Bidirectional XML Transformation with Bi-X*

Dongxi Liu Yasushi Hayashi Keisuke Nakano Zhenjiang Hu Masato Takeichi
Department of Mathematical Informatics, University of Tokyo

Abstract

Bi-X is an expressive bidirectional XML transformation language, featuring that one program can be executed in two directions. In one direction, a Bi-X program transforms source XML documents into target XML documents, while in the other direction, it transforms the updated target documents together with original source documents into updated source documents. In this paper, we give a summary of the language Bi-X and introduce some applications of using Bi-X.

1 Introduction

XML is a widely used format of exchanging data over network. When an XML document is exchanged between different systems, it always needs to be transformed, and the target document may have different structure and content. For example, a source XML document may be transformed into a smaller HTML document, so that this target document can be displayed in web browsers and contain only interesting contents to users.

There have been many languages, such as XSLT and Java, designed specially to or with libraries to support XML transformation. These languages reduce the burden of programmers to develop XML transformation programs. However, they cannot provide help for maintaining content consistency between the source and target XML documents if the latter are changed. For example, we transform a source XML document into a target document with an XSLT transformation, and later, we notice a data error in the target document and naturally make a correction to that data, resulting in inconsistent source and target documents. Unfortunately, the XSLT transformation we used cannot help solve this problem. In this paper, we introduce the bidirectional XML transformation language Bi-X developed in our PSD project. Bi-X programs can not only implement XML transformation as the existing transformation languages but also offer the ability to synchronize the source and target XML documents after transformation.

Bi-X is a bidirectional language in the sense that a program can be executed in two directions. In one direction (forward direction), the program transforms a source XML document into a target XML document, while in the other direction (backward direction), the program takes as input the target document and the original source document and produces a new source document. If the target document is changed, backward execution will reflect these changes into the new source document.

This work is supported by Comprehensive Development of e-Society Foundation Software Program of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

Main = <xseq><xchild/> *GetTitles SetTitles* </xseq>
GetTitles = <xmap><xif>P_0 X_0 <xconst/></xif></xmap>
where P_0 = <xwithtag>sec</xwithtag>
X_0 = <xseq><xchild/><xmap> X_{00} </xmap></xseq>
X_{00} = <xif><xwithtag>title</xwithtag><xid/><xconst/></xif>
SetTitles = <xlet><var>$titles</var><xseq>$X_1$ X_2</xseq></xlet>
where X_1 = <xconst><titles/></xconst>
X_2 = <xsetcnt><xvar>$titles</xvar></xsetcnt>

Figure 1: Bi-X Code For Section Titles

There have been several bidirectional transformation languages [1, 2]. Compared with these exsiting bidirectional transformation languages, Bi-X is quite expressive and we have demonstrated its expressiveness by using it to interpret XQuery, which is a powerful functional language for querying XML data [3].

2 An Example of Bi-X

Suppose you are revising a paper, which is stored in the following XML document.

```
<paper>
  <title>Bi-X</title>
  <sec>
    <title>Introduction</title>
    <paras><txt>text1</txt></paras>
  </sec>
  <sec>
    <title>An Example</title>
    <paras><txt>text2</txt><fig>fig2</fig></paras>
  </sec>
  <sec>
    <title>Bi-X</title>
    <paras><tab>tab3</tab><txt>text3</txt><paras>
  </sec>
</paper>
```

Depending on your particular situation, you may revise the paper in different ways. For example, you may go out of your office for a walk and hence it is convenient for you to revise only the section titles, for example, with your mobile phone, since the whole paper may be too big to be put into your phone. For this purpose, we could generate the following small document for you by the Bi-X code in Figure 1.

```
<titles>
  <title>Introduction</title>
  <title>An Example</title>
  <title>Bi-X</title>
</titles>
```

On this target document, you can edit the titles, delete titles and insert new titles. After going back to your office, you just need to execute backward the code in Figure 1 and those changes on the small document will be automatically reflected into the document of the whole paper. For example, if you change the second title from "An Example" into "An Example of Bi-X", then after backward execution the title of the second section in the paper is also changed into "An Example of Bi-X"; if you insert a new `title` element containing the string "Applications" after the third title, then after backward execution the updated paper will include the fourth section, which contains

X ::= <xid/> | <xconst>*Val*</xconst> | <xchild/>
| <xsetcnt>$X_1 ... X_n$</xsetcnt> | <xif>P X_1 X_2</xif>
| <xseq>$X_1 ... X_n$</xseq> | <xmap>X</xmap>
| <xlet><var>*Var*<var>X</xlet> | <xvar>*Var*</xvar>
| <xfunapp><name>*fname*<name>
<args>$X_1 ... X_n$<args></xfunapp>
P ::= <xwithtag>*str*</xwithtag> | <xeq>X_1 X_2</xeq>

Figure 2: Bi-X Constructs

the newly inserted `title` element but no `paras` element since the updated target document does not provide such information.

3 Language Constructs in Bi-X

Bi-X contains a collection of constructs for writing bidirectional XML transformation programs. Some constructs are given in Figure 2. Each construct is an XML element. Below, we will explain those constructs.

The constructs `xid` and `xconst` implement identity and constant transformation, respectively. The constant returned by `xconst` is its argument *Val*. XML elements can be deconstructed by `xchild` and constructed by `xsetcnt`. That is, `xchild` returns the content of the source element, and `xsetcnt` sets the new contents computed by its argument transformations X_i ($1 \leq i \leq n$) into the source element.

The conditional construct `xif` executes X_1 if P holds, otherwise executes X_2; the predicate `xwithtag` holds if the source data of `xif` has the tag specified by its argument *str*, and the predicate `xeq` holds if its two argument transformations return identical XML data.

A sequence of transformations can be composed by using `xseq`, which executes its argument transformation sequentially with the output of the preceding transformation as the input of the succeeding one. The construct `xmap` apples its argument transformation onto each item in the source data, and their results are concatenated in order as the transformation result.

The construct `xlet` binds the source data to the variable *Var* and then executes its argument transformation, and the bound variable can be referenced by `xvar`. This variable binding mechanism contributes much to the expressiveness of Bi-X.

The function call is implemented by the `xfunapp` construct. A function is declared in the form below, where *fname* is the function name, Var_i the parameters and X the function body.

```
<function name="fname"
  arg1="Var1" ... argn="Var2">
  X
</function>
```

Functions can be reused by different transformation programs by putting them into Bi-X libraries. A library is imported into a Bi-X program by either of the following XML processing instructions. The second form is for accessing a library over network.

<?import library = "*filename*"? >

<?import library = "*URI*"? >

Bi-X is a typed language. The types in Bi-X have the same expressiveness as the widely-used Document Type Definition (DTD). Given the type of source data and a Bi-X program, the Bi-X interpreter automatically infers the type of the target document. Moreover, it guarantees if the source data conforms to the declared type, then the target document generated by forward execution also has the inferred type for it, and if the updated target data is well-typed, then after backward execution, the updated source data is also well-typed.

During type inference, Bi-X interpreter annotates some constructs with type information. These type information will play a role of guiding backward executions when updated target data includes inserted elements. For example, `xif` is annotated with the view types of its two branches and during backward transformation these types will be used to determine which branch will be used to process the newly inserted target data when this data does not have corresponding source data to check the predicate of `xif`.

The implementation of Bi-X is available publicly at [4], where a user manual describes all language constructs and type definitions in Bi-X.

4 Applications of Bi-X

We have applied Bi-X in several applications. Our experience demonstrates that Bi-X can be used in practical applications that need bidirectional transformation. Three applications are introduced below.

Vu-X [5] is a WYSIWYG web site maintainer based on bidirectional transformation. We have generated all web pages of our lab from a central XML document by using Bi-X transformation. By this way, those web pages can be maintained conveniently with Vu-X.

XQuery is a powerful functional language for querying XML data. We have used Bi-X as a target language to interpret XQuery [3], and hence provided a way of addressing the view update problem of XQuery.

PSD and PSD Schema are structured documents or schema for structured document with computation embedded to maintain data dependency within documents. We have used Bi-X as a language to express such embedded computation [6]. Therefore, we can maintain mutual data dependency within structured documents.

References

[1] J. Nathan Foster, Michael B. Greenwald, Jonathan T. Moore, Benjamin C. Pierce, and Alan Schmitt. Combinators for bi-directional tree transformations: a linguistic approach to the view update problem. In *POPL*, 2005.

[2] Zhenjiang Hu, Shin-Cheng Mu, and Masato Takeichi. A programmable editor for developing structured documents based on bidirectional transformations. In *PEPM*, 2004.

[3] Dongxi Liu, Zhenjiang Hu, and Masato Takeichi. Bidirectional interpretation of xquery. In *PEPM*, 2007.

[4] Bi-X. http://www.ipl.t.u-tokyo.ac.jp/~liu/BiXQuery.html.

[5] Keisuke Nakano, Dongxi Liu, Yasushi Hayashi, Zhenjiang Hu, and Masato Takeichi. Bidirectional transformation based web publishing support system Vu-X. In *IPSJ*, 2008.

[6] Yasushi Hayashi, Dongxi Liu, Keisuke Nakano, Zhenjiang Hu, and Masato Takeichi. An extended schema for describing dependencies in structured documents. In *JSSST*, 2007.