Automatic Generation of XSLT Stylesheets Using Evolutionary Algorithms

P. García-Sánchez
Department of Architecture and Computer Technology
University of Granada (Spain)
pgarcia@geneura.ugr.es

J.J. Merelo
Department of Architecture and Computer Technology
University of Granada (Spain)
jmerelo@geneura.ugr.es

J.P. Sevilla
Department of Architecture and Computer Technology
University of Granada (Spain)
jpsevilla@geneura.ugr.es

J.L.J. Laredo
Department of Architecture and Computer Technology
University of Granada (Spain)
juanlu@geneura.ugr.es

A.M. Mora
Department of Architecture and Computer Technology
University of Granada (Spain)
amorag@geneura.ugr.es

P.A. Castillo
Department of Architecture and Computer Technology
University of Granada (Spain)
pedro@atc.ugr.es

ABSTRACT

This paper introduces a procedure based on genetic programming to evolve XSLT programs (usually called stylesheets or logicsheets). XSLT is a general purpose, document-oriented functional language, generally used to transform XML documents or, in general, solve any problem that can be coded as an XML document. The proposed solution uses a tree representation for the stylesheets as well as diverse specific operators in order to obtain, in the studied cases and a reasonable time, a XSLT stylesheet that performs the transformation.

Categories and Subject Descriptors

I.2.2 [**Computing Methodologies**]: Artificial Intelligence - *Automatic Programming*; I.7 [**Computing Methodologies**]: Document and Text Processing

General Terms

Algorithms

Keywords

Stylesheets, XML, XSLT, Evolutionary Computation Techniques

1. INTRODUCTION

Since the Information technology (IT) industry has adopted the XML dialects as standard information exchange format, there is a business need for programs that transform one XML set of tags to another, extracting information or combining it in many possible ways. XSLT stylesheets (XML Stylesheet Language for Transformations), also called *logicsheets*, were designed for this purpose: applied to an XML document, they produce another.

The objective of this work is to find the XSLT logicsheet that, from one input XML document, is able to obtain a desired output XML document, which contains exclusively the information that is considered important from the first one. To perform this task we have designed a genetic programming algorithm where this logicsheet will be yielded, using evolutionary operators.

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.

2. XSLT USAGE

XSLT provides a general mechanism for the association of patterns in a source XML document to the application of format rules to these elements, but in order to simplify the search space for the evolutionary algorithm, only three instructions of XSLT will be considered in this work: template, that sets which XML fragment will be included when the element in its match attribute is found; apply-templates, which is used to select the elements to which the transformation is going to be applied and delegate control to the corresponding templates; and finally value-of, which simply includes the content of an XML document into the output file. This implies also a simplification of the general XML-to-XML transformation problem: we will just extract information from the original document, without adding new elements (tags) that did not exist in the original document. In fact, this makes the problem more similar to the creation of an *scraper*, a program that extracts information from legacy websites or documents. Thus, we intend this paper just as a proof of concept and initial performance measurement, whose generalization, if not straightforward, is at least possible.

3. PROPOSED METHOD

We have implemented an evolutionary algorithm, converting XSLT stylesheets into tree structures and making them evolve using variation operators. Every XSLT stylesheet is evaluated using a fitness function that is related to the difference between generated XML and output XML associated to the example (the desired to obtain). The solution has been programmed using JEO, an evolutionary algorithm library developed at University of Granada as part of the DREAM project, which is available at `http://www.dr-ea-m.org` together with the rest of the project. The algorithm works as shown in Figure 1.

The operators may be classified into two different types: the first one consists in operators whose assignment is to modify the XPath routes that contains the attributes of the

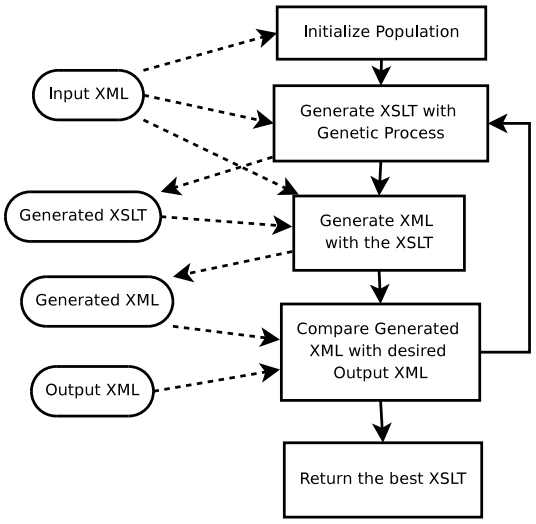

Figure 1: This figure shows how the algorithm works. Each individual of the population is an XSLT whose fitness is the comparison between the XML that is generated and the desired output XML

XSLT instructions. The second type of operators are used to modify the XSLT tree structure and take different shape in each of them (so that the structure is kept). In order to ensure the existence of the elements (tags) added to the XPath expressions and XSLT instruction attributes, every time one of them is needed it is randomly selected from the input file.

Fitness is related to the differences between the desired and the obtained output, but it has been also designed so that evolution is helped. Instead of using a single aggregative function, fitness is a vector that includes the number of deletions and additions needed to obtain the target output from the obtained output, and the resulting XSLT stylesheet length.

4. EXPERIMENTS AND RESULTS

To test the algorithm we have performed several experiments with 7 different XML input and output files, and using two different XSLT structures. The algorithm has been executed 30 times for each input XML and type of structure. All files are available at the previously commented site.

In these initial experiments we have found which kind of XSLT template structure is the most adequate for evolution, namely, the one that matches the select attribute in apply-templates with the match attribute in templates, and an indeterminate number of value-of instructions within each template. By constraining evolution in this way, we restrict the search space to a more reasonable size, and avoid the high degree of degeneracy of the problem, with many different structures yielding the same result, that, if combined, would result in invalid structures.

In general, we have also demonstrated that a XSLT logicsheet can be found just from an input/output pair of XML documents for a wide range of examples, some of them particularly difficult.

5. CONCLUSIONS

We present the results of an evolutionary algorithm designed to yield the XSLT logicsheet that is able to make a particular transformation from a XML document to another (target or desired XML file). One of the advantages of this application is that resulting logicsheets can be used directly in a production environment, without the intervention of a human operator; besides, it tackles a real-world problem found in many organizations.

There are some questions and issues that will have to be addressed in future papers:

- Using the DTD (Document Type Definition) associated to a XML file as a source of information for conversions between XML documents and for restrictions of the possible variations.
- Testing evolution with other kind of tools, such as a chain of SAX filters.
- Obviously, testing different kinds and increasingly complex set of documents, and using several input and output files at the same time, to test the generalization capability of the procedure.
- Using the identity transform as another frame for evolution, as an alternative to the types of structures of this work. The identity transform puts every element found in the input document in the output document; elements can then be selectively eliminated via the addition of single statements.
- Tackle difficult problems from the point of view of a human operator. In general, the XSLT stylesheets found here could have been programmed by a knowledgeable person in about an hour, but in some cases, input/output mapping would not be so obvious at first sight. This will mean, in general, increase also the XSLT statements used in the stylesheet, and also in general, adding new types of operators.

All source code for the programs is available from `http://forja.rediris.es/websvn/wsvn/geneura/GeneradorXSLT/` under an open source licence (GPL).

6. ACKNOWLEDGMENTS

This work has been supported by the Spanish MICYT project TIN2007-68083-C02-01, the Junta de Andalucia CICE project P06-TIC-02025 and the Granada University and Intecna Soluciones SL. project OTRI-1515.