

Representing, indexing and mining scientific data with XML and RDF: Golem and CrystalEye

Andrew D. Walkingshaw[1,2], Toby O. H. White[2], Nick Day[1], Jim Downing[1], Peter Murray-Rust[1]

[1] Unilever Centre for Molecular Science Informatics, Chemical Laboratory, University of Cambridge, Lensfield Road, Cambridge, CB2 1EW

[2] Department of Earth Sciences, University of Cambridge, Downing Street, Cambridge, CB2 3EQ

andrew@lexical.org.uk

Introduction

Overview

We briefly introduce chemical informatics, and some of the technical and social issues we have to consider in the design of chemical data representations and chemical information systems. We then introduce

some of our work in this area; using the Chemical Markup Language and our Golem toolkit, we have been automatically constructing ontologies from corpora of CML documents. Following from this, we have been using RDF and SPARQL in collecting, querying and mining data obtained from two sources: atomistic simulation of molecules and materials and, secondly, from experimental data published by journals in crystallography. We then show some of the new questions this lets us answer; in particular, how the global distribution of work in crystallography has changed between 2000 and 2007.

Chemical informatics

Chemical informatics is a young field. In a sense, one of the stories of the 20th century in science has been the subdivision of researchers into communities of practice; first the split between theoreticians and experimentalists, then the computer-driven rise of scientists working primarily with simulation. However, an even more profound transformation has been made possible by computer technology; scientists can produce much more data than was ever feasible beforehand, through mechanisation, robotics and digital sensors enabling automated high-throughput experimental procedures, and through ubiquitous high-performance computing letting us perform more, and bigger, simulations.

In the light of that, a new community of practice has sprung up, dealing with the huge volumes of data produced by all of the rest of science; scientific informatics. The goal of scientific informatics is to enable the integration of, and extraction of knowledge from, the many disparate data sources which are now available to us.

Using scientific data

The first question we have to ask ourselves, as informaticists building tools for working scientists to use, is what chemists might want to do with this data. Mostly that's going to be finding, or validating, patterns in it – confirming or developing hypotheses. Abstractly, you can look at this as a kind of visualization problem; transforming data

from its native form into one where any patterns become more apparent.

In his recent book [FRY07], Ben Fry, an artist and programmer, wrote of the seven stages of visualization: he identified them as

- acquisition of data
- parsing
- filtering
- mining
- finding a good representation - model - for the data,
- refining that model, and
- adding interactivity.

This is a good model for how we approach problems in chemical informatics. First, we acquire data, by performing experiments, running simulations, and mining the literature; then we parse the data and transform it into a common format, filter out the subset of the data pertinent to the problems we are interested in, and then start building models by a combination of theoretical considerations and statistical/data-mining techniques. We then iterate through this process, refining our understanding; finally, we present our conclusions.

Tools and protocols

This flags up the key areas where chemical informaticists borrow other fields' approaches; in particular, we learn from data and knowledge representation, from data-mining, and from the Web and Web standards. We can reuse a lot of the standards developed by the Web community, in particular; the Web architecture is a natural fit for the kind of data-driven science we're trying to do. As such, we have HTTP and REST as a way of delivering services, and HTML, CSS and Javascript as a cross-platform user interface format: the rise of

lightweight web frameworks like Django (<http://djangoproject.org/>) or Ruby on Rails (<http://rubyonrails.org/>) has been a significant factor in making it easier to deploy software over the Web. Furthermore, when representing our data, we have XML document formats; and, especially, we have RDF and the SPARQL query language as the beginnings of a distributed database.

The move to reusing open structured data formats in chemical informatics is not new. Chemical Markup Language (CML) [MUR99] was one of the first XML languages to formally adopt a DTD. However, CML, itself, was following in the footsteps of the earlier structured document formats adopted in crystallography in the early 90s, particularly CIF (the Crystallographic Information File format) [HAL91].

There are extensive libraries available to chemical informaticists these days. Many of these, like the Chemistry Development Kit [STE03], are directly designed for the manipulation of datasets focussing on small, often drug-like, molecules; this particular subfield of chemical informatics is very well-developed and industrially highly significant. Much of the software being developed, however, is usable in attacking the more general problem of how to design systems for storing and indexing chemical data – the OSCAR3 chemical named entity recogniser [COR07] being a prominent example of this. We also benefit from the scientifically-focussed general-purpose software and libraries which have been developed, such as Numpy (<http://scipy.numpy.org/>), a fast array/matrix library, and Scipy (<http://scipy.org/>), an algorithmic toolbox built on it.

In general, control of the representation of our data, and statistical approaches to assigning meaning to that data, are the two most powerful general approaches we can make use of in designing chemical information systems; both of these are well-served by the tools we now have.

Our goals

Writing on his weblog, Tim O'Reilly reported an interview with Reuters' CEO from the MoneyTech conference

<http://radar.oreilly.com/archives/2008/02/reuters-ceo-sees-semantic-web.html>):

... Reuters' news is the raw material for analysis and application by investors and downstream news organizations. Adding metadata to make that job of analysis easier for those building additional value on top of your product is a really interesting way to view the publishing opportunity. If you don't think of what you produce as the 'final product' but rather as a step in an information pipeline, what do you do differently to add value for downstream consumers? In Reuters' case, Devin thinks you add hooks to make your information more programmable.

In this sense, our task, when designing systems to store our data, is to make that data easier to program against – easier to consume. However, we also have to have data to aggregate in the first place, so we also need to make it easy to author; furthermore, whatever we do has to fit in with how scientists work now. If our systems are too unfamiliar, they won't be adopted.

Sources of data

Scientists can glean data from many different sources. The obvious route is through carrying out experiments and simulations themselves, or having them carried out on their behalf; this, in itself, can now produce enough data to tax our ability to cope with it. Additionally, however, we can extract data from databases or from the literature. In this paper, we consider examples of both of these sources; data from atomistic simulations of molecules and materials, and published supplemental data from crystallography journals.

Simulations

FoX and CML

Many computational chemistry programs have a heritage dating back to the 1970s, and as a result, they are almost always written in

Fortran. This makes a lot of sense: Fortran is fast, has exceptional numerical libraries and the atomistic simulation community has a lot of experience with it. However, there are problems; Fortran's I/O facilities are very restricted – outputting much more than plain text is hard work – and many of these programs have evolved their (often idiosyncratic) output formats over time, according to whatever the needs of the developer of the time were.

Of course, we would prefer them to have structured and semantically-rich output, but in order to achieve that, we have to work with these legacy codebases; we have to make it easy to add structured output in a way that Fortran programmers find intuitive.

In that spirit, we can use the FoX library; FoX [FOX, WHI06] is the first fully-validating XML parser and writer to be written in Fortran 95. Therefore, it can be used directly with these programs, rather than having to either post-process existing output or use compiler-specific foreign-function interfaces. Furthermore, it ships with a library, WCML, which allows developers to output a subset of the Chemical Markup Language, informally referred to as CMLComp (<http://cmlcomp.org/>), simply by making the appropriate function calls, such as:

```
call cmlAddProperty(xmlfile, 200, dictRef="castep:Etot", &
                    units="castepunits:ev")
```

Using this library, authors don't need to have any knowledge of XML to generate output in this format: they just need to know what library calls to make, which is much easier to explain to developers who may have no prior knowledge of XML.

Golem

However useful the ability to output structured data is, we still need to make it easier to process. The approach we take to this is best shown by example. The code above produces the following markup:

```
<property dictRef="castep:Etot">
  <scalar units="castepunits:ev" dataType="fpx:real">200
</scalar>
```

</property>

Here, the CML “dictRef” attribute is a reference to a CML dictionary entry; CML dictionary entries give documentation on what a specific CML element is about. Here, the markup tells us we have a property, which is a real number with a specific value and units, but not what it means; that comes from the dictionary entry. (As such, the “dictRef” attribute here has something in common with the “property” attribute in the RDFa specification [RDFa].)

Here is the relevant entry from the CASTEP dictionary:

```
<entry id="Etot" term="Total energy">
  <annotation />
  <definition>Total energy of the
    calculation.</definition>
  <description>
    <h:div class="dictDescription">
      It should be noted that total energies in DFT
      calculations are relative rather than absolute...
    </h:div>
  </description>
  <metadataList>
    <metadata name="dc:author" content="golem-kiln" />
  </metadataList>
  <golem:xpath>
    /cml:cml/cml:propertyList[@id="finalProperties"]/cml
    :property[@dictRef="castep:Etot"]
  </golem:xpath>
  <golem:template call="scalar" role="getvalue"
  binding="pygolem_serialization" />
  <golem:implements>value</golem:implements>
  <golem:implements>absolute</golem:implements>
  <golem:childOf>id_finalProperties</golem:childOf>
</entry>
```

As can be seen, the “definition” and “description” elements provide human-readable semantics here; however, we need machine-readable information as well. Those are supplied by our Golem language and toolkit [GOLEM]. The Golem language annotates CML dictionary entries, giving the location where concepts occur in the document (via XPath, via <golem:xpath>), relationships between concepts in the dictionary (<golem:childOf>), optionally bounds and type information, and transformations of the object via XSLT into a JSON serialization (<golem:template>, through the “call” attribute) – the above markup would be transformed into:

```
[200, "castepunits:ev"]
```

These annotations are interpreted, and transformations performed by the Golem toolkit, implemented in Python; the CML processing in this paper is performed by programs written using this library.

This leaves the problem of creating the dictionary, but much of that can be automated. Taking a corpus of output files from a given code, one can search it for all the dictRefs that appear; constructing an XPath for each instance by simply backtracking from the current node, such that

```
<a>
  <b>
    <c dictRef="target" />
  </b>
</a>
```

would give the XPath

```
/a/b/c[@dictRef="target"]
```

we can deduce a location for every instance of each dictRef in our corpus. Using that, we can find the longest common XPath in the corpus for instances of each particular dictRef; in other words, the most specific location for each concept in the set of documents we are analysing. Furthermore, as the output comes from the WCML library, there is a limited library of possible serializations (somewhat like microformats); assuming our corpus-analyser can recognise these serializations, the appropriate transform can be assigned to the

dictionary entry automatically. Therefore, we can infer the value for the <golem:xpath> and <golem:template> elements; a tool to do this is supplied with Golem.

Therefore, by using FoX and Golem, machine-readable semantics can be easily attached to the output of simulation programs, without any specific XML knowledge being needed by the simulation program's developers.

Journals

Another source which scientists might wish to mine is the huge amount of data published every year by journals. We will focus on crystallography – the analysis and measurement of the positions of atoms in molecules and crystals. Crystallography is actually one of the most rigorous areas of science in terms of shared data formats and semantics; since the early 1990s, a structured-text output format called the Crystallographic Information File, developed under the aegis of the International Union of Crystallography, has been in wide usage. Crystal structures are typically submitted in this format for publication in journals; this raw data is then published as supplemental data alongside the paper, and is typically free to download. Furthermore, facts aren't copyrightable: these files are sets of facts, so we can aggregate them.

This gives rise to CrystalEye.

CrystalEye

CrystalEye [CRYSTALEYE] is an aggregator for published crystallographic data; it spiders content from publishers' websites, converts it from CIF to CML, and then processes the CML to add a rich Web interface to the data; it also calculates various aggregate statistics over the dataset (such as, for instance, distributions of bond lengths). It also provides a chemical search interface, allowing researchers to search for examples of parts of molecules (chemical substructure search). This, in itself, adds quite a lot to the raw data.

[OPEN DATA](#)

[<< Table of Contents](#)

Publisher: Acta Crystallographica
Journal: Section A
Year/Issue: 2006/03-00

Article (via DOI): [10.1107/S0108767306010336](https://doi.org/10.1107/S0108767306010336)
Compound Class: organic
Date Recorded: 2006-01-13

Contact Author:
e-mail:

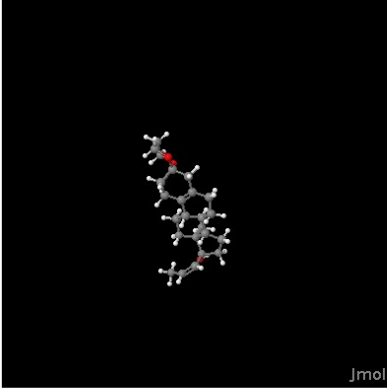
Data collection parameters

Chemical formula sum	C ₂₄ H ₃₆ O ₃
Chemical formula moiety	
Crystal system	Orthorhombic
Space group H-M	P2(1)2(1)2(1)
Space group Hall	
Data collection temperature	100.0

Refinement results

R Factor (Obs)	0.0381
R Factor (All)	0.0497
Weighted R Factor (Obs)	
Weighted R Factor (All)	0.0363

Available Resources



Show no. of unit cells along axis:

a:
b:
c:

Enter Jmol script:

```
load./cn5008supl_I.complete.cml.xml
```

crystaleye.png

Figure 1. A screenshot of CrystalEye, with an embedded Jmol [JMOL] applet.

It is, however, very focussed on the structures; for instance, it doesn't do anything with the bibliographic data embedded in the CIF file, other than converting it to CML – who wrote a paper, what their institution was, when they wrote it. However, as all the underlying data is in CML, one can use the tools introduced earlier to infer a CML/Golem dictionary for the dialect of CML which CrystalEye produces, enabling us to write tools to further mine the data aggregated by CrystalEye – including converting parts of it, such as the bibliographic data, to RDF, the underlying data model of the Semantic Web [RDF].

RDF

Automatically generating RDF from CrystalEye

CrystalEye exposes the data it aggregates as an Atom [RFC4287] feed using the Feed Paging and Archiving extension [RFC5005], so when programming against it, this is an obvious place to start. However, we need to specify a transform from CML to RDF in the context of dictionary entries. Here is some markup from CrystalEye – a list of author names. By analysing the corpus, we have determined that this dictionary reference occurs only once per file, and always at the same XPath. It is, therefore, unambiguous in its meaning: it does not depend on the surrounding elements for context.

```
<array dataType="xsd:string"
  dictRef="iucr:_publ_author_name" delimiter="|">
  Volker Bohmer|Michael Bolte|Crenguta Danila|
</array>
```

Now, the “iucr” namespace is associated in the CML dictionary with a namespace URI; thus, we can choose to construct an RDF predicate like

```
iucr:_publ_author_name
```

for the data contained by this element. We can then use the previously associated XSLT transformation to contain the element's data to JSON, and using that as an RDF literal, assign it as the object of the triple. In NTriples, the result is:

```
<document> iucr:_publ_author_name "[["Volker Bohmer",
"Michael Bolte", "Crenguta Danila"],
"units:undefined"]"^^<http://example.org/json>
```

Again, here, we have invented an URI to denote the type of the objects in our triples as JSON. Admittedly, this is not exactly the acme of linked data – but it is valid RDF, which we can start to query using

SPARQL. What's more, we can do this for all the concepts in our dictionary which are unambiguous by the above rules, giving us an automatic approach to converting much of the CrystalEye database into RDF. The result is that data becomes much, much more searchable – and thus far we've not tuned any of our tools to the data we're manipulating.

We can then move on to enhancing our RDF – by converting data to commonly used ontologies, for example – and exploiting it by building applications which give us new ways of browsing the data. Firstly, as we are dealing with bibliographic metadata, we can add Dublin Core [RFC2413] terms: for instance, the above becomes, again as NTriples,

```
<document> dc:contributor "Volker Bohmer" .  
<document> dc:contributor "Michael Bolte" .  
<document> dc:contributor "Crenguta Danila" .
```

We can now query CrystalEye to find all the papers written by one of these authors. The SPARQL [SPARQL] for that is:

```
PREFIX dc: <http://purl.org/dc/terms/>  
DESCRIBE ?file WHERE {  
  ?file dc:contributor "Crenguta Danila" .  
}
```

The next stage is to put the RDF on the web behind a SPARQL endpoint; we used a Talis platform store (<http://n2.talis.com/>). Adding a web interface onto this, using the store as a back-end, we can easily make a browsable author index for CrystalEye. We used Django with rdflib (<http://rdflib.org/>), querying our platform store by SPARQL over HTTP; an example of the results can be seen at <http://ceview.appspot.com/author/Jones,%20William/>.

We can make our index expose the underlying RDF in several methods; by content negotiation (application/rdf+xml, text/rdf+n3) and as RDFa (<http://triplr.org/ntriples/ceview.appspot.com/author/Jones,%20William/>)

Additionally, we can now answer questions like “Who are a given author's coauthors?”; the query for that is:

```
PREFIX dc: <http://purl.org/dc/terms/>
SELECT DISTINCT ?coauthor WHERE {
  ?file dc:contributor '"Crenguta Danila"' .
  ?file dc:contributor ?coauthor .
  FILTER (?coauthor != '"Crenguta Danila"') .
}
```

So we can now make reasonably sophisticated bibliometric queries of CrystalEye – in other words, by interrogating the data which underlies the published articles in crystallography journals rather than interrogating the journals themselves. However, it would be good if we could query CrystalEye by scientific topic; to get that, we need to add more triples to our store.

OSCAR and natural-language processing

OSCAR 3 is a chemical named entity recogniser; given free text, it extracts the names of chemicals and chemical techniques and other pieces of scientific terminology. As such, we can use this to annotate resources according to which named entities they refer to.

The CrystalEye RDF includes triples like:

```
<document> iucr:_publ_section_title '"1,3-Alternate
calix[4]arenes, selectively functionalized by amino
groups"'
```

which contain the papers' titles. Here, “amino groups” and “calix[4]arenes” are likely to be key terms, for instance; so by passing the titles through OSCAR, parsing OSCAR's annotations, and nominating a namespace for those annotations, we can create the following triples:

```
<document> iucr:oscarAnnotation
<http://wmm.ch.cam.ac.uk/crystaleye/ontology/CM/amino>
```

```
<document> iucr:oscarAnnotation
<http://wwmm.ch.cam.ac.uk/crystaleye/ontology/CM/calix%5B4
%5Darenes>

<document> iucr:oscarAnnotation
<http://wwmm.ch.cam.ac.uk/crystaleye/ontology/ONT/amino>

<document> iucr:oscarAnnotation
<http://wwmm.ch.cam.ac.uk/crystaleye/ontology/ONT/amino%20
groups>

<document> iucr:oscarAnnotation
<http://wwmm.ch.cam.ac.uk/crystaleye/ontology/ONT/groups>
```

This gives us a method of automatically annotating CrystalEye documents with, effectively, machine-generated tags related to their content. We can then exploit these in our browsing interface, producing the following:

CrystalEye query for: Jones, William

Get this page as: [Google Earth/Maps format](#)

[1,4-Benzenedimethanol \(raw CML\)](#)

published in *Acta Crystallographica, Section E* on 2001-11-20

Authors:

- [Jones, William](#)
- [Shan, Ning](#)

Chemical named entities

- [1,4-Benzenedimethanol](#)
-

[1-Hydroxy-2\(1H\)-pyridinethione \(raw CML\)](#)

published in *Acta Crystallographica, Section C* on 1999-05-14

Authors:

- [Bond, Andrew](#)
- [Jones, William](#)

Chemical named entities

- [1-Hydroxy-2\(1H\)-pyridinethione](#)
-

[2-\(p-Nitrophenoxy\)tetrahydropyran \(raw CML\)](#)

published in *Acta Crystallographica, Section E* on 2004-09-23

[crystaleye-browser.png](#)

Figure 2: a faceted browser for CrystalEye

Thus, we have managed to produce a browsable index of CrystalEye by author and by key chemical term, without any hand-curation of data; and as the underlying representation is RDF and can be queried using SPARQL, it is much easier for someone else to come along later and mine the data in ways which we haven't foreseen.

Geocoding and the global distribution of crystallography

Finally, we move on to an example of one of these richer queries; how has the geographic distribution of publications in crystallography changed since the start of the decade?

Our data includes the institutional affiliations of the authors of papers, and also the date of publication: by geocoding the names of institutions and cities therein against the GeoNames dataset (<http://geonames.org/>), we can get approximate latitudes and longitudes (to the resolution of a city, more or less) for around 30,000 papers (a little under half our dataset). Storing these results in our triple store as before, we can therefore get (as far as possible) the locations of the authors of papers published in a given time interval:

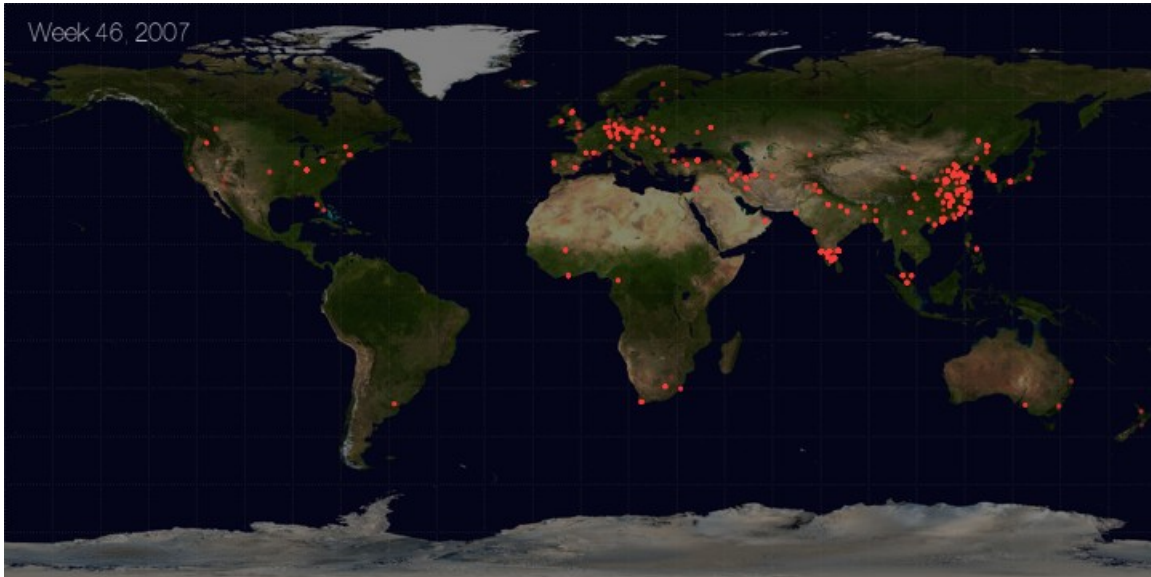
```
PREFIX ce:
<http://wmm.ch.cam.ac.uk/crystaleye/dictionary#>
PREFIX fn: <http://www.w3.org/2005/xpath-functions>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT DISTINCT ?place ?lat ?lon ?date
WHERE {?url ce:AcceptanceDate ?date .
      ?url ce:location ?loc .
      ?loc ce:address ?place .
      ?loc ce:latitude ?lat .
      ?loc ce:longitude ?lon .
      FILTER (xsd:date(?date) >= xsd:date("START")) .
      FILTER (xsd:date(?date) < xsd:date("END")) .
}

ORDER BY ?date
```

This gives us an approach to visualizing the answer to our question; script this query to obtain the locations of papers published in fortnightly intervals, and then present the results on a map. We chose

to map the results using a sketch written in the Processing environment [PROCESSING]; as a result, we have made a video of the locations of crystallography papers between 2000 and 2007, which can be seen at <http://vimeo.com/846349>.



screen-3515.tif

Figure 3: The distribution of authors in crystallographic journals in late 2007

Conclusions

In conclusion, visualizations like this go to demonstrate the connections between the choice of data representation and the ability to visualize data; in a way, that is the underpinning principle of the work that we are presenting. In particular, SPARQL opens up the possibility for people to reuse data in new ways which neither the original author nor the aggregator have foreseen; we can hope that the rise of Linked Open Data continues to expand the range of resources available to us.

Finally, predicting the future of scientific data capture and scientific publishing is almost impossible, but it is becoming clearer that the pressing issues are primarily cultural and economic; we have all the technology we need to make very rich Web-centric datasets available. Where we go from here is up to us all.

Acknowledgements

Andrew Walkingshaw would like to acknowledge the advice of his colleagues Prof. Martin Dove, Dr Dan Wilson, Dr Joe Townsend, Dr Nico Adams, Dr Lezan Hawizy and Dr Peter Corbett. The present work depends on access to the Talis Platform beta, kindly provided by Talis Information Ltd (<http://www.talis.com/>). Andrew is part of the MaterialsGrid project (<http://www.materialsgrid.org/>), a project of the Technology Strategy Board in partnership with Accelrys and IBM.

CrystalEye is supported by the International Union of Crystallography and the Royal Society of Chemistry; the development of OSCAR is supported by the Royal Society of Chemistry and Nature Publishing Group.

Bibliography

- [FRY07] Fry B, Visualizing Data, O'Reilly Media, Sebastopol, CA, United States of America, 2007
- [MUR99] Murray-Rust P, Rzepa H S, Chemical Markup, XML and the World Wide Web, 1. Basic Principles, Journal of Chemical Information and Computer Sciences, volume 39 issue 6, pages 928-942, American Chemical Society, 1999
- [HAL91] Hall S R, Allen F H, Brown ID, The crystallographic information file (CIF): a new standard archive file for crystallography, Foundations of Crystallography, A47 pages 655-685, International Union of Crystallography, 1991
- [COR07] Corbett P T, Batchelor C R, Teufel S, Annotation of Chemical Named Entities, Association for Computational Linguistics, 2007 (<http://acl.ldc.upenn.edu/W/W07/W07-1008.pdf>)
- [STE03] Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttmann E, Willighagen E L, Journal of Chemical Information and Computer Sciences, volume 43 issue 2, American Chemical Society, pages 493-500, 2003
- [FOX] White T, FoX: a FortranLibrary for XML, <http://www.uszla.me.uk/FoX/>
- [WHI06] White T O H, Murray-Rust P, Couch P A, Tyer R P, Bruin R P, Todorov IT, Wilson DJ, Dove MT, Austen KF, "Application and Uses of CML within the eMinerals project", Proceedings of the UK e-Science All Hands Meeting 2006, pages 606 to 613, 2006
- [RDFa] Adida B, Birkbeck M, RDFa Primer: Embedding Structured Data in Web Pages, W3C Working Draft, 2008, <http://www.w3.org/TR/2008/WD-xhtml-rdfa-primer-20080317/>
- [GOLEM] Walkingshaw A, The Golem ontology system, 2007, <http://www.lexical.org.uk/golem/>

- [CRYSTALEYE] Day N E, Downing O J, Murray-Rust P, Crystaleye, 2007, <http://wwmm.ch.cam.ac.uk/crystaleye/>
- [JMOL] Jmol: an open-source Java viewer for chemical structures in 3D, <http://www.jmol.org/>
- [RDF] Manola F, Miller E, RDF Primer, theW3 Consortium, 2004, <http://www.w3.org/TR/REC-rdf-syntax/>
- [RFC4287] Nottingham M, Sayre R, RFC 4287: The Atom Syndication Format, the Internet Society, 2005, <http://www.ietf.org/rfc/rfc4287>
- [RFC5005] Nottingham M, RFC5005: Feed Paging and Archiving, the Internet Society, 2007, <http://www.ietf.org/rfc/rfc5005>
- [RFC2413] Weibel S, Kunze J, Lagoze C, WolfM, RFC 2413: Dublin Core Metadata for Resource Discovery, the Internet Society, 1998, <http://www.ietf.org/rfc/rfc2413>
- [SPARQL] Pru'dhommeaux E, Seaborne A, SPARQL Query Language for RDF, the W3 Consortium, 2008, <http://www.w3.org/TR/rdf-sparql-query/>
- [PROCESSING] Fry B, Reas C, the Processing programming language and environment, <http://processing.org/>