XML-BASED INTEROPERABILITY AMONG SYMBOLIC COMPUTATION SYSTEMS

M. Andrés, F.J. García, V. Pascual, J. Rubio
Departamento Matemáticas y Computación. Universidad de La Rioja.
Edificio Vives. Calle Luis de Ulloa, s/n. 26004 Logroño (La Rioja, Spain)

ABSTRACT

In this paper, the state of a project to interoperate among some complex symbolic computation systems is presented. We briefly put into context our problem, stressing the difficult points (essentially, the need of exchanging information which encodes infinite data structures) and showing the limitations of the current standard XML-based languages to communicate mathematical knowledge, as MathML.

KEYWORDS

XML, MathML, symbolic computation, interoperability, mathematical knowledge.

1. INTRODUCTION

With the intensive use of the World Wide Web (WWW), the trend to interoperate among heterogeneous systems through computer networks has been accelerated. The WWW is used as a mechanism to communicate and exchange information between different platforms and languages. The information which can be transferred is not only documentary but also data which can be used to carry out computations. In other words, Internet appears also as a tool for doing scientific computations in a distributed and collaborative way.

One of the fields where these ideas can be applied is that of Symbolic Computation. Computer Algebra packages are being extended to interconnect them. In fact, some Computer Algebra Systems (CAS) are already enabled to perform distributed computing. An example is Distributed Maple [Schreiner2003], which is an environment for executing parallel computer algebra programs on multiprocessors and heterogeneous clusters. One of the key aspects of distributed computing is, as it is well-known, the data interchange among the distributed nodes. Some of the distributed CAS have an own proprietary format to exchange their data. But if we want to increase the degree of interoperability between different CAS, it is necessary a common format for representing the mathematical data. XML (eXtensive Markup Language) [XML] comes up as an acceptable alternative to such proprietary formats.

This problem of universal representation of data is central in all the WWW research area. XML is being used as a *de-facto* standard protocol, allowing systems to exchange structured data. XML has some well-known features that have supported its adoption in a lot of data interchange initiatives. In short, it is readable and human-friendly (only a simple text editor is necessary to write and read and XML document); the cost of transformation from an XML dialect to another format (being XML-based or not) is minimal (only an XSLT style sheet is necessary); and, at last, it is extendable (if the available set of element tags is neither enough nor suitable for a particular application, users can add new ones).

In our particular problem (symbolic computation) the data to be transferred must encode mathematical knowledge. Fortunately, we dispose of some standard XML-based tools in this field. One of the dialects of the XML language family is MathML (Mathematical Markup Language) [MathML]. This language was authored with the goal of enabling mathematics to be served, received and processed through the WWW, in the same way that HTML had enabled it for text. MathML, which was developed by the World Wide Web Consortium [W3C], is an XML application and it was the first standard for representing mathematics. MathML allows the user to describe the symbolic meaning of the mathematics, not just its notation; so it is

adequate to exchange mathematical content between applications such as web browsers, computer algebra systems and other scientific software. Another XML application to represent mathematical data is OpenMath [OpenMath2000]. OpenMath is also a standard, and it was specifically developed to encode the semantics (i.e., the meaning) of mathematics; it is also used to exchange mathematical objects between applications. The meaning of the mathematical objects included in OpenMath is defined by means of content dictionaries.

Therefore, MathML and OpenMath give XML representations for mathematical objects. While MathML tries to be used only to represent mathematics (in two possible ways: as presentation or content elements), OpenMath provides an extensible mechanism to describe mathematical semantics. Both of them are very related and they complement each other. In fact, MathML can include OpenMath elements to define the mathematical meaning of its content elements and, reciprocally, OpenMath can use the representation of an object given by MathML. These tools are obtaining a great attention in last years (see, for instance, the conferences [Buchberger2001], [Asperti2003]) and are the basis for our work, which is presented in the following Section.

2. INTEROPERABILITY AND SPECIALIZED CAS: BEYOND MATHML

In some previous papers ([Lamban1999], [Domínguez2001a], [Domínguez2001b], [Lamban2003]), we undertook the formal analysis of two symbolic computation systems, EAT (Effective Algebraic Topology) [EAT] and Kenzo [Kenzo]. These systems, written under the direction of Sergeraert, are devoted to symbolic computation in Algebraic Topology, and, in particular, they have been useful to compute homology groups of infinite topological spaces, namely loop spaces. Both systems are written in the Common Lisp programming language and have obtained some results (specifically, homology groups) which had never been determined before using neither theoretical nor computational methods. For this reason, it was considered interesting to carry out the formal analysis of the systems and to explore the application of their characteristics to other systems and environments.

One of the conclusions of this analysis task was the convenience of making distributed the running of these software systems. Being devoted to intensive symbolic computation (usually, interesting groups need several days of CPU time over very powerful workstations), the possibilities opened by distributed computing are not at all negligible. Our first attempt was to directly reconstruct the systems in the Java language. But two evident difficulties appear. On one hand, the different type discipline in both languages. On the other hand, the paradigm shifts between both programming languages (in order to encode infinite data structures, the functional programming features of Common Lisp are essentially used in EAT and Kenzo; on the contrary, Java is object-oriented). In order to confront these two difficulties, we introduced an intermediary step: we rebuilt (some fragments of) EAT in the ML programming language [Paulson2000]. ML is not so far of Common Lisp because it is a functional programming language; but due to its type system, it is a good bridge towards more standard languages. With respect to the implementation of functional programming in object-oriented languages, in [Aransay2001] we showed a solution that allowed us to represent lexical closures in Java.

Thus, at this moment we have implemented three symbolic computation systems for Algebraic Topology, respectively, in Common Lisp, ML and Java. The following step we undertook was to interoperate with these systems through the net. For this, we needed to exchange complicated mathematical data among them. Our first attempt was to use MathML as exchange language, but it turns out to be not adequate. The elements of the algebraic structures which EAT works with are more complicated than those of the MathML. But besides in our symbolic computation problem, we need to represent complex algebraic structures of infinite nature. These are very different from the elementary data (equations, polynomials and so on) which can be managed in MathML in the well-known CAS (as Mathematica or Maple, for instance). One alternative to MathML is to use OpenMath whose *content dictionaries* are provided, in principle, to achieve this kind of extension. Nevertheless, content dictionaries in OpenMath are strictly organised, so making difficult its application in a flexible and absolutely new field, as it is symbolic computation in Algebraic Topology. Thus, we have been obliged to define our own DTD, but trying to design it as a proper extension of the MathML DTD, in order to keep further compatibility.

It is important to note that in our symbolic computation setting, there are always two *layers* of data structures (see [Lamban1999], for instance). In the first layer, we need, as usual, elementary data as symbols,

integers, lists of them, trees and so on. These data can be encoded without difficulty in MathML, and we can use them directly to communicate through the net. The second layer of data structures is intended to capture algebraic structures, as groups, rings, vectorial spaces, etc. These data can be (and they are frequently in symbolic computation in Algebraic Topology) of infinite nature, and their implementations require functional programming. The data from the first layer act as *elements* (in the set-theoretical sense) for the data in the second layer. For instance, a list can represent a *vector* which belongs to a given vector space (represented on computer memory in a functional way). This means that any intermediary structure consists of: (1) some *first-layer* data, and (2) one algebraic structure, where the data from (1) must be interpreted. This *mixed* organisation makes difficult to represent in an XML-flat document any relevant information.

We have found a partial solution to this problem, taking profit of some internal organization details of the original Kenzo system. There, for maintenance purposes, each algebraic structure is endowed with the tree of calls which generated it. Now, this textual information can be encoded in an XML format, allowing another system to recover the exact algebraic structure where the computation is being carried out. This approach has two evident drawbacks. First, it is only applicable to interoperate among *isomorphic* software systems (that is to say, the constructors have the same names and the same parameters). Second, it can be very inefficient to reconstruct a complete algebraic structure each time that a computation is borrowed.

With respect to the first point, let us note that our systems are, for the moment, isomorphic (because the ML and Java versions have been re-implemented from the Common Lisp Kenzo). In general, the solution can be based on *ontology mappings*; in other words, it would be necessary to indicate how each particular service in a system can be translated in another one. The second point should be dealt with a careful management of the computations already done for the system.

At present, we have implemented a prototype which allows the three previously evoked systems (the Common Lisp, ML and Java versions) to interchange XML documents (between each one of them and also each program with itself), with the aim of doing collaborative computation. More concretely, one of the programs can stop temporarily its current computation and transfer the intermediate results in XML format (including also the necessary information on the "ambient" infinite algebraic structure where the computation is being carried out). Then itself or another of the programs can recover these data and continue the computation. For the moment, the prototype works in a local, no-distributed manner. One of the next steps is to organise a web service to transmit this information through Internet, allowing several computers to collaborate in a same computation. We are planning to use the Linda coordination language in order to achieve this goal. Some experiences in this field, in a different computing context, have been already documented for our team in [Alvarez2003]. As usual, Java is the core language for all this web-services approach, and this justifies our efforts to translate from a typically scientific programming language, as Common Lisp, to an object-oriented standard as Java.

3. CONCLUSION

Our work focusses on a very important problem in intensive scientific computation: the use of Internet to make collaborative computations, in order to decrease the generally exponential computation times reached in some areas. One of these areas is Symbolic Computation (also known as Computer Algebra). In our concrete application domain (symbolic computation in Algebraic Topology), we must face an even more difficult question: the dealing with infinite (algebraic) data structures. Our current prototype shows that, in principle, distributed and heterogeneous symbolic computing in this area can be carried out, by using XML-based communication. Another conclusion of our in-progress work is that interoperability between different programming languages (and even between different programming paradigms) can be also achieved. This is specially promising due to the intensive use of functional programming in the original systems EAT and Kenzo. This result can be understood as a practical realization of the theoretical results from [Lamban2003], where the implicit object-oriented aspects of EAT were elucidated.

In a wider context, our research can have some value in understanding the expressiveness and adequacy of XML-based communications, in general. It is well-known that mathematical language is a good laboratory to experiment with general knowledge representation. This is illustrated, in the WWW domain, by the fact that MathML was the first accepted protocol satisfying the XML specifications. Therefore, knowing the

power and limitations of MathML (and, in particular, our work on very specialized CAS) becomes important for the whole XML community.

As a consequence of our research, the need of enriching the MathML-extension features is enlighted. This need has been also documented by several authors. But our application field reveals a challenging problem: to find an extension allowing the programmer to encode infinite data structures, or even more generally, to represent the full functional programming constructs.

ACKNOWLEDGEMENT

Partially supported by MCyT, project TIC2002-01626 and by Comunidad Autónoma de La Rioja, project ACPI-2002/06.

REFERENCES

[Alvarez2003] Álvarez, P. et al, 2003. Generative Communication with Semantic Matching in Distributed Heterogeneous Environments. *To appear in Lecture Notes in Computer Science*.

[Aransay2001] Aransay, J. et al, 2001. Object-oriented implementation of algebraic structures: a case-study in Java and C++ *(in Spanish)*. *Proceedings of Séptimo Encuentro de Álgebra Computacional y Aplicaciones.* Universidad de La Rioja, pp. 78-82.

[Asperti2003] Asperti, A. et al (eds), 2003. Mathematical Knowledge Management. *Lecture Notes in Computer Science*, Vol. 2594.

[Buchberger2001] Buchberger, B. and Caprotti, O. (eds), 2001. *Electronic proceedings of the first International Workshop on Mathematical Knowledge Management.*

[Dominguez2001a] Domínguez, C. and Rubio, J., 2001. Modeling inheritance as coercion in a Symbolic Computation System. *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 2001).* ACM Press, pp. 107-115.

[Dominguez2001b] Domínguez, C. et al, 2001. Hidden specification of a functional system. *Lecture Notes in Computer Science*, Vol.2178, pp 555-569.

[EAT] Rubio, J. et al, 1997. *EAT: Symbolic Software for Effective Homology Computation.* ftp://fourier.ujf-grenoble.fr/pub/EAT.

[Kenzo] Dousson, X. et al, 1999. *The Kenzo program.* http://www-fourier.ujf-grenoble.fr/~sergerar/Kenzo/.

[Lamban1999] Lambán, L. et al, 1999. Specifying implementations. *Proceedings of the International Symposium on Symbolic and Algebraic Computation (ISSAC 1999).* ACM Press, pp. 245-251.

[Lamban2003] Lambán, L. et al, 2003. An object-oriented interpretation of the EAT system. *To appear in Applicable Algebra in Engineering, Communication and Computing*.

[MathML] Ausbrooks, R. et al, 2001. *Mathematical Markup Language (MathML) Version 2.0*. http://www.w3.org/TR/2001/REC-MathML2-20010221/.

[OpenMath2000] Caprotti, O. et al (eds), 2000. *The OpenMath Standard*. http://www.openmath.org/standard.

[Paulson2000] Paulson, L.C., 2000. *ML for the working programmer- 2nd edition*. Cambridge University Press, Cambridge, United Kingdom.

[Schreiner2003] Schreiner, W. et al, 2003. Distributed Maple: Parallel Computer Algebra in Networked Environments. *In Journal of Symbolic Computation*, Vol. 35, No. 3, pp. 305-347.

[XML] Bray, T. et al (eds), 2000. *Extensible Markup Language (XML) 1.0 (Second edition)*. http://www.w3.org/TR/REC-xml/.

[W3C] World Wide Web Consortium. http://www.w3.org.