gccXfront: Exploiting gcc as a front end
for program comprehension tools via XML/XSLT

Mark Hennessy*
Computer Science Dept.
National University of Ireland
Maynooth, Co. Kildare, Ireland
markh@cs.may.ie

Brian A. Malloy
Computer Science Dept.
Clemson University
Clemson, SC, USA
malloy@cs.clemson.edu

James F. Power
Computer Science Dept.
National University of Ireland
Maynooth, Co. Kildare, Ireland
jpower@cs.may.ie

Abstract

Parsing programming languages is an essential component of the front end of most program comprehension tools. Languages such as C++ can be difficult to parse and so it can prove useful to re-use existing front ends such as those from the GNU compiler collection, gcc. *We have modified* gcc *to provide syntactic tags in XML format around the source code which can greatly enhance our comprehension of the program structure. Further, by using XML transformation stylesheets, the XML outputted by our modified* gcc *can be translated into a more readable format. Our tool,* gccXfront *leverages the power and portability of the* gcc *suite, since any C, C++, Objective C or Java program can be processed using* gcc. *Our tool can thus act as a bridge between* gcc *and other program comprehension tools that accept XML formatted input.*

1 Introduction

In [?] an approach to the automatic generation of XML [?] from *GNU gcc* was formulated. This modified version of *gcc* outputs syntactic tags belonging to the program in XML format. The XML files that are outputted contain the full representation of the source code as it has been processed by *gcc*. Even for a simple "Hello World" program there are over 130 lines of XML. The object of *gccXfront* is to use XSLT (Extensible Stylesheet Language Transformation) to produce a more human readable and more relevant XML representation of the parse to facilitate comprehension of C, C++ and Java programs. This can be represented on many different levels of abstraction from the raw outputted XML all the way up to the XML representing the ideal concepts of grammar in its most abstract form e.g. of the form *stmts ::= stmt ; stmts* etc.

gccXfront has been written in the Java programming language. It utilizes the very latest Java APIs for XML Processing (JAXP) along with the more established "swing" components to realize the GUI. The GUI is designed around two main components (See Fig. 1):

- **Parse Tree Browser:** This allows the user to navigate through the XML file visually through the use of a tree which mimics the directory-like structure of the corresponding XML file. This greatly aides the user in two ways; firstly in the way that it offers an exact visualisation of the parse tree and secondly in that the tree nodes can be selected to bring the user directly to the corresponding code within the XML file. There are also options relating to the tree that can be selected to aid the viewing of the tree. These stem from the type of XML parser used (the DOM model) and can allow the user to ignore trivial aspects of the XML such as white space, comments etc. However these options do enhance the aesthetics of the tree view.

- **XML Pane:** This component contains three tabs. The first tab holds the XML output by *gcc*. The second tab contains our XSLT stylesheet that we use to transform our XML to the desired domain. Our third tab stores the output of the transformed XML.

As mentioned previously the front end was written in the Java programming environment. This was chosen for its platform independence, not only in terms of program execution but also in terms of XML parser independence and XSLT transformer independence. The JAXP allows any XML parser or any XSLT translator to be used either by default or at the request of the user.

2 gccXfront and Program Comprehension

The net goal of this tool is to give the user a concise, clear view of the constituents of a C, C++ or Java program. This

---

*Address all correspondance to Mark Hennessy

type of tool fits in the category of tools used for program analysis such as visualisation and metrication. These type of tools can also act as the basis for more comprehensive tools designed to re-factor and re-engineer code.

Most parser generators are capable of dumping some sort of output, such as the grammar rules chosen or some kind of parse tree. These outputs however can sometimes be hard to comprehend and the nature of their outputs (usually dumped to the standard error) does not lend itself well to any further analysis from any other tools. Furthermore the size and the amount of this "dumped" data can mitigate against any form of analysis. However *gccXfront* utilises the XML outputs of the modified *gcc* parser to aid comprehension in two ways:

1. **Parse Tree:** A visual representation of the parse tree is constructed. This allows for immediate and concise analysis of the parse tree in terms of production rules used. However the parse tree outputted is quite large and unwieldy even for the most basic of C++ programs. Thus the XSLT transformations are utilised to pare down the XML(e.g. stmt ::= simple_stmt) thus allowing a view of only the relevant tags for the program.

2. **XML Tags:** The XML tags correspond to grammar non-terminals and, as such represent syntactic categories such as declarations, statements etc. As stated previously, the initial output is quite large and does not lend itself well to analysis. Once the XSLT transformations are performed the user is left with a far more human readable format for the grammar.

One of the key features of this front end is its portability. Our tool is independent of the XML and XSLT used. At no stage of operation (either in generation of the XML with *gcc* or in the use of the tool) does the user ever have to write code. Instead XSLT stylesheets are used to define all our interactions with the output XML. This provides the possibility of compatibility with other tools and formats such as GXL [**?**] and XMI[**?**]. In fact our front end provides a bridge between *gcc* and other back-end program comprehension tools.

3 Current Work

Once any C,C++ or Java program has had XML tags annotated to the source code using the modified *gcc*, it is then possible, using XSLT stylesheets, to transform our XML into other exchange formats, notably XMI and GXL. Other program tools such as Rigi [**?**] can then exploit this GXL when it is converted to Rigi standard format using XSLT. Thus *gccXfront* achieves the goal of providing a bridge between *gcc* and the many reverse and reengineering tools available.
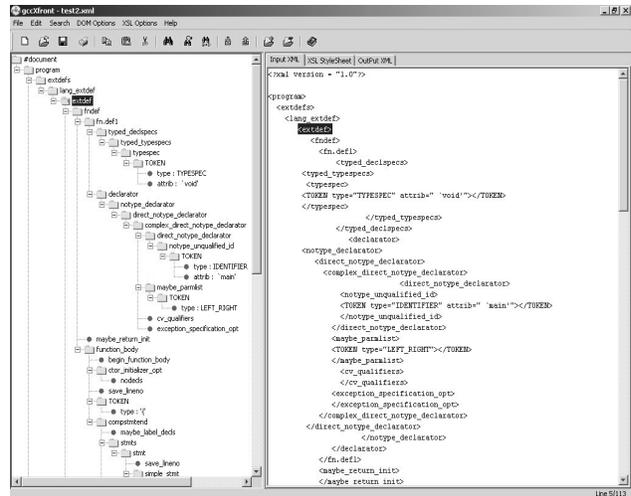

Figure 1. The GUI Front End with "Hello World"Parse.

gccXfront can be downloaded from http://www.cs.may.ie/ markh/gccXfront

References

[1] R.C. Holt, A. Winter, and A. Schurr. GXL: toward a standard exchange format. In *Seventh Working Conference on Reverse Engineering*, pages 162 171, Brisbane, Queensland, Australia, 23-25 November 2000.

[2] Object Management Group. XML Metadata Interchage (XMI). http://www.omg.org, Revision 1.2 2002.

[3] James F. Power and Brian A. Malloy. Program annotation in XML: a parse-tree based approach. In *9th Working Conference on Reverse Engineering*, Richmond, VA, November 2002.

[4] Tilley S. R., Wong K., Storey M.-A. D., and Muller H. A. Rigi: a visual tool for understanding legacy systems. *International Journal of Software Engineering and Knowledge Engineering*, December 1994. http://www.rigi.csc.uvic.ca.

[5] WorldWideWeb Consortium. eXtensible Markup Language (XML). http://www.w3.org/XML/, Revision 1.225 23 April 2002.