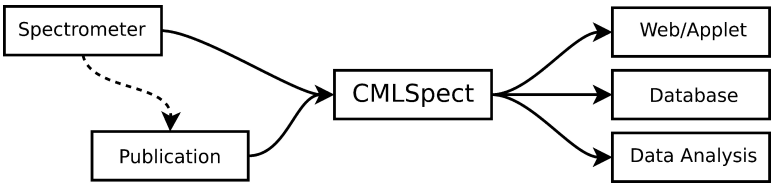
JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data

Stefan Kuhn, Tobias Helmus, Robert J. Lancashire, Peter Murray-Rust, Henry S. Rzepa, Christoph Steinbeck, and Egon L. Willighagen

J. Chem. Inf. Model., **2007**, 47 (6), 2015-2034 • DOI: 10.1021/ci600531a



More About This Article

Additional resources and features associated with this article are available within the HTML version:

- Supporting Information
- Links to the 1 articles that cite this article, as of the time of this article download
- Access to high resolution figures
- Links to articles and content related to this article
- Copyright permission to reproduce figures and/or text from this article

View the Full Text HTML

Chemical Markup, XML, and the World Wide Web. 7. CMLSpect, an XML Vocabulary for Spectral Data

Stefan Kuhn,[†,⊥] Tobias Helmus,[†] Robert J. Lancashire,[‡] Peter Murray-Rust,*[,§] Henry S. Rzepa,[||] Christoph Steinbeck,[†] and Egon L. Willighagen[†]

Cologne University Bioinformatics Center (CUBIC), Cologne, Germany, University of the West Indies, Mona Campus, Jamaica, Unilever Centre for Molecular Informatics, Cambridge, U.K., and Department of Chemistry, Imperial College London, London, U.K.

Received November 24, 2006

CMLSpect is an extension of Chemical Markup Language (CML) for managing spectral and other analytical data. It is designed to be flexible enough to contain a wide variety of spectral data. The paper describes the CMLElements used and gives practical examples for common types of spectra. In addition it demonstrates how different views of the data can be expressed and what problems still exist.

1. INTRODUCTION

The vision of the semantic Web is of a global knowledgebase where humans and machines are jointly able to understand and process information.[1] To accomplish this it is necessary to formalize the representation of machine-readable information and to support it with software systems that can process it automatically without guidance from humans. There is now a large global activity in implementing these ideas in scientific disciplines (eScience, cyberinfrastructure, scientific semantic grid, etc.)[2] and a corresponding requirement for supporting chemistry in this environment.

In previous papers[3−8] we have described the concept of the chemical semantic Web and developed an XML framework, the Chemical Markup Language (CML). This has been designed to support commonly encountered chemical concepts, and we have presented full formal specifications for molecules and chemical reactions. In those papers we also outlined a broader framework including chemical computation, solid state, and spectral information. In this paper, we add a full formal specification for spectral data, called "CMLSpect" (see section 2). The applications of this are shown in sections 3 and 4. We compare our concept to existing formats in section 5. We also introduce some changes and additions to the original concept of CML (see section 6) to adopt it better to the needs of its users, which were evaluated constantly and became much clearer over the time of usage of CML than they were at the beginning.

CML is now being adopted in a number of areas including formal publication of chemistry as in primary research articles, storage of chemical information in databases, and support for automation of chemical computation.

The term "spectroscopy" describes a very important category of chemical information. The term is imprecise, and we do not attempt to define it too narrowly as the methodology we present is flexible and extensible. In particular we do not enumerate a fixed set of spectroscopic techniques but provide a general framework. A brief, noncomprehensive and overlapping list of terms (from Wikipedia[9]) shows the wide range of techniques that are relevant to chemistry and which might be categorized by the term "spectroscopic": atomic absorption spectroscopy, reflectance spectroscopy, circular dichroism, electron paramagnetic spectroscopy, electronic spectroscopy, Fourier transform spectroscopy, gamma-ray spectroscopy, infrared spectroscopy, laser spectroscopy, mass spectrometry, multiplex or frequency-modulated spectroscopy, raman spectroscopy, X-ray spectroscopy. Spectroscopic methods are universally available, and it is effectively mandatory to report spectroscopic measurements in the publication of a novel compound. The motivation is partly to act as a fingerprint for characterizing the compound and partly to understand its chemical structure and properties. We estimate that millions of instances of such measurements are published annually in peer-reviewed journals.

The information in many spectral techniques is very large, and when different methods for a compound are combined the insight provided can be very significant. Moreover when the spectra of related compounds are compared, systematic chemical principles can be discovered. We believe that if all the spectra for published compounds were available in machine-understandable form, then many new chemical discoveries would be possible.

However, the traditional publication process has led to almost all spectral information being lost before publication. This was originally inevitable in paper-based publishing but is no longer necessary. All modern instruments produce a digital output, and the storage requirements and bandwidth are trivial compared with many modern activities such as video and mobile telephony. In this paper we urge that all such data are routinely captured, stored, and disseminated and present an information architecture that supports this.

A major challenge is that over the past two centuries powerful textual and graphical metaphors have been developed for communicating chemistry. Although these can be

* Corresponding author phone: +44 (0) 1223 763069; fax: +44 (0) 1223 763076; e-mail: pm286@cam.ac.uk.
[†] Cologne University Bioinformatics Center (CUBIC).
[‡] University of the West Indies.
[§] Unilever Centre for Molecular Informatics.
[||] Imperial College London.
[⊥] Present address: Leibniz Institute of Plant Biochemistry, Halle, Germany.

10.1021/ci600531a CCC: $37.00
Published on Web 09/22/2007

rendered and transmitted in electronic form ("machine-readable"), the results often have little or no semantic content. However, due to established practices in chemistry, this is still the predominant form of publication.

A spectrum normally involves the response of a sample as one or more variables are systematically changed. Most frequently it is the intensity of electromagnetic radiation as the energy (or equivalently the frequency) is varied. However, there are other techniques such as thermogravimetric analysis (variation of mass with temperature), differential scanning calorimetry (heat capacity against temperature), cyclic voltammetry (a kind of potentiodynamic electrochemical measurement), and inelastic neutron scattering (absorption of energy from thermal neutrons). Although these are not "spectroscopy" in the purest sense they are frequently copublished and can be adequately supported by the CMLSpect technology we report here. It is even possible to hold chromatographic data in the infrastructure, and we are confident that there will be other techniques which CMLSpect can support. The examples in this paper, however, will be drawn from techniques such as NMR, IR, UV/vis, and mass spectrometry. We also distinguish between a phenomenological description of a spectrum (as for example recorded by an instrument) and a theoretical model describing the underlying physics and from which a simulated spectrum could be constructed if wished so. The current article addresses the former and not the latter.

CMLSpect is continuously developed. All project documents, including schemas and software, are hosted as the CML project at sourceforge.net (http://sourceforge.net/projects/cml). This article refers to version 2.5 of the CML schema (module schema25 in the sourceforge CVS tree) and JUMBO version 5.3 (module jumbo53). These releases will only get bugs fixed but will not make any functional changes. So if looking for the software described in this article we recommend using these releases; however, there might be new CML releases in the future with major changes and features added. We advise users to use the latest stable release for integration into other software projects.

2. CML(SPECT) ARCHITECTURE

Although all CML components (CMLElements and attributes) can be included in a compound document (e.g., a primary publication) there is a de facto "core set" which is necessary or desirable in most spectroscopic applications. [XML uses "elements" to describe the components of a document which can cause confusion with chemical elements. We shall therefore use "CMLElement" to denote elements of CML and the more general "XMLElement" to describe any XML element. Attributes will be marked with an @ in front, which is not used in CML documents.] The examples will highlight the use of most of these, but the following brief descriptions will act as an introduction for those not familiar with CML. In a number of cases the community semantics for an element may have evolved, and this will be pointed out.

A small CML example could look like Chart 1.

Readers not already familiar with the concept of CML should keep this example in mind when going through this section. The following sections will highlight the general concepts in a CMLSpect document, among those in the previous example.

2.1. Containers and Primitives. There is no required top-level CML container, but <cml> is useful for this purpose. It is easily identified as specifying that the contents are mainly CML elements and has no built-in semantics. For lists of components it is sometimes useful to use the <list>, which again has no semantics and which can contain heterogeneous children. However, many CML elements have specific containers such as <parameterList>, <moleculeList>, <propertyList>, <spectrumList>, or <conditionList>. Each can only contain specific children: thus <propertyList> can only contain <property> elements or further nested <propertyList>s. <conditionList> does not (yet) have specific <condition> children and can hold any CML elements that make sense.

The primary scientific data are usually held in one of <scalar>, <array>, or <matrix>. [We use "array element" to indicate the individual components a[i], m[i][j], and "<array> element" or "<matrix> element" (or simply "<array>" or "<matrix>") to indicate the CMLElements.] These are respectively zero-, one-, and two-dimensional homogeneous arrays of data representable as plain text strings; they cannot hold complex objects. For array and matrix the data elements are separated by a delimiter. This is normally "whitespace" with the convention that concatenated whitespace act as a single delimiter pseudocharacter. This allows humans to format the data prettily (e.g., with CR/LF and multiple spaces). Tab characters are allowed but deprecated as fragile. If the data elements contain whitespace, then another delimiter must be used (e.g., "|"). This should be set through the delimiter attribute.

<array> is normally used without explicit indication of the number of array elements, though @size can be used to check data integrity. However, to save space and to improve robustness a regular array such as an *x*-axis can be specific by the generating functions start, end, and size. Thus, <array start="3.0" end="10.0" size="8"/> will generate the implicit array <array>3.0 4.0 5.0 6.0 7.0 8.0 9.0 10.0</array>.

This implicitly generates a stepSize of 1.0, but the triple of (@start, @end, @stepSize) is extremely fragile and may generate different @sizes due to rounding errors. In discussion of arrays and matrices the XPath convention is used: all indices start at 1. Thus the above array runs from 1 to 8 with array[1] = 3.0 and array[8] =10.0, etc. The following representations are semantically equivalent to the above:

```
<array>
3.0   4.0
5.0   6.0
7.0   8.0
9.0  10.0
</array>
```

or <array delimiter="|">|3.0|4.0|5.0|6.0|7.0|8.0|9.0|10.0|</array>. (Note the leading and trailing delimiters which ensure that XML prettifiers do not add whitespace.)

<matrix> always represents a rectangular 2D-array of homogeneous data elements and must have @rows and @columns attributes which determines the number of data elements.

<scalar>, <array>, and <matrix> are the primary method for communicating scientific data which do not have spe-

Chart 1. CMLSpect Example of an NMR Spectrum with Peak Assignments to the Corresponding Molecular Structure

```
<cml>
  <molecule id="mol1">
    <atomArray>
      <atom id="a1" elementType="C" x2="4.231" y2="1.1423"/>
      <atom id="a2" elementType="C" x2="4.231" y2="1.9673"/>
      <atom id="a3" elementType="C" x2="3.5167" y2="2.3797"/>
      <atom id="a4" elementType="C" x2="2.8022" y2="1.9673"/>
    </atomArray>
    <bondArray>
      <bond id="b1" atomRefs2="a1 a2" order="D" />
      <bond id="b2" atomRefs2="a2 a3" order="S" />
      <bond id="b3" atomRefs2="a3 a4" order="D" />
    </bondArray>
  </molecule>
  <spectrum type="NMR" moleculeRef="mol1">
    <parameterList>
      <parameter dictRef="jcamp:averages" title="averages">32</parameter>
      <parameter dictRef="jcamp:MMR_OBSERVEFREQUENCY">
        <scalar units="units:mhz">400</scalar>
      </parameter>
    </parameterList>
    <conditionList>
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">500</scalar>
      <scalar dataType="xsd:string" dictRef="cml:temp" units="siUnits:k">Unreported</scalar>
    </conditionList>
    <peakList>
      <peak xValue="3.36" xUnits="units:ppm" peakShape="sharp" id="p0" atomRefs="a1"/>
      <peak xValue="3.60" xUnits="units:ppm" peakShape="sharp" id="p1" atomRefs="a2" />
      <peak xValue="3.77" xUnits="units:ppm" peakShape="sharp" id="p2" atomRefs="a3"/>
    </peakList>
  </spectrum>
</cml>
```

cific elements in CML. Their semantics are provided through a @dictRef attribute which should identify an <entry> in a dictionary. Thus, <scalar dictRef="jcamp:NMR_OBSERVEFREQUENCY" dataType="xsd:double" units="units:mhz">400.01</scalar indicates that the quantity 400.01 is to be interpreted with the semantics described in the dictionary with the jcamp namespace. Dictionaries are conrolled vocabulary agreed upon by a community.[10] Examples of dictionaries are given in subsection 2.2. In general all primitive data containers (or their ancestral containers) should carry @dictRef attributes. Data containers are also typed, using either the standard W3C dataTypes (e.g., xsd:double), types in the CMLSchema (e.g., cm1:elementType), an element symbol in the IUPAC periodic table, or created by the author (e.g., nmr:frequencyType). The @dataType is extremely valuable in helping software decide how to process CML files; thus a standard XSD processor could throw an error for

```
<array dataType="xsd:nonNegativeInteger">3 7 -1 5 3</array>
```

Most numeric data are associated with quantities linked to scientific units of measurement. For describing units the CML dictionary mechanism can be used. There exist multiple dictionaries for units. Thus the units in <scalar dictRef="jcamp:NMR_OBSERVEFREQUENCY" dataType="xsd:double" units="units:mhz">400.01</scalar> are defined by the entry with id='mhz' in the nmrUnits units dictionary. Even in a subdomain such as "spectroscopy" it will take considerable time to aggregate and agree on a communal system of units, so we expect most CMLSpect documents to reference one or more specialist units dictionaries.

Alternatively the evolving UnitsML specification (http://unitsml.nist.gov/) could be used.

2.2. Chemical Compounds, Other Materials, and Context. Spectroscopy is carried out on substances, which may or may not be pure compounds describable by a connection table.

If it is a pure compound, then it can be identified or described at a number of levels:

- Identifier (IUPAC InChI, CAS registry number, etc.): This should be a precise agreed string which can be resolved to an agreed chemical concept.
- Formula: Often the chemical composition of a pure substance is known but not its structure. The <formula> element supports this. Simple constitutional formulas are best managed with the following concise attribute: <formula

concise="C 10 N 2 H 6"/>. Other approaches are allowed with inline and convention, but there is no guarantee that a processor can understand them.

• <molecule>/<atomArray>/<atom> and <molecule>/<bondArray>/<bond>: This is the normal method of representing pure materials defined at the atomic level (bonds are optional in extended solids, e.g., NaCl). All atoms and all bonds (if present) have ids which are unique within the molecule. Ids need not be ordered or meaningful, but it helps humans. This allows other elements (e.g., <peak>) to reference them directly and accurately. Chart 1 has an example for this (in this case a ^{13}C NMR assignment):

```
<molecule id="mol1">
  <atomArray>
    <atom id="a1" elementType="C" x2="4.231" y2="1.1423"/>
    ...
  </atomArray>
</molecule>
<peakList>
  <peak xValue="3.36" xUnits="units:ppm"
                peakShape="sharp" id="p0" atomRefs="a1"/>
  ...
</peakList>
```

• <atomSet>, <bondSet> and @atomRefs, @bondRefs: CML provides tools for mapping one set of elements to another. The @atomRefs and @bondRefs attribute on peaks and other spectral features map these to a set of atoms directly. In addition the <atomSet> element provides a mechanism for identifying a set of atoms with some common semantics (a functional group, major contributors to a normal mode, isotopically labeled, etc.). There are corresponding <bondRefs> and <bondSet> components. There can be any number of (overlapping) atomSets in a molecule. Furthermore elements can have any number of <label> children which allows their annotation in chemical or spectroscopic terms.

If, on the other hand, the substance is not a pure compound, CML has a series of components to support this. <sample> is a container with no other controlled semantics. In other words authors can put anything they like in <sample>, but they cannot rely on machines understanding the content. An example (using standard CML elements) might be

```
<sample>
  <object title="silica cell" dictRef="mydict:silicaCell">
    <scalar dictRef="mydict:cellLength" units="cml:cm">1</scalar>
  </object>
</sample>
```

suggesting that the sample was contained in a cell of a particular material and size. A generic CML processor would not understand the semantics, but a specialized community could create their own software that recognizes and processes the attributes.

<substance> refers to a physical material, and if contained within <spectrum> is the material being studied. In some cases there will be mixtures, for which <subtanceList> with child <substance>s should be used. In general <substance>s (e.g., Nujol, polystyrene) will not have simple connection tables. Together with <sample>, <substance> can be used for an example like this

```
<sample id="abc123"
   xmlns:spect="http://www.blueobelisk.org/dict/spect"
   xmlns:spectsubst="http://www.blueobelisk.org/dict/spectsubst">
  <substanceList title="Nujol mull" dictRef="spect:mull">
    <substance title="my compound">
      <molecule ref="#ab12"/>
    </substance>
    <substance dictRef="spectsubst:nujol" title="Nujol"/>
  </substanceList
</sample>
```

Here there is a list of two substances in the sample. Their relationship ("mull") is described in a dictionary of spectroscopic terms. The compounds are first our own substance, identified by a molecule reference, and second Nujol, which is described in a dictionary of spectroscopic substances, since it is an "external" substance. Both of these dictionaries are contained in the Supporting Information of this article. Note these dictionaries are prototypes to clarify examples rather than real world dictionaries.

Apart from the substance, an experiment is characterized by its conditions. In CML, these can be contained within <conditionList>, already developed for CMLReact, and <parameterList>, widely used in CMLComp. The borderline between <parameterList> and <conditionList> is arbitrary, but chemical conditions (pH, temperature, concentration, etc.) are probably best as conditions, while some machine settings may be better as parameters. At present <conditionList> does not have <condition> children and takes objects directly. Again we look at Chart 1:

```
<spectrum>
  <parameterList>
    <parameter dictRef="jcamp:averages"
                    title="averages">32</parameter>
    <parameter dictRef="jcamp:MMR_OBSERVEFREQUENCY">
      <scalar units="units:mhz">400</scalar>
    </parameter>
  </parameterList>
  <conditionList>
    <scalar dataType="xsd:string" dictRef="cml:field"
                  units="siUnits:hertz">500</scalar>
    <scalar dataType="xsd:string" dictRef="cml:temp"
                units="siUnits:k">Unreported</scalar>
  </conditionList>
</spectrum>
```

2.3. Spectra. CML contains elements which are specific to spectra and will be found inside of <spectrum>. Important attributes for <spectrum> are @id, @type, and @convention. @id should be a unique identifier for the spectrum, which can be used to reference it. @type gives the type of the spectrum (currently possible values are NMR, massSpectrum, infrared, UV/vis, others can be introduced via a dictionary). @convention is explained in section 2.7. <spectrum> can also hold a number of siblings, the most important

ones being <peakList> and <spectrumData>, since these hold the actual analytical data. <spectrumData> is for continuous data, whereas a <peakList> can hold peaks. Either one of them or both can be contained in a <spectrum>.

The existence of both possibilities makes CMLSpect flexible to hold a wide range of data, from "raw" results to a highly processed peak spectrum. If both are in, the <peakList> is supposed to hold the peaks in the continuous spectrum, although this is not (and cannot be) enforced by the specification.

The <spectrumData> holds an <xaxis> and a <yaxis>. Inside these <array> and <matrix>, as explained above, are used to store values. If <xaxis> and <yaxis> both hold an <array> of values, then we have a continuous spectrum, although in the strict sense such a spectrum is discrete as well, but it can become pseudocontinuous by making the steps small enough. The arrays hold a size attribute, which, if set, must match for a valid spectrum. This is the most generic way of giving a spectrum since in <array> the attributes units and dataTypes are used to specify the axis. A <matrix> can be used to handle two-dimensional spectra.

A <peakList> holds the peaks in a spectrum, which is a feature aimed at the high practical relevance of peaks in chemical spectroscopy. Mostly a <peak> will be a maximum, but it could also be any other point of interest. The peak can be defined not only as a point with the attributes xValue and yValue but also as a range with xMin/xMax and yMin/yMax. An important concept here are assignments of peaks, which are necessary for rich spectral information. CMLSpect uses atomRefs, bondRefs, and moleculeRefs for this. The atomRefs could be used, e.g., in a one-dimensional NMR spectrum, whereas the bondRefs could be used in, e.g., an IR spectrum. Again the concept is generic and can hopefully be used to encode any chemical concept. A <peak> can have a <peakMultiplicity>, e.g., triplet in an NMR spectrum.

Associated with <peak> are two important concepts: <peakGroup> and <peakStructure>. A <peakGroup> combines several peaks, which have a chemical relationship (as opposed to <peakList>, which is a structural container). A <peakStructure> gives details about peaks, mainly, e.g., NMR couplings (@type="coupling"). Value and units specify then the coupling constant and atomRefs specify to which atom the peak couples. It is also important to distinguish between a <peak> and a <transition>, the former belonging to a spectrum, while the latter is more properly considered a feature of a <model>. The mapping of, e.g., <peakStructure> to the predictions of a specific model is not considered in the present article.

2.4. Annotation. Current practice in annotating spectra has several levels beyond the simple record of the data emitted from an instrument. Most commonly there is an attempt to rationalize the spectrum as a series of peaks and other subsidiary features (e.g., shoulders). This arises because many spectra are explained as the cumulative effect of individual components. In CMLSpect "peak" means simply that the author has identified a region of the spectrum and wishes to comment on it. At the simplest level the peak is simply a (lossy) method of communicating the complete spectrum in a small set of text and numeric strings. Before the advent of computers this also proved useful as an indexing method—volumes of common peak values could be consulted to identify the material or parts of its structure. With the cheap storage and transmission of data we deprecate the peak list as the only method of disseminating spectral information. Although peak lookup is still powerful and valuable, it will necessarily be enhanced by having the full spectral data. Moreover the human abstraction of peaks is often highly error-prone and subjective (for example only the peaks fitting a particular theoretical model might be reported, and reviewers cannot see the additional effects due to (say) impurities).

Peaks are often interpretable in terms of the (presumed) structure of the material and annotated with terms such as

- C═O stretching
- Aromatic protons
- Porphyrin chromophore
- Octahedral Fe(II)

CMLSpect therefore supports the annotation of peaks in terms of molecules and parts of molecules. In a general form a molecular feature gives rise to a series of peaks (a <peakGroup>) and is composed of a series of atoms (<atomSet>) and or bonds (<bondSet>). By using the CMLElement, any set of atoms, bonds, or molecules can be mapped to any set of peaks or peakGroups, and examples are given later. It is important to use the appropriate ontology to indicate that the fundamental physicochemical description and the vernacular usage may be different. We note that CMLSpect may in many cases be useful for holding a subset of the results of chemical calculations but that it does not seek to provide a complete description of the phenomenon or of the model underlying the phenomenon. However, a considerable degree of the latter could be supported by the use of <scalar>, <array>, <matrix>, <eigen>, when linked to domain-specific dictionaries. Thus an observed infrared spectrum with <peakGroup>s might be accompanied by the results of harmonic vibrational analysis (model) from a normal coordinate program recorded as <eigen>. Mapping (<map>) between the peakGroups and components of the <eigen> is possible.

The usage of <peak>, <peakGroup>, and <peakList> is subjective. In general a <peakList> is a list of peaks and/or peakGroups which benefit from being corecorded but which have no strong semantic relationship other than (perhaps) emanating from the same experiment. A peak is often a single feature whose first and second derivatives cross the baseline once and twice, respectively. However, peaks may be "split", and this effect may be either disregarded by the author or have an explanation whose language uses "peak" to describe this. For this we provide ancillary annotation tools such as <peakStructure>. This is a (non-CML-controlled) vocabulary which can be used to reflect the domain of discourse through dictionaries. A <peakGroup> normally comprises peaks which have some chemical semantic relationship. This might be a single main phenomenon (e.g., a vibration) affected by isotopic variation or perturbation ("coupling") to another phenomenon. In some cases the structure of the spectrum is sufficiently fine and understood that it makes sense to have nested <peakGroup>s.

CMLSpect enables human and machine readers to extract annotated information (or conversely add to it). This should allow machines to (say) carry out a complete analysis of the coupling constants in a well-resolved organic ^{1}H NMR spectrum. It also should allow heuristic or ab initio programs to make assignments such as IR frequencies and NMR nuclei.

In this way, with well reported spectra, machines should be capable of carrying out much of the initial understanding of the molecule.

2.5. CMLSpect scope. CMLSpect is designed to capture most, but not all, types of spectra. At present CMLSpect can support the following:

• A continuous spectrum ($y = f(x)$). This is the commonest type. CMLSpect can hold both real (frequency) and imaginary (FT) spectra (but does not require users to have software for transformation). We use the CMLElement <array> to hold both x and y. Where the x elements are regularly spaced it is possible to define the step size and number of steps (this is a new feature of CML compared to ref 8).

• A discrete spectrum ((x1, y1) (xn, yn)). This is exemplified by peaks in a mass spectrum or C NMR spectrum where the peak widths are so small that it is customary only to report peaks. There need not be any restriction on monotonicity for x or y so that cyclic voltammograms can be represented.

• A two-dimensional spectrum ($z = f(x, y)$). This requires a rectangular matrix (<cm1:matrix>) to hold the values.

In principle CML can hold parts of a spectrum (it is common to publish small snippets of a spectrum at higher magnification than the rest). This would be enabled by a spectrum containing multiple <spectrumList>s. However, we would deprecate this in favor of publishing the complete spectrum data from which the magnified snippets can always be determined.

Some experiments, especially modern gas-chromatography and mass spectroscopy, can contain several components which are chained in some manner. CMLSpect currently provides no explicit support for multiple methods, and, as this is an evolving field, it is likely that many of these will be manufacturer-dependent.

Note that the basic components of CML for managing data are very flexible and can be combined in many new and meaningful ways. It is easy to store multiple versions of a spectrum, perhaps to analyze errors or time-dependent phenomena. In principle <array> allows users to add error information to a spectrum (rarely done at present). We have also used CMLSpect to contain derivatives of a spectrum, baseline corrected data, etc. all in the same container. The semantics of these will be local and will require human documentation and software support, but they may then evolve as CMLConventions in their own right.

2.6. Semantics and Ontology. CML provides a number of tools for supporting semantic relationships. In general CMLSchema does not provide explicit markup for most concepts but delegates this to dictionaries. These can be developed independently of the schema and will generally contain hundreds or more <entry>s providing semantics for a given community. By default CML uses the @dictRef mechanism, and element or attributes are only created if they require special support in the language. Thus "temperature" needs no specific language-support in CML.

CML tries to avoid controlled vocabularies where the information is representable as <scalar>, <array>, and <matrix>. Even an enumerated list of spectral types would be out of date very soon. An important criteria for inclusion in CML itself is whether the concept is meaningless without supporting code or hardcoded semantics (as in the <peak*> elements).

Relationships between XML elements are becoming increasingly important, and generic approaches such as RDF support arbitrary types. However, these do not yet easily support procedural code, and CML therefore has a number of tools to support relationships. These include the following:

•@ref. This is a reference (by id) to an element present elsewhere in the document. This has similarities to href in HTML and typed pointers/references in Object Oriented languages. Thus

```
<molecule id="tms">
  <atomArray>
    <atom id="a1" elementType="Si"/>
    <atom id="a2" elementType="C" hydrogenCount="3"/>
    <atom id="a3" elementType="C" hydrogenCount="3"/>
    <atom id="a4" elementType="C" hydrogenCount="3"/>
    <atom id="a5" elementType="C" hydrogenCount="3"/>
  </atomArray>
  <bondArray>
    <bond id="a1 a2" atomRefs2="a1 a2" order="1"/>
    <bond id="a1 a3" atomRefs2="a1 a3" order="1"/>
    <bond id="a1 a4" atomRefs2="a1 a4" order="1"/>
    <bond id="a1 a5" atomRefs2="a1 a5" order="1"/>
  </bondArray>
</molecule>
<molecule id="cdcl3">
  ...
</molecule>


<spectrum>
  <molecule id="m1">
    <formula concise="C 22  H 33 N 5 O7"/>
    <metadataList>
      <metadata name="dc:title">Unknown active
                        constituent of ...</metadata>
    </metadataList>
  </molecule>
  <substanceList>
    <substance role="nmr:marker">
      <molecule ref="tms"/>
    </substance>
    <substance role="solvent">
      <molecule ref="cdcl3"/>
    </substance>
    <substance>
      <molecule ref="m1"/>
    </substance>
  </substanceList>  ...
</spectrum>
```

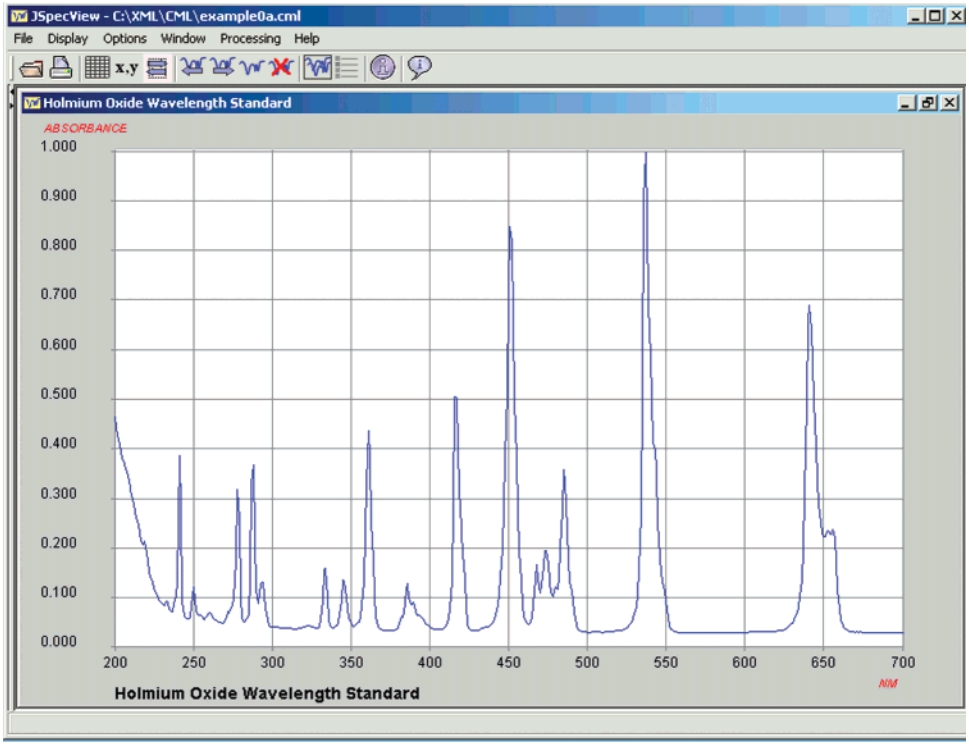


Figure 1. This shows Chart 2 opened in the JSpecView program.

has semantics as if the 3 complete molecules had been included at the ref.

• <link>, <map>. <link> provides a semantic link between a set(from) of 1 or more element(s) and a set(to) 1 or more of element(s). <map> can contain many semantically related links. Later examples should show the approach.

"Metadata" is now highly valuable for Web and other applications. It has many uses including discovery, typing, rights, and provenance, some of which are provided by Dublin Core. CML allows it to be inserted as a child element almost anywhere but more will occur under important containers (<cml>, <molecule>, <spectrum>, etc. and their <*List>s). There can be no absolute agreement on the difference between <parameter> and <condition>. Our very rough approach is that it is something that the library/Web community cares about and will index or use, while <parameter>/<condition> is more likely to be used by processing software. Processing tools should be able to inspect all containers for a variety of information.

CML does not prescribe practice, and we hope that communities will develop their own consistent semantics and conventions. The @convention attribute supports such communities. In essence it announces the following: "this element and its descendants have semantics understood and agreed by the community owning the convention. These semantics cannot override builtin CML semantics (e.g. a <peak>cannot be redefined to have an @elementType) but can resolve uncertainties. Subscribers to the convention are likely to need to provide additional software to understand it. Nonsubscribers can ignore it without losing the semantics inherent in CML".

The use of @conventions is still evolving. We expect it to be used within spectroscopy to give "hints" as to how something is to be processed or rendered. Thus <spectrumList> has no controlled semantics. <spectrumList convention="nmr:overlay"> might indicate that the spectra within the list can be overlaid in a diagram (this might require them all to have consistent axes). Alternatively, <spectrumList convention="acmePubs:compound"> might indicate that the components are complementary (IR, NMR, UV, vis) and should be laid out in the ACME publishing style. In both cases processing software could ignore the @convention and process the spectra individually without corruption.

2.7. Defining Conventions. There is no global agreement on what a spectrum should contain and how it should be interpreted. Some of the current authors wish to insist that a spectrum contains required information, while others want to leave it open to cater for unanticipated uses. Writers of software must rely on certain component information being present and valid. The XML community in general has no mechanism for addressing this, though some individuals have recognized the problem and devised local schema metalanguages to provide greater functionality than XML Schema and Relax-NG. We therefore have to develop our own approach using Schematron, XSLT and XPath. This approach is still evolving, but two conventions are already being used for the NMRShiftDB.org and JSpecView. We anticipate to cover most of the required chemical validation. Here are some examples:

• If the spectrum is continuous and one-dimensional, then the number of xvalues (existing or generatable) and yvalues must be equal.

• If the spectrum is an IR transmittance spectrum, then the "peaks" are local minima.

• For an ^{1}H NMR spectrum the frequency of the chemical shift is obtained by multiplying value in ppm by the machine frequency $*10^{6}$.

A generic CMLSpect processor cannot be expected to support such a variety of constraints and conversions; moreover, they may not be generally agreed in the com-

munity. Hence, we allow any author to assert that their information obeys a qualified convention (typically an attribute on the root element). Here we illustrate this with two such conventions created by the authors: JSpecView and NMRShiftDB. The NMRShiftDB convention can be found at http://nmrshiftdb.sourceforge.net/nmrshiftdb-convention.html, and the JSpecView convention can be found at http://jspecview.sourceforge.net/convention.html. Both these conventions are defined in an informal, human-readable documents and backed-up with machine-readable Schematron files, to allow automated validation (see section 6 for details of schematron usage). An example schematron file for NMRShiftDB can be found in the Supporting Information.

An author applying a convention attribute is asking the reader (human or machine) to find out whether it can support that convention. As all conventions map to namespaced strings, there is no lexical confusion. Our examples might be

```
<spectrum convention="JSpecView"
 xmlns:JSpecView="http://jspecview.sf.net/convention.html"
                                              type="infrared">
  <spectrumData>
    <xaxis>
      <!-- in the JSpecView convention, the xaxis is
           given by first value - last value - size -->
      <array units="jcampUnits:xUnits" start="firstX"
        end="lastX" size="npoints" dataType="xsd:double" />
        ....
      </array>
    </xaxis>
    <yaxis> ... </ydata>
  </spectrumData>
</spectrum>


<spectrum convention="nmrshiftdb"
 xmlns:nmrshiftdb="http://nmrshiftdb.sourceforge.net/
                                 nmrshiftdb-convention.html"
 type="NMR">
  <!-- in NMRShiftDB, the conditonList must contain
                                          field strength -->
  <conditonList>
    <scalar dataType="xsd:string" dictRef="cml:field"
      units="siUnits:hertz">Unreported</scalar>
    ...
  </conditionList>
  <!-- in NMRShiftDB, a spectrum must have a peakList -->
  <peakList>...</peakList>
</spectrum>
```

A processing engine (user agent) reading the first might take the following actions:

- Recognize the JSpecView namespace and process the information on the assumption it is JSpecView-compliant. If it finds an inconsistency with the JSpecView convention, then it can and should throw an error or warning.
- Discover that it knows nothing about the JSpecView namespace and try to process the components in a context-independent manner. In practice this might mean that it recognizes and depicts the spectrum and might label the axes with title attributes. In most cases this might be sufficient, but it would probably not use common display conventions such as direction of axes for IR spectra. Context-independent processing is unlikely to crash but may be unacceptable to sections of the community.
- Discover that it knows nothing about the JSpecView namespace and assert that it cannot process it.

Since the convention attribute is available for anyone to use, its values should be selected to maximize interoperability. This is a sociopolitical problem not a technical one. If every manufacturer, every publisher, etc. creates its own conventions for CMLSpect, then we shall have achieved syntactic coherence but semantic tag soup. Even if ontological mappings can be devised (e.g., with RDF and OWL) it will be difficult to manage the many:many conversions.

We therefore suggest an Open list of conventions that we believe will be useful and supportable. Besides promotion here, they will be honored in the Blue Obelisk Open chemical community (http://www.blueobelisk.org/)[11] who will attempt to use them in all cases of spectral interchange. We hope that they will anneal to a system which one agreed convention for a given type of spectrum (e.g., transmission-ir, continuous ^{1}H NMR, etc.). Each will be defined by an <u1:appinfo>-like ruleset defining what elements must, may, and must not occur. More complex relations can also be described, and we hope that wherever possible these are in the formal language. This should allow developers to add validation tools to their processing software (or provide validation services), to create mutators and accessors automatically, and to apply some of the simpler transformations through communally agreed methods.

3. EXAMPLES OF CMLSPECT FILES

Our first example (see Chart 2 and Figure 1) is a UV/vis spectrum. It has been generated by the CML export of JSpecView (see 4.3), as one can see from the @convention attribute. As root element, we have a <cml> element, which contains the <spectrum>. The <spectrum> shows the @type and the @convention. Furthermore, there is an @id and a @title. The |id is a real id, identifying the spectrum when referencing to it, whereas the @title is intended for human reading and can contain any information considered useful. The <cml>element also contains the schema and references to several dictionaries. The core element of the <spectrum> are the <spectrumData> (the other elements will be explained in later examples). The spectrum just contains continuous data; therefore, we have an <xaxis> and a <yaxis> inside the <spectrumData> but no peaks. Both axis have units and a dataType. The <xaxis> has 501 values in equal distances from 200 to 700 nm. The <yaxis> also has 501 values, corresponding to the values in <xaxis> (... indicates values left out here for shortness). In this way we have defined a continuous spectrum.

Note that all examples in this text are shortened; the complete code for all examples (together with the documents

Chart 2. Example Showing the Spectrum of a Compound with CAS Registry Number 12055-62-8, along with Metadata[a]

```
<cml xmlns="http://www.xml-cml.org/schema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:siUnits="http://www.xml-cml.org/units/siUnits"
    xmlns:units="http://www.xml-cml.org/units/units"
    xmlns:jspecview="http://jspecview.sf.net/convention.html"
    xmlns:cml="http://www.xml-cml.org/dict/cml"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:jcamp="http://www.xml-cml.org/dict/jcampDict"
    xsi:schemaLocation="http://www.xml-cml.org/dict/jcampDict dict/jcampDict.xml ...>
  <spectrum id="UV_VIS" title="Holmium Oxide Wavelength Standard" convention="JSpecView" type="UV/VIS">
    <metadataList>
        <metadata name="jcamp:origin" content="Lambda 900"/>
        <metadata name="jcamp:owner" content="NIST-Gaithersburg"/>
    </metadataList>
    <parameterList>
      <parameter dictRef="jcamp:SpectrometerDataSystem" title="Spectrometer/Data System" value="Lambda 900" />
      <parameter dictRef="jcamp:resolution" title="resolution">
          <scalar units="units:nm">2.0</scalar>
      </parameter>
    </parameterList>
    <sample>
        <molecule>
            <formula inline="Ho2O3"/>
            <name convention="jcamp:casregistryno">12055-62-8</name>
        </molecule>
    </sample>
    <spectrumData>
        <xaxis>
            <array units="units:nm" size="501" dataType="xsd:double">
            200 201 202 ...</array>
        </xaxis>
        <yaxis multiplierToData="1.000">
            <array units="units:absorbance" size="501" dataType="xsd:double">
            0.46341657 0.44507496 0.42653419 ...</array>
        </yaxis>
    </spectrumData>
  </spectrum>
</cml>
```

[a] Most peaks in the spectrumData have been omitted for brevety.

referenced in schemaLocations) can be found in the Supporting Information.

The next example (Chart 3) is a proton NMR spectrum, which comes from NMRShiftDB (the entry can be viewed via http://www.nmrshiftdb.org/portal/js_pane/P-Results/nmrshiftdbaction/showDetailsFromHome/molNumber/10026026.The cml has been generated via JUMBO Java DOM.[12] The <cml> container contains a <molecule> and a <spectrum>. <cml>contains the NMRShiftDB convention mentioned above. We leave out details of the molecule, but note that the molecule itself and each atom has an @id. The @id of the molecule is used in the attribute @moleculeRef of the <spectrum> to tell which molecule the spectrum belongs to. In a <conditionList> the measurement conditions of the spectrum are defined and specified via dictionary entries. Here the importance of agreed vocabularies can be seen: If everybody would use his own terms for conditions, interchangeability would be impossible. On the other hand, we cannot specify all conditions which might be needed for all sorts of spectra in the CMLSpect definitions. By reference to dictionaries we can keep CMLSpect flexible and at the same time stick to conventions. We also have a <substanceList>, which contains the solvent used, and a <metadataList>, which tells us the assignment method and the spectrum type ("1H" means proton spectrum). Again these use dictionaries. The spectrum itself only consists of <peak>s in a <peakList>, meaning we have no continuous data (this is because NMRShiftDB only saves peak lists). Note the peaks do not have @yValues, which is common in reported NMR data, where the *y*-axis, the intensity, is only of minor importance (although clearly of importance to any model of the spectrum) and therefore often not stored in NMRShiftDB. All peaks are assigned to atoms via an @atomRef. Furthermore, they have a <peakStructure>with @type of value coupling, which means that coupling constants are also encoded. E.g., the first peak (at 3.36 ppm) is assigned to atom a25 (which is, not much surprising in a proton spectrum, a hydrogen). We even are told it is has multiplicity "ddd", and it couples with atom a32 with 1.8 hertz and with atom a22 with 10.3 hertz.

Chart 3. Example CMLSpect Document Containing a Molecular Structure and Assigned NMR Spectrum, Including Metadata, and peakShape or Multiplicity[a]

```
<cml xmlns="http://www.xml-cml.org/schema" ... convention="nmrshiftdb"
 xmlns:nmrshiftdb="http://nmrshiftdb.sourceforge.net/nmrshiftdb-convention.html" ...>
  <molecule title="1H-Indol-3-yl-beta-D-ribohexo-3-ulopyranoside" id="nmrshiftdb10026026">
    <atomArray>
      <atom id="a1" elementType="C" x2="4.231" y2="1.1423" formalCharge="0" hydrogenCount="0" />
      <atom id="a2" elementType="C" x2="4.231" y2="1.9673" formalCharge="0" hydrogenCount="0" />
      <atom id="a3" elementType="C" x2="3.5167" y2="2.3797" formalCharge="0" hydrogenCount="0" />
      <atom id="a4" elementType="C" x2="2.8022" y2="1.9673" formalCharge="0" hydrogenCount="0" />
...
    </atomArray>
    <bondArray>
      <bond id="b1" atomRefs2="a1 a2" order="D" />
      <bond id="b2" atomRefs2="a2 a3" order="S" />
      <bond id="b3" atomRefs2="a3 a4" order="D" />
...
    </bondArray>
  </molecule>
  <spectrum id="nmrshiftdb10074894" moleculeRef="nmrshiftdb10026026" type="NMR">
    <conditionList xmlns="http://www.xml-cml.org/schema">
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">500</scalar>
      <scalar dataType="xsd:string" dictRef="cml:temp" units="siUnits:k">Unreported</scalar>
    </conditionList>
    <substanceList>
      <substance role="subst:solvent" title="DMSO-d6"/>
    </substanceList>
    <metadataList xmlns="http://www.xml-cml.org/schema">
      <metadata name="nmr:assignmentMethod" content="1D shift positions, HMBC, HMQC" />
      <metadata name="nmr:OBSERVENUCLEUS" content="1H" />
    </metadataList>
    <peakList xmlns="http://www.xml-cml.org/schema">
      <peak xValue="3.36" xUnits="units:ppm" peakShape="sharp" id="p0" atomRefs="a25">
        <peakStructure type="coupling" peakMultiplicity="nmr:ddd" atomRefs="a22"
         units="unit:hertz" value="10.3" />
        <peakStructure type="coupling" peakMultiplicity="nmr:ddd" atomRefs="a32"
         units="unit:hertz" value="1.8" />
      </peak>
      <peak xValue="3.60" xUnits="units:ppm" peakShape="sharp" id="p1" atomRefs="a31" />
...
    </peakList>
  </spectrum>
</cml>
```

[a] For brevity, not all atoms/bonds and peaks are given.

Our third example (Chart 4) is an infrared spectrum (see the @type attribute). As opposed to the previous example, there is no <cml> root element, but <spectrum> forms the root directly. This is possible, if we do not want to combine several elements. The spectrum, as one can see from the @convention attribute, comes from JSpecView. A structure is given via a <sample> CMLElement. The structure is not fully specified, but a CAS registry number and a chemical formula are given. Also the name of the substance is used as the title of the spectrum, which is not appropriate in a strict sense, but possible since the title is not strictly defined but is open for any human-readable entries. It makes sense to use the <sample> attribute (and not a separate <molecule> with a @molRef), since this is more a reference to an outside structure, whereas in the second example the structure was fully specified in the cml file. We again have conditions, actually the same pressure in two units. The metadata are defined via dictionaries, in that case using Dublin core. The spectrum is given as a continuous spectrum and some peaks. <spectrumData>contains the continuous data, and <xaxis> and <yaxis> both have an <array>. @size is identical, both are doubles, but the units are

different. We also have a <peakList>, which contains three <peak>s in a <peakGroup> and one single peak.

The next example (Chart 5) is a mass spectrum (shown by the @type of <spectrum>). It has also been generated via an export from JSpecView. It has metadata and parameters, which come from JCAMP and have therefore been generated in the JCAMP namespace. The substance is given in <sample> via a CAS number. The spectrum itself is shown as a collection of (around 90) peaks (some omitted here). The spectrum is described in this way (and not via <xaxis> and <yaxis>), since mass spectra are normally not given as a continuous line but as a collection of lines resembling a bar chart.

Our last example (Chart 6) shows how an experimental section of a journal article can be stored in CML. We extracted the experimental section of a paper[13] into a CML file, whereas in a CML-based workflow the procedure would be the other way around. The author would produce a CML, and out if this the experimental section (and other documents) would be created automatically (or at least with computer support). The CML file contains all data from the experimental section of the article. It has <molecule> and <spectrum> entries, which are enriched by the rest of the data. The example only shows the first two of the six structures mentioned in the paper, but it could be done for the rest as well. All spectra reference the substance via an id. There are seven spectra for each compound. Some things to mention are as follows:

- Textual content of the experimental section has been extracted to metadata, like the instruments used. Dictionaries serve for explaining the meaning.
- A BibTeXML entry has been added for keeping the literature information.
- The accuracy of the peak information depends on the information in the paper. CML could hold much more information, e.g., the couplings in the ^{1}H NMR spectra could also be fully assigned by @atomRefs.
- Some ids have been changed to comply with CML rules.

4. PRACTICAL APPLICATIONS AND USE CASES

This section shows a few applications using CMLSpect and illustrative use cases, where CMLSpect is used for building workflows and solving real world problems.

4.1. Application 1: NMRShiftDB. An application using CMLSpect is NMRShiftDB.[14,15] NMRShiftDB is an open-access, open-source, and open-content database of small organic molecules and their assigned NMR spectra. A Web interface is available at http://www.nmrshiftdb.org. Internally NMRShiftDB uses a relational database for storage, but it has different CML interfaces: First, all data can be downloaded as CML (example 2 was such a download), and, second, there are some Web services available using CML. An example for this is a Web service for predicting spectra (called doPrediction), which takes as an input a molecule in CML format and a spectrum type (as a string) and returns a predicted peak spectrum for this. Since Web services are XML-based messages sent via the Internet, CML can easily be integrated in these messages without conversions. If an application outputs CML (as done by NMRShiftDB) and the client works with XML, they can easily interact. Furthermore since CML is XML based and a schema exists for it, the CML schema can directly be used to to define data types in the WSDL (Web Services Definition Language). A WSDL (see http://www.w3.org/TR/wsdl) file for the prediction service looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://www.nmrshiftdb.org/ws/NMRShiftDB/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:cml="http://www.xml-cml.org/schema"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  name="NMRShiftDB"
  targetNamespace="http://www.nmrshiftdb.org/ws/NMRShiftDB/">
  <wsdl:types>
    <xsd:schema
      targetNamespace=
                  "http://www.nmrshiftdb.org/ws/NMRShiftDB/"
      xmlns:cml="http://www.xml-cml.org/schema">
      <xsd:import namespace=
               "http://schemas.xmlsoap.org/soap/encoding/" />
      <xsd:import namespace=
                       "http://schemas.xmlsoap.org/wsdl/" />
      <xsd:import namespace=
                       "http://www.w3.org/2001/XMLSchema" />
      <xsd:import
        namespace="http://www.xml-cml.org/schema"
        schemaLocation=
             "http://almost.cubic.uni-koeln.de/schema.xsd" />
      <xsd:element
        name="doPredictionResponse"
        type="cml:spectrumTypeType" />
      <xsd:element
        name="doPredictionRequest"
        type="tns:DoPredictionParameters" />
      <xsd:complexType name="DoPredictionParameters">
        <xsd:sequence>
          <xsd:element
            ref="cml:molecule"
            minOccurs="1"
            maxOccurs="1" />
          <xsd:element
            name="spectrumTypeName"
            type="xsd:string" />
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>
...
</wsdl:definitions>
```

Note that we have defined a CML namespace with @xmlns:cml="http://www.xml-cml.org/schema" and later define the input of the service as having a CML <molecule> with @ref="cml:molecule". Due to the easy combinations of XML formats, we can extend our cml files by other XML types. For example, NMRShiftDB also offers a Web service for submitting data to the database. CML <molecule> and <spectrum> are used to define the core data. We also want to submit a literature reference. With BibteXML, an established XML format exists for this. So we can embed it like this:

```
<cml xmlns="http://www.xml-cml.org/schema">
  <molecule id="m1">
...
  </molecule>
  <spectrum format="MASS"
      id="sid_4100e4c136f730d9c1fa27448dec9c2538c958b5"
   type="NMR">
    <peakList>
...
    </peakList>
    <bibtex:entry id="maaranolide"
                xmlns:bibtex="http://bibtexml.sf.net/">
      <bibtex:article>
        <bibtex:author>Jui-Hsin Su, Atallah F. Ahmed,
         Ping-Jyun Sung, Chih-Hua Chao, Yao-Haur Kuo,
         and Jyh-Horng Sheu</bibtex:author>
        <bibtex:title>Manaarenolides A-I, Diterpenoids from
         the Soft Coral Sinularia manaarensis</bibtex:title>
        <bibtex:journal>Journal of Natural
         Products</bibtex:journal>
        <bibtex:year>2006</bibtex:year>
        <bibtex:volume>69</bibtex:volume>
        <bibtex:pages>1134-1139</bibtex:pages>
        <bibtex:number>8</bibtex:number>
      </bibtex:article>
    </bibtex:entry>
  </spectrum>
</cml>
```

Here, a BibteXML entry (having namespace http://bibtexml.sf.net/) is embedded in a CML file (namespace http://www.xml-cml.org/schema). We also can see that the literature refers to the spectrum (and not, e.g., the molecule), because it is inside a <spectrum>.

4.2. Application 2: Bioclipse. Bioclipse (http://www.bioclipse.net/)[16] is a bio- and chemoinformatics workbench based on the Eclipse rich client platform. Following the Eclipse philosophy it contains many largely independent components (called plug-ins). CML, especially CMLSpect, is used as the "glue" between different components. Some components deal with spectra (charts for continuous and peak display, a table for editing peaks ...), and they all rely on the file browser to open files. The file browser reads all spectra files (currently CML and JCAMP-DX) into JUMBO objects, and the components all work on top of this. Here CML serves not only as a file format but also (together with JUMBO) as an internal data format. Furthermore Bioclipse integrates several of the NMRShiftDB Web services. Since everything there is in CML as well, the integration is done easily. E.g., we can handle a peak spectrum returned by a prediction without conversion.

4.3. Application 3: JSpecView. JSpecView evolved from code originally designed to display JCAMP-DX files in a browser plug-in. In 2006 it was released as an Open Source project,[17] and the decision was taken to extend the file types to read AnIML and CML documents. Unlike the original plug-in, JSpecView can open and display multiple files either as a set of tabbed displays or overlaid (if axes and units are similar). Other new features include the following: the ability to highlight areas in the spectrum, to be able to zoom in and out to many levels, to convert Absorbance to Transmittance or vice versa, and to calculate and display an integral curve for H NMR spectra. For JCAMP-DX files it is expected that a set of XY data is present in the form of continuous fixed increment/decrement X values, continuous XY points, or noncontinuous XY points in a table. For AnIML and CML documents, it again expects XY data so that the same routines can be used for display. This places some restrictions on its usefulness with CML documents where there may not be any displayable XY data available.

4.4. Use Case 1: Digial Repository. In the SPECTRa project[18] the chemistry department and librarians at the University of Cambridge and Imperial College London have developed a system for the automatic capture and archival of spectroscopic data. The spectroscopist uses the manufacturers software to create JCAMP5 files which are then automatically converted into CMLSpect using the JCAMP-DX library,[19] originally developed by Creon Lab Control. The spectra software then extracts key metadata from the CMLSpect file (e.g., frequency, nucleus, chemical shift range, standard, etc.) and integrates this with chemical metadata (e.g., <molecule> and <formula>) and also conventional metadata (e.g., author, date, digital rights, etc.). The data and metadata are then deposited into the DSpace institutional repositories, which can be harvested using the OAI-PMH protocol. In this way robots such as Google and Oaister can harvest spectra based on metadata such as spectrum type and chemical structure.

4.5. Use Case 2: Extracting Spectral Information from Publications. A reader is concerned about the assignment of chemical shifts for compounds published in an article. The downloaded article is run through OSCAR1, which extracts the peak data from the free text and emits it as a structured CMLSpect list of <PeakList>s. This has been implemented in Bioclipse, where the user can open an HTML version of a published paper and extract information from the experimental section, such as the molecular structure, NMR, MS, and IR spectra, and the elemental analysis details (see Figure 3). Using a wizard, the user can take the molecular structure and one of the NMR spectra to create a new CMLSpect resource. Either manually, or using a another wizard, the user can do a peak assignment (see Figure 2). Optionally, he may associate the bibliographic information of the article with the CMLSpect resource (using XML Namespace technologies) and prepare the CMLSpect file for submission to the NMRShiftDB.

Chart 4. IR Spectrum for a Compound with Concise Formula and CAS Registry Number[a]

```
<spectrum id="but2" title="2-Butanol" convention="JSpecView" type="infrared" state="gas"
   xmlns="http://www.xml-cml.org/schema" ...>
    <metadataList>
        <metadata name="jcamp:origin" content="Sadtler Research Labs Under US-EPA Contract"/>
        <metadata name="jcamp:owner" content="NIST Standard Reference Data Program"/>
        <metadata name="dc:identifier" content="No.424 (EPA Vapor Library"/>
        <metadata name="dc:source" content="I am guessing this means vapor phase"/>
    </metadataList>
    <sample>
        <molecule>
            <formula concise="C 4 H 10 O 1"/>
            <name convention="jcamp:casregistryno">78-92-2</name>
        </molecule>
    </sample>
    <conditionList>
        <scalar dictRef="units:bar">1.2345</scalar>
        <scalar dictRef="cml:press" units="siUnits:pascal">12345</scalar>
    </conditionList>
    <spectrumData>
        <xaxis>
            <array units="units:cm-1" size="227" dataType="xsd:double">734 738 ...</array>
        </xaxis>
        <yaxis multiplierToData="0.000109021">
            <array units="units:absorbance" size="227" dataType="xsd:double">42 37 43 ...</array>
        </yaxis>
    </spectrumData>
    <peakList>
        <peakGroup id="pg1" xMax="3040" xMin="2800">
            <peak id="ch1" title="CH-stretch-1" peakMultiplicity="singlet" peakShape="sharp" xUnits="units:cm-1"
              xValue="2974" yUnits="units:absorbance" yValue="1.0921"/>
            <peak id="ch2" title="CH-stretch-2" peakShape="shoulder" xUnits="units:cm-1"
              xValue="2938" yUnits="units:absorbance" yValue="0.653"/>
            <peak id="ch3" title="CH-stretch-3" xUnits="units:cm-1"
              xValue="2890" yUnits="units:absorbance" yValue="0.470"/>
        </peakGroup>
        <peak id="oh1" title="CH-stretch???" peakShape="broad" xUnits="units:cm-1"
          xValue="3657" yUnits="units:absorbance" yValue="0.1092"/>
    </peakList>
</spectrum>
```

[a] The spectrum is both given as peak list with peak annotation (in <peakList>) and as full spectrum (in <spectrumData>). Only the first few values of the IR spectrum are given.

5. EXISTING FORMATS

Ever since computers have been involved in the process of measurement and analysis of spectroscopic data, there has been the need for an easy way to store, archive, and interchange these data between different systems and platforms. Traditionally the storage of spectroscopic data is done by proprietary software which comes with the spectrometer and quite often stores this information in a proprietary data format as well which therefore cannot or just in a limited way be used for the exchange to other systems. This again makes displaying, processing, and printing outside the parent system very difficult. These formats cannot easily be used for the archiving of the information, because it will not be possible to combine data from different sources in a consistent way, which is of growing importance as the collection of experimental data is expanding and large data repositories emerge. Therefore, there are already a number of data format standards used, which try to handle and solve the problems coming with the proprietary formats. The following list is not exhaustive but gives an overview on the open standards we found to be the most important ones for the field of spectroscopy.

5.1. JCAMP-DX.[21] The Joint Committee on Atomic and Molecular Physical Data started as a Task Force on Spectral Data Portability under the direction of Paul A. Wilks, Jr., at the Pittsburgh Conference (Pittcon) of 1983. The scope of JCAMP was originally as follows: "The Joint Committee will generate, collect, evaluate, edit, and approve the publication and encourage the distribution of atomic and molecular physical data in suitable form to serve as references for pure compounds and mixtures" (Bob McDonald, http://members.aol.com/rmcdjcamp/faq.htm#JCAMP). JCAMP (the organization) was initially sponsored by the following: American Chemical Society (ACS), American Physical Society (APS), American Society for Mass Spectrometry (ASMS), American Society for Testing and Materials (ASTM), Optical Society

Chart 5: Mass Spectrum for 4-Vinylbenzyl Chloride (1592-20-7) Given as Peak List

```
<cml xmlns="http://www.xml-cml.org/schema" ...>
  <spectrum id="MS_4-vinylben" title="4-vinylbenzyl chloride" convention="JSpecView" type="massSpectrum">
    <metadataList>
        <metadata name="jcamp:origin" content="PSLC - Univ of Wisconsin-Stevens Point"/>
        <metadata name="jcamp:owner" content="Robert Badger"/>
    </metadataList>
    <parameterList>
      <parameter dictRef="jcamp:SpectrometerDataSystem" title="Spectrometer/Data System" value="unknown" />
    </parameterList>
    <sample>
        <molecule>
            <formula inline="C9H9Cl"/>
            <name convention="jcamp:casregistryno">1592-20-7</name>
        </molecule>
    </sample>
     <peakList>
       <peak id="a1" xUnits="units:moverz" xValue="26.05"
        yUnits="units:relabundance" yValue="0.86"></peak>
       <peak id="a2" xUnits="units:moverz" xValue="27.05"
        yUnits="units:relabundance" yValue="2.34"></peak>
...
      </peakList>
  </spectrum>
</cml>
```

of America (OAS), Society for Applied Spectroscopy (SAS), and Spectroscopy Society of Canada (SSC). The objective of the Task Force was to design a standard file format for exchange of infrared spectra between vendor data systems that used different proprietary file formats. Data exchange capability was in demand by end-users who wished to transfer spectra between different spectrometers in their own and other laboratories.

In 1995 IUPAC took over the responsibility for the JCAMP-DX range of scientific standards from the Joint Committee on Atomic and Molecular Physical Data (JCAMP). At that time an IUPAC Working Party had responsibility for the support and development of the JCAMPDX scientific data standards and this evolved into the Subcommittee for Electronic Data Standards (SEDS). All spectral data are stored as labeled fields of variable length using ASCII characters. The JCAMP-DX file is thus a text file that is human readable and can be edited and annotated using standard text editors.

The JCAMP-DX format has many advantages. It is open-source, using standard terms, so that data from any instrument, or simulated data, can be freely exchanged. Nonproprietary software is available for conversion from other file formats, file compression, Internet transmission, and visualization of the data. One criticism of the JCAMP-DX standard is that "the specifications are both complicated and incomplete. As the documentation grew to allow other types of instrument data, peak table data, and chemical structures, among other things, programmers had trouble interpreting how the tags should be used to write out their data. This led to the situation where JCAMP files created by a given software system could not be read by any other system because of inconsistent software implementations. This was exacerbated by the fact that there was (and still is) no way of validating the adherence to the JCAMP standards for files created by different software packages. There is no testing software available, and there is no overseeing body organizing round-robin testing among the vendors and policing their efforts." [see ref 22, p 3]. Despite this essentially all spectroscopic instruments currently available have routines for the export of JCAMP-DX files, which work well due to long usage in the practical world and improvements based on that.

5.2. AnIML.[23] The Analytical Information Markup Language (AnIML) is a *Web-aware* mechanism for the data interchange from instrument-to-instrument, application-to-application, and instrument-to-application being developed by the ASTM subcommittee E13.15. It is partly based on the SpectroML markup language from NIST and Thermo Electron's Generalized Markup Language (GAML). Additionally it borrows heavily from older interchange formats like JCAMPDX and ANDI, for existing data dictionaries and relevant markup languages. AnIML can be divided into two major parts, the core part, described by an XML (Extensible Markup Language) schema and the Technique Layers. The first can hold arbitrary analytical data, whereas the latter defines the structure of data and metadata for specific analytical techniques. The Technique Layers are extensible to permit vendor, enterprise, and/or user extensions to the data representation. The utilization of extension data will need custom software but should not break the general readability of the AnIML file. There is a validation software (the AnIML Validator) available to check an AnIML document for completeness, correct syntax, and, in a limited way, for semantic content.

We have worked closely with the authors of the emerging AnIML specification. Compared to CML, it is a heavily engineered specification with 5 layers (data, technique, vendor, enterprise, and audit). We expect CMLSpect and AnIML to complement each other in the following ways:

Chart 6. CMLSpec Document Containing the Molecules and Spectra Extracted from the Publication for which the Bibliographic Information Is Marked up in BibTeXML (http://bibtexml.sf.net/) Namespace

```
    <peakList>
      <peak xValue="99.4" xUnits="units:ppm" id="p1" atomRefs="a8" />
...
    </peakList>
  </spectrum>
  <spectrum title="Table 1.1" moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-1" type="NMR">
    <conditionList>
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">400</scalar>
    </conditionList>
    <substanceList>
      <substance role="subst:solvent" title="D2O"/>
    </substanceList>
    <metadataList>
      <metadata name="nmr:OBSERVENUCLEUS" content="1H" />
      <metadata name="nmr:INSTRUM" content="Varian Unity" />
    </metadataList>
    <peakList>
      <peak xValue="5.33" xUnits="units:ppm" id="p1" atomRefs="a40" peakMultiplicity="nmr:d">
        <peakStructure type="coupling" units="unit:hertz" value="3.6" />
      </peak>
...
    </peakList>
  </spectrum>
...
  <spectrum moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-1" type="UV/VIS">
...
  </spectrum>
  <spectrum moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-2" type="UV/VIS">
...
  </spectrum>
  <spectrum moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-1" type="spect:IR">
    <metadataList>
      <metadata name="nmr:INSTRUM" content="Perkin Elmer GS II" />
    </metadataList>
    <peakList>
      <peak xValue="3422" xUnits="units:wavenumber_energy"/>
...
    </peakList>
  </spectrum>
...
  <spectrum moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-1" type="massSpectrum">
    <metadataList>
      <metadata name="nmr:INSTRUM" content="Finnigan TSQ 1700" />
      <metadata name="spect:exactType" content="ESI-MS" />
      <metadata name="spect:treatment" content="[M-H]-" />
    </metadataList>
    <peakList>
      <peak xValue="417" xUnits="units:moverz"/>
    </peakList>
  </spectrum>
...
</cml>
```

Chart 6 (Continued)

```
<cml xmlns="http://www.xml-cml.org/schema" ...>
  <entry xmlns="http://bibtexml.sf.net/">
    <article>
        <author>Xiaozhe Zhang, Qing Xu, Hongbin Xiao, Xinmiao Liang*</author>
        <title>Iridoid glucosides from Strychnos nux-vomica</title>
        <journal>Iridoid glucosides from Strychnos nux-vomica</journal>
        <year>2003</year>
        <volume>64</volume>
        <pages>1341-1344</pages>
    </article>
  </entry>
  <molecule id="m10.1016_S0031-9422_03_00501-6-structure-1">
    <atomArray>
      <atom id="a1" elementType="C" x2="429.8481473023461"
       y2="601.3343638274541" formalCharge="0" hydrogenCount="0" />
...
    </atomArray>
    <bondArray>
      <bond id="b1321194" atomRefs2="a1 a2" order="S" />
...
    </bondArray>
  </molecule>
  <molecule id="m10.1016_S0031-9422_03_00501-6-structure-2">
...
  </molecule>
  <spectrum title="Table 2.1" moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-1" type="NMR">
    <conditionList>
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">100.57</scalar>
    </conditionList>
    <substanceList>
      <substance role="subst:solvent" title="D2O"/>
    </substanceList>
    <metadataList>
      <metadata name="nmr:OBSERVENUCLEUS" content="13C" />
      <metadata name="nmr:INSTRUM" content="Varian Unity" />
    </metadataList>
    <peakList>
      <peak xValue="99.5" xUnits="units:ppm" id="p1" atomRefs="a8" />
...
    </peakList>
  </spectrum>
  <spectrum title="Table 2.2" moleculeRef="m10.1016_S0031-9422_03_00501-6-structure-2" type="NMR">
    <conditionList>
      <scalar dataType="xsd:string" dictRef="cml:field" units="siUnits:hertz">100.57</scalar>
    </conditionList>
    <substanceList>
      <substance role="subst:solvent" title="D2O"/>
    </substanceList>
    <metadataList>
      <metadata name="nmr:OBSERVENUCLEUS" content="13C" />
      <metadata name="nmr:INSTRUM" content="Varian Unity" />
    </metadataList>
```

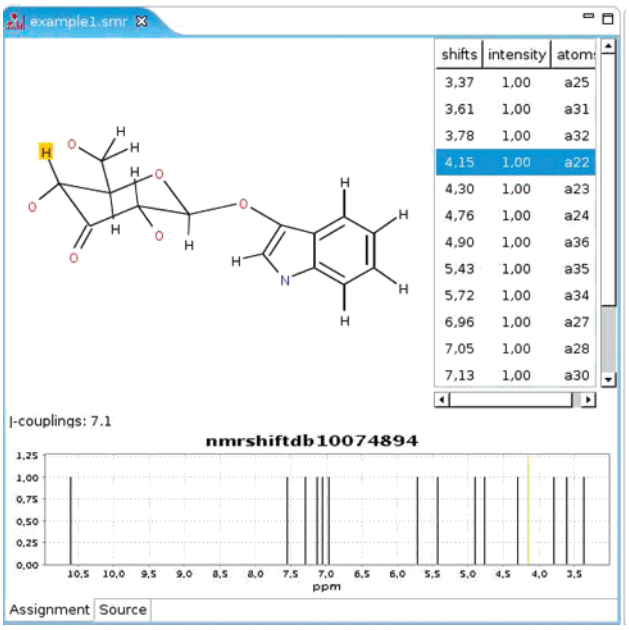


Figure 2. This shows Chart 3 opened in the SpecMol editor of Bioclipse. The structure and the spectrum are shown graphically; the assignments of peaks are given and highlighed if a peak is selected. For this peak, also the coupling is shown.

• AnIML is designed for instrument manufacturers to install and maintain. CML users might expect to consume AnIML and translate parts of it into CMLSpect.

• CMLSpect maps onto the discourse of chemical publishing and is sufficiently lightweight that it is accessible to (say) under- and postgraduates.

• AnIML is geared to supporting the reporting of validated information in regulatory processes and contains much metadata and authentication material that CML does not.

• CML has an intimate relationship between spectra and molecular structure and has been designed for annotation. AnIML has no built-in chemical structure language in XML, and users may wish to use CML with AnIML for this purpose

We would hope that the vocabularies defined by CMLSpect and by AnIML would become widely distributed and would jointly lead to a greater semantic formalism of chemical spectroscopy.

5.3. ANDI.[24] The Analytical Data Interchange format developed by the Analytical Instrumentation Association (AIA) is a standardized data interchange format mainly for mass spectrometry and chromatography, that not only just focuses on the easy exchange of analytical data between instruments and applications but also tries to realize this while maintaining the GLP (Good Laboratory Practice) and GMP (Good Manufacturing Practice) integrity of the data. Because of this an ASCII format like JCAMP-DX was decided to be too insecure, and a binary format which could not be manipulated was needed. For sustaining the portability of the data between different operating systems and software platforms Unidata's netCDF format was chosen for binary encoding. This format is supported by free, public domain software and source code which can be compiled for cross-platform compatibility. To properly design the formats for specific analytical disciplines the AIA organized a number of committees formed by instrument vendors and users. These focused on the definition and creation of dictionaries defining as many terms as possible to ensure a completely GLP-compliant way of storing the data. Because this results in extremely large dictionaries, simplification was needed and achieved by the definition of five levels of compliance. In order to be compliant with a certain level of completeness, a file must include all the tags defined for that level and the prior levels as well. The project was turned over to the American Society of Testing and Materials (ASTM) in the mid 1990s and is now maintained by the ASTM E13.15 subcommittee.

5.4. Galactic SPC.[25] SPC is the file format used in all Galactic and Thermo Galactic products as exchange and storage format, respectively. Since its invention and inception, the format's description was published in Galactic's documentation and via other public domain sources. As Galactic is one of the major OEM suppliers of instrument sofware, their software is used in many spectroscopic instruments resulting in a large amount of data being available in this format. This resulted in a lot of other software packages supporting export to SPC files. The format was designed to meet the needs of a user who wants to view, process, and print the data outside the instrument vendor's software but is not that well suited for archiving the data. It uses a binary encoding but is still flexible enough to hold many different types of data. In addition to that it has a flexible mechanism for the storage of parameter information in unformated ASCII named "Log Text" for carrying along important information from the source instrument file format. Another problem is, that because of its "flat-file'ness" it is very difficult to extend the file format with the evolution of the spectroscopic methods and instruments.

5.5. Comparison to CMLSpect. As we have seen some of the older formats carry disadvantages due to their history, whereas the newer formats are often restricted to particular types of data. CMLSpect is able to overcome these:

• CML can hold spectral and structural data and relationships between them in a uniform way.

• Close integration of spectra and structures is possible, e.g., for assignments (difficult if embedding CML structures into another XML-based format).

• Being XML it can be embedded in other XML documents and can be processed as XML. For example, CML can be transformed to SVG via an XSL Transformation.

• As opposed to flat files, XML and therefore CML can be easily extended with new elements and/or attributes without disturbing processing by existing programs

• With XML, parsers error checking and validation is easy (and can be supplemented by conventions, schematron, etc.).

6. CHANGES TO OVERALL ARCHITECTURE OF CML

This section shows changes and developments of the overall CML architecture since the last CML paper[8] was published. They are not specific to CMLSpect but influence all of the CML components.

CML is evolving to an increasingly unconstrained system, and there are now no absolutely required CMLElements in any document. To distinguish between different XML vocabularies (languages, schemas) it is now almost universal to add namespaces. In the past we suggested modular namespaces for the different subdomains of CML [see ref 6, p 758], but these have proved unworkable. Chemistry does not separate into simple independent subsets, and the

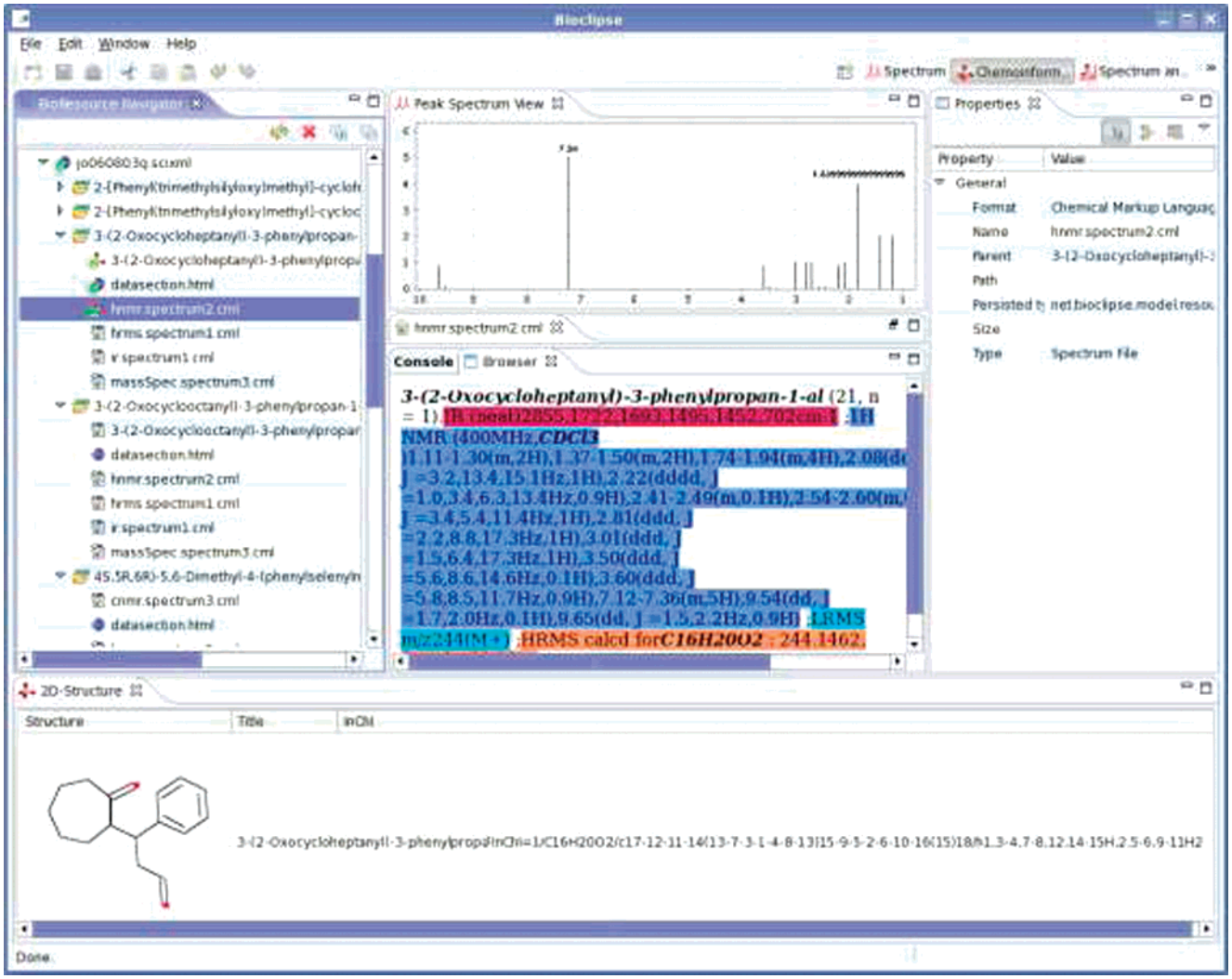


Figure 3. Bioclipse showing the chemical content from a paper in *J. Org. Chem.*[20] The information is extracted from the Experimental Section of 3-(2-oxocycloheptanyl)-3-phenylpropan1-al highlighted along with a colored version of the original text.

technology for managing these was unnecessarily complex and effectively unworkable. The previous terms ("CMLCore", "CMLReact", "CMLComp") and now "CMLSpect" are therefore not formal collections of CMLElements in different namespaces but flexible and fuzzy concepts to describe the particular subdisciplines being supported. It is, for example, likely that <molecule>, originally part of "CMLCore" will frequently co-occur with <spectrum>. We also attempted to provide versioning through namespaces (e.g., http://www.xml-cml.org/cml2), but this also became impossible as every time a new version is released all existing software has to be immediately updated (as namespaces are used to recognize vocabularies). We have therefore decided that all CMLElements share a single immutable namespace: http://www.xml-cml.org/schema. At present we intend that CML semantics and syntax are backwardly compatible, and to support this we are introducing a version attribute. Note that before namespaces were widely used, file extensions were used to distingish XML dialects. Since namespaces do (or at least should do) this now, CML files should have a.xml file extension (.cml is deprecated). If a mime-type should be used (e.g., for a browser download), the mime-type "chemical/x-cml" is recommended.

CML is designed to interoperate with other namespaces, and common W3C examples for this are XHTML, SVG, and MathML. We also expect that from chemistry and science ThermoML and AnIML (and perhaps UnitsML and MatML) will commonly co-occur. For example it might make sense for a CMLSpect element to contain AnIML elements or vice versa. It will be common to have examples such as

```
<cml:spectrum xmlns:cml="http://www.xml-cml.org/schema">
  <xhtml:p xmlns:xhtml="http://www.w3.org/1999/XHTML">This spectrum
    was originally only available in image form
    <xhtml:img src="spectrum.gif"/>, but we now report the peaks in
    CMLSpect format</xhtml:p>
  <cml:peakList> </cml:peakList>
</cml:spectrum>
```

For convenience in this paper we shall omit prefixes for CML by taking the default namespace to be http://www.xml-cml.org/schema, but it is essential to remember that the CML namespace should always occur in any document containing CML.

CML now contains over 100 CMLElements and over 200 attributes. It is also now not mandatory to use a full CMLSchema (in fact we expect that only CML developers are likely to do this). Users can pick and choose what

CMLElements they use for their chemical domain. Attributes and their types give few problems in creating a subset, but formal content models are more problematic. However, we find that CML content models in XSD are increasingly becoming obsolete. It is, for example, impossible to say what must, may, and must not be contained within a <spectrum>. Any of the following might make sense: <sample>, <molecule>, <svg:svg>, <xhtml:div>. Alternatively there may be no content as in the following: <h:div>. We measured the <spectrum type="IR"/> of benzene</h:div> which represents semantic markup of the running text "We measured the infrared spectrum of benzene".

It is therefore possible for a user to select those elements required for their problem. Since we do not intend to continue to use XSD content validation, the problem of a variable set of CMLElements disappears. Instead we have developed more flexible and powerful rule-based approaches using Schematron (http://www.schematron.com/). This technology allows users to generate their own content rules such as "A spectrum must have either a sample child or a molecule child but not both", which cannot be easily expressed in XML Schema but is easy in Schematron:

```
<rule id="spectrum" context="cml:spectrum"
  xmlns:cml="http://www.xml-cml.org/schema">
  <assert id="spectrum.content"
    test="count(cml:molecule) + count(cml:sample) = 1"/>
</rule>
```

We shall continue to use XML Schema as the apparatus for the definition of CML and to include such rules in the <u1:appinfo> of the definition. The "schema", however, consists of a series of individual <u1: element>s from which a user can pick those required.

In our JUMBO program it is now routine to autogenerate code for all CMLElements. This is inspired by the IETF's successful mantra "rough consensus and running code". Thus <peakList> (which can have (say) children <peak>, <peakGroup>, etc. and attributes title, id) would generate a code such as

```
List<CMLPeak> CMLPeakList.getPeakElements();
CMLPeakList.addPeakGroup(CMLPeakGroup);
CMLPeakList.setTitle(String)
String CMLPeakList.getId();
```

At present this is only done routinely in the Java language, but we have prototyped it in C++, Python, and FORTRAN. Consequently all new and existing CMLElements have been deployed in procedural code and have been used to validate examples relating to their attributes and content. This means that many design inconsistencies and infelicities have been eliminated. It also tends to keep the language simple; if a proposed feature is too difficult for the authors to write code easily, then they will think if the complexity is necessary.

Many CMLElements (including several in CMLSpect) have functionality which cannot be fully expressed in accessors and mutators (get, set, add, remove, etc.), and code for these CMLElements has been handcrafted.

6.1. Problems with Micro- and Macroscopic Views. An emerging challenge in CML and the formalization of semantic chemistry is the difference between the microscopic and macroscopic. In some disciplines, such as synthetic organic chemistry, the microscopic (represented by a connection table and 3 D structure) maps well to the macroscopic (pure compound in a bottle). CMLCore was based on this simplicity and has, in general, worked well. But many substances can only be explained in terms of an ensemble of components (typical problems being tautomerism, conformational variability, impurities, racemates, extended solids, defects, etc.). Many of these concepts are essential in understanding spectra (e.g., the equivalence of protons in the NMR spectrum of a compound may be due to rapid conformational interchange.) It can be very difficult to describe these with a single connection table.

In addition there can be several levels of understanding in describing spectra. Beyond the purely phenomenological, spectra are used to interpret the properties of molecules, and each spectrum has its own domain of discourse. Thus an organic chemist might say "This peak is due to the C=O vibration", while a molecular spectroscopic might say "This peak is the envelope of transitions between the ground vibrational state and the first excited level with symmetry A2. The normal mode associated with this excited level contains large displacement vectors for the atoms C and O". Similarly the *coupling constant* measured as the interpeak distance in a spectrum is not the coupling constant calculated from the Bloch equations for a molecule; the latter constituting instead a *model* underlying the observable spectrum rather than necessarily describing the measured spectrum itself. In general spectroscopists and chemists use many different ontologies which may be fundamentally irreconcilable. We therefore do not prescribe a vocabulary for annotating spectra but provide places where annotation can be precisely added.

ACKNOWLEDGMENT

We greatly appreciate financial support from the DAAD (Deutscher akademischer Austauschdienst, German Academic Exchange Service) for the Cologne−Cambridge cooperation.

Supporting Information Available: Files example0.cml to example4.cml (full examples of section 3) viewable in any text or XML viewer [more specific application, recommendation−use Bioclipse (http://www.bioclipse.net/) or JSpecView (http://jspecview.sourceforge.net/) for viewing]; example4.cml [renamed example4.smr for viewing in Bioclipse (limitation of Bioclipse, not connected to cml), example not viewable in JSpecView]; and other examples viewable in JSpecView when renamed to .cml: dictionaries in the dict subdirectory referred to by these files (serve as examples of dictionaries), schema.xsd (CML schema in version 2.5, referenced by the example files), nmrshiftdb-convention.schematron (example for a convention defined via Schematron), and spect.xml and spectsubst.xml (examples for dictionaries, explanation in subsection 2.2). This material is available free of charge via the Internet at http://pubs.acs.org.

REFERENCES AND NOTES

(1) Berners-Lee, T.; Hendler, J.; Lassila, O. The Semantic Web A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. *Sci. Am.* **2001**, *284*.

(2) Hey, T.; Trefethen, A. E. UK e-Science Programme: Next generation grid applications. *Int. J. High Perform. Comput. Appl.* **2004**, *18*, 285−291.
(3) Murray-Rust, P.; Rzepa, H. Chemical Markup XML, and the Worldwide Web. 1. Basic Principles. *J. Chem. Inf. Comput. Sci.* **1999**, *39*, 928−942.
(4) Murray-Rust, P.; Rzepa, H. Chemical Markup XML, and the Worldwide Web. 2. Information Objects and the CMLDOM. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1113−1123.
(5) Gkoutos, G. V.; Murray-Rust, P.; Rzepa, H.; Wright, M. Chemical markup, XML, and the World-Wide Web. 3. Toward a signed semantic chemical web of trust. *J. Chem. Inf. Comput. Sci.* **2001**, *41*, 1124−1130.
(6) Murray-Rust, P.; Rzepa, H. Chemical Markup XML, and the Worldwide Web. 4. CML Schema. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 757−772.
(7) Murray-Rust, P.; Rzepa, H.; Williamson, M.; Willighagen, E. Chemical Markup, XML, and the World Wide Web. 5. Applications of Chemical Metadata in RSS Aggregators. *J. Chem. Inf. Comput. Sci.* **2004**, *44*, 462−469.
(8) Holliday, G. L.; Murray-Rust, P.; Rzepa, H. S. Chemical Markup XML, and the Worldwide Web. 6. CML React, an XML Vocabulary for Chemical Reactions. *J. Chem. Inf. Comput. Sci.* **2006**, *46*, 145−157.
(9) Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Main_Page/ (accessed April 22, 2007).
(10) Murray-Rust, P.; Rzepa, H. S. Scientific publications in XML towards a global knowledge base. *Data Sci. J.* **2002**, *1*, 84−98.
(11) Guha, R.; Howard, M. T.; Hutchison, G. R.; Murray-Rust, P.; Rzepa, H.; Steinbeck, C.; Wegner, J.; Willighagen, E. L. The Blue Obelisk-interoperability in chemical informatics. *J. Chem. Inf. Model.* **2006**, *46*, 991−8.
(12) Zhang, Y.; Murray-Rust, P.; Dove, M.; Glen, R.; Rzepa, H.; Townsend, J.; Tyrell, S.; Wakelin, J.; Willighagen, E. JUMBO An XML infrastructure for eScience. *Proc. UK e-Science All Hands Meeting 2004* **2004**, 930−933.
(13) Zhang, X.; Xu, Q.; Xiao, H.; Liang, X. Iridoid glucosides from Strychnos nux-vomica. *Phytochemistry* **2003**, *64*, 1341−1344.
(14) Steinbeck, C.; Kuhn, S.; Krause, S. NMRShiftDB Constructing a Chemical Information System with Open Source Components. *J. Chem. Inf. Comput. Sci.* **2003**, *43*, 1733−1739.
(15) Steinbeck, C.; Kuhn, S. NMRShiftDB Compound identification and structure elucidation support through a free community-build web database. *Phytochemistry* **2004**, *65*, 2711−2717.
(16) Spjuth, O.; Helmus, T.; Willighagen, E. L.; Kuhn, S.; Eklund, M.; Wagener, J.; Murray-Rust, P.; Steinbeck, C.; Wikberg, J. E. S. Bioclipse: an open source workbench for chemoand bioinformatics. *BMC Bioinformatics* **2007**, *8*, 59.
(17) Lancashire, R. JSpecView Applet Specification. http://jspecview.sourceforge.net/ (accessed April 22, 2007).
(18) SPECTRa Submission, Preservation and Exposure of Chemistry Teaching and Research Data A Digital Repository for the Chemical Community. http://www.lib.cam.ac.uk/spectra/index.html (accessed April 22, 2007).
(19) JCAMP-DX SourceForge Project Page. http://sf.net/projects/jcamp-dx (accessed April 22, 2007).
(20) Kagawa, N.; Ihara, M.; Toyota, M. Convergent total synthesis of (+)-mycalamide a. *J. Org. Chem.* **2006**, *71*, 6796−6805.
(21) IUPAC CPEP Subcommittee on Electronic Data Standards. http://www.iupac.org/standing/cpep/wp_jcamp_dx.html (accessed April 22, 2007).
(22) An XML-Based File Format for Archival Storage of Analytical Instrument Data. http://www.gaml.org/Documentation/XML%20Analytical%20Archive%20Format.pdf (accessed April 22, 2007).
(23) AniML. http://animl.sourceforge.net/ (accessed July 22, 2006).
(24) SourceForge.net: Analytical Data Interchange. http://sourceforge.net/projects/andi (accessed April 22, 2007.
(25) Thermo Electron Corporation SPC File Format. http://www.thermo.com/com/cda/resources/resources_detail/1,2166,112125,00.html-?fromPag (accessed April 22, 2007).

CI600531A