E/R Schema for the Datrix C/C++/Java Exchange Format

Richard C. Holt and Ahmed E. Hassan
Software Architecture Group (SWAG)
Dept. of Computer Science
University of Waterloo
Waterloo, Ontario N2L 3G1
CANADA
+1 (519) 888-4567 x 4671
{aeehassa, holt}@plg.uwaterloo.ca

Bruno Laguë, Sébastien Lapierre, and Charles Leduc
Quality Engineering and Research
Bell Canada
1050 Beaver Hall Hill, 2nd floor
Montréal, Québec, H2Z 1S4
CANADA
+1 (514) 786-6442
{bruno.lague, sebastien.lapierre, charles.leduc}@bell.ca

ABSTRACT

A SEF (software exchange format), such as GXL [6], TA [4] or RSF [7], is used to exchange data between tools that analyze software. Researchers at Bell Canada have specified the Datrix [3] SEF in TA (and soon to be, GXL) for C, C++ and Java. It is designed so that a parser for the language, C, C++ or Java, can read a source program and emit the program's Abstract Syntax Tree (AST) in the Datrix format. This note explains how an entity/relation (E/R) schema [2] was extracted for Datrix, and gives this schema as an E/R diagram.

Keywords

Software exchange format, SEF, abstract syntax tree, AST, schema, data exchange

1. INTRODUCTION

Most of reverse engineering tools use a parser to extract *facts* about the source code. There is an ongoing international effort to standardize the means of representing such facts, using a software exchange format (SEF). In the case of Bell Canada's Datrix system, facts about C++ (including C) and Java are exchanged in the TA format (and soon to be, in the proposed standard GXL format). The exchanged information is the Abstract Syntax Tree (AST) for the source code.

The form of the Datrix exchange data is described in Bell Canada's documentation [1] by means of examples and natural language (English). Since the data is represented in TA, mathematically speaking, the data is a typed, directed, attributed graph, which we will call simply a *typed graph*. We can use an entity/relation (E/R) diagram to specify the set of all graphs that a legal Datrix parser could produce. In this note, we call an E/R diagram a *schema*. Using an E/R diagram, we can specify: (1) the allowed types and attributes for nodes, and (2) the allowed types and attributes of edges as well as the allowed edge connections between node types. (Note that both TA and GXL include notation for specifying such schemas.)

The goal of the work described here is to enhance that documentation by providing a schema. The Datrix schema presented actually specifies an Abstract Semantics Graph (ASG), which includes the AST as well as interconnecting edges giving dependencies among nodes.

2. EXTRACTION PROCESS

In order to "extract" an E/R schema for Datrix, we began by reading the Datrix documentation, which describes the form of output a parser is expected to produce for each kind of C++ fragment (we did not consider Java).

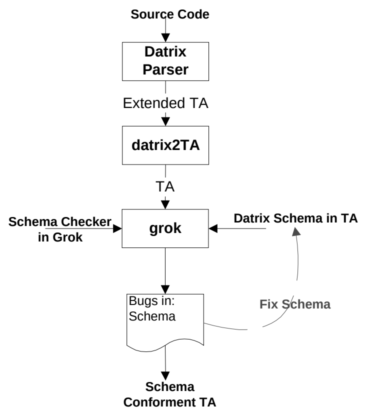

Figure 1: Testing the Datrix Schema

From this documentation, we were able to surmise the set of node types and edge types. From the examples and some study, we were also able to surmise the node and edge attributes, as well as the edge connectivities. Since the AST for C++ is inherently rather complex, we expected that our proposed Datrix schema was not completely correct, so we wanted to be able to test its validity. However, it is not immediately obvious how a schema could be tested.

Figure 1 shows how this testing was done. We ran the Datrix parser provided by Bell Canada. The parser produced TA corresponding to the source program. This TA is actually an enhanced version of TA, so we translated it to "pure" TA using a filter called datrix2TA. We used a 4 KLOC C++ AVL tree C++ library, provided by Kostas Kontogiannis as our test data (as our source program).

We took our E/R schema (see Figures 2 and 3) and wrote it in TA (recall that a part of the TA SEF is used to specify schemas); see "Datrix Schema in TA" in Figure 1.

We used a "fact calculator" called Grok, which reads facts (streams of data as well as schemas in TA) and manipulates them using Tarski algebra [5]. We already had a Grok script which checks to see if a set of TA facts conforms to a given TA schema; see "Schema Checker in Grok" in Figure 1.

As Figure 1 shows, we used the Schema Checker to test if the TA facts from the AVL tree library conformed to the Datrix schema. Not surprisingly, initially the Checker produced a listing of a number of errors. We analyzed each of these errors to see where our proposed Datrix schema was in error (or possibly where the Datrix parser was in error). We then corrected the schema and re-ran the test until no errors were reported. The result, when again analyzed by hand, seems to be a good and useful schema for Datrix, and ideally will eventually be incorporated into the Datrix documentation and TA/GXL output.

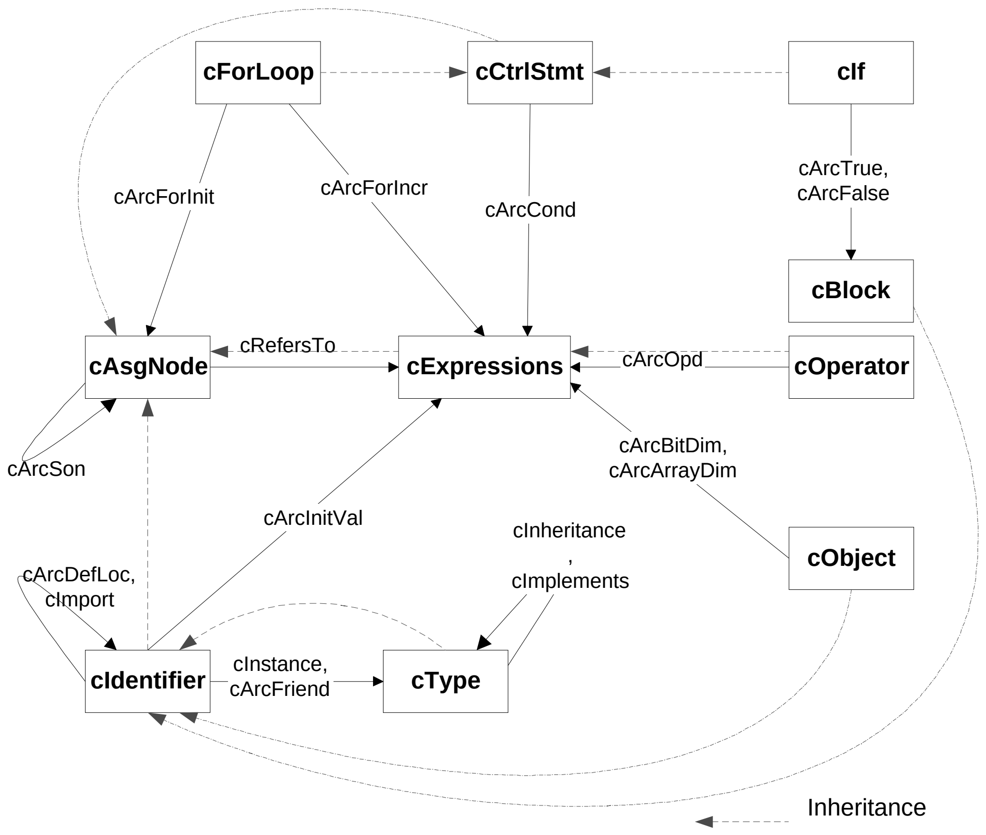

Figure 2: Partial Datrix Schema (Relations and Inheritance)

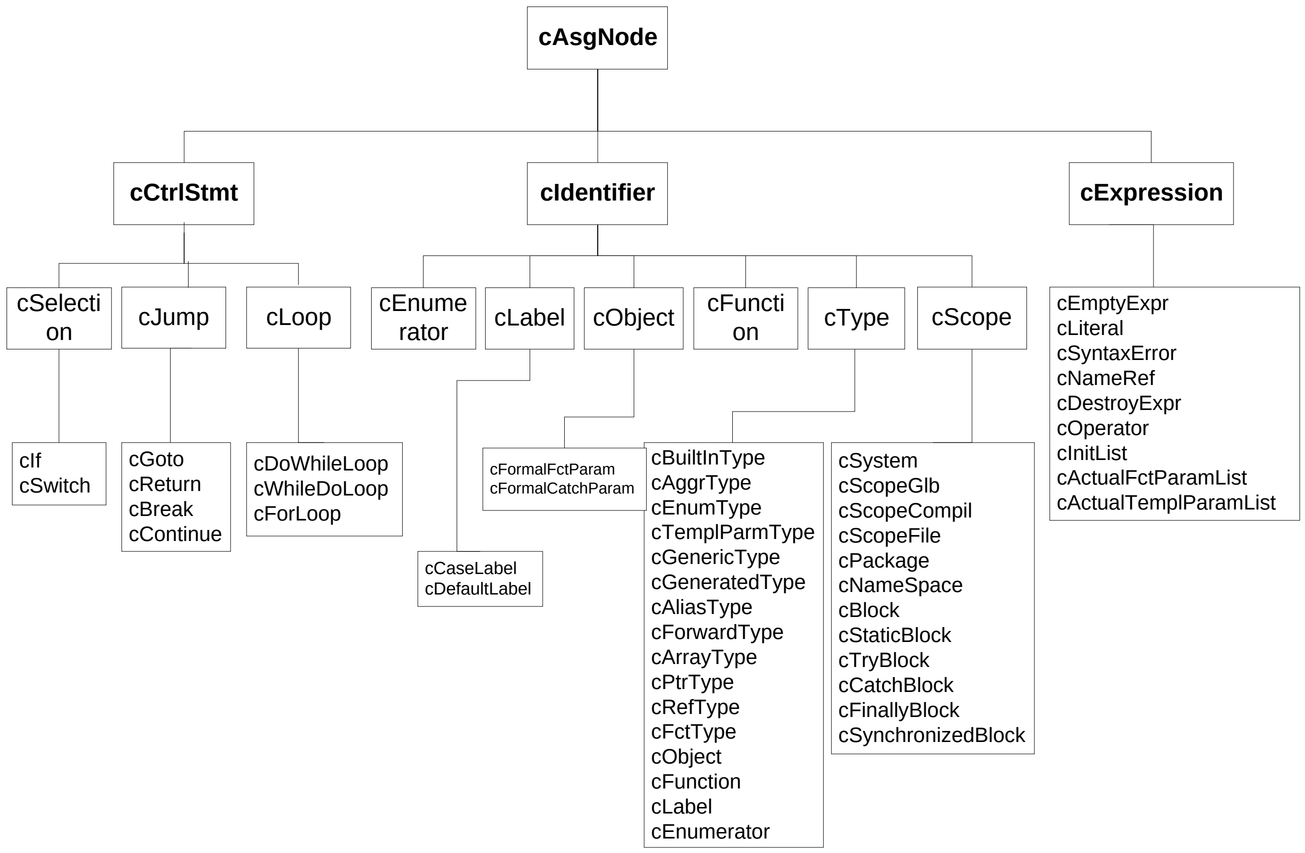

Figure 3: Partial Datrix Schema (Inheritance)

3. THE DATRIX SCHEMA

Figures 2 and 3 give the Datrix schema, somewhat simplified to meet space restrictions, as an E/R diagram. Figure 2 shows allowed relations among nodes. For example, the node for an If statement *cIf* is connected by *cArcTrue* and *cArcFalse* edges to corresponding nodes *cBlock* representing *true* and *false* clauses.

Figure 3 shows all nodes, with extensive use of inheritance to show common attributes and connectivity. This has been simplified by omitting some nodes, notably those for comments, assembly blocks, throw statements and SQL statements.

References

[1] Bell Canada, DATRIX^{TM} Abstract Semantic Graph: Reference Manual http://www.casi.polymtl.ca/casibell/datrix/refmanuals/asgmodel-1.4.pdf

[2] P. Chen. The Entity-Relation Model – Toward a Unified View of Data. *ACM Transactions on Database System*, 1(1):9-36, March 1976.

[3] Datrix online at: http://www.iro.umontreal.ca/labs/gelo/datrix/ http://www.casi.polymtl.ca/casibell/datrix/

[4] R. C. Holt, “An Introduction to TA: the Tuple-Attribute Language”, March 1997.

[5] R. C. H. Structural Manipulation of Software Architecture using Traski Algebra. Working Conference on Reverse Engineering, Honolulu, October 1998.

[6] R. C. Holt, A. Winter, A. Schürr, GXL: Towards a Standard Exchange Format. Online at: http://www.gupro.de/techreports/RR-1-2000/

[7] K. Wong. Rigi User’s Manual. Version 5.4.1, July 1996 http://www.rigi.csc.uvic.ca/rigi/manual/user.html