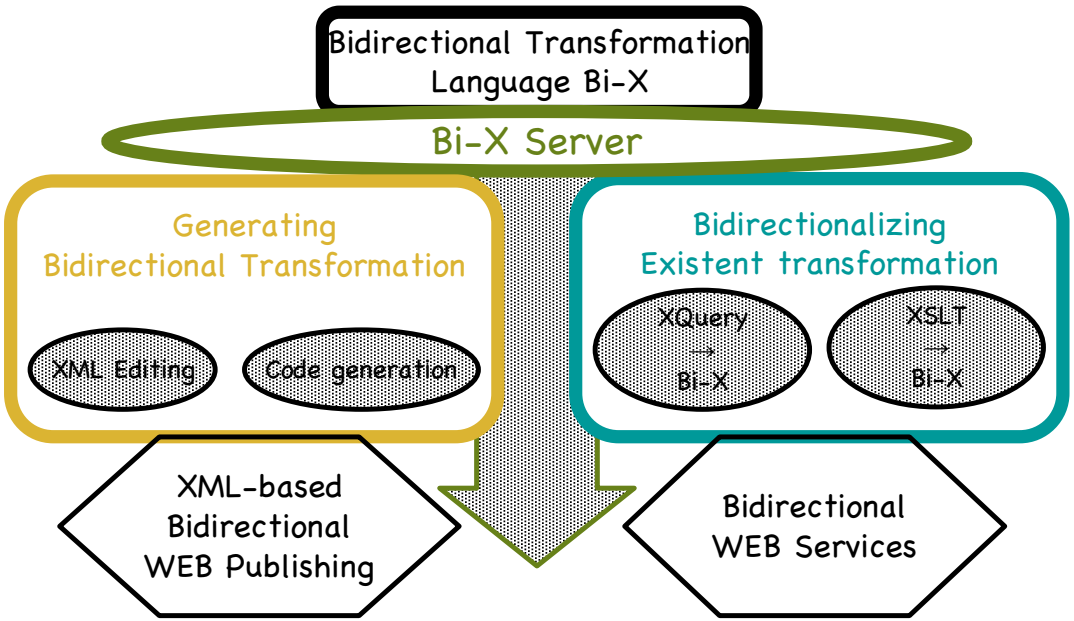
Documents:

- Document Source — XML source
- Document View — HTML
- View Generator — XSLT

Documents are of much simpler structures than softwares.

The PSD Project

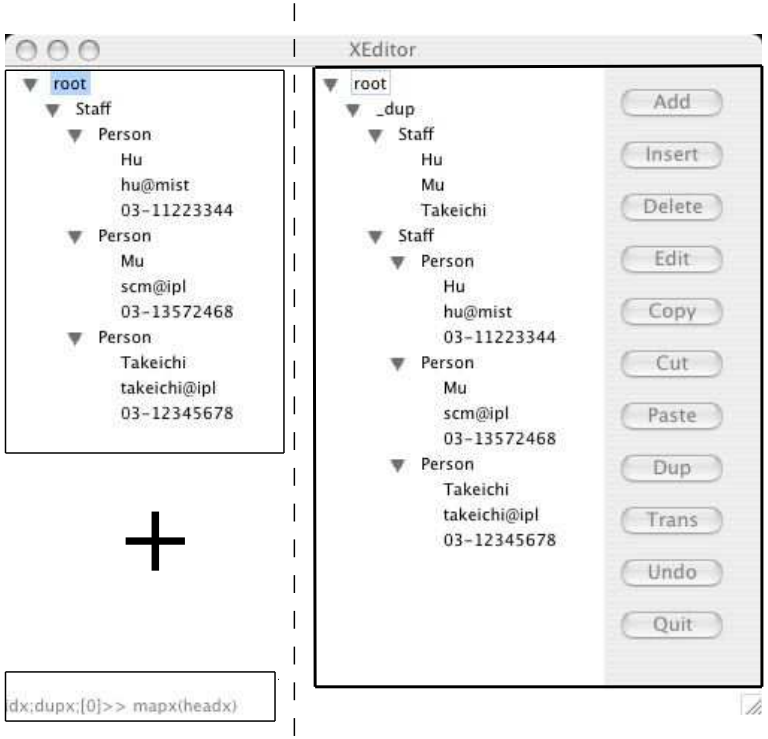
A bidirectional transformation framework for supporting formal creation and maintenance of documents.



Motivation: Applying program transformation techniques to document transformation.

Document Creation

XEditor: Document Construction = View Editing

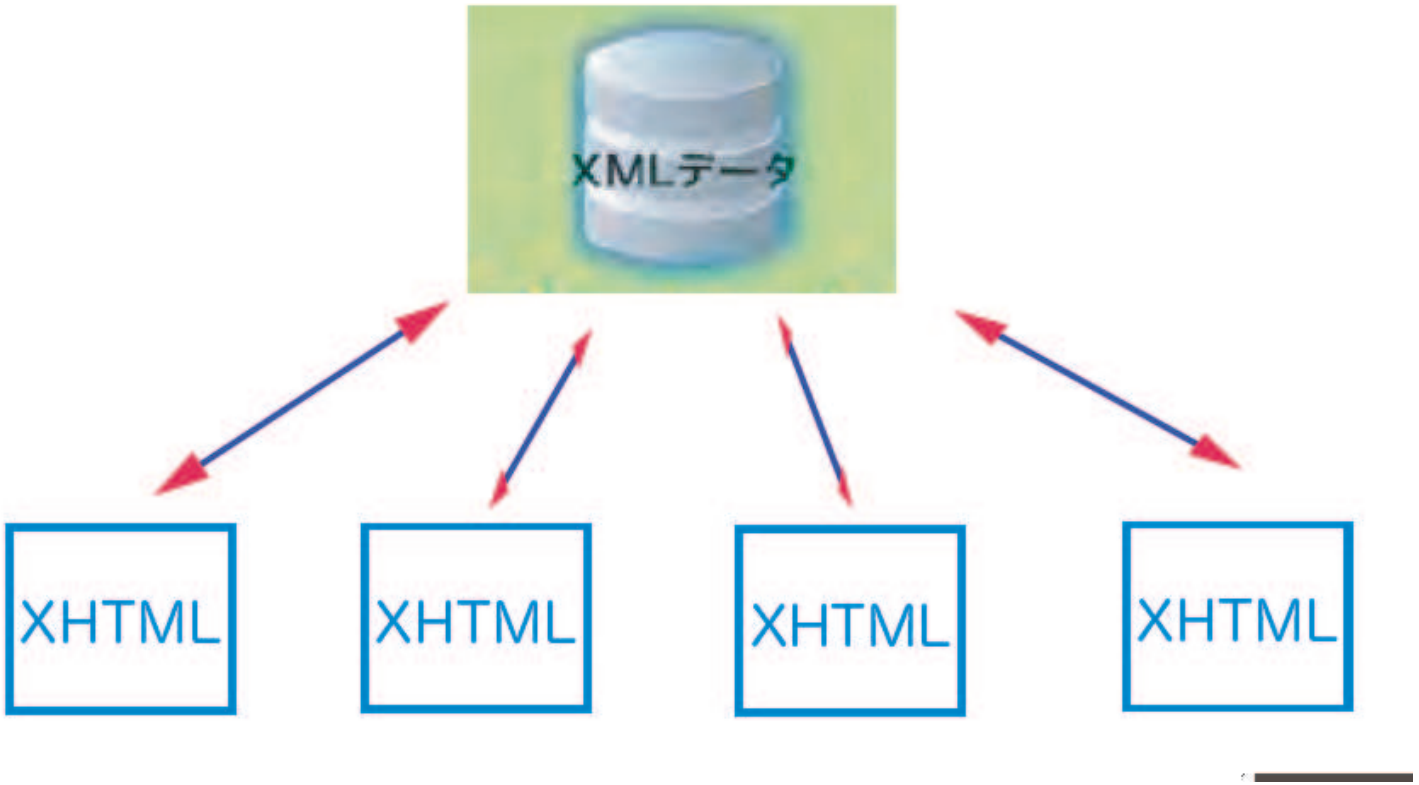


XML + Transformation in X/Inv ⇐ Editing on view

Ref: [MPC'04, PEPM'04, HOSC:07].

Document Maintenance

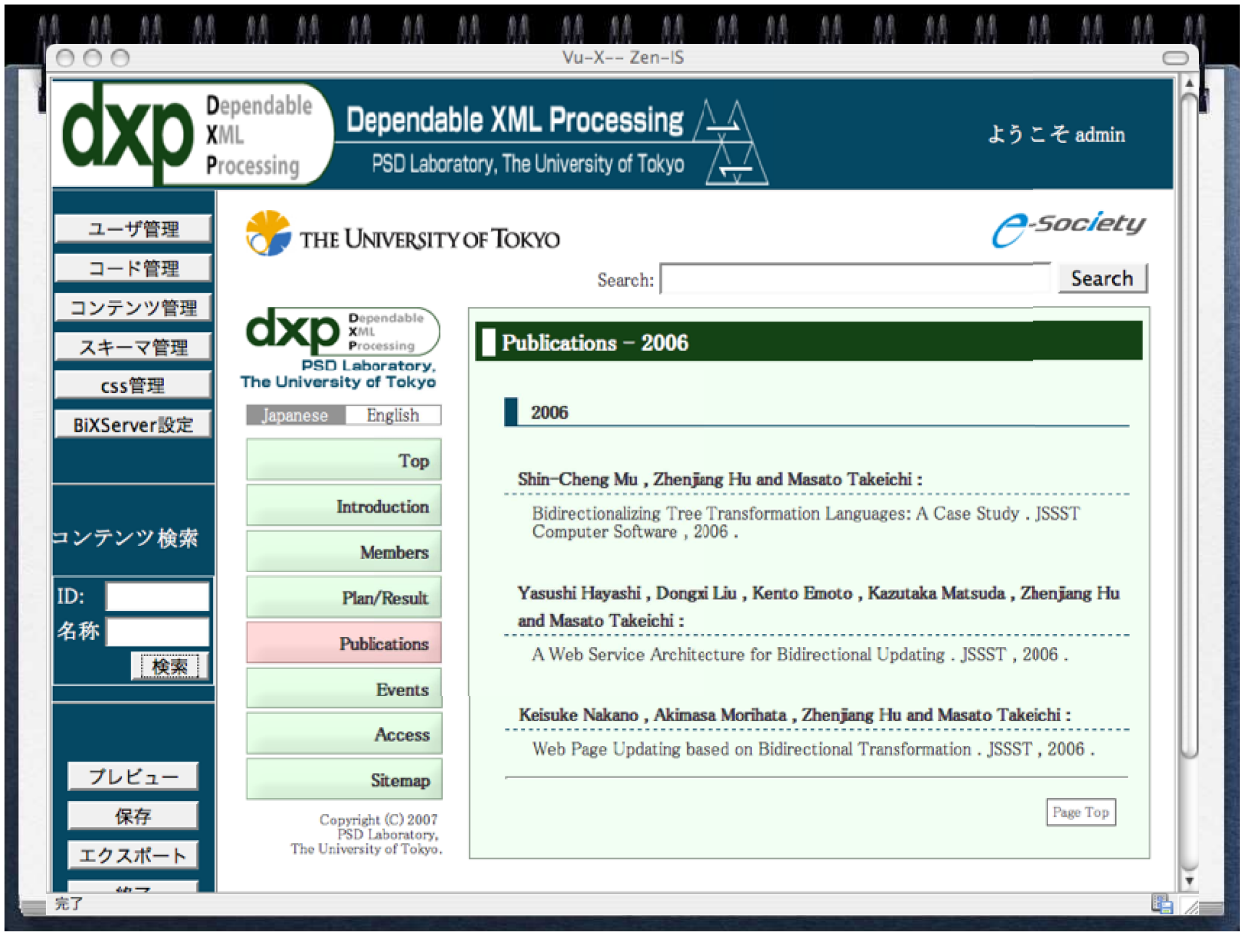
Vu-X: update web pages on browsers in the WYSIWYG manner.



Ref: [JSSST'06, JSSST'07, PEPM'07, APWeb'07].

Document Maintenance

Vu-X: update web pages on browsers in the WYSIWYG manner.



http://www.psdlab.org/vux/

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

- X: a domain-specific bidirectional language for interactive document construction (with the dup primitive)

  [ACM PEPM'04, HOCS:07]

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

- X: a domain-specific bidirectional language for interactive document construction (with the dup primitive)

  [ACM PEPM'04, HOCS:07]

- BiX: a bidirectional transformation language for general XML processing (with function definitions and local bindings)

  [ACM PEPM'07]

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

- X: a domain-specific bidirectional language for interactive document construction (with the dup primitive)

  [ACM PEPM'04, HOCS:07]

- BiX: a bidirectional transformation language for general XML processing (with function definitions and local bindings)

  [ACM PEPM'07]

- ArchX: a domain-specific bidirectional transformation language with regular patterns

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

- X: a domain-specific bidirectional language for interactive document construction (with the dup primitive)

  [ACM PEPM'04, HOCS:07]

- BiX: a bidirectional transformation language for general XML processing (with function definitions and local bindings)

  [ACM PEPM'07]

- ArchX: a domain-specific bidirectional transformation language with regular patterns
- Bi-Haskell: a general bidirectional functional language

Bidirectional Languages

- Inv: an injective language for revertible computation

  [MPC'04, APLAS'04]

- X: a domain-specific bidirectional language for interactive document construction (with the dup primitive)

  [ACM PEPM'04, HOCS:07]

- BiX: a bidirectional transformation language for general XML processing (with function definitions and local bindings)

  [ACM PEPM'07]

- ArchX: a domain-specific bidirectional transformation language with regular patterns
- Bi-Haskell: a general bidirectional functional language

Write forward transformation and get backward transformation for free!

Bidirectionalization

- Bidirectionalization of *XSLT* [JSSST-CS:06]

Bidirectionalization

- Bidirectionalization of *XSLT* [JSSST-CS:06]
- Bidirectional Interpretation of *XQuery* [ACM PEPM'07]

Bidirectionalization

- Bidirectionalization of *XSLT* [JSSST-CS:06]
- Bidirectional Interpretation of *XQuery* [ACM PEPM'07]
- Bidirectional Interpretation of *ATL* [ACM/IEEE ASE'07]

Bidirectionalization

- Bidirectionalization of *XSLT* [JSSST-CS:06]
- Bidirectional Interpretation of *XQuery* [ACM PEPM'07]
- Bidirectional Interpretation of *ATL* [ACM/IEEE ASE'07]
- Bidirectionalization of a small general functional language [ACM ICFP'07]

Outline

BiX

- A Combinator-based language

BiX

- A Combinator-based language
  - Primitive bidirectional transformations for tree manipulation, which is extensible.

BiX

- A Combinator-based language
  - Primitive bidirectional transformations for tree manipulation, which is extensible.
  - Combinators for composing smaller bidirectional transformations

BiX

- A Combinator-based language
  - Primitive bidirectional transformations for tree manipulation, which is extensible.
  - Combinators for composing smaller bidirectional transformations

We only need to prepare a pair of transformations for each primitive bidirectional transformation, other backward transformations are obtained for free.

Primitive Bi-Transformations

Primitive bidirectional transformations for tree manipulation:

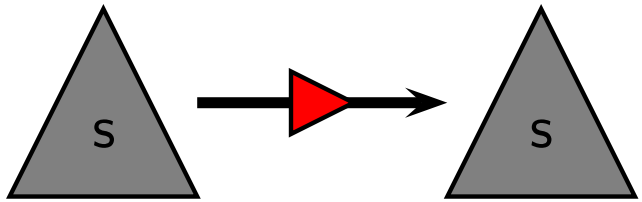
- for tree destruction, e.g.,
  - *xid*
  - *xleftchild*
- for tree construction, e.g.,
  - *xconst t*
  - *xdup*
  - *xnewroot n*

xid

- Identity Transformation

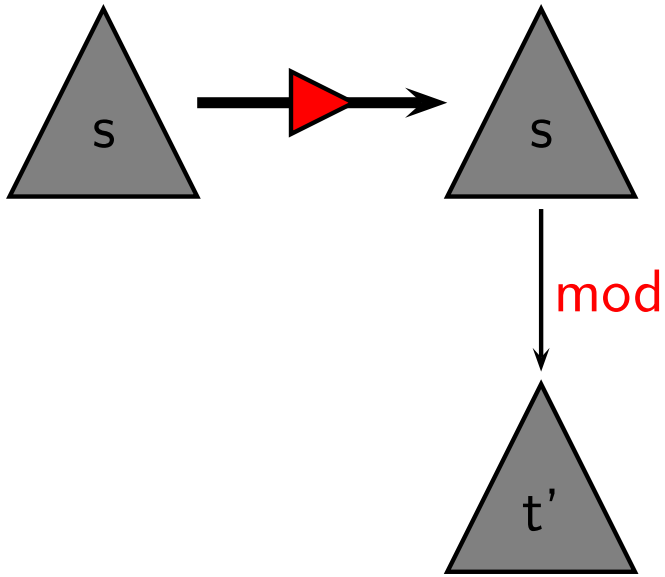
xid

- Identity Transformation



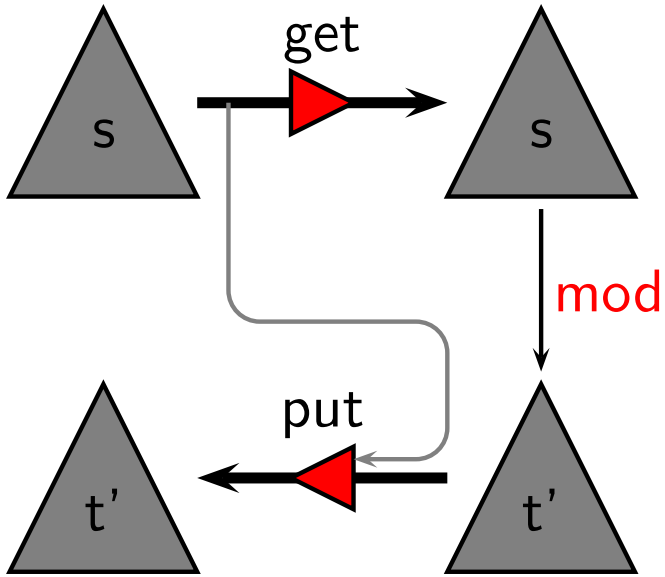
xid

- Identity Transformation



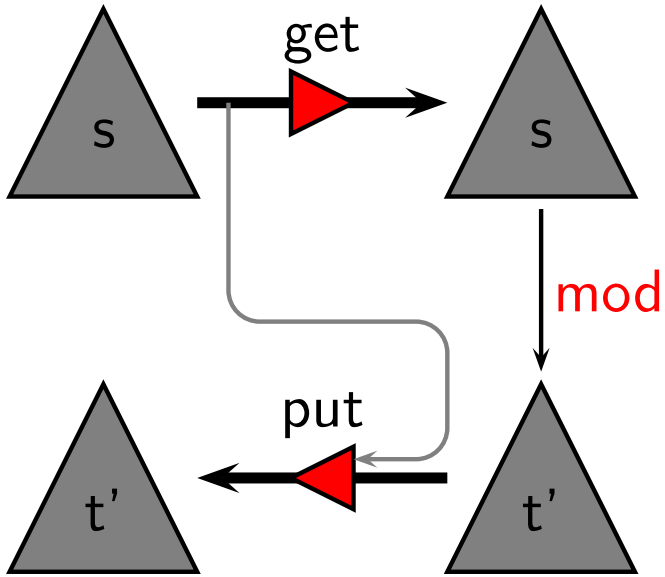
xid

- Identity Transformation



xid

- Identity Transformation



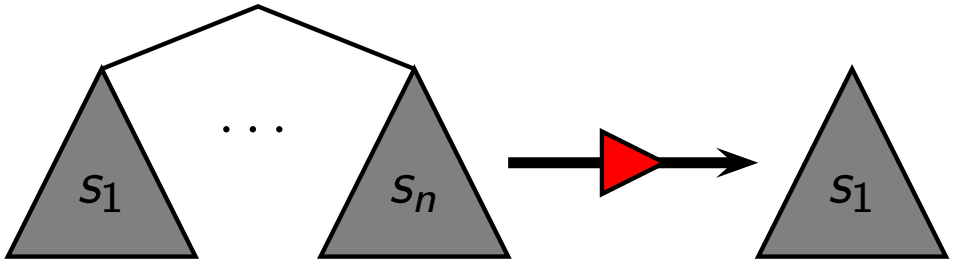
$$\begin{aligned} \text{get } s &= s \\ \text{put } t'\ s &= t' \end{aligned}$$

xleftchild

- Select the leftmost child of the root

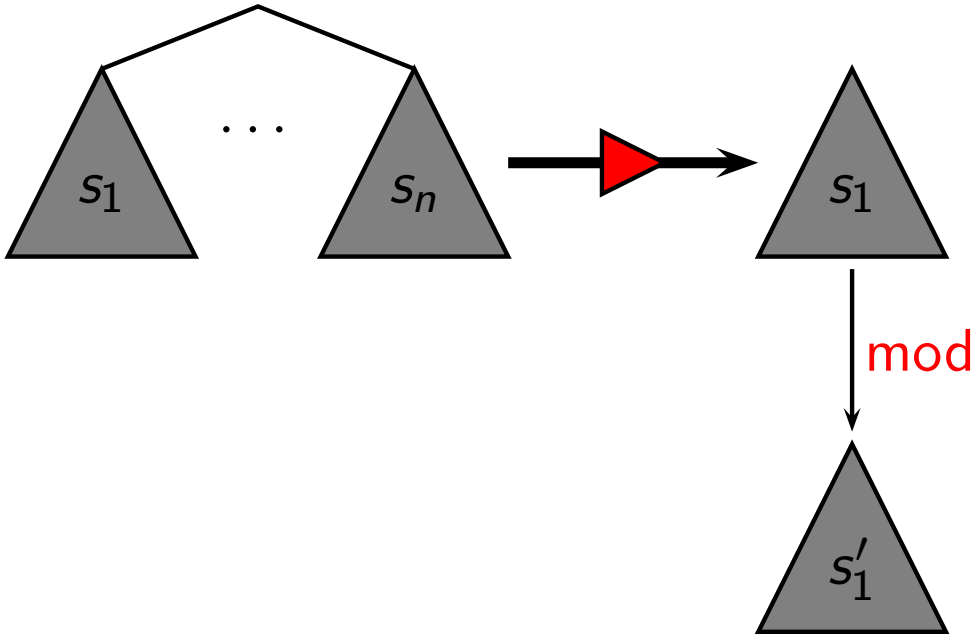
xleftchild

- Select the leftmost child of the root



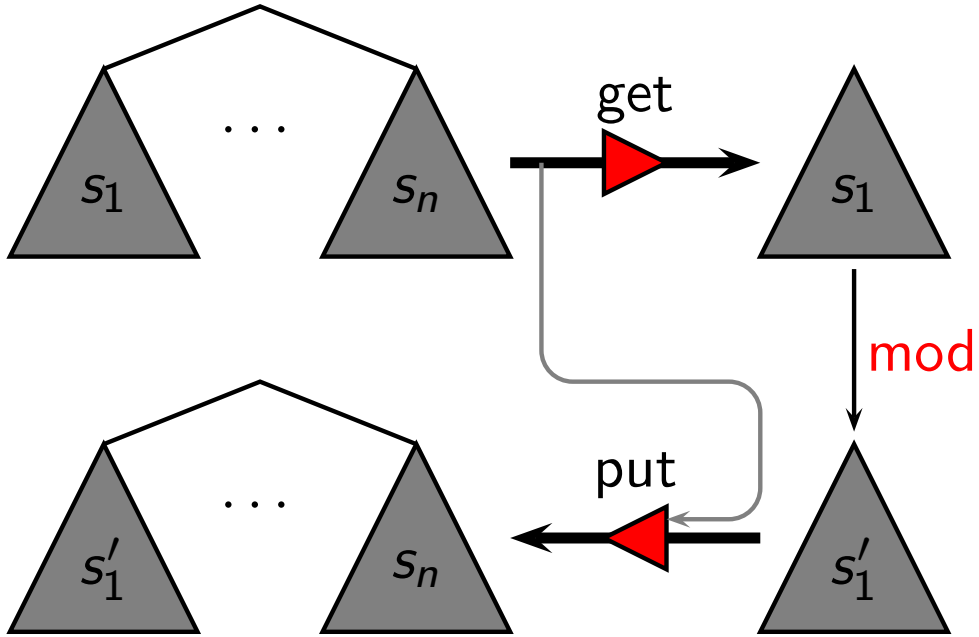
xleftchild

- Select the leftmost child of the root



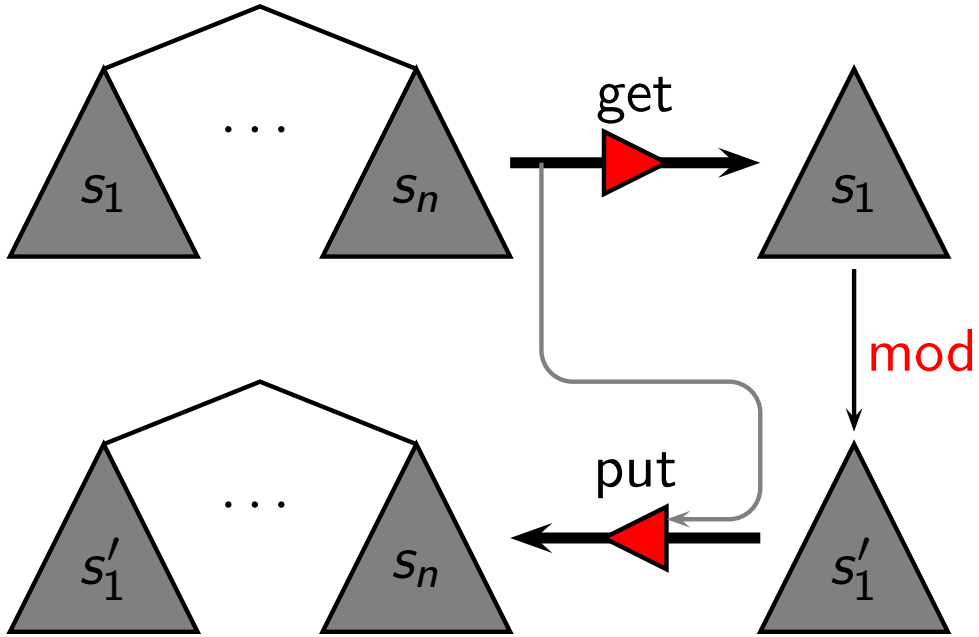
xleftchild

- Select the leftmost child of the root



xleftchild

- Select the leftmost child of the root



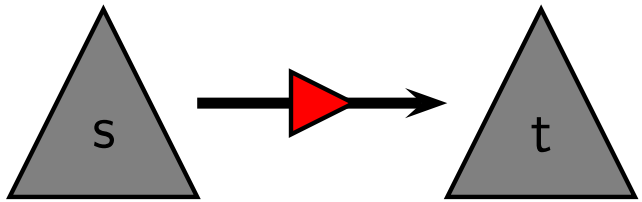
$$\text{get}\ (\mathit{Node}\ n\ [s_1, \ldots, s_n]) = s_1$$
$$\text{put}\ s'_1\ (\mathit{Node}\ n\ [s_1, \ldots, s_n]) = \mathit{Node}\ n\ [s'_1, \ldots, s_n]$$

xconst t

- Constant Transformation:

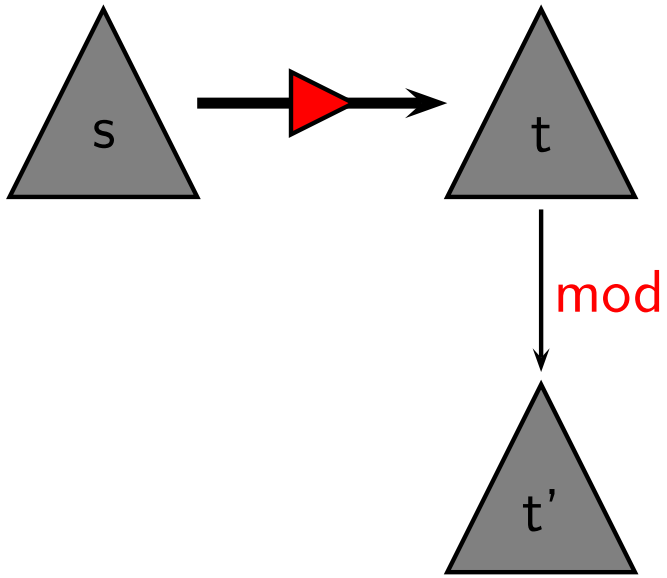
xconst t

- Constant Transformation:



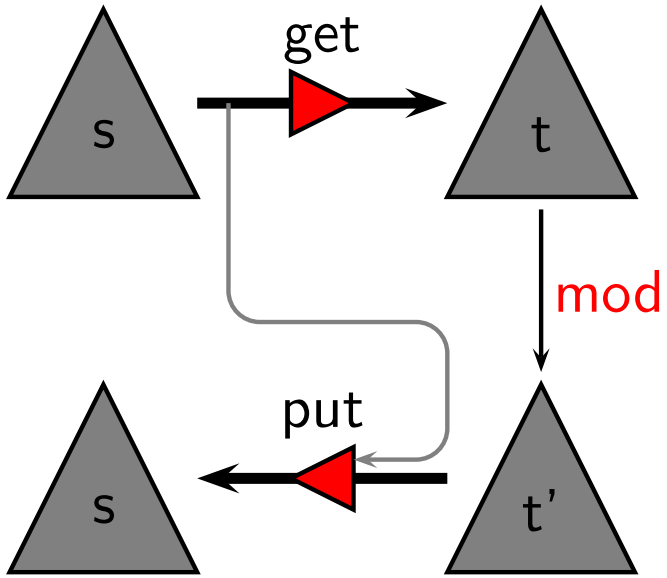
xconst *t*

- Constant Transformation:



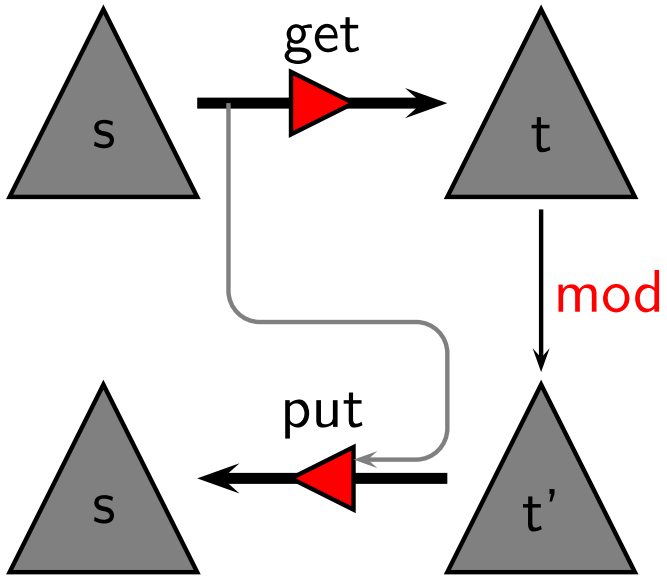
xconst t

- Constant Transformation: disallow any modification on the target.



xconst t

- Constant Transformation: disallow any modification on the target.



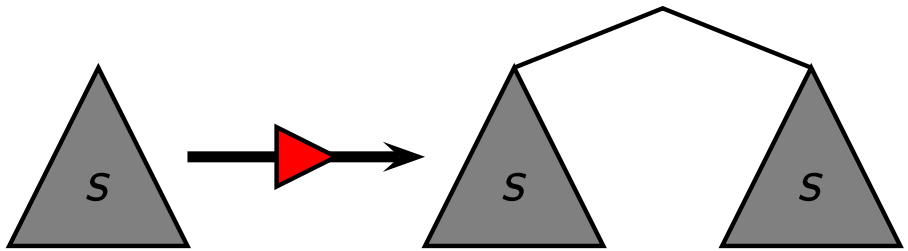
$$\begin{aligned} \text{get } s &= t \\ \text{put } t' \ s &= s \end{aligned}$$

xdup

- Replicate data:

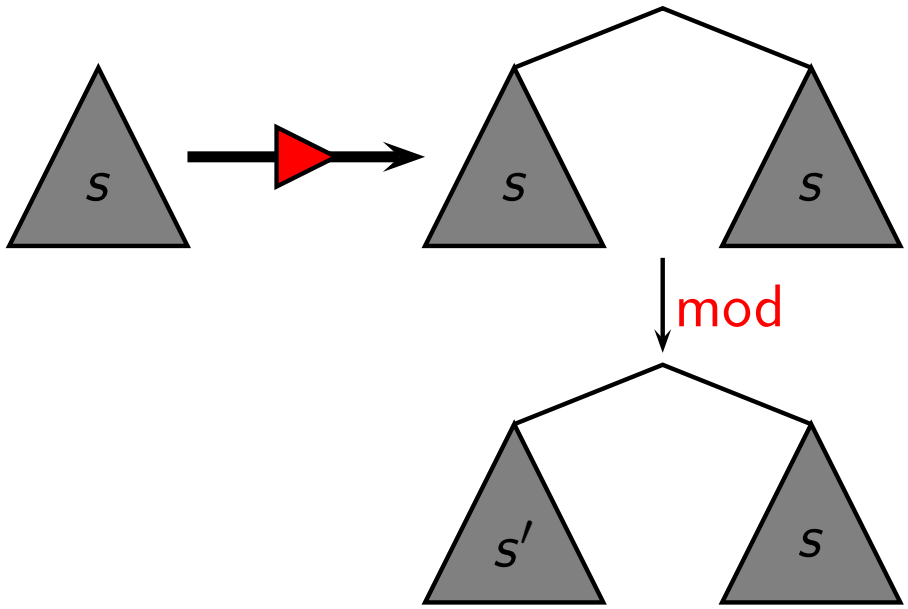
xdup

- Replicate data:



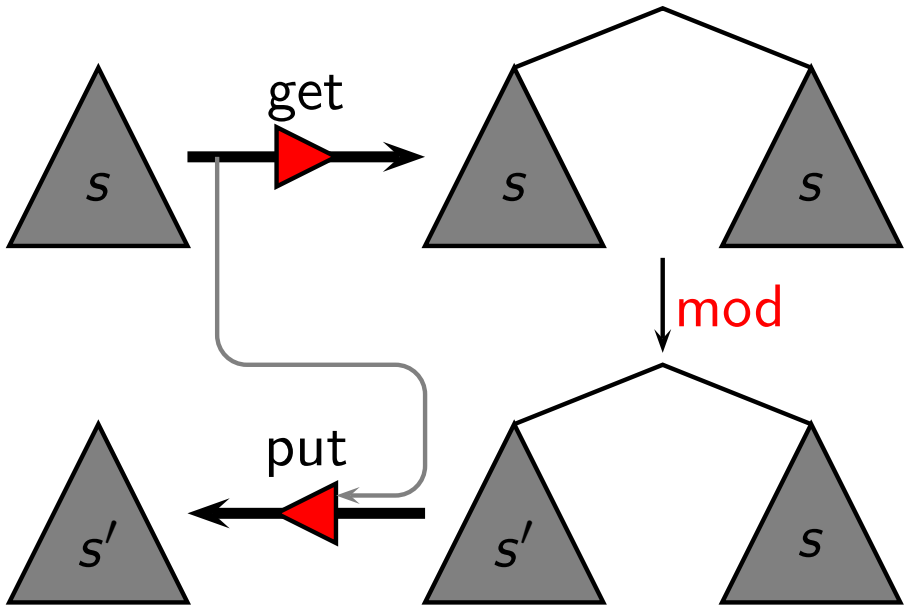
xdup

- Replicate data:



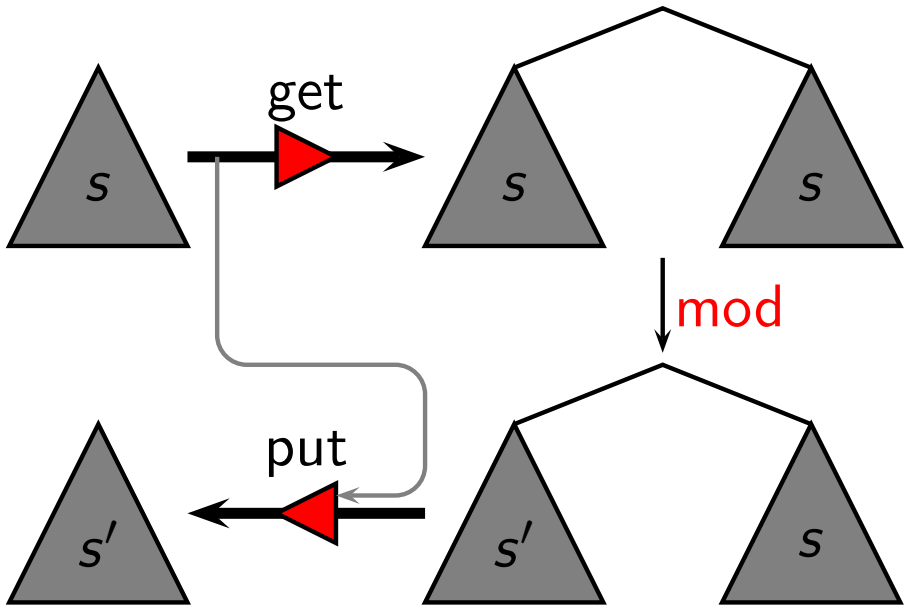
xdup

- Replicate data:



xdup

- Replicate data: allow data dependency in the target



$$\begin{aligned} \text{get } s &= \mathit{Node}\ \_\ [s, s] \\ \text{put } (\mathit{Node}\ \_\ [s_1, s_2])\ s &= \textbf{if } s = s_1 \textbf{ then } s_2 \textbf{ else } s_2 \end{aligned}$$

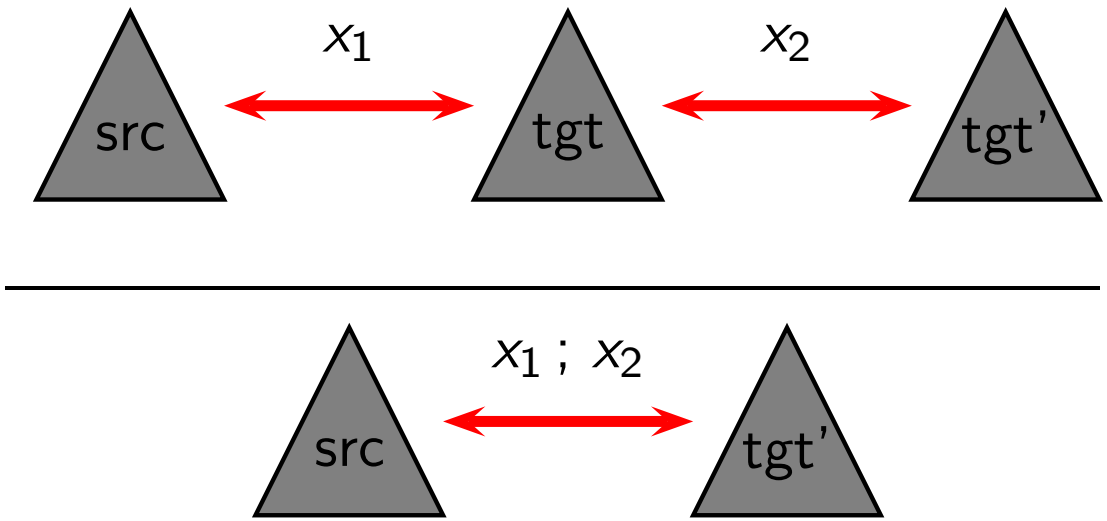
Assume we have only one modification.

Combinators

Used to construct bigger bidirectional transformations by composing smaller bidirectional transformations.

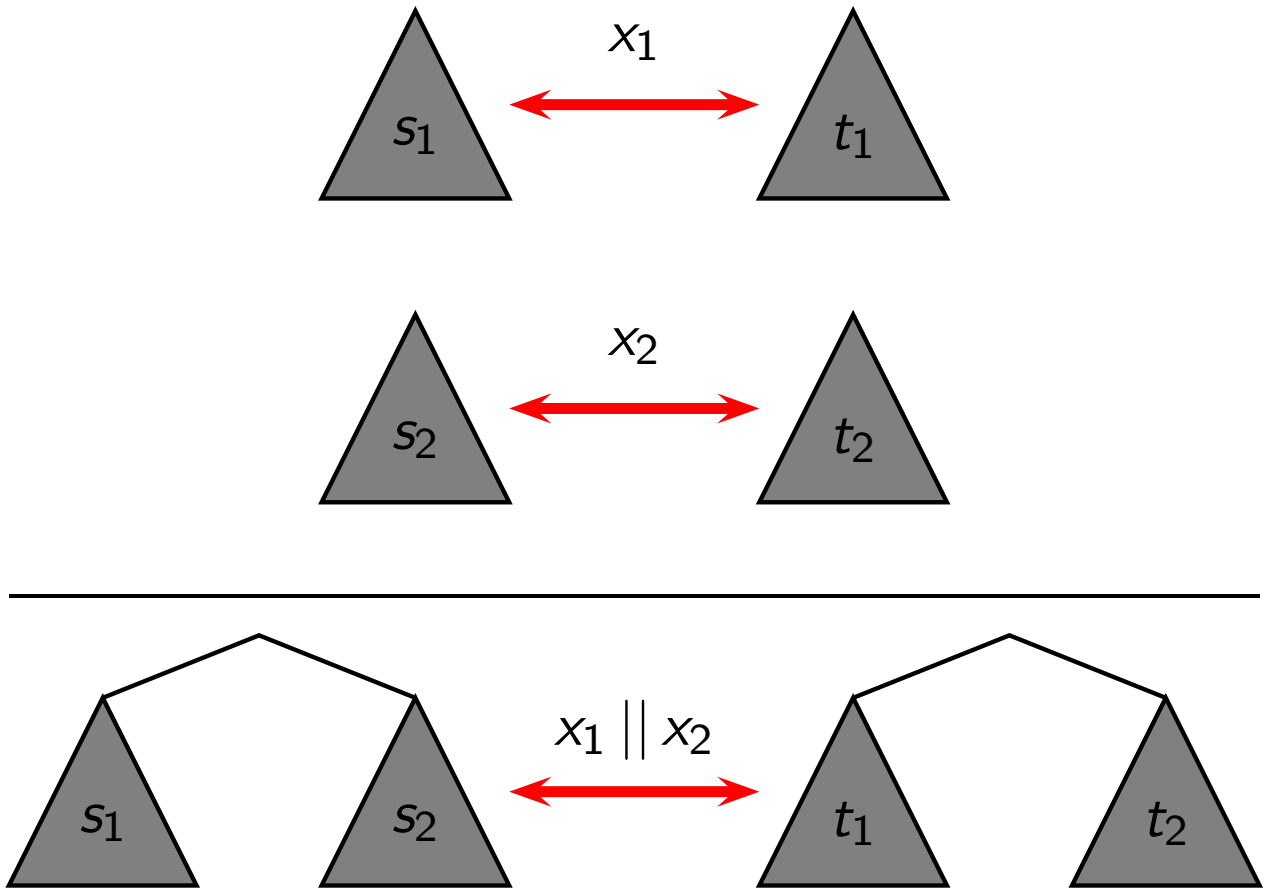
- Sequential composition: $x_1 \; ; \; x_2$
- Parallel composition: $x_1 \; || \; x_2$
- Conditional: xif $p \; x_1 \; x_2$
- Map to each: xmap x

Sequential Composition

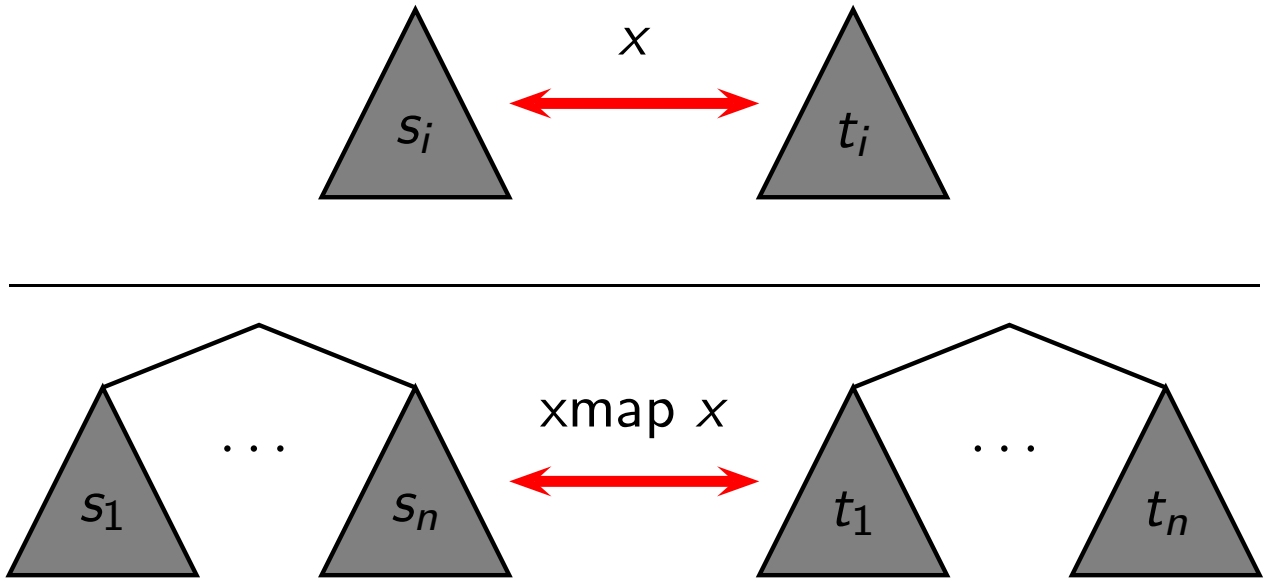


$$\begin{aligned} \text{get}_{(x_1\,;\,x_2)}\ s &= \text{get}_{x_2}(\text{get}_{x_1}\ s) \\ \text{put}_{(x_1\,;\,x_2)}\ t'\ s &= \text{put}_{x_1}\ (\text{put}_{x_2}\ t'\ (\text{get}_{x_1}\ s))\ s \end{aligned}$$

Parallel Composition

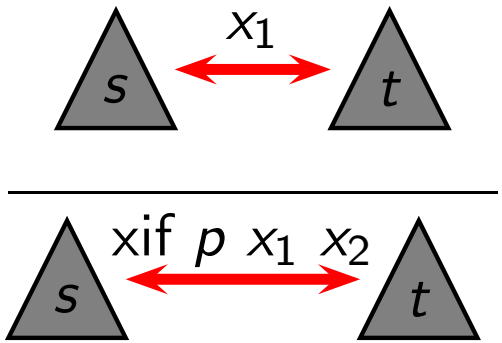


Map-to-each

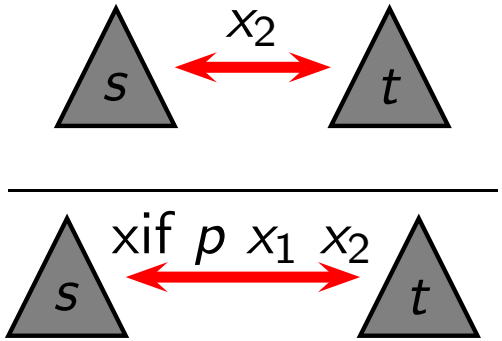


Conditional

If $p\ s$ holds, then



otherwise,



Summary of the Core BiX

$$
\begin{array}{lll}
X & ::= & \mathit{xid} \mid \mathit{xconst}\ S \mid \mathit{xchild} \mid \mathit{xsetcnt}\ X \mid \ldots \\
 & & \mid X_1; X_2 \mid X_1 || X_2 \mid \mathit{xmap}\ X \mid \mathit{xif}\ P\ X_1\ X_2 \\
 & & \mid \mathit{xlet}\ \mathit{Var}\ X \mid \mathit{xvar}\ \mathit{Var} \\
 & & \mid \mathit{xfunapp}\ \mathit{fname}\ [X_1, ..., X_n] \\
 & & \\
P & ::= & \mathit{xwithtag}\ \mathit{str} \mid \mathit{xistext} \mid \mathit{xiselement} \mid X \\
 & & \\
\mathit{Def} & ::= & \mathit{fun}\ \mathit{fname}(\mathit{Var}_1, ..., \mathit{Var}_n) = X
\end{array}
$$

A Programming Example: toc

- Source Data:

```
<book>
  <title>Data on the Web</title>
  <author>Serge</author><author>Peter</author>
  <author>Dan Suciu</author>
  <section id="intro" difficulty="easy">
    <title>Introduction</title><p>Text ... </p>
    <section>
      <title>Audience</title><p>Text ... </p>
    </section>
  </section>
  <section id="syntaxnew" difficulty="medium">
    <title>A Syntax For Data</title><p>Text ... </p>
  </section>
</book>
```

A Programming Example: toc

- Target data we want to have is a table-of-contents.

```
<toc>
  <section id="intro">
    <title>Introduction</title>
    <section><title>Audience</title></section>
  </section>
  <section id="syntaxnew">
    <title>A Syntax For Data</title>
  </section>
</toc>
```

A Programming Example: toc

- The transformation in BiX to produce the table of contents from a book:

```
fun toc($book-or-section) =
  xvar $book-or-section; xchild;
  xmap (xif (xwithtag ‘‘section’’) X0 (xconst ()))
where
  X0 = xlet $section
            (xconst <section>[]; xsetcnt (X1||X2))
  X1 = xvar $section; xchild;
       xmap (xif (xwithtag ‘‘title’’) xid (xconst ()))
  X2 = xfunapp toc [xvar $section]
```

We can obtain BiX from XQuery

- Making the table-of-contents from a book.

```
declare function local:toc($book-or-section)
{
  for $section in $book-or-section/section
  return
    <section>
      {$section/@id, $section/title,
       local:toc($section)}
    </section>
};
<toc>
   { for $s in doc("book.xml")/book
     return local:toc($s)}
</toc>
```

Outline

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12:

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1:

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4:

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4: We started a joint project (in Japan) for it.

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4: We started a joint project (in Japan) for it.
  - 2007.11:

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4: We started a joint project (in Japan) for it.
  - 2007.11: We announced our first result in ASE'07.

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4: We started a joint project (in Japan) for it.
  - 2007.11: We announced our first result in ASE'07.
  - 2007.12:

Software Engineering Wants Bidirectional Transformation

- Software companies ask for it!
- Many researchers in ASE are talking about it!

- Specifically, ABC would be empowered by it!
  - 2003.12: We thought it at Robust Software Construction Workshop.
  - 2006.1: We started a joint project (in China) for it.
  - 2007.4: We started a joint project (in Japan) for it.
  - 2007.11: We announced our first result in ASE'07.
  - 2007.12: We are at our first joint workshop on it.

Bidirectional Software Engineering is Challenging

- Software development requires precise but understandable semantics of bidirectional transformations for communication among developers at different stages.

Bidirectional Software Engineering is Challenging

- Software development requires precise but understandable semantics of bidirectional transformations for communication among developers at different stages.
- Transformation in software engineering is mostly applied to graphs instead of trees.

Bidirectional Software Engineering is Challenging

- Software development requires precise but understandable semantics of bidirectional transformations for communication among developers at different stages.
- Transformation in software engineering is mostly applied to graphs instead of trees.
- Usefulness of bidirectional software engineering cannot be convinced if it is not applied to large-scale applications.

Bidirectional Software Engineering is Challenging

- Software development requires precise but understandable semantics of bidirectional transformations for communication among developers at different stages.
- Transformation in software engineering is mostly applied to graphs instead of trees.
- Usefulness of bidirectional software engineering cannot be convinced if it is not applied to large-scale applications.

Two Questions for this Workshop

- What kind of transformation languages are required for describing software development process (from specification to final implementation) in ABC?

Two Questions for this Workshop

- What kind of transformation languages are required for describing software development process (from specification to final implementation) in ABC? Is ATL fine, if we do not think about backward transformation?

Two Questions for this Workshop

- What kind of transformation languages are required for describing software development process (from specification to final implementation) in ABC? Is ATL fine, if we do not think about backward transformation?
- Which part of ABC is better to be used as the first experiment of "fusion" with bidirectional transformation?

Two Questions for this Workshop

- What kind of transformation languages are required for describing software development process (from specification to final implementation) in ABC? Is ATL fine, if we do not think about backward transformation?
- Which part of ABC is better to be used as the first experiment of "fusion" with bidirectional transformation?

software engineering +
programming languages +
transformational programming $\Rightarrow$ a new paradigm