View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Apollo

The use of XML and CML in Computational Chemistry and Physics Programs

A. García[1], P. Murray-Rust[2], **J. Wakelin**[3]

[1] Departamento de Física de la Materia Condensada, Facultad de Ciencias, Universidad del País Vasco, E-48080 Bilbao, Spain.
[2] Unilever Centre for Molecular Science Informatics, Chemistry Department, University of Cambridge, Cambridge CB2 1EW, UK.
[3] Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EQ, UK.

Abstract

This work addresses problems associated with data exchange and data representation in the computational chemistry and physics communities. Recent computational developments, such as Condor and the Grid, have paved the way for new kinds of simulations that demand more rigorous data handling. To this end, the paper discusses the use of XML and the Chemical Markup Language (CML) in theoretical chemistry and physics. Extensions to the core CML language, known as CMLComp, are also discussed. However, the majority of atomic scale simulation software is written in Fortran. Fortran's lack of XML support represents a potential barrier to the adoption of CML in these fields. This has prompted the authors to develop XML and CML processing tools for Fortran, including native SAX and DOM implementations, as well as libraries for generating well formed XML and CML. These libraries have been used to extend existing simulation packages to work with the CML and CMLComp languages. Finally, we give a practical example that highlights how these XML aware applications can be effectively used as workflow components in complex chemical and physical simulations.

1. Introduction

Data exchange has always been an important issue for the computational chemistry and physics communities. However, it has normally been tackled on an informal or *ad hoc* basis and there is no agreement upon standards for data representation in these fields. Consequently, the majority of data is represented by bespoke or legacy formats, intimately tied to the software that produces them. Traditionally, our data sets and data processing requirements have been on a scale such that an informal treatment has been possible. However, recent computational advances, such as the Grid and Condor, that allow the user to perform high-throughput simulations and/or create complex workflow schemes, have forced us to think more carefully about data representation. Indeed, data handling is becoming one of the most important issues of scientific simulation.

2. Data Exchange in Computational Chemistry and Physics

Conventionally, atomic scale simulation programs have relied on bespoke text and binary formats for data input and output (see figure 1). These formats are rarely well specified and have little, if any, syntactic commonality between them. This presents a major barrier to data exchange and software interoperability in these fields. However, while there is little commonality between them at the syntactic level, at a semantic level the content of these files is often extremely similar. Moreover, in comparison to many disciplines the semantics in chemistry and physics are well established. For instance, a single set of atomic coordinates could be used in many hundreds of different programs regardless of the methodology on which they are based or what they are used to calculate – this is because concepts such as "atomic coordinates" have strict meanings, when we obtain a set of atomic coordinates we "know what we are getting". There are many more examples and some are listed in Table 1. The clear semantics in these fields mean that it is *possible* to take data from one source and use it in another. But it is the fact that much of this data can be readily re-purposed that actually makes it useful or desirable for us to exchange it. For instance, the same set of atomic coordinates could be used as the basis of many different studies (e.g. a molecular dynamics run, a phonon calculation, a band structure calculation).

- Atomic Coordinates
- Connectivity Tables
- Lattice Parameters
- Atomic Forces
- Forcefields
- Basis Sets
- Pseudopotentials
- Wavefunctions
- Charge Densities

Table 1. Examples of the kinds of data that are frequently transferred between atomic scale simulation programs.

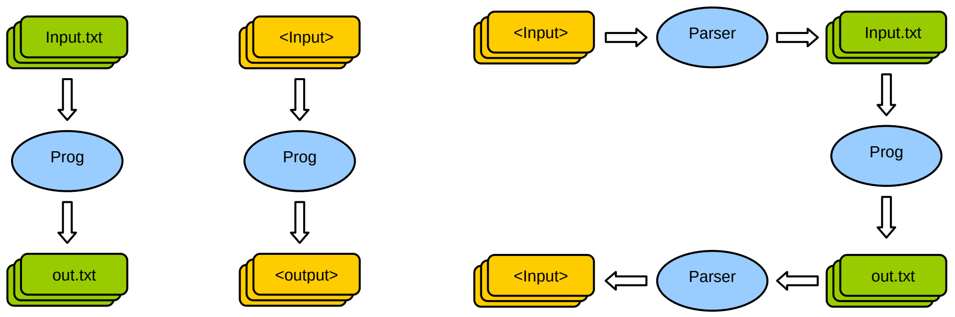

Figure 1. The traditional set-up of atomic/molecular scale simulation software (left) and the way our chemistry and physics programs now work (centre). Where the source code is not available it is necessary to parse input and output text to generate XML (right) – In these cases the developer can use the JUMBO CML libraries available for a wide range of programming languages, details of which are given in an accompanying manuscript.

3. CML and CMLComp

The Chemical Markup Language was the first example of an XML application and is becoming the *de facto* standard for XML-based interchange of chemical information [1,2].

However, while the core CML language was able to cover many of the necessary concepts it became apparent that certain concepts, central to solid state physics or computational chemistry, could not easily be accommodated in the existing CML framework. This lead to the development of a new markup language designed specifically to cover the needs of chemical and physical simulations. The language, now known as CMLComp, comprises a subset of core CML and a relatively small number of new elements introduced specifically to cover the needs of the atomic scale simulation communities. Currently these extensions are incorporated into the CMLComp XML schema and more generally can be re-used in other applications using JUMBO [3].

4. XML and Fortran

The majority of computational chemistry and physics software is written in Fortran, a language eminently suitable for high-performance numerical work but with little support for other application domains. Unfortunately, the lack of XML support for Fortran presents a major barrier to the adoption of CML or XML in these fields. To this end, the development of robust XML processing libraries for Fortran has been one of the major successes of this work. We have implemented the two main XML processing APIs, the Simple API for XML (SAX) and the W3C's Document Object Model (DOM) [4,5]. In addition we have developed libraries for generating well-formed XML and CML directly from Fortran.

The SAX parser is implemented using a finite state machine, which processes the input sequentially, with a very small memory footprint. The standard callbacks of the SAX specification are supplemented with helper routines to ease the manipulation of numerical datasets. Well-formedness is checked and errors in the file tagged with line/column information, but no DTD processing or validation in general is attempted. The SAX parser is actually written in the F subset of Fortran [6], which enforces very clean coding constructs and for which free compilers exist.

Our DOM implementation builds on the SAX implementation, using callbacks to fill the appropriate in-memory data structures. The DOM Core API provides two different sets of interfaces to an XML document; one presenting an "object oriented" approach with a hierarchy of inheritance, and a "simplified" view that allows all manipulation to be done via the node interface [5]. As casting and inheritance are not supported in F95 we provide only a "flattened" implementation in which all DOM nodes are described identically (regardless of what they represent). However, even though all data are stored in identical structures internally, we still provide all of the DOM 1.0 methods, so that from the point of view of the end-user there is no difference between this implementation and any other. The actual nodes tree is constructed using a linked-list/pointers strategy similar in spirit to the C GDOME implementation. In addition to the node interface there are two other interfaces central to the DOM representation of an XML document: *namedNodeMaps*, which are used for accessing attributes and *nodeLists* which are used when accessing the child nodes of a given node, or indeed any collection of related nodes. Again, both interfaces have been implemented using linked lists. There are still a number of missing features; most notably there is no validation. In addition DOM should be able to handle different text encodings and any DOM implementation should use 16-bit strings unfortunately there is no simple way to do this in Fortran and we only support 8 bit character encodings, such as ASCII/ISO 8859-1 (although we do not expect this to be a problem for most work). It is important to note that

Fortran does not offer a native variable length character type (or character array type) therefore variable length strings are handled in a separate module [7]

For convenience we have provided libraries for generating well formed XML and CML. The XML formatting library (WXML) can be used to manage multiple XML files and will generate well-formed XML or else inform the user when well-formedness rules are violated. It also allows the user optionally to indent output. The CML library extends the base XML formatting modules, again checking and enforcing well-formedness, but in addition providing convenience routines for generating large CML elements. In general the CML routines reflect the underlying CML and CMLComp schemas. Routines are typically overloaded to take any of the allowed data types, and use Fortran's optional arguments to represent optional CML attributes.

In the near future we plan to offer an Xpath API built on top of the DOM subsystem (there is already a very simple but useful "stream-Xpath" module based on the SAX parser). The complete Fortran XML set of libraries is open-source, distributed with the BSD license [8,9].

5. Chemistry and Physics Software

Using the libraries described in section 4 we have extended a range of solid-state chemistry/physics software packages to use the CML and CMLComp languages described in section 3. We have initially focused on three programs [10-12].

- SIESTA: an *ab initio* simulation package based on density functional theory and the pseudopotential approximation. SIESTA is used to perform accurate and detailed, but computationally expensive, electronic structure calculations.
- GULP: a classical mechanics simulation program that uses empirically parameterised potentials to describe the interaction between atoms, making it computationally less demanding than *ab initio* methods. GULP can calculate lattice energies, optimised structures, elastic and vibrational properties for large and complex systems. GULP may also be used for developing potentials.
- DL_POLY: also uses a classical description of the atomic interactions based on interatomic potentials, including those developed in GULP. DL_POLY main role is to investigate the dynamic and thermodynamic properties of extremely large systems.

6. A Real World Example: Pollutants in Soils

One of the main aims of our work is to investigate the mechanisms by which pollutant molecules such as DDT, dioxins and biphenyls, become bound to soil minerals. It is the complex data processing requirements of these simulations that have driven much of this work. It is not the purpose of this paper to discuss the scientific details of this work rather the aim of the following discussion is to highlight the extreme combinatorial nature of the problem.

From a scientific point of view, we hope to determine where and how these molecules become bound to soils, systematically searching for trends across families of related compounds. In practical terms this is an extremely large and open-ended task. There are potentially thousands of candidate molecules. Concentrating on the provisional list above there are 420 molecules (including 210 dioxin and 209 biphenyl congeners). There is an equally large choice of minerals, and even for a given mineral there may be many relevant surfaces to investigate. Moreover, even for a single choice of molecule, mineral and surface, we need to investigate the potential energy surface of the system (i.e. how the energetics of the problem change as function of the molecules' position above the mineral). This in itself requires a large number of calculations. In fact, the nature of the problem requires that calculations be performed, not only on the whole system, but also on molecules and minerals separately, in order to assess their individual contributions to the energetics. It should be clear that any systematic analysis would require many thousand of calculations.

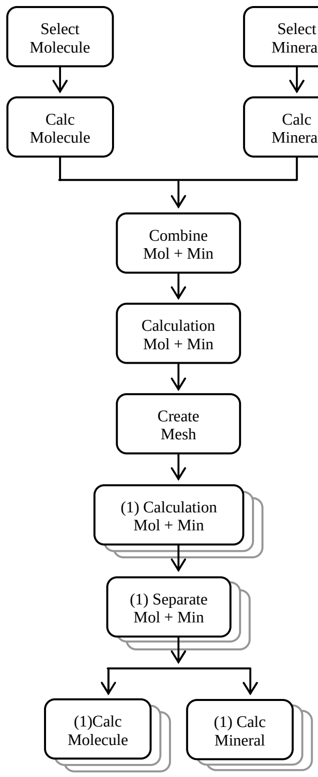

Figure 2. A diagram showing the sequence of simulation that we are using to model the interactions between pollutants and soil minerals.

The series of events described in the preceding paragraph are illustrated in the flow diagram in Figure 2. The diagram highlights all of the major features of this complex simulation scheme: (1) there are dependencies between calculations, i.e. certain calculations can not be performed until other calculations have finished (2) every time a major calculation is completed there is a small post-processing step needed before the next simulation(s) can be performed and (3) the vast majority of calculations can be performed in parallel. We aim to automate this series of events using the Condor DAGMan utility [13]. However, while it is possible to automate the execution of jobs with off-the-shelf workflow tools, it is still necessary to deal with data as it passes from program to program. The creation of CML aware applications has been essential in tackling this problem. Moreover, the use of XML allows one to efficiently separate the workflow problem from the underlying simulations, so that it is possible to replace the simulation software with *any other* CML aware application (that calculates the necessary data) in a component-like fashion. Initial calculations have been performed using the e-Minerals Condor pools at UCL and Cambridge. Full-scale calculations will be performed using the e-Minerals minigrid.

7. Conclusions

Computational advances have paved the way for new kinds of chemical and physical simulations that have forced us to re-evaluate the way in which we deal with our data. The authors believe that a move, away from the traditional data representation formats of chemistry and physics, to more robust XML based description of data, will improve software interoperability and data exchange more generally. The creation of CML aware scientific software has allowed us to efficiently implement multipart scientific workflow schemes. Moreover, standardization of data representation (within our project) allows us to treat these applications as interchangeable workflow components. The other major success of this work is the development of free, open-source XML processing libraries for Fortran. The libraries offer implementations of the SAX, DOM and Xpath standards, thus providing a useful resource for the entire Fortran community.

References

[1] P. Murray-Rust and H. S. Rzepa, *Chemical Markup Language and XML Part I. Basic principles*, J. Chem. Inf. Comp. Sci., **39**, 928 (1999).

[2] P. Murray-Rust and H. S. Rzepa, *Chemical Markup, XML and the World Wide Web. Part II: Information Objects and the CMLDOM*, J. Chem. Inf. Comp. Sci., **41**, 1113 (2001).

[3] Y. Zhang, P. Murrary-Rust, M.T. Dove, R.C. Glen, H.S. Rzepa, J.A. Townsend, S. Tyrrell, J. Wakelin, E.L. Willighagen, *JUMBO – An XML Infrastructure for eScience*, Proceedings of UK e-Science All-Hands Conference, September 2004.

[4] http://sax.sourceforge.net/

[5] http://www.w3c.org/DOM

[6] http://fortran.com/imagine1

[7] http://nn-online.sci.kun.nl/fortran/xml

[8] http://lcdx00.wm.lc.ehu.es/ag/xml

[9] After completing the initial stages of the project we learnt of two other Fortran XML initiatives (see http:// sourceforge.net/projects/xml-fortran/ and ref [7]). However, we offer both SAX and DOM interfaces and an Xpath like interface to XML. In addition we provide a basic XML formatting library, making this the most complete and robust Fortran-XML library that we are aware of.

[10] J.M. Soler, E. Artacho, J.D. Gale, A. García, J. Junquera, P. Ordejón and D. Sánchez-Portal, *The Siesta method for ab initio order-N materials simulation*, J. Phys.: Condens. Matter, **14**, 2745 (2002).

[11] J.D. Gale, *GULP - a computer program for the symmetry adapted simulation of solids*, JCS Faraday Trans., **93**, 629 (1997)

[12] I.T. Todorov and W. Smith, *DL_POLY_3: The CCP5 National UK Code for Molecular Dynamics Simulations*, Phil. Trans. (in press).

[13] http://www.cs.wisc.edu/condor

Acknowledgments

JW would like to thank the NERC for funding.