Slicing XML Documents

Josep Silva [2]

DSIC, Technical University of Valencia
Valencia, Spain

Abstract

Program slicing is a well-known technique to extract the program statements that (potentially) affect the values computed at some point of interest. In this work, we introduce a novel slicing method for XML documents. Essentially, given an XML document (which is valid w.r.t. some DTD), we produce a new XML document (a *slice*) that contains the relevant information in the original XML document according to some criterion. Furthermore, we also output a new DTD such that the computed slice is valid w.r.t. this DTD. A prototype implementation of the XML slicer has been undertaken.

Key words: XML, DTD, Program Slicing.

1 Introduction

The increasing complexity of websites demands for tools which are able to aid Web designers in their construction and maintenance. Systematic, formal approaches can bring many benefits to quality website development, giving support for automated website transformations.

One of the most widely used program transformations is program slicing. It consists of a decomposition technique for the extraction of those program statements that (potentially) affect the values computed at some point of interest. Program slicing was originally introduced by Weiser [6] and has now many applications such as debugging, code understanding, program specialization, etc. See [4] for a survey.

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC under grant TIN2004-00231, by the *Generalitat Valenciana* GRUPOS03/025, by the MCYT under grant HU 2003-0003, and by the ICT for EU-India Cross-Cultural Dissemination Project ALA/95/23/2003/077-054.

[2] Email: jsilva@dsic.upv.es

```
<PersonalInfo>
   <Contact>
      <Status> Professor </Status>
      <Name> Ryan </Name>
      <Surname> Gibson <Surname>
   </Contact>
   <Teaching>
      <Subject>
         <Name> Logic </Name>
         <Sched> Mon/Wed 16-18 </Sched>
         <Course> 4-Mathematics </Course>
      </Subject>
      <Subject>
         <Name> Algebra </Name>
         <Sched> Mon/Tur 11-13 </Sched>
         <Course> 3-Mathematics </Course>
      </Subject>
      ...
   </Teaching>
   <Research>
      <Project
         name = "SysLog"
         year = "2003-2004"
         budget = "16000€" />
      ...
   </Research>
</PersonalInfo>
```

(a) XML document with the contents of a webpage

```
<!ELEMENT PersonalInfo (Contact,
                        Teaching,
                        Research)>
<!ELEMENT Contact (Status,
                   Name,
                   Surname)>
<!ELEMENT Status ANY>
<!ELEMENT Name ANY>
<!ELEMENT Surname ANY>
<!ELEMENT Teaching (Subject+)>
<!ELEMENT Subject (Name,
                   Sched,
                   Course)>
<!ELEMENT Sched ANY>
<!ELEMENT Course ANY>
<!ELEMENT Research (Project*)>
<!ELEMENT Project ANY>
<!ATTLIST Project
   name CDATA #REQUIRED
   year CDATA #REQUIRED
   budget CDATA #IMPLIED
>
```

(b) DTD defining personal info

Fig. 1. XML document (a) and its associated DTD (b)

In this work, we present a slicing technique which is applicable to XML documents [1]. The result of slicing an XML document is a new XML document (a *slice*) composed by those parts of the original document satisfying some criterion (the *slicing criterion*). We also produce a new DTD such that the computed slice is well-formed and valid with respect to this DTD.

2 XML

XML [1] was developed by an XML Working Group formed under the auspices of the World Wide Web Consortium (W3C) in 1996. XML documents are made up of basic units called "elements" according to a set of restrictions specified on an independent specification called *Document Type Definition* (DTD). Figure 1 shows an example of XML (a) and DTD (b) documents with some staff information from a university (for the time being, the reader can ignore the differences between black and grey text). An XML document is "well-formed" if it conforms to the standard XML syntax rules described in [1]. A "valid" XML document is a well-formed XML document, which also conforms to the rules of a DTD.

XML documents can easily be transformed to a webpage by means of

the *Extensible Stylesheet Language Transformations* (XSLT) [2]. It specifies the presentation of XML documents by describing how their instances are transformed to an XML document that uses a formatting vocabulary, such as (X)HTML or XSL-FO, that can be interpreted by any standard browser. For instance, the XML document in Figure 1 can be automatically translated to a standard webpage showing the information structured with tables and colored text. Different slices of an XML document could be used to produce (keeping the same XSLT transformation) distinct versions of the generated webpage.

3 The Slicing Technique

Program slicing techniques have been typically based on a data structure called *Program Dependence Graph* (PDG). In the case of XML, we take advantage of the tree-like structure of XML documents and DTDs. Despite the fact that DTDs can be graphs, we only consider trees without lost of generality, since they can easily be converted to trees by duplicating those elements producing cycles. Every vertex in the tree represents a single element of the XML document of type 'ELEMENT' (we refer the reader to [1] for a more detailed explanation about XML elements) containing all its attributes. Edges represent aggregation relations between elements.

In our setting, the slicing criterion consist of a set of elements from one of the trees (the XML or its associated DTD). As a consequence, we distinguish between two types of slicing:

DTD slicing: Given a set of elements from a DTD, we extract from this DTD those elements which are strictly necessary to maintain the tree structure, i.e., all the DTD elements that are in the path from the root to any of the elements in the slicing criterion.
Once the DTD slice is produced, the set of elements in the slice is used as a slicing criterion in order to produce the associated XML slice (see below).

Example 3.1 Consider the XML and DTD documents in Figure 1. Let the set containing the single element "Course" be the slicing criterion. The slices computed for the two documents are highlighted in Figure 1 with black color.

the user interested in that appears in the case of graphs (consider for instance "Name" as a slicing criterion)

XML slicing: Analogously to DTD slicing, given a set of XML elements, we extract from the XML document those elements which are in the path from the root to any of the elements in the slicing criterion.
Note that this criterion could be more fine-grained than the previous one. While a slicing criterion in a DTD selects a type of elements, a slicing criterion in an XML document can select only some particular instances of

this type.

Example 3.2 Consider the XML and DTD documents in Figure 1. Let the element "3-Mathematics" of type "Course" be the slicing criterion. The slice computed for the DTD is exactly the same than the one in Example 3.1. However, the XML slice only contains the information related to the "Algebra" subject, being the information related to the "Logic" subject removed.

The slicer proceeds as follows:

1) Compute the set of relevant elements with respect to the slicing criterion

2) The computed set is modified in order to satisfy all the restrictions of the DTD (for instance, the * and + relations). Attributes are also included

3) The result is used as a slicing criterion for the associated DTD/XML

Let $\mathcal{D}$ be a well-formed DTD, $\mathcal{X}$ an XML document valid with respect to $\mathcal{D}$, $\mathcal{D}'$ and $\mathcal{X}'$ two slices of $\mathcal{D}$ and $\mathcal{X}$ respectively, computed with a DTD-slicing criterion $\mathcal{C}$, and $\mathcal{D}''$ and $\mathcal{X}''$ two slices of $\mathcal{D}$ and $\mathcal{X}$ respectively, computed with a XML-slicing criterion $\mathcal{C}'$, we have proven the following properties:

a) $\mathcal{D}'$ is well-formed and $\mathcal{X}'$ is valid with respect to $\mathcal{D}'$

b) $\mathcal{D}''$ is well-formed and $\mathcal{X}''$ is valid with respect to $\mathcal{D}''$

If all the elements in $\mathcal{C}'$ are of one of the types in $\mathcal{C}$, then

c) $\mathcal{D}' = \mathcal{D}''$

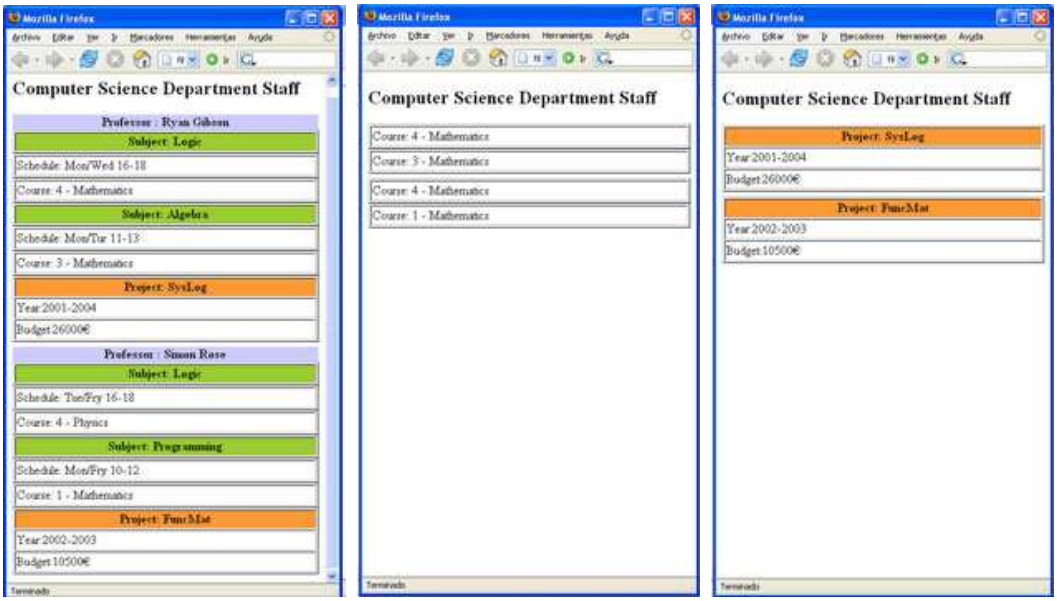
d) $\mathcal{X}''$ is a subtree of $\mathcal{X}'$

4 Implementation

We have implemented our prototype in Haskell [3]. Haskell provides us a formal basis with many advantages for the manipulation of XML documents such as the *HaXml* library [5]. It allows us to automatically translate XML or HTML documents into a Haskell representation. In particular, we use the following data structures that can represent any XML/HTML document:

```
data Element = Elem Name [Attribute] [Content]
data Attribute = (Name, Value)
data Content = CElem Element
             | CText String
```

One of the main possible applications of XML slicing is webpages slicing, because XML slices have a direct representation in webpages, producing slices of the original webpages. For instance, Figure 2 (a) shows the webpage that is automatically generated with an XSLT file from the XML document of Figure 1. Figure 2 (b) shows the webpage that is automatically generated



(a) WebPage (b) "Courses" Slice (c) "Projects" Slice

Fig. 2. Web Pages automatically generated from the XML in Figure 1

with the same XSLT file from the XML slice of Figure 1. Figure 2 (c) shows the webpage that is automatically generated with the same XSLT file from the slice of Figure 1 computed with a DTD slicing criterion formed by the "Project" element.

Preliminary results are very encouraging, showing that the slicing technique can bring many benefits when applied in conjunction with XSLT being able to slice complex webpages. The implementation and some other materials are publicly available at: `http://www.dsic.upv.es/~jsilva/xml`.

References

[1] Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler, and Franois Yergeau. Extensible markup language (xml) 1.0 (third edition). Available at: http://www.w3.org/TR/REC-xml/, February 2004. W3C Recommendation.

[2] James Clark. Xsl transformations (xslt). Available at: http://www.w3.org/TR/xslt, November 1999. W3C Recommendation.

[3] S. Peyton Jones, editor. *Haskell 98 Language and Libraries: The Revised Report*. Cambridge University Press, 2003.

[4] F. Tip. A Survey of Program Slicing Techniques. *Journal of Programming Languages*, 3:121–189, 1995.

[5] Malcolm Wallace and Colin Runciman. Haskell and XML: Generic combinators or type-based translation? In *Proceedings of the Fourth ACM SIGPLAN International Conference on Functional Programming (ICFP'99)*, volume 34–9, pages 148–159, N.Y., 1999. ACM Press.

[6] M.D. Weiser. Program Slicing. *IEEE Transactions on Software Engineering*, 10(4):352–357, 1984.