Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles

Peter Murray-Rust*,† and Henry S. Rzepa‡

Virtual School of Molecular Sciences, School of Pharmaceutical Sciences, University of Nottingham, U.K., and Department of Chemistry, Imperial College of Science, Technology and Medicine, London SW72AY, U.K.

Received May 26, 1999

Chemical markup language (CML) is an application of XML, the extensible markup language, developed for containing chemical information components within documents. Its design supports interoperability with the XML family of tools and protocols. It provides a base functionality for atomic, molecular, and crystallographic information and allows extensibility for other chemical applications. Legacy files can be imported into CML without information loss and can carry any desired chemical ontology. Some applications of CML (Markush structures, chemical searching) will be discussed in later articles. An XML document type declaration (DTD) for CML is included as a Chart.

INTRODUCTION TO XML TERMINOLOGY AND SYNOPSIS OF TERMS

Prior to introducing XML (extensible markup language) and CML (chemical markup language),[1] we feel it important to define some key XML terminology (Table 1).[2] We assume the reader is familiar with the basis of HTML (hypertext markup language) and its application in chemistry.[3] XML uses the same syntactic approach but, deliberately, has less flexibility and requires more precise application. This makes it much easier to write parsing software for well-formed documents (first three rules below). The most important parts of the XML syntax and name space specifications are as follows:

(1) All tags must be balanced (⟨FOO⟩...⟨/FOO⟩). Tags can contain any alphanumeric character and "−", "_" and ":" but must not contain white space.

(2) The shorthand ⟨FOO/⟩ is equivalent to ⟨FOO⟩⟨/FOO⟩ (an empty element).

(3) All attributes must be quoted; foo="bar".

(4) All names are case sensitive (e.g. ⟨p⟩ and ⟨P⟩ are deemed distinct).

(5) Comments can be inserted in most places as a string within the delimiters, i.e., ⟨!- - ... - -⟩. The string "- -" may not occur internally.

(6) Processing instructions (strings within ⟨?...?⟩) are application-specific and, apart from those required by XML itself, are not discussed in this article.

(7) The ":" character in tags is reserved for name spaces (e.g. ⟨cml:molecule⟩. The prefix is equivalenced to URI to preserve global uniqueness. Thus ⟨cml:molecule xmlns:cml="http://www.xml-cml.org"⟩ and ⟨z23:molecule xmlns:z23="http://www.xml-cml.org"⟩ are equivalent.

(8) Name spaces are inherited by child elements; e.g. in ⟨cml:molecule xmlns:cml="http://www.xml-cml.org"⟩ ⟨atom⟩...⟨/atom⟩⟨/cml:molecule⟩, the atom child is equivalent to ⟨cml:atom⟩...⟨/cml:atom⟩.

(9) Name spaces can be nested and the youngest ancestor with a name space declaration determines the current scope

(10) A DTD may, but need not, be provided for a document instance. If it is, a validating parser can check whether the document is valid (i.e. it conforms).

A number of related applications are under active development,[2] all expressed in XML itself. The most relevant to this article are as follows:

XSL. An XML-based language allowing document transformation (filtering, reordering, etc.) and subsequent rendering. Modern browsers will provide native XSL support. XSL has a UNIX-like syntax for navigating to ancestors and descendants.

XQL. A powerful XML-based query language specifically designed for structured documents. Queries can be based on any combination of (a) element names, (b) attribute names, (c) attribute values, and (d) element content. This is further enhanced by the ability to query the context of an element (ancestor, sibling, etc.) and to interrogate order ("next element"). At present the XQL syntax is similar to XSL. It is likely that XQL will allow extensions to user-defined functions.

XLINK and Xpointer. This is a general mechanism for addressing any element within a document. It will manage addressing to URIs (uniform resource identifiers) and URNs (uniform resource names) and is likely to have a syntax similar to XSL and XQL for addressing within documents. XLINK can be thought of as a greatly extended set of the HTML hyperlinking facility and inter alia allows (a) selective transclusion of documents, (b) creation of a database of links for a set of documents, (c) treatment of XLINKs as elements (i.e. first-class information objects), and (d) multiended links.

RDF. This is an XML application for creating metadata, using property triples (element1, property, element2), where the properties are first-class elements. It is expected to use the Dublin Core approach frequently.[4]

[XML is developing very rapidly, and the W3C publishes frequent updates to the XML family of activities (http://www.w3.org/), especially recommendations and drafts of

† University of Nottingham.
‡ Imperial College of Science, Technology and Medicine.

10.1021/ci990052b CCC: $18.00
Published on Web 09/18/1999

Table 1. Key Terms of XML Terminology

document type declaration (DTD)	formal (BNF-like) specification of the allowed components and structure of a document conforming to that DTD (a DTD should include ontological information about the elements and attributes)
XML schema	extension of DTD functionality, written in XML[a]
XML	extensible markup language
XSL	extensible stylesheet language[a]
XQL	XML query language[a]
XLINK	linking scheme (hypermedia) for XML[a]
Xpointer	Xpointer provides the addressing mechanism[a]
markup	introduction of special characters into documents to identify content and structure
element	component of a document delimited by a start tag and end tag
attribute	name−value pair located in the start tag of an element
element name	name of an element
tag	syntactic construct indicating the start or end of an element (the element includes its tags and their content)
tagset	informal term for the collection of element names in a document or set of documents
#PCDATA	character (string) data in element content
subelement/child	element completely contained within an other element in a hierarchical manner
element content	anything within the start and end tags of an element (normally a mixture of #PCDATA and child elements but may include comments and/or Pis or may be empty)
descendant; ancestor; sibling	terms describing the relation of elements in the tree/hierarchy of a document
document instance	XML document, containing exactly one root element which may (and usually has) subelements
well-formed	document which conforms to XML syntax (e.g. elements nest correctly, end tags are present, and attribute values are quoted)
valid	well-formed document whose structure, element names, attribute names, and values are consistent with a particular DTD
W3C	worldwide web (WWW) consortium, a vendor-led, vendor-neutral consortium for the development of protocols for using the WWW
CML	chemical markup language, a conforming application of XML (CML is released as a series of drafts; this being the first comprehensive publication)

[a] Specification under development.

these. It is almost certain that some details in this article will be obsolete when it is published on paper, and the W3C site should always be taken as providing the authoritative view on XML protocols. In addition, the W3C site contains many excellent introductory and reference textbooks on XML, many including software, and readers may find these helpful in reading parts of the article. It is a principle of the XML activity that working software and examples should be available wherever possible, and a large amount of OpenSource and commercial software is now available. The nonprofit OASIS organization provides support for XML, and its pages (http://www.oasis-open.org) contain the most comprehensive set of XML resources of all kinds. The authors maintain a web site for CML activities at http://www.xml-cml.org/, and the latest developments in CML specifications, examples (including those shown in this article), and CML-aware software will be found there. CML evolves in step with developments in XML and, like XML, progresses through a series of drafts identified by date (i.e. CML-1999-05-15). These drafts are independent of any CML-aware software. Because of the extensive set of XML protocols and their terminology, we present them before the introduction. All XML examples in this article are well-formed, but some are deliberately not valid CML in that they contain tags invented for that example. Finally, readers should note that the term "element" almost always refers to an XML element and not a chemical element.]

INTRODUCTION. XML AND THE WORLDWIDE WEB

The Worldwide Web (WWW) is radically changing the way that we structure and present our information. The technological developments are remarkable, for example the easy creation, modification, reuse, and linking of documents, but there are even greater sociological implications. Since the new technology allows anyone to be their own publisher, there is an enormous challenge to the conventional institutions of publication at all levels. Chemistry will be considerably affected by this, and this article addresses how the current thinking on the WWW will impinge on chemistry. It outlines the strategy for using the new WWW markup languages based on the extensible markup language (XML) to create a markup strategy.

In keeping with the philosophy of other W3C protocols and markup languages, chemical markup language (CML) is issued as a series of drafts. This represents the first full publication of CML and describes the basic elements. CML is designed to be extensible, and we propose that it could form the core of other markup activities in chemistry. This article concentrates on the basic chemical information components (atoms, bonds, electrons) and crystallography. CML can also support more complex chemical concepts such as reactions, chemical grammars (e.g. Markush structures and combinatorial chemistry), and chemical queries (substructure searches). The final design of these will depend on the syntax and support for XLINK, and when these W3C recommendations are available, these topics will be published in following articles.

Components and Information. In paper-based publication the medium and message are inextricably linked and can normally only be processed by humans. The development of electronic scientific publishing has often involved conventional "paper" documents with associated electronic data. Thus the electronic deposition of crystal structures and macromolecular sequences is now a routine part of publication.[5] In general, however, the document part of the paper, though often carried in electronic form, is conceptually based on a conventional paper-based document structure. A major feature of this approach is that form and content are mixed.

For documents to be reliably machine-processable the paper image is not sufficient. It is necessary to identify the

various components of a document, both in their intrinsic nature and their role in the document structure. This process is termed *markup* and has been adopted by many publishers through the standard generalised markup language (SGML, ISO:8879).[6]

Markup has now been extended to become a central tool for the exchange of information over electronic networks. The introduction of hypertext markup language (HTML) was the first step in producing a globally accepted nonproprietary method for transmitting machine-processable electronic documents. Typical markup elements of HTML include components such as ⟨IMG⟩ or ⟨ADDRESS⟩, document structure elements such as ⟨HEAD⟩, ⟨BODY⟩, ⟨TITLE⟩, ⟨P⟩, local structure elements such as ⟨UL⟩/⟨OL⟩ + ⟨LI⟩ and ⟨TABLE⟩ + ⟨TR⟩ + ⟨TD⟩.

The use of HTML made a critically important contribution to hypermedia by introducing (unbounded) hyperlinks or anchors (⟨A⟩) and thus encouraged the use of both hyperdocuments and active components. We developed the use of these for chemistry by proposing the use of MIME types to label chemically significant components of hyperdocuments.[7] More generally, the adoption of HTML has promoted the idea that documents are not monolithic objects but can be regarded as built from smaller components with defined and varied content and functionality. It is generally recognized, however, that HTML has weak support for structure and poor tools for specific markup and functionality. In scientific disciplines there is a key need to exchange "data" such as numeric quantities with units and ranges and domain-specific objects such as mathematical equations or chemical reactions. HTML cannot address these, so the Worldwide Web Consortium (W3C) has undertaken a major program to support robust, extensible markup.

The members of the W3C are (primarily commercial) organizations who have agreed to create communal, nonproprietary protocols for, inter alia, the exchange of information over the WWW. The W3C's processes are confidential, but its results are open. The cornerstone of this is XML, a "very simple subset" of SGML. One of us (P.M.-R.) was invited to be part of the initial working group on XML, and as a result we suggest that some aspects of the process, as well as the end product, may be of value to the chemical informatics community.

Syntax, Semantics, and Ontology. The representation of information in electronic form usually involves several layers: encoding, syntax, semantics, and ontology. Each of these is discussed separately below.

(a) Encoding. This specifies the method for mapping bytes (octets) or similar concepts onto characters. Thus ASCII (the American Standards Committee for Information Interchange) has specified that the character "a" is represented by the byte with value 65. Commonly used supersets of ASCII are ISO-8859, ISO-Latin-1, and ISO-10646 ("Unicode"). The latter is based on 16 bits and endeavors to support all the major character sets in the world. XML (and Java) are designed to be Unicode-compliant.[8] It is critical to define the character set used in a document, and XML documents should start with a declaration such as

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

Lack of understanding of encoding can lead to serious corruption of information; we know of cases where characters for degrees (superscript zero) and micro (Greek μ) have been corrupted by incorrect assumption of character sets.

(b) Syntax. This specifies how the byte stream should be tokenized. It is dependent on the application and is frequently underspecified. An example of a syntactic problem from an MDL molfile is

```
YOHIMBINE
GTMACCS-II111109515132D 1   0.00479     0.00000     0    GST

 29 33  0  0  1  0              1 V2000
0.4699    2.1336    0.0000 C   0  0  0  0  0  0
0.6808    1.2945    0.0000 C   0  0  1  0  0  0
[...]
```

Here the string in the second line contains information on how the molecular information was created, the date, the dimensionality, etc. Without a manual it is impossible to identify the tokens (e.g. as "GT" "MACCS-II" "112395" "1513" "2D"). Errors in parsing (the process of tokenizing and structuring this information) are therefore common and can be extremely damaging. We know for example of cases where "CL" (chlorine) has been converted to "C" (carbon) by incorrectly written parsers for the ubiquitous PDB format.

XML enables authors to remove all syntactic ambiguity, and this, in itself, is an undramatic but major step forward for chemical informatics. XML has solved many troublesome syntactic problems (e.g. how to include end-of-line characters, to quote quotes, and so forth).

(c) Semantics. This allows meaning to be added to tokens or larger components of documents. The above example could be rewritten as

```
<PROG>MACCS-II</PROG>
<DATE>111095</DATE>
<TIME>1513</TIME>
<COORDS>2D</COORDS>
```

Note that semantics provide a method for adding meaning or behavior and do not in themselves indicate the meaning. XML describes this as a DATE element with #PCDATA (i.e. string) content "111095", but this does tell us what a "DATE" is or how to interpret "111095". Humpty-Dumpty can choose to interpret ⟨GLORY⟩ as meaning a "nice knockdown argument", and this is a central concern for this article. Humans are sometimes good at guessing the meaning of tag sets, but machines are not. There are at least three approaches to adding semantics.

(a) Creating *a generally agreed tag set*. HTML is the best example of this, and all HTML-compliant software treats ⟨A href="foo"⟩ in the same way, i.e. a hyperlink. Other tag sets (formally referred to as document type definitions or DTDs) may treat ⟨A⟩ in other ways (author, answer, etc.). XML itself does not create tag sets, but some of its applications (such as XSL, the extensible stylesheet language, and SVG, scaleable vector graphics) do. It is likely that the W3C will sanction in the region of 10−20 DTDs for use in horizontal application, including mathematics, but this will not extend to chemistry. In this article we propose a base set of tags for markup in chemistry. Meaning can be added to a fixed tag set either by reformatting the input so that humans instinctively understand it (e.g. bullets can be added before ⟨LI⟩ elements) or by attaching programmatic functionality. Many browsers interpret ⟨IMG SRC="foo.gif"⟩ to

read the contents of the file foo.gif as a GIF image and display it at the current position in the document. Other browsers e.g. concerned with accessibility may simply print an alternative text string ⟨IMAGE alt="help"⟩ or pipe the text to a speech synthesiser.

(b) Adding *programmatic functionality through (modular) code*. Thus a browser encountering ⟨molecule href="foo.pdb" mimeType="chemical/x-pdb"/⟩ might interpret this as a command to regard the contents of foo.pdb as a Protein DataBank file and display it with appropriate software. We expect that authors will wish to attach programmatic functionality to some of the proposed tag sets in this article.

(c) Linking *to glossaries/ontologies/metadata*. This is an almost essential approach for robust markup. Thus the DATE element above cannot be universally interpreted without explicit or implicit links to a formal description. ⟨DATE convention="ISO-8601"⟩1999-11-10⟨/DATE⟩ would specify the object as a date with month = "11" and day = "10". Similarly the TIME field is meaningless without specifying the (probably default) time zone information and other constraints. ISO11179 is likely to become an important standard for describing metadata in XML.

(d) Ontologies. These represent formal descriptions of terms, concepts, behavior, etc. Creating and using useful ontologies is an extremely challenging task for those exchanging and reusing data. In some cases—such as ⟨DATE⟩ or ⟨FLOAT⟩—it is likely that we can identify existing standards (ISO, IEEE, IETF, etc.) which encapsulate what we wish to describe and provide enough power to do it precisely. If the concepts are independent of human language, then a global consensus may be possible, as the task is primarily to alert the electronic community to adopt a common approach. In many cases more than one ontology may be in common use for a particular concept. In an ideal case the curators of the ontologies will have provided equivalences between different ontologies. Thus ISO standards often reference each other or other standards bodies. For an ontology to be valuable it must be widely used and provided by a curatorial process, which is widely accepted. An example is the CIF dictionaries from the International Union of Crystallography (IUCr).[5] These have been worked out in public view, triennially reviewed by the Union, and published in electronic form.[9] The crystallographic community is sufficiently cohesive that it is likely that the CIF ontology and associated metadata (e.g. ranges, units) will be universally accepted in appropriate publications or computer programs. Rather than using the ambiguous term "temperature factor", crystallographers can globally refer to a construct such as "atom_site_U_iso_or_equiv", the metadata for which includes ranges, units, and the algorithm for converting to other representations. In many cases, however, terms are used for concepts whose interpretation differs between communities or between individuals. Thus almost every chemist will have a slightly different personal interpretation of what is, or is not, a "molecule". Where we can we should reference accepted authorities (e.g. IUPAC, ACS, CAS, IUCr, IUPAB, etc.). By its nature, however, the complete ontology of chemistry cannot be captured by any person or organization. We regard Linus Pauling's "*The Nature of the Chemical Bond*"[10] as representing the last time that a chemical ontology was feasible, and then only by a genius. *In this article we very deliberately do not attempt to develop or reconcile chemical ontologies.*

We remark that in knowledge management a parallel hierarchy is often given as data, information, knowledge, and wisdom.

There is a rough correspondence between this and the above analysis: representing data and syntactic analysis; structured information and semantic analysis; the meaning of information (knowledge) and ontologies.

Information Loss. Current molecular "file formats" and database entries normally choose a subset of available information that can be captured. Many older formats are based on fixed length records (often 80 characters) and a restricted order for those records, and this limits or completely denies extensibility. In general two different formats have different ontologies and cover a different subset of chemical information space.

Every piece of chemical software uses its own ontology, usually implicitly. The relevant information has to be supplied in the input files, but because the implementation of ontologies is very expensive, the program is usually built to accept a small number of file types. When chemical information is passed from one program to another, ontological conversion is necessary. However only the concepts present in both ontologies can be passed, and this normally leads to *information loss*. For example, a PDB file does not explicitly hold bond orders, while an MDL molfile does not hold occupancies. Both these concepts are therefore lost in an interconversion. There is also often an *ontological loss* since the meaning of a common concept (e.g. bond order) may be different in both.

CML has been designed to allow conversion from legacy files to CML *without information loss*. In some cases this is because the information can be represented in an abstract, convention-free form. In other cases, however, this is essentially a syntactic conversion with annotation of the original *convention* (i.e. ontology) used. While we do not believe a single ontology is possible for chemistry, the use of CML may highlight agreement on some ontological subsets. We continue to emphasize that conversion from CML to other formats will probably involve information loss. The use of CML as input to programs should make it easier to identify chemical information in files and to convert where this is possible.

Documents and Data. Currently "documents" and "data" use distinct technologies that do not easily interoperate. Documents are mainly managed by word processing packages, while data are managed by relational or object-oriented databases. Holding data in a word processor format is at best proprietary, while databases do not have good facilities for structured text. Thus in a conventional "publication", the data and its textual description are often separated. In the WWW context this often appears frustrating because one or another of the components is not on free public access or in a form not easily associated with the other. In databases we cannot refer to the "full text" of the article the data relates to. In full-text journals, the "data" are often missing, relegated to supplemental information or expressed as scanned images of pages. A key aspect of XML is that it allows conventional "documents" and "data" to be seamlessly integrated into a single XML document instance. Thus

```
<analytical>The <link id="23"> final product</link> had a
<item type="float" title="melting point" glossary="terms.xml#mpt"
units="K"> <value>345.6</value><min>343</min><max>347</max></item>which
agrees well with <link href="citations.xml#23">other measurements</link>
for this material.</analytical>
```

contains running text (which could be styled and printed) as well as precisely identified and ontologically described numeric data. With XSL the fragment above could be rendered into a human readable report or entered into a database. When XML authoring tools allow the facile creation of such documents, the community will have easy access to a much higher quality of reusable published information. As an example, we could retrieve all the melting points in a given document by an XQL search of the form

```
//item[@glossary='terms.xml#mpt']/value!text()
```

which means "find all ⟨items⟩ with a glossary attribute corresponding to a melting point, extract the child of element name value, and return its text value". Note that we do not have to know the structure of the document and can ignore other elements we are not interested in. In this way the document becomes a portable database of considerable power. The author may not even be aware that their document may be reused in this or other ways. The greatest hurdle to reuse is not technology but the (semialtruistic) commitment to providing a free service to others in the community. J. D. Bernal[11] urged us to "get the best information in the minimum quantity in the shortest time, from the people who are producing the information to the people who want it, *whether they know they want it or not*" (our emphasis).

MARKUP LANGUAGES FOR CHEMISTRY

Creating markup languages for a domain is challenging because fuzzy ontological concepts have to be mapped onto precise syntax and semantics. The biggest opportunity, and also the problem, is that the success of XML has enabled effective semantic support for ontologies but that most users of information have never been exposed to these disciplines or concepts. In chemistry there has been a graphical approach to the ontology of chemical structure for over 150 years, and most chemists are trained to think in this way. More recently it has been possible to devise machine processable ontologies for restricted domains such as organic molecules, crystal structures, proteins, and nucleic acids.

Graphical approaches normally hide many ontological problems, and, for example, machine processing of graphical representations is not generally successful. A similar problem exists in mathematics where typography and layout have been a key approach for several hundred years. This expressiveness is essential for developing new ideas in mathematics, but it makes it difficult for machines to process typeset mathematics, as presented in the TeX system.

The W3C decided that mathematics was a key activity and set up a working group, with membership from learned societies, publishers, software houses, etc. This meets regularly both face-to-face and virtually to develop the language and to make a vendor-neutral recommendation for MathML. A fundamental feature of MathML is that it must interoperate with other W3C protocols, including a set of initial design requirements. At an early stage it was decided that MathML must support both presentational aspects and semantic content. The presentation could approach the expressive power of TeX (for human readers), while the semantics should support the use of mathematics for technical disciplines and math education, and be processable by symbolic algebra packages. In the recent recommendation, both approaches are supported and authors have a choice of which to use.

CML has been developed in a similar spirit, though so far without a formal procedure. Several CML drafts have been published on the Internet[12] with invitations to comment solicited and responses to comments made. It is our intention that CML or its descendant(s) is ultimately adopted by appropriate trusted vendor-neutral organization(s) in the chemical community. The design features have been abstracted from similar W3C activities, with chemical additions where appropriate.

Before introducing the design, we stress some key criteria:[13]

(a) *CML must be ontologically neutral.* At present most molecular "file formats" contain complex, often implicit, ontologies which are not necessarily reconcilable between different formats. Thus many systems can describe "aromatic bonds" or "formal charges", but these concepts are not completely portable. In other words simply mapping (say) bonds designated of order 4 (MDL molfile) to the designation −5 (CCDC file) may be ontologically unsound since the algorithms used to create these may be different. Hydrogen counts, formal charges, atom "types", bond orders, and valences are related in a complex manner and are not always used consistently. In some systems bond orders are related to measured lengths, in others to calculated properties, and in others to electron counts. Therefore while CML must support the concept of bond order, it cannot arbitrate between different uses. CML has a minimal ontology[14] that is primarily for convenience in managing the components. Thus ⟨cml:molecule⟩ has no chemical significance and functions purely as both a handle for the author to use for a concept and as a container for ⟨atom⟩s, ⟨bond⟩s ⟨electron⟩s, other ⟨molecule⟩s, and anything else. Most relevantly it announces the components around which software may be built.

(b) *CML cannot anticipate the uses it is put to.* An author may wish to describe a hypothetical molecule; CML must not forbid this, although it may not actively support it. A theoretical article may wish to discuss the difference between a Kekule structure and the "equivalent" delocalized one.

(c) *CML cannot provide chemical perception.* Chemists are trained to recognize features in molecules and frequently to make mental translations between them. Common examples might be tautomers, delocalized systems, conformations, and representations of bond types (double, dative, bipolar). It is dangerous to design any interconversion protocol that would be automatically applied.

There is also often considerable fuzziness in chemical information. For example the reaction schemes

$$C_6H_5CH_3 + Cl_2 \rightarrow C_6CH_5CH_2Cl + HCl$$

$$C_6H_5CH_3 + Cl_2 \rightarrow C_6CH_5CH_2Cl + ClC_6H_4CH_3$$

make different semantic use of the "+" on the right hand side, meaning either products in equimolar amounts or alternative products. Chemists are trained to make the

Table 2. Chemically Relevant Data Types and Associated XML-Based Protocols

data type	XML-based protocol
running text, with inserted objects	HTML or XHTML (XML conforming HTML)[a]
numeric data with units and metadata	XMLSchema[a]
graphical and image data	PNG, SVG[a]
mathematical variables, equations, and other quantities	MathML
tabular data	HTML, CALS
graphs	SVG[a]
molecular structures and crystallographic data	CML[a]
reactions	CML[a] + XLINK/XMLSchema[a]
structured figures (e.g. reaction schemes)	SVG[a] + CML[a] + XLINK[a]
macromolecular objects	BSML[a], BioML[a], CML[a]
metadata	RDF[a], ISO11179/XMI[a]
hypermedia	XLINK[a] + XPointer[a]
citations and other bibliographic material	DOCBOOK
terminology	MARTIF/ISO12200, VHG/ISO12620

[a] Specification under development.

distinction (often subconsciously), but this is a difficult context-sensitive analysis which machines cannot be expected to make. It takes considerable effort to design a general reaction-based representation.

Other areas of fuzziness often arise from graphical representations introduced primarily for printed representation of molecular structures. Examples are as follows: wiggly bonds (does this mean "unknown stereochemistry" or "both epimers present"?); curly arrows (electron movement or atom movement?); thickened line (nearness to viewer or wedge bond?); relative or absolute stereochemistry (does the object describe a racemate or a single enantiomer?); stereochemistry (this is a particularly difficult area since it brings together the "2D" and "3D" representations of molecules; some concepts (e.g. wedge bonds) only have meaning when related to 2D graphics, and there are a variety of descriptions that are very difficult to interconvert in a machine: wedge/hatch notation, R/S notation, D/L notation, *endo*/*exo* notation, and *cis*/*trans* and *E*/*Z* notation).

In CML we have taken an approach to global and local stereochemistry based on the chiral volume (or other scalar quantity) related to identified points (atoms or dummy atoms). It can be extended to cover metal coordination spheres. This has the value of being independent of drawing conventions, recursive algorithms for atom priority, or local stereochemistry.

CML in other words has been designed to support the molecular components in structured documents. It would be impossible, indeed arrogant, to try to devise an all-embracing language, and many complex areas, e.g. combinatorial chemistry, are not yet specifically supported. However the simple building blocks in CML can be combined to create languages of greater power when required.

PRESENTATION VS CONTENT

This is a key feature of XML and central to CML. The XML philosophy encourages designers of DTDs to separate content and presentation wherever possible. This means that the content can be reused or re-presented at a later stage if required. This is often difficult, and for CML we have been helped by the design of MathML. MathML allows for presentational mathematics; i.e. it dictates the layout of the symbols and their typesetting and is heavily influenced by Knuth's TeX system, the lingua franca of mathematics publication. However TeX as a presentational format cannot always be parsed unambiguously into its semantic components; thus "fx" could mean the variable "fx", $f \cdot x$, or $f(x)$. Therefore MathML introduced additional support for "content", where the markup could be algorithmically parsed by symbolic algebra packages. The primary aim for CML, therefore, is to capture the content of chemistry rather than its presentation.

CHEMICAL DOCUMENTS AND THEIR COMPONENTS

XML "documents" can describe a very wide range of chemical information, including "publications" in journals, reports, memos, patents, regulatory processes, etc. Entries in databases could include the following: data captured in laboratories or by instruments; output from programs (data analysis, theoretical, simulation, etc.); input to programs; searches; catalogs and dictionaries of chemicals.

From an extensive study of these and other chemical objects we have identified a number of generic component types. In many cases these are isomorphous component types being used in other disciplines. Thus a melting point (see above) has the same abstract structure as the price of an item or the age of a person. The abstract data type might be "floating point number, with units, allowed range and links to metadata". Since the W3C is actively developing several types of abstract XML information object, it makes great sense to reuse these wherever possible, rather than reinventing them in a chemical context. If we can do this, we greatly widen the applicability, support, and tools available and drastically reduce the acceptance and training required. As a general principle, therefore, we *reuse concepts and protocols from the W3C and other bodies wherever possible*. It is entirely possible in the future that some of the current components in CML might then be replaced at an appropriate date by accepted W3C approaches. The abstract data types we have encountered widely in "chemical publications" are listed in Table 2. In each case we have listed one or more XML-based protocols to support them.

XML is designed to be extensible in several ways. We envisage at least four such mechanisms.

(a) *Allowing communities to develop their own tag set* (e.g. MathML, CML). A community can decide to use an established ontology and represent it in XML or to create a new one. In the area of biological sequences there are currently two proposals for managing sequence data (BSML, BioML); it will be important to ensure that they have

ontological interoperability with e.g. the CORBA-based systems of many of the (inter)national data providers.

In developing tagsets there are several strategic decisions to be made:

The granularity of the markup. Do we wish to address a single macromolecular sequence or every atom in a sequence? Should every person mentioned in a document be captured?

The flexibility allowed to authors. Must certain data fields or elements be present (e.g. must a ⟨molecule⟩ contain ⟨atom⟩s)? In general documents which are part of required, controlled processes such as safety sheets, clinical chemical data will have little flexibility, while journal articles may have more.

The use of attributes or subelements. This is a hotly debated (but usually arbitrary) decision: ⟨person lastname="Pauling"/⟩ and ⟨person⟩⟨lastname⟩Pauling⟨/lastname⟩⟨/person⟩ are essentially identical, but one may have minor advantages over the other in ease of authoring or other processes.

(b) *Allowing different tag sets to be combined in a document instance, through XML name spaces.* This is an important development in XML because components from different DTDs can be combined without risk of tag collision. Thus CML might define an ⟨element⟩ element (for "chemical element"), while XSL already defines an ⟨element⟩ for "XML element". These collisions can be avoided with name spaces so that the document might contain

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
[…]
<xsl:element>
<cml:element xmlns:cml="http://www.xml-cml.org">...
</cml:element>
</xsl:element>
</xsl:stylesheet>
```

There is guaranteed planetwide uniqueness because the name space prefixes (xsl, cml) are mapped to the unique URIs shown. In this way we can confidently include all the DTDs mentioned in the above list as collision-free, even without knowing their individual tagsets.

(c) *Complex and irregular data structure.* Conventional approaches (RDBs or tables have difficulty including information that does not map to rectangular form. XML can include nested trees to any depth. Thus, for example, a molecule that contains ^{13}C and ^{35}Cl could include a chemical shift for the carbon and a quadrupole moment for the chlorine. ⟨molecule⟩s could contain information from a wide range of disciplines or be similarly contained.

(d) *Arbitrarily complex linking.* Any element(s) in an XML document can be linked to any other element(s) with user-defined roles. In this way complex relationships can be built, while standard XLINK approaches are used. A chemical reaction can often be described by identifying the reactant molecule(s), the solvent and conditions, and the product molecule(s). Every ⟨atom⟩ in every molecule can be uniquely identified (e.g. by the XML id attribute). The mapping of atoms in the reactant to those in the product is a list of XLINKs.[14] In a similar way a reaction scheme can be thought of as a collection of simpler objects: molecules, numeric values, descriptive annotations, etc.

DESIGN CRITERIA FOR CML

The WWW process usually includes initial goals, and these have been outstandingly useful. Those for XML were as follows:

(1) XML shall be straightforwardly usable over the Internet.

(2) XML shall support a wide variety of applications.

(3) XML shall be compatible with SGML.

(4) It shall be easy to write programs that process XML documents.

(5) The number of optional features in XML is to be kept to the absolute minimum, ideally zero.

(6) XML documents should be human-legible and reasonably clear.

(7) The XML design should be prepared quickly.

(8) The design of XML shall be formal and concise.

(9) XML documents shall be easy to create.

(10) Terseness in XML markup is of minimal importance.

We have taken a primary goal of CML as follows: *"CML shall be compatible with XML and shall reuse its ideas and technology"*. Other goals derived from the XML philosophy are listed below.

CML shall support a wide variety of applications. This is an essential part of CML, and the strategy has been to define a small language that can be reconfigured for different applications. In some cases CML supports the application, while in others it enables the author to describe the information in their own scheme. In general CML only supports a very small number of very commonly used applications such as connection tables or crystallography.

It shall be easy to write programs which process CML documents. It is relatively easy to write programs that will read and render CML documents in a general fashion. It is, however, very challenging to write programs which provide chemical perception, whatever form the document is written in. No attempt is therefore made for CML to provide chemical perception, e.g. conversion between one representation of a molecule and another, but CML is designed to make it easier to access the information that is required for this perception, such as formal bond orders.

XML documents should be human-legible and reasonably clear. Many users of chemical files wish to be able to "touch and feel" their data, and it is important that raw CML files are understandable. We anticipate that some users may wish to edit CML files with text-based tools, and although there are dangers in this, it is a safer process than editing most current molecular files. In many cases, however, users will never see the content of the CML files and will certainly not edit them.

The CML design should be prepared in an open process. It will take time to arrive at a CML design that has widespread acceptance. For this an open process is important. It is probable that (like MathML) it will proceed in stages with working drafts issued for public comment.

The design of CML shall be formal and concise. CML can only cover a wide range if it is small and generic, so only a small tag set is defined. Most other applications can be built using these components. Often they require additional semantics, but there are generic mechanisms for this such as the use of glossaries. This has so far not posed a problem for the number of file types we have analyzed.

CML documents shall be easy to create. CML documents cover a variety of disciplines, and the most productive approach will be merging components from different sources. This makes creation very much easier as it is often a question of writing MYFILE2XML converters. After that generic XML editing tools will be available for operations such as merging, restructuring, textual editing, and the insertion of hyperlinks. An example of this is the OpenSource application OSMOLE for creating and editing simple molecules.[15] OSMOLE is written as a set of Java classes (components) and is designed to be extensible both in XML and in software.

Terseness in CML markup is of minimal importance. The main purpose of this in XML has been to make it much easier to write simple reliable tools for processing XML. There is less chance of error if a document means what it says rather than relying on implicit semantics in the specification. It also means that components of documents can be extracted without subtle contextual dependencies.

CML does not provide any default conventions. CML must be capable of transmitting accurately whatever the author wishes to communicate and makes no stipulations on how the receiver should interpret it. In some cases generic CML software will be adequate, e.g. displaying bonds as single, double lines, etc., but in other cases the author may require the reader to use specific software for the rendering or processing. This design criterion is critically important since it provides a clean division between the encoding of molecular information and its interpretation.

CML is a neutral format. Molecular science has many current conventions that are not always easily interoperable. CML avoids creating yet another convention and allows any current approach to provide the semantics. It provides a keyword (CONVENTION) which identifies the convention; the element can be linked to a glossary if it exists. Any current file type can be converted to CML without loss of information. We note in particular that the reverse is not true, since CML can support more complex constructs than most current file types.

CML must be capable of evolution. In designing conventional DTDs, it is usual to create a restricted content model for an element. Thus a BOOK might contain CHAPTERs but not MOLECULES. However it is unrealistic to decide now what the content of CML elements can be. Thus a MOLECULE might contain (i.e. consist of) other MOLECULES or HTML components as well as atoms and bonds. There are other changes, mainly simplifications, from earlier CML drafts.

CML must be able to separate content from presentation. Some chemical formats consist mainly of typesetting information. It is important that the chemical nature of molecules and atoms be abstractable from the document without having to analyze graphical primitives.

ONTOLOGY IN CML

The current version of CML has been created to support a wide number of concepts in chemistry. In this article we address the simpler aspects of the ontology and leave more complex issues to a later publication. In some cases this is so we can gather feedback about the use of the current draft, in others it is to await forthcoming constructs of XML such as XLINK and XML schemas, and for others it is because there is little shared ontology within the chemical community. The following are some of the topics we reserve for a future article.

(1) Content models for ⟨reaction⟩. The simplest approach to representing reactions is to label molecules as products or reactants and either to include them explicitly within the ⟨reaction⟩ element or provide links. An example is shown in the DTD (Chart 1). Atom mapping can be elegantly represented by XLINK, and we have used an early version of this but shall wait until the next recommendation.

(2) Electronic properties. We provide a placeholder for ⟨electron⟩ but no ontology at present. This can be regarded as a tool for experimentation (a proven approach in XML drafts). ⟨electron⟩ has a count attribute so lone pairs, spin states, etc., can be represented and can be linked to ⟨atom⟩s or ⟨bond⟩s. Since all elements (but not attributes) are first class objects, ⟨atom⟩, ⟨bond⟩, and ⟨electron⟩ can be the target of links, so that it is attractive to use them for "curly-arrow" representations.

(3) Searching. Substructures can be represented by appropriate use of attributes for ⟨atom⟩ and ⟨bond⟩ and then used for searching XML documents or repositories of documents. When combined with XQL, this can make a very effective alternative to current chemical databases and will be explored in a subsequent article.

(4) Generic structures. There are many occasions where a chemical structure represents a range of possible molecules, such as Markush and other formalisms. CML can be extended to support these, but again the most elegant approaches will require XLINK.

(5) Complex structures. CML is primarily atom-based and does not intrinsically contain different levels of granularity except for ⟨sequence⟩. We are developing a general grammar to represent hierarchies in CML, e.g. for carbohydrates, polymers, macromolecules, etc., and we invite contributions from the community.

(6) Stereochemistry. We have developed a grammar for the use of the atomParity and stereo values but defer discussion of these to a later article.

CML TAG SET

Following these criteria we propose a set of components for CML (Table 3). A DTD is included (Chart 1) which lists all the tags and their attributes, but mainly has content models of "ANY", i.e. that element can contain any number of any other elements (or #PCDATA) in any order. Essentially the DTD declares the tag set and attributes but leaves open how they are structured. CML is intended to be combined with elements from other relevant XML DTDs.

Discussion of the CML Elements. We note first that CML elements will often be prefixed with a namespace, e.g. ⟨cml:molecule⟩ but these prefixes are omitted in the entries shown in Table 3. Elements either contain other elements (an element content) or strings (called #PCDATA in XML). Note also that many of the atom- and bond-based properties are added through primitive data types (e.g. ⟨float⟩) with the CML functionality determined by the value of the built-in attribute (Table 4).

A specific problem arises for the basic data types other than strings, because these are not yet finalized in XML.

Chart 1. Formal DTD (Document Type Definition) for CML[18]

```
<!-- Appendix A - CML DTD-1999-05-15 -->
<!-- Authors:
                P.Murray-Rust
                H.Rzepa
This DTD is fully described in Journal of Chemical Information
                and Computer Science, Vol xxx, 1999, pp. xxx
-->

<!-- ==========================================================-->
<!-- PARAMETER ENTITIES                                        -->
<!-- ==========================================================-->

<!-- ======attributes found on almost all elements ========== -->

<!ENTITY %      title         'title      CDATA       #IMPLIED'>
<!ENTITY %      id            'id         CDATA       #IMPLIED'>
<!ENTITY %      convention    'convention CDATA       "CML"'>
<!ENTITY %      dictRef 'dictRef    CDATA       #IMPLIED'>

<!-- ======linking information ============================== -->

<!ENTITY %      simpleLink    'href       CDATA       #REQUIRED'>

<!-- ======quantifiers and constraints on some primitives == -->

<!ENTITY %      count         'count      CDATA       "1"'>
<!ENTITY %      size          'size       CDATA       #IMPLIED'>
<!ENTITY %      rows          'rows       CDATA       #REQUIRED'>
<!ENTITY %      columns 'columns    CDATA       #REQUIRED'>

<!-- ======constraints on some numeric data primitives    ===== -->

<!ENTITY %      min           'min        CDATA       #IMPLIED'>
<!ENTITY %      max           'max        CDATA       #IMPLIED'>
<!ENTITY %      units         'units      CDATA       #IMPLIED'>
<!ENTITY %      angleUnits    'units      (degrees | radians) "degrees"'>
<!ENTITY %      unitsRef      'unitsRef   CDATA       #IMPLIED'>

<!--  values which may be useful in min and max attributes  -->
<!ENTITY %      integer.zero        '0'>
<!ENTITY %      integer.max         '2147483647'>
<!ENTITY %      integer.min         '-2147483648'>
<!ENTITY %      float.zero          '0.0'>
<!ENTITY %      float.max           '8.98846567431158E307'>
<!ENTITY %      float.min           '4.9E-324'>

<!-- ======= builtin values for any element ================== -->
<!ENTITY %      builtinId           'id'>

<!-- ======= builtin values for atoms ======================== -->
<!ENTITY %      elementType   'elementType'>
<!ENTITY %      atomId  'atomId'>

<!ENTITY %      x2            'x2'>
<!ENTITY %      y2            'y2'>
<!ENTITY %      x3            'x3'>
<!ENTITY %      y3            'y3'>
<!ENTITY %      z3            'z3'>
<!ENTITY %      xy2           'xy2'>
<!ENTITY %      xyz3          'xyz3'>
<!ENTITY %      xFract  'xFract'>
<!ENTITY %      yFract  'yFract'>
<!ENTITY %      zFract  'zFract'>
<!ENTITY %      xyzFract      'xyzFract'>
<!ENTITY %      occupancy     'occupancy'>
<!ENTITY %      isotope 'isotope'>
<!ENTITY %      formalCharge  'formalCharge'>
<!ENTITY %      nonHydrogenCount    'nonHydrogenCount'>
<!ENTITY %      hydrogenCount 'hydrogenCount'>
<!ENTITY %      atomParity    'atomParity'>

<!ENTITY %      residueType   'residueType'>
<!ENTITY %      residueId     'residueId'>

<!ENTITY %      atomStringBuiltin   '
                        %elementType; | %atomId; |
                        %residueType; | %residueId;
                        '
>
<!ENTITY %      atomFloatBuiltin    '
                        %x2; | %y2; |
                        %x3; | %y3; | %z3; |
                        %xFract; | %yFract; | %zFract; |
                        %occupancy; | %isotope; |
                        %formalCharge; | %hydrogenCount; |
%nonHydrogenCount; |
                        %atomParity;
                        '
>
<!ENTITY %      atomIntegerBuiltin  '
                        %isotope; |
                        %formalCharge; | %hydrogenCount; |
%nonHydrogenCount; |
                        %atomParity;
                        '
>
<!ENTITY %      atomCoordinate2Builtin    '
                        %xy2;
                        '
>
<!ENTITY %      atomCoordinate3Builtin    '
                        %xyz3; | %xyzFract;
                        '
>
<!-- ======= builtin values for bonds ======================== -->
<!ENTITY %      atomRef       'atomRef'>
<!ENTITY %      builtinAtomRefs     'atomRefs'>
<!ENTITY %      length        'length'>
<!ENTITY %      order         'order'>
<!ENTITY %      stereo        'stereo'>
<!ENTITY %      atomRefs            'atomRefs   CDATA #IMPLIED'>

<!ENTITY %      bondStringBuiltin   '
                        %atomRef; | %builtinAtomRefs; |
                        %order; |
                        %stereo;
                        '
>
<!ENTITY %      bondFloatBuiltin    '
                        %length;
                        '
>
<!ENTITY %      bondIntegerBuiltin  ''

>
<!-- ======= builtin values for crystal ====================== -->
<!ENTITY %      acell               'acell'>
<!ENTITY %      bcell               'bcell'>
<!ENTITY %      ccell               'ccell'>
<!ENTITY %      alpha               'alpha'>
<!ENTITY %      beta                'beta'>
<!ENTITY %      gamma               'gamma'>
<!ENTITY %      z             'z'>
<!ENTITY %      spacegroup          'spacegroup'>
<!ENTITY %      crystalStringBuiltin      '
                        %spacegroup;
                        '
>
<!ENTITY %      crystalFloatBuiltin '
                        %acell; | %bcell; | %ccell; |
                        %alpha; | %beta; | %gamma; |
                        %z;
                        '
>
<!ENTITY %      crystalIntegerBuiltin     '
                        %z;
                        '
>
<!-- ======================================================== -->
<!ENTITY %      stringBuiltin '
                        builtin (
                        %builtinId; |
                        %atomStringBuiltin; |
                        %bondStringBuiltin; |
                        %crystalStringBuiltin;
                        )      #IMPLIED    '
>
<!ENTITY %      floatBuiltin  '
                        builtin (
                        %atomFloatBuiltin; |
                        %bondFloatBuiltin; |
                        %crystalFloatBuiltin;
                        )      #IMPLIED    '
>
<!ENTITY %      integerBuiltin      '
                        builtin (
                        %atomIntegerBuiltin; |
                        %crystalIntegerBuiltin;
                        )      #IMPLIED    '
>
<!ENTITY %      coordinate2Builtin  '
                        builtin (
                        %atomCoordinate2Builtin;
                        )      #IMPLIED    '
>
<!ENTITY %      coordinate3Builtin  '
                        builtin (
                        %atomCoordinate3Builtin;
                        )      #IMPLIED    '
>

<!-- ========================================================== -->
<!-- ELEMENTS for widely used data primitives                   -->
<!-- ========================================================== -->

<!ELEMENT       string        (#PCDATA)>
<!ATTLIST       string
                        %title;
                        %id;
                        %stringBuiltin;
                        %dictRef;
                        %convention;
>
<!ELEMENT       link                (#PCDATA)>
<!ATTLIST       link
                        %title;
                        %id;
                        %simpleLink;
                        %convention;
>
<!ELEMENT       float               (#PCDATA)>
<!ATTLIST       float
                        %title;
                        %id;
                        %floatBuiltin;
                        %min;
                        %max;
                        %units;
                        %unitsRef;
                        %dictRef;
                        %convention;
>
<!ELEMENT       integer       (#PCDATA)>
<!ATTLIST       integer
                        %title;
                        %id;
                        %integerBuiltin;
                        %min;
                        %max;
                        %units;
                        %unitsRef;
                        %dictRef;
                        %convention;
>
<!ELEMENT       stringArray         (#PCDATA)>
<!ATTLIST       stringArray
                        %title;
                        %id;
                        %stringBuiltin;
                        %size;
                        %min;
                        %max;
                        delimiter   CDATA       #IMPLIED
                        %dictRef;
                        %convention;
>
<!ELEMENT       floatArray          (#PCDATA)>
<!ATTLIST       floatArray
                        %title;
                        %id;
                        %floatBuiltin;
                        %size;
                        %min;
                        %max;
                        %units;
                        %unitsRef;
                        %dictRef;
                        %convention;
>
<!ELEMENT       integerArray  (#PCDATA)>
<!ATTLIST       integerArray
                        %title;
                        %id;
                        %integerBuiltin;
```

Chart 1 (continued)

```
                        %size;
                        %min;
                        %max;
                        %units;
                        %unitsRef;
                        %dictRef;
                        %convention;
>
<!ELEMENT       floatMatrix         (#PCDATA)>
<!ATTLIST       floatMatrix
                        %title;
                        %id;
                        %rows;
                        %columns;
                        %min;
                        %max;
                        %units;
                        %unitsRef;
                        %dictRef;
                        %convention;
>

<!ELEMENT       coordinate2         (#PCDATA)>
<!ATTLIST       coordinate2
                        %title;
                        %id;
                        %coordinate2Builtin;
                        %unitsRef;
                        %dictRef;
                        %convention;
>

<!ELEMENT       coordinate3         (#PCDATA)>
<!ATTLIST       coordinate3
                        %title;
                        %id;
                        %coordinate3Builtin;
                        %unitsRef;
                        %dictRef;
                        %convention;
>
<!ELEMENT       angle               (#PCDATA)>
<!ATTLIST       angle
                        %title;
                        %id;
                        %atomRefs;
                        %angleUnits;
                        %min;
                        %max;
                        %dictRef;
                        %convention;
>

<!ELEMENT       torsion       (#PCDATA)>
<!ATTLIST       torsion
                        %title;
                        %id;
                        %atomRefs;
                        %angleUnits;
                        %min;
                        %max;
                        %dictRef;
                        %convention;
>

<!ELEMENT       list                ANY>
<!ATTLIST       list
                        %title;
                        %id;
>

<!-- ======================================================-->
<!-- ELEMENTS for chemical and crystallographic concepts    -->
<!-- ======================================================-->
<!-- NOTE
                for elements which have element-specific values for the
                builtin attribute, those values are already listed as
                entities
-->
<!-- ======================================================-->

<!ELEMENT       molecule            ANY>
<!ATTLIST       molecule
                        %title;
                        %id;
                        %count;
                        %dictRef;
                        %dictRef;
                        %convention;
>

<!-- ======================================================-->
<!ELEMENT       formula       ANY>
<!ATTLIST       formula
                        %title;
                        %id;
                        %count;
                        %dictRef;
                        %convention;

>

<!-- ======================================================-->
<!ELEMENT       atom                ANY>
<!ATTLIST       atom
                        %title;
                        %id;
                        %count;
                        %dictRef;
                        %convention;
>

<!-- ......................................................-->
<!ELEMENT       atomArray           ANY>
<!ATTLIST       atomArray
                        %title;
                        %id;
                        %dictRef;
                        %convention;
>


<!-- ======================================================-->
<!ELEMENT       bond                ANY>
<!ATTLIST       bond
                        %id;
                        %atomRefs;
                        %dictRef;
                        %convention;
>

<!-- ======================================================-->
<!ELEMENT       bondArray           ANY>
<!ATTLIST       bondArray
                        %id;
                        %dictRef;
                        %convention;
>
<!-- ======================================================-->
<!ELEMENT       electron            ANY>
<!ATTLIST       electron
                        %id;
                        %count;
                        %dictRef;
                        %convention;
>
<!-- ======================================================-->
<!ELEMENT       reaction            ANY>
<!ATTLIST       reaction
                        %id;
                        %dictRef;
                        %convention;
>
<!-- ======================================================-->
<!ELEMENT       crystal       ANY>
<!ATTLIST       crystal
                        %title;
                        %id;
                        %dictRef;
                        %convention;
>
<!-- ======================================================-->
<!ELEMENT       sequence            ANY>
<!ATTLIST       sequence
                        %title;
                  %id;
                        %dictRef;
                        %convention;
>

<!-- ======================================================-->
<!ELEMENT       feature       ANY>
<!ATTLIST       feature
                        %title;
                        %id;
                        %dictRef;
                        %convention;
>
```

We have therefore included the following "obvious" element types which are needed for XML and will be provided by the emerging XML schema activity:

```
<float> <integer> <string> <date>
<boolean> <list> <enumeration> <array>
```

Because of the design of XML and the power of XSL, it will be possible to algorithmically convert these elements to whatever element types are defined in XML schema. CML can be kept small because it describes just the generally agreed chemical components of a system. The other basic criterion for inclusion in the CML tag set is that the element or attribute gives information about what the chemical component(s) is (are). The most obvious are atom identity, position in space, and atom properties that affect the identity; examples would be isotope and occupancy but specifically not chemical shift, formal charge, hydrogen count, etc. Measured properties such as e.g. vibrational frequencies are expected to be held in ⟨float⟩ or similar elements and linked to glossaries. Accordingly, we propose the following fundamental components:

```
<atom>  <bond> <electron> <molecule>
<crystallography> <reaction> <sequence> <feature>
```

Because so many chemicals are categorized in the solid state, support for crystallography is included, as are placeholders for reactions and also for macromolecular sequences and feature. We have also created a small number of additional primitive data types such as ⟨angle⟩, ⟨torsion⟩, ⟨coordinate2⟩, and ⟨coordinate3⟩. In some cases where there are large numbers of primitives such as ⟨float⟩, it is useful to compress them into a compacted arrangement, so ⟨floatArray⟩, etc.,

Table 3. CML Element Set

element name	description	type of content (cf. Table 1)
molecule	generic container for any assemblage of atoms, bonds, or other molecules	element content, most normally, atomArray, bondArray, and other molecules
formula	generic container for molecular constitution	content and semantics are undefined
atom and atomArray	describes an atom or set of atoms can be extensibly qualified with properties	element content
bond and bondArray	describes a bond or set of bonds can be extensibly qualified with properties	element content
crystal	container for crystallographic information such as space group, cell dimensions, and symmetry that is required to build an extended structure from fractional coordinates (it includes cell parameters, space group, and entities/cell; symmetry operators can be included as ⟨floatMatrix⟩ if required, and all experimental crystallographic information such as wavelength of radiation would use ⟨float⟩, etc., linked to appropriate dictionaries)	element content
reaction	container for the components of a reaction (semantics are not defined in CML-1999-05-15, although a suggestion is given below)	element content
electron	container for information on electrons (the semantics are not defined CML-1999-05-15, but allow ⟨link⟩s to point to atoms and bonds associated with the electrons)	element content
sequence and feature	these support protein sequence and its annotation (sequence can be qualified by convention to describe the convention (e.g. one-letter codes) and holds the sequence as a single #PCDATA string; feature is a placeholder to annotate sequence- and structure-specific sites (e.g. mutations and active sites); it is likely to be implemented by contained XLINKs)	element content
float, integer, string, floatArray, integerArray, stringArray, floatMatrix date, boolean, list, enumeration	primitive data types, mainly for components of another CML element	usually #PCDATA content
coordinate2, coordinate3, angle, torsion	specialized primitive data types for supporting Cartesian, fractional, and internal coordinates for molecular structure	#PCDATA content
link	simple hyperlink to act as a pointer between molecular components (e.g. linking electrons to the associated atoms or bonds)	#PCDATA content at present, though probably extended to XLINK syntax for extended links when appropriate

are provided. In general these hold the string content of white-space-separated primitives; for ⟨stringArray⟩ we allow for a user-defined delimiter character.

There are a small number of properties or built-in attributes associated with these elements that are hard coded into the language. However most other properties and the relationships and annotations described above are provided by generic XML mechanisms. A rule of thumb is that software to process CML documents should provide support for the built-in properties. This will be essential if the document is to be converted algorithmically to another chemical file type such as e.g. one of the chemical/* MIME types).

Small molecules are representable in CML with the ⟨atom⟩ and ⟨bond⟩ elements, and these should be used where possible as they give both precision and flexibility. However for large molecules (e.g. proteins) this would result in a very large number of elements, perhaps over 100 000. Building such large trees is often expensive in time and memory, and because of the essentially rectangular nature of the ⟨atom⟩ and ⟨bond⟩ information, we have created a syntactic equivalent through ⟨atomArray⟩ and ⟨bondArray⟩.[16] These two elements are used as follows:

All ⟨atom⟩ children of a ⟨molecule⟩ are grouped and replaced by a single ⟨atomArray⟩ element. Its children correspond to the ⟨string⟩, ⟨float⟩, etc., children of each atom and are represented by ⟨stringArray⟩, ⟨floatArray⟩, etc. Thus HOCl can be represented using either ⟨atom⟩ and ⟨bond⟩ elements or the corresponding arrays (Chart 2) can also be expressed as in Chart 3. Note that this should only be used for large files with many elements. XML documents have been shown to compress very efficiently, and there is unlikely to be any gain in using the array format purely for this process. Note also that generic XML tools will not be able to carry out some types of operation on the array format such as e.g. fine-grained atomistic searches.

CML Element Attributes. In XML, functionality is often added through attributes and these are used widely in CML. There are many generic attributes that can be found on most elements, and the main ones are as follows:

id. XML uses unique identifiers as the target of links, and it is useful good practice to label all information components with an id. This makes it easy to abstract and reuse components, such as molecules. XML has a convention and syntax (Xpointer) for locating information components in documents and id is strongly recommended. Note that the value of an XML id is restricted to alphanumeric characters and a very few punctuation characters. Where published ids do not fit this convention (e.g. "CA 3", which contains white space), other attributes or elements are provided, e.g. atomId for ⟨atom⟩. These should not be used for linking.

convention. This allows CML authors to identify the convention used for a given element. There is no defined list of values, but it will certainly be useful to include the major file types in the chemical/* MIME list.[7] A convention attribute on an element is assumed to be inherited by all the descendants of the element. Thus the ⟨atom⟩ elements in ⟨molecule convention="PDB"⟩⟨atom⟩...⟨/atom⟩...⟨/molecule⟩ would be assumed to use the PDB convention unless

Table 4. Builtin Attributes in CML

CML element attributes	description	data type
	Children of atom and atomArray	
x2, y2 (or xy2)	two-dimensional Cartesian coordinates for chemical structure diagrams; arbitrary units, until the screen and other coordinate system standards are defined for Web use	float/integer
x3, y3, z3 (or xyz3)	three-dimensional Cartesian coordinates in angstroms (10^{-10} m); if other units are required, a units attribute can be added	float
xFract, yFract, zFract, or xyzFract	fractional coordinates relative to a cell length from crystallography	float
elementType	any string describing the element, but the element symbol from IUPAC is recommended (in cases where a range of elements is possible, a separate element may need to be defined; nonstandard element symbols can be used for pseudoelements, e.g. Me for methyl or dummies, R1 in Markush formulas)	string
atomId	atom identifier (this is NOT the same as the id attribute (which should obey XML id rules and must be unique within an XML document); atomId can be any string, which the authors use to identify an atom; significant white space as in "CA 1" (PDB syntax) and punctuation is allowed but should be avoided if possible)	id
isotope	mass of the isotope	float/integer
occupancy	fraction of atom present (primarily from crystallography)	float
hydrogenCount	total hydrogen count on atom whether explicit or not	float/integer
atomParity	atom-based stereochemistry derived from chiral volume of ligands (convention-dependent)	float/integer
formalCharge	allows an electron count of the atom and molecule to be made (unrelated to any experimental or theoretical charge (which could be added by an additional atomArray))	float/integer
	Children of bond and bondArray	
atomRef and atomRefs	string (Ids for atoms involved in the bond; which end of the bond is given first depends on the convention and may not matter)	id
order	formal bond order, e.g. as in a chemical structure diagram (there is currently a confusion of conventions and CML does not intend to arbitrate; thus some systems have a provision for an aromatic bond with arbitrary numeric descriptor while others use alternating single and double bonds; it is recommended that convention attributes be used frequently to avoid misunderstandings)	string
stereo	some systems use wedge and hatch bonds to denote relative stereochemistry (for many conventions this is only usable with a two-dimensional structural diagram and is open to misinterpretation; e.g. different coordinate systems could invert the stereochemistry; moreover it is not as powerful as the atom- and dummy-atom-centered parity)	string
	Children of crystal	
acell, bcell, ccell	cell lengths, default units are angstroms (float alpha, beta, gamma); cell angles, default units are degrees (float z); number of entities per cell	integer/float
alpha, beta, gamma	cell angles; default units are degrees	float
z	number of entities per cell	integer/float
spacegroup	space group convention should be given	string

Chart 2

```
<molecule>
          <atom id="a1">
             <string builtin="elementType">H</string>
          </atom>
          <atom id="a2">
             <string builtin="elementType">O</string>
          </atom>
          <atom id="a3">
             <string builtin="elementType">Cl</string>
          </atom>
          <bond id="b1" atomRefs="a1 a2">
             <string builtin="order">1</string>
          </bond>
          <bond id="b2" atomRefs="a3 a2">
             <string builtin="order">1</string>
          </bond>
</molecule>
```

Chart 3

```
<molecule>
        <atomArray>
           <stringArray builtin="id">a1 a2 a3</stringArray>
           <stringArray builtin="elementType">H O Cl</stringArray>
        </atomArray>
        <bondArray>
           <stringArray builtin="id">b1 b2</stringArray>
           <stringArray builtin="atomRef">a1 a2</stringArray>
           <stringArray builtin="atomRef">a2 a3</stringArray>
           <stringArray builtin="order">1 1</stringArray>
        </bondArray>
</molecule>
```

overridden by another convention attribute. Implementers of CML-aware software may be able to take advantage of the conventions.

builtin. This adds essential properties to atoms and bonds (Table 4). The key intention is to describe the identity of molecular and other chemical species, while further extension is done through the convention attribute.

count. Atoms, molecules and electrons may be qualified by a multiplier, which may be useful for describing stoichiometry. The default is 1.0, count attributes in descendants are multiplied by count attributes on ancestors.

title. The value of this attribute is primarily for display, and many browsing tools will show the title by default. It may also be used for adding semantics and can usefully serve as a search field. It is of limited use for adding ontology unless linked to a dictionary (e.g. through dictRef) or convention. Thus ⟨item title="foo" convention="bar"⟩ suggests that a definition of foo can be found in the convention bar.

size, rows, columns. These are constraints on the contents of array and matrix primitives.

min, max. These minimum and maximum allowed values of *y*.

href, atomRef, atomRefs, dictRef, unitsRef. Many elements have links to others, such as glossary elements. CML also uses links to describe basic structures such as bonds and reactions. Thus a bond consists of two or more links to atoms, which must have declared unique *id*s in the document. Linking allows complex structures, so that e.g. a three-center bond can be described by links to three atoms (in an *atomRefs* attribute). *dictRef* is provided for linking to

glossaries or dictionaries. *unitsRef* gives a link to scientific units for a quantity. Note that since *id*s should not contain white space, the *atomRefs* value will consist of a series of white-space-separated ids, where the white space is normalized to a single space.

Built-in Attributes and Their Values. The following case-sensitive built-in attributes are currently hard coded into CML.[17] They are normally attached to an element which takes PCDATA content and in some cases determine which primitive dataTypes (e.g. ⟨float⟩) are appropriate. The default content is string. Example:

```
<atom>
          <float builtin="occupancy">0.5</float>
</atom>
```

For explicit examples of the use of the built-in attribute, see examples 1−4 below. The possible values of this attribute are listed by the element name of the parent (Table 4).

ROLE OF SOFTWARE

CML has been designed so that it is as easy as possible to write processing software. It is not expected that all CML-aware applications should implement all the functionality described in the DTD. When a CML application cannot process an element, an attempt will be made to route it unchanged to any output. XML provides a rich set of generic tools for this sort of process.

In general the CML-specific functionality is indicated by the built-in elements; CML-aware software should address some or all of these. Thus an application receiving an input of ⟨floatArray builtin="xFract"⟩ should be aware that it should search the neighboring elements for a ⟨crystal⟩ element if it wishes to orthogonalize these values. This search facility can be provided by embedding XSL or XQL functionality in the program or using these tools as a preprocessor.

Many conventional housekeeping operations in normal software can be provided by generic XML functions. Deleting an atom from a conventional connection table, and associated atom and bond properties, is often tedious and error-prone. In CML it may be sufficient to locate all elements which contain or reference a given atom id and to amend or delete them.

We expect that as generic software becomes XML-aware, and there are signs that this is happening rapidly, there will be a move to providing CML-aware interfaces to programs, instruments, databases, etc. Until that is universal it will be necessary to have two-way legacy conversion programs. As noted before, conversion to CML is usually without loss, but conversion in the other direction may lose information.

We expect to build a library of CML-aware tools and to provide examples of how generic XML software can be used in CML applications.

EXAMPLES OF CML[18]

Example 1. This describes an ensemble of one oxygen atom, two deuterium atoms, and nine electrons (which might be a singly positively ionized heavy water molecule) (Chart 4). Although it is more verbose than normal chemical notation, it is searchable by generic tools and available for algorithmic analysis without worrying about syntax. An alternative syntax below for large molecules is given in example 3.

Chart 4

```
<!DOCTYPE molecule SYSTEM "CML-1999-05-15.dtd" []>
<molecule id="deuteriumOxidelon">
  <atom count="2">
    <string builtin="elementType">H</string>
    <float builtin="isotope" >2</float>
  </atom>
  <atom count="1">
    <string builtin="elementType">O</string>
  </atom>
  <electron count="9"/>
</molecule>
```

Chart 5

```
<!DOCTYPE molecule SYSTEM "CML-1999-05-15.dtd" []>
<molecule title="salt" id="sodiumChloride">
  <crystal>
    <string builtin="spacegroup" convention="HM">Fm3m</string>
    <float builtin="acell">5.8</float>
  </crystal>
  <atom>
    <string builtin="elementType">Na</string>
    <float builtin="xFract">0.0</float>
    <float builtin="yFract">0.0</float>
    <float builtin="zFract">0.0</float>
    <integer builtin="formalCharge">+1</integer>
  </atom>
  <atom>
    <string builtin="elementType">Cl</string>
    <coordinate3 builtin="xyzFract">0.5 0.0 0.0</coordinate3>
  </atom>
</molecule>
```

Example 2. A crystallographic example. From this information and the space group it is possible to recreate the whole unit cell and contents. Note that *xfract*, etc., are fractional coordinates; *acell* is the cell dimension in angstroms (by default, others are possible) (Chart 5).

Example 3. Although ⟨atom⟩ and ⟨bond⟩ may be used for small molecules, it is often more compact and more robust to link all atoms together by property (Chart 6). The atomArray and bondArray elements are used for this. For large molecules this provides a concise, fully marked up syntax. All ⟨atomArray⟩s must be the same length and so, separately, must ⟨bondArray⟩s. The formatting of the ⟨atomArray⟩ is not mandatory, and any whitespace simply acts as a delimiter. Each atom must have an id. Note that hydrogen atoms can be given explicitly or implicitly through hydrogenCount, which allows for flexibility in representation.

Example 4. Advanced use of the ⟨atom⟩ element is shown in Chart 7.

Example 5. Advanced use of the ⟨bond⟩ element is given in Chart 8.

Example 6. An example of a possible reaction definition is provided in Chart 9.

VALIDITY AND PROCESSING SOFTWARE

CML documents often have implicit internal relationships that cannot be expressed in XML. For example bonds represent links to two atoms, so the atomRef values must link to valid atoms. The arrays in the example must correspond to a valid connection table that makes chemical sense. Because of this, it is likely that CML documents will require specific software to check validity or calculate other properties. An important role will be conversion between conventions.

EXTENSIONS

CML is extensible in several ways such as by adding customized content to elements. Thus the connection table

Chart 6

```
<!DOCTYPE molecule SYSTEM "CML-1999-05-15.dtd" []>
<molecule id="dimethylformamide">
  <atomArray>
    <stringArray builtin="id">            H1 C1 O1 N1 Me1 Me2</stringArray>
    <stringArray builtin="elementType ">  H  C  O  N  C   C</stringArray>
    <integerArray builtin="hydrogenCount">0  1  0  1  3   3</integerArray>
  </atomArray>
  <bondArray>
    <stringArray builtin="atomRefs">C1 C1 C1 N1 N1</stringArray>
    <stringArray builtin="atomRefs">H1 O1 N1 Me1 Me2</stringArray>
    <stringArray builtin="order">1 2 1 1 1</stringArray>
  </bondArray>
<!--this is a comment. The molecule is:
      Me1    H1
        \    |
         N1-C1=O1
        /
      Me2
-->
</molecule>
```

Chart 7

```
<!-- ==START==========Example 4==================-->
<!--  use of <atom> -->
<!DOCTYPE atom SYSTEM "CML-1999-05-15.dtd" []>
<atom           id="a.001">
  <string       builtin="elementType">C</string>
  <coordinate2  title="Screen coordinates (arbitrary)"
                builtin="xy2">100.0 200.0</coordinate2>
  <coordinate3  title="Cartesian 3D coordinates (pm)"
                unitsRef="units.pm" builtin="xyz3">1.234
2.345 4.456</coordinate3>
  <coordinate3  title="xyzFractional 3D coordinates"
                builtin="xyzFract">0.234 -0.123
0.347</coordinate3>
  <float        builtin="occupancy">0.57</float>
  <integer      builtin="isotope">13</integer>
  <string       builtin="residueType">Gly</string>
  <string       title="residueId"
                builtin="residueId">G32A</string>
  <float        title="B-factor"
                dictRef="_atom_b_iso">10.8</float>
  <integer      builtin="formalCharge">0</integer>
  <integer      builtin="hydrogenCount">3</integer>
</atom>
<!-- ==END============Example 4==================-->
```

Chart 8

```
<!-- ==START==========Example 5==================-->
<!--  use of <bond> including internal coordinates -->
<!DOCTYPE molecule SYSTEM "CML-1999-05-15.dtd" []>
<molecule>
<!-- atoms omitted - assumed to have IDs a.001, a.002, a.003 a.004 -->
  <bond id="b1" atomRefs="a.001 a.002">
    <float title="equilibrium bond length" units="pm">123.4</float>
    <string  title="formal bond order" builtin="order">2</string>
  </bond>
  <bond id="b2" atomRefs="a.003 a.002">
    <float title="equilibrium bond length" units="pm">144.4</float>
    <string  title="formal bond order" builtin="order">1</string>
  </bond>
  <angle id="ang12" atomRefs="a.003 a.002 a.001"
    title="equilibrium bond angle" units="degrees">144.4</angle>
  <torsion id="tor1" atomRefs="a.003 a.002 a.001 a.004"
    title="gauche conformation" units="degrees">-55.5</torsion>
</molecule>
<!-- ==END============Example 5==================-->
```

already used as an example can be extended so the molecule contains additional information (Chart 10).

The molecule contains additional information to simulate an internal catalogue entry. It could easily include, or be linked to, spectra, commercial details, local uses, etc. Extending properties of atoms or bonds is also possible. Thus if the molecule had been analyzed crystallographically, each atom might have an isotropic "temperature factor" *B*. These could be included by an inserted atomArray:

```
<floatArray title="B-factor" convention="PDB"
type="float" delimiter=";">23.0;25.0;33.0;45.0;45.0</floatArray>
```

The values are specifically delimited, and the example shows that the first value for the H atom is missing/null. For more complex objects, e.g. anisotropic *B*-factors, which require a matrix, a ⟨list⟩ may be used. This example shows a null

Chart 9

```
<!-- ==START==========Example 6==================-->
<!--  use of <reaction> -->
<!DOCTYPE reaction SYSTEM "CML-1999-05-15.dtd" []>
<!-- possible example of a reaction:
define a set of molecules first (mol1 ... mol4) -->
<reaction>
  <list title="reactants">
    <link href="mol1"/>
    <link href="mol2"/>
  </list>
  <list title="products">
    <link href="mol3"/>
    <link href="mol4"/>
  </list>
  <list title="conditions">
    <float title="temperature" units="K">180</float>
    <link title="solvent" href="solv03"/>
  </list>
</reaction>
<!-- ==END============Example 6==================-->
```

Chart 10

```
<!DOCTYPE molecule SYSTEM "CML-1999-05-15.dtd" [
<!ATTLIST molecule xmlns CDATA #IMPLIED>
<!ELEMENT h:html ANY>
<!ATTLIST h:html xmlns:h CDATA #IMPLIED>
<!ELEMENT emph ANY>
<!ELEMENT p ANY>
<!ELEMENT pre ANY>
]>
<molecule xmlns="http://www.xml-cml.org" id="formamide">
<!--start of connection table-->
  <atomArray>
    <stringArray builtin="id">            H1 C1 O1 N1 Me1 Me2</stringArray>
    <stringArray builtin="elementType ">  H  C  O  N  C   C</stringArray>
    <integerArray builtin="hydrogenCount">0  1  0  1  3   3</integerArray>
  </atomArray>
  <bondArray>
    <stringArray builtin="atomRefs">C1 C1 C1 N1 N1</stringArray>
    <stringArray builtin="atomRefs">H1 O1 N1 Me1 Me2</stringArray>
    <stringArray builtin="order">1 2 1 1 1</stringArray>
  </bondArray>
<!--end of connection table-->
<!--this is a comment. The molecule is:
      Me1    H1
        \    |
         N1-C1=O1
        /
      Me2
-->
<!--additional information to show how the molecule XML-element can be
extended-->
  <h:html xmlns:h="http://www.w3.org/TR/html20">
<p>Formamide is the simplest amide ...</p>
<!--(This is a comment) Note how HTML text can be included in a CML
molecule XML-element. The CML namespace applies to the atomArray and
bondArray XML-elements, but is overridden by the HTML namespace
declaration.-->
<p>This represents a <emph>connection table</emph> for formamide. The
structure corresponds to the diagram:</p>
<pre>
      H3       H1
        \     /
         N1-C1=O1
        /
      H2
</pre>
  </h:html>
<!-- This is a comment. We are back in CML namespace -->
  <float title="molecularWeight" units="g">45.03</float>
  <list title="local information">
    <link title="safety" href="/safety/chemicals.xml#formamide"/>
    <string title="location">Storeroom 12.3</string>
  </list>
</molecule>
```

matrix for the first atom, a complete upper triangular matrix for the second. Later atoms are omitted.

```
<atomArray title="aniso-B-factor">
  <list>
    <matrix rows="3" cols="3"></matrix>
    <matrix rows="3" cols="3" type="upperTriangular">
      10.0 20.0 10.0 23.0 33.0 44.0
    </matrix>
<!-- … etc …→
  </list>
</atomArray>
```

USE, UPDATES, AND FURTHER INFORMATION

In the spirit of the WWW, the latest version of the CML draft will be published on http://www.xml-cml.org/. Versioning will be by date (in ISO 8601 format). It is intended that these drafts remain current for sufficient time for readers to use or implement prototype CML systems.

It is particularly difficult for software implementers to have to support several moving drafts. The XML effort has emphasized this, with ambition sometimes running ahead of the corporate resources of the community to implement and test software. It is therefore important that there is an initial simple version of CML that can be supported and is guaranteed free from change, even though later versions may be more powerful. For this reason we have deliberately omitted complex operations such as reactions and combinatorial chemistry.

XSL has implied support for transformation between different document types. Therefore if syntax changes in the future, there will be an algorithmic method for transforming older versions to newer ones.

The language presented here is capable of supporting all the information in the common chemical/* MIME types. CML is not a software system and does not yet specify application programming interfaces (APIs) or algorithms for validity or transformation. A set of free software will mounted at the xml-cml.org site, although readers are welcome to develop their own. This software will help to define the interpretation of fuzzy parts of the CML draft.

REFERENCES AND NOTES

(1) The CML project was first described in: Murray-Rust, P.; Leach, C.; Rzepa, H. S. *Abstr. Pap.—Am. Chem. Soc.* **1995**, *210*, 40-COMP. Murray-Rust, P.; Rzepa, H. S. *Abstr. Pap. Am. Chem. Soc.* **1997**, *214*, 23-COMP. Preliminary specifications, software components, and examples were published on CD-ROM in: *Proceedings of the Electronic Conference on Trends in Heterocyclic Chemistry (ECHET96)*; Rzepa, H. S., Snyder, J., Leach, C., Eds.; The Royal Society of Chemistry: Cambridge, U.K., 1997; ISBN 0-85404-894-4. Also: *Proceedings of the Electronic Conference on Trends in Organometallic Chemistry (ECTOC3)*; Rzepa, H. S., Leach, C., Eds.; The Royal Society of Chemistry: Cambridge, U.K., 1998; ISBN 0-85404-889-8.

(2) This is more fully described in the XML1.0 Recommendation; http://www.w3.org/xml/ and in particular http://www.w3.org/TR/xmlschema-1/ and http://www.w3.org/TR/xmlschema-2/. See also: Graham, I., Quin, L. *1.0. XML Specification Guide*; Wiley: New York, 1999; ISBN 0471327530.

(3) Murray-Rust, P.; Rzepa, H. S.; Whittaker, B. J. *Chem. Soc. Rev.* **1997**, *27*, 1−10.

(4) Weibel, S. *Bull. Am. Soc. Inf. Sci.* **1997**, *24*, 9−11. A model for medical metadata has been proposed; Malet, G.; Munoz, F.; Appleyard, R.; Hersh, W. *J. Am. Med. Inf. Assoc.* **1999**, *6*, 163−172. For metadata in chemistry, see: Gkoutos, G. V.; Murray-Rust, P.; Turner, A. N.; Rzepa, H. S. To be submitted for publication.

(5) Hall, S. R.; Allen, F. H.; Brown, I. D. *Acta Crystallogr.* **1991**, *A47*, 655−685.

(6) See e.g.: Bryan, M. *An Author's Guide to the Standard Generalized Markup Language*; Addison-Wesley: Reading, MA, 1988. Goldfarb, C. F.; Runinsky, Y. *The SGML Handbook*; Clarendon Press: Oxford, U.K., 1990. Goldfarb, C. F.; Prescod, P. *The XML Handbook*; Prentice Hall: Englewood Cliffs, NJ, 1998.

(7) Rzepa, H. S.; Murray-Rust, P.; Whitaker, B. J. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 976−982.

(8) For further details of Unicode, see: http://www.unicode.org/. *The Unicode Standard.* Version 2.0; NISP Press: 1996.

(9) See for example: http://www.iucr.ac.uk/.

(10) Pauling, L. *The nature of the chemical bond and the structure of molecules and crystals: An introduction to modern structural*, 2nd ed.; Cornell University Press: Ithaca, NY, 1942.

(11) Bernal, J. D. *Science in History*, 3rd ed.; C. A. Watts: London, 1965.

(12) Murray-Rust, P. http://www.xml-cml.org/.

(13) These are elaborated in more detail at: http://www.w3.org/TR/REC-xml.

(14) This will be explored in more depth in part 2 of this series.

(15) Murray-Rust, P. Unpublished work.

(16) Some early versions of CML used only these elements (see also ref 1).

(17) For readers familiar with earlier versions of CML, note that some built-in values are obsolete since they do not describe the essential identity of the components. The current values are limited to those for which processing software needs to be provided.

(18) These examples, together with the CML DTD shown in Chart 1, are included in Supporting Information with this Journal and are also available at: http://www.xml-cml.org/.

CI990052B