DDLm comments (summary)

This document contains an edited version of the comments received by I.D.Brown on the Discussion Paper concerning decisions that COMCIFS needs to make related to the *methods* Dictionary Definition Language (DDLm). Not all these comments were posted on the discussion list and so will not have been seen before.

The comments are edited according to subject into a readable (?) sequence so as to provide a lead into the discussions at the closed COMCIFS meeting at the IUCr Congress in Osaka in August 2008. Parts of the original Discussion Paper are included to provide context. [Editorial changes and comments are enclosed in square brackets.]

Anyone interested in pursuing these discussions in Osaka is invited by the Chair to attend the closed COMCIFS meeting to be held during the IUCr Congress.

2008-08-25:1300 in room 804
2008-08-28:1300 in room 804

0. ***There are two important CIF principles that are relevant.***

Original:

1. CIF (normally) defines only one data item for a given piece of information and the format of that item is the one closest to the natural representation of the quantity (rather than a format in common use or one designed for ease in computation). For example, the CIF specifies which units that are to be used and alternative units are not permitted.

2. A given piece of information may appear only once in a given datablock. This is necessary to ensure uniqueness of the reported value. If the same information is given twice there could be an ambiguity if the two values do not agree. This restriction follows necessarily from 1, but if 1 is relaxed, this principle could still be preserved.

Discussion

Bernstein:

Certainly it is desirable to avoid easily misunderstood choices
of representation and it is desirable to avoid pointless repetition
of the same information, but we already treat these more as sensible
guidance than as rigid rules.

Even the generally well-accept "principle" of not allowing alternate
units is "violated" in the PDB Exchange dictionary, and will be
in the next imgCIF dictionary.
[Note; the PDB Exchange dictionary is not a COMCIFS approved dictionary and the 'next imgCIF dictionary' has not yet been presented for approval]

One could also question how "natural" the choices of units are.
Consider, for example, cell parameters. The "natural" units for
angles are radians, but out of respect for common practice, we

use degrees.

The second principle is also violated, for example by allowing cell volumes as well as cell edge lengths and angles. The volume is derived and duplicative, and it certainly is possible to generate a CIF in which the cell volume is inconsistent with the cell edge lengths and angles, and, unlike U's vs. B's it is highly likely that both will have been given, along with, in the case of mmCIF, several transformation matrices that also need to be cross-checked.

It would suggest replacing the two principles with the following three guidelines:

1. When defining new data items, dictionary developers are advised to avoid unnecessarily duplicative definitions, e.g. two definitions of the same item that differ only in the units used. Exceptions should be justified and fully documented.

2. When relationships exist among multiple definitions, those relationships should be stated clearly, and, if at all possible, algorithmically, preferably using DDLm, to allow for automatic validation, and, when, necessary, generation of values for missing data items.

3. An existing tag should never be used in a way that is inconsistent with definitions used by journals and archives.

[This raises an interesting point not yet mentioned. As long as each dataname is unique the letters that compose it have no significance to the computer which would look to the DDLm method for a definition, but datanames are highly significant for the user who may bring an incorrect preconception about the nature of the definition without checking the algorithm. One way of voiding this is to use datanames that have no human significance.]

1. Virtual dictionaries.

Original:

DDLm is designed to allow dictionaries to be merged. This is achieved through *import* items that allow all or parts of dictionaries stored at the specified URIs, to be imported into a higher level dictionary, with a calling 'head dictionary' as the top level. Because merging will be simple, it is likely that the current dictionaries, such as the Core CIF dictionary, will be broken into several smaller, more specialized, dictionaries during their conversion to DDLm. One consequence of this approach is that each CIF will specify the contents of its own run-time virtual dictionary. This may range between a small head dictionary containing only *import* statements or a complete existing dictionary in which no further merging is needed. The CIF items needed to identify the head dictionary are already present in the audit_conform category.

Currently these items are rarely used, but in the future CIFs will be expected to contain the _audit_conform items needed to identify the (head) dictionary to be used for reading the CIF as well as the URI where it can be found. Any *import* items in the head dictionary will in turn have pointers to where the required subdictionaries can be found. For most applications this dictionary will be one of a small number stored on the IUCr web site where the COMCIFS-approved subdictionaries will also reside.

However, if CIF is to exploit its potential for flexibility, there are other possibilities that some people may occasionally wish to use. For some specialist application, the originator of a CIF may wish to create a custom head dictionary, particularly if a local dictionary is to be incorporated in the virtual dictionary since the standard head dictionaries on the IUCr web site clearly will not contain calls to local dictionaries. This is not a problem if the CIF is only to be used locally since the head dictionary can be stored on a local server. However, if the CIF is to be passed to another institution the question arises as to where the relevant head dictionary (and any required local dictionaries) are to be stored. There are a number of possibilities:

A, ***These dictionaries are stored on a local web server.***
Problems: The dictionaries would have to remain in the server with a stable URI as long as the CIF was in existence, but if the CIF were circulated to other institutions the originator would no longer have control over when or whether the last copy of the CIF was deleted. Therefore the head and local dictionaries would need a permanent and stable URI or the CIF would need an expiration date. Are such conditions easily met?

B. ***The dictionaries could be stored on the IUCr web server.***
Advantages: This would be stable,
Problems: Would the IUCr be willing to provide a home for such personal files with no indication of how long they would need to remain active?

C. ***The head dictionary could be included in the same file as the CIF that calls it.***
Advantages: The dictionary would travel with the CIF and no permanent URI would be needed.
Problems: There is a risk that the CIF and dictionary could become separated over time, making reading the CIF problematic.

D. ***The head dictionary could be included as a text field within the CIF itself as an item called _audit_conform_included_dictionary.***
Advantages: The head dictionary is an integral part of the CIF and would remain with it. The head dictionary could incorporate any required items from a local dictionary or the _audit_conform_included_dicrtionary could be looped.
Problems: There may be some technical issues in this solution but these should not be insurmountable. Mixing different file types within a datablock may be considered inappropriate and possibly dangerous. However, there is a precedent in imgCIF which includes a representation of a binary file within the ASCII text field of a CIF

E. ***CIF rules would disallow the export of a CIF that calls a head dictionary without a***

stable URI.
Advantages: It solves the problem
Problems: It would be difficult to enforce. It would effectively limit uses 6 and 7 above to in-house use or to institutions that can provide stable URIs

Discussion
[The discussion of this item is extensive and has been split into three sections.
A. a discussion of the solutions listed above (D seems to be favoured).
B. a description of SGML as an analogue to CIF
C. a discussion of the technical issues of including dictionaries as a text field in a CIF]

A. Discussion of the solutions listed above
Bernstein:
The layering of dictionaries and hierarchy of methods can be viewed as bugs or as features. A CIF being used for a journal submission or an archive submission has to be a complete fully documented package. In that case, it would not be appropriate for the CIF to depend on a local dictionary not submitted with the data CIF, e.g. using David's option D of _audit_conform_included_dictionary, but clearly it would be helpful to the community to have the IUCr start and archive of "local" dictionaries, preferably with namespace controls, e.g. using the current system of prefixes.

For work within a lab, however, we cannot and should not act as the "CIF police". If someone has a scientific use for odd layering and real-time assembly of virtual dictionaries, why should they not do it? When the CIF is ready to be moved elsewhere it needs to be cleaned up and documented, e.g. by creating the expanded dictionary from the local pieces.

Hall:
Nick and I ... are
likely to advise that the level of delivery and centralised support be kept as transparent and as simple as possible. Certainly DDLm will allow considerable flexibility in dictionary creation and application, particularly at the lab level, but administering its global archival/publication application will require precise (perhaps, rigid) guidelines. Of course some will argue that the CIF dictionary equivalent of Wikipedia is needed but I seriously doubt it would work in science (as indeed there is increasing doubt about the value of open online encyclopedias, beyond superficial!).

Hester:
I favour option D (the dictionary is contained within the datablock as

the text value of a dataitem). This ensures that the dictionary remains as close as possible to the dataitems it is concerned with. Note that creating a CIF with such a built-in definition in no way forces the IUCr to condone the content of that definition: if someone were to define betas in their CIF and then submit a CIF with betas rather than UIJs, the IUCr remains free to act as it has in the past (especially if the beta definition boils down to a description along the lines of "beta i j as conventionally defined").

I would note in passing that Option B (the IUCr accepts and administers deposited dictionaries) is appealing in that the IUCr could take care of checking dictionary correctness and keeping track of what was defined by whom and when. But if DDLm were actually to expand beyond the confines of the IUCr this would not be a useful option, as other fields may not have a well-developed central organisation and at the same time would be in even greater need of a method of passing around dataitem definitions.

Bernstein:
Allowing a dictionary to be the value of a tag is a reasonable extension of the range of possible URI's for dictionaries, but to avoid ambiguities we need to include such tag-located dictionaries within the DDLm import paradigm, so that we know precisely where within the composite virtual dictionary the tag-located dictionaries should be placed.

Krahn:
I suggested something like [option D] recently on this list, but got no interest. Arguments against it were that non-standard data just makes things complicate database management. I favor it because not all data is database oriented, and that this would allow easy addition of non-standard data, which is often needed when experimenting with new methods.

The contained virtual dictionary can specify the parent dictionary. The same syntax could be used to just specify the associated dictionary that the data block conforms to. The parent dictionary referenced could also be a partial dictionary which references another higher-level dictionary to establish a dictionary hierarchy.

Yes, we should come up with a way of referring to such embedded dictionaries in an import statement, or else we could use the dictionary URI defined within the dictionary itself. For this latter to work, any item names corresponding to dictionaries need to be pre-parsed before the full dictionary processing step is embarked upon

- a reasonable requirement I think. Those programs wishing to be dictionary aware would follow the following steps:

1. Parse the CIF file
2. Pre-process (ie remove escapes from) the data item(s) containing dictionaries
3. Parse these data item(s)
4. Add the parsed dictionary URIs to the list of known URIs
5. Locate dictionaries based on the contents of the _audit_conform data items and proceed as usual

du Boulay:
>> 2. The CIF dictionary that is used to read a CIF may be a virtual
>> dictionary created at run time by
>> merging a number of smaller CIF dictionaries according to
>> instructions included in the CIF itself.
>> This works in the following way: The CIF is first read in as a STAR
>> file in which no meaning as
>> attached to the datanames or their values except for those items
>> in the audit_conform category.
>> The program then uses the information in the _audit_conform items
>> to direct the assembly of the
>> virtual dictionary. Finally the program uses the virtual dictionary
>> to interpret the other items in
>> the CIF. The reading program must be hardcoded for reading STAR
>> files and interpreting items in the audit_conform category.

Its not exactly an efficient processing mechanism.
Compare that to SGML/XML where the second line says:
"The following document conforms to the structure of 'this DTD'"
or in XML(only):
"Until further notice, all XML elements including and contained within this element belong to 'this namespace' which refers to 'this schema URI'"

The whole notion of having to parse an entire data block prior to identifying the dictionary to which it conforms is just seems awkward and inefficient (in my opinion). Since you already have a CIF format Magic Number comment, maybe you could just introduce a leading CIF Conformance comment also, or is this too reminiscent of PDB REMARKs?

>> DDLm offers the following additional possibilities that are not
>> available under DDL1 or DDL2.
>> 6. Including local dictionaries in the virtual dictionary used
>> to read a CIF.

I was under the impression 6. was already possible in DDL1 and DDL2 in the
form of simple concatenation, though seldom used?
[It is seldom used because there is no software for merging dictionaries, although there is a protocol for doing so.]

>>DDLm is designed to allow dictionaries to be merged. This is achieved through import items
>>that allow all or parts of dictionaries stored at the specified URIs ...

Possibly there's scope for confusion about URIs, URLs and URNs.
A URI can be any unique identifying string that is either a URN or a URL.
Dereferencing the URI is often shortcut by using an explicit URL,
but generally the dereferencing process can/should be externalised, ideally
through the use of catalogues, e.g.
http://www.oasis-open.org/committees/download.php/14041/xml-catalogs.html
Maybe theres an opportunity to leverage some existing XML technology?

Also, why not add:

F. In the absence of audit_conform and/or the available head, default to
assuming some standard dictionary, exactly as is done currently.
advantages:
Well, you are going to handle this anyway in coping with existing
archived CIFs that don't specify conformance.

I had a lot of trouble rationalizing the whole dictionary conformance issue.
On the one hand, baring syntax issues like repeated names in a data block,
CIFs containing unofficial data names are just as conformant as those which
don't. Even a CIF containing no data names from a given dictionary is
considered conformant to that dictionary.
For that matter, I was led to believe that the virtue of CIF not declaring
conformance to a specific dictionary, was that it could then be
associated with any dictionary, such as new and improved dictionaries.

So, I guess I am wondering what the rationale is for suddenly caring
about sundry arbitrarily defined items in some obscure customised
dictionaries? Are these definitions really going to be so important?
Is there some intention to tighten up the conformancy checks and
reject cifs containing unrecognised data items?
[A program would not be able to exploit the methods unless it had access to the appropriate dictionary which might be a local dictionary. The conformance only notes the dictionary used to prepare the CIF. Later (though not earlier) versions of official dictionaries can be substituted, e.g., DDLm dictionaries can be used to access DDL1 conformant CIFs]

Spadaccini:
I think I made it pretty clear [in an earlier email]

that [embedded dictionaries] was a very big mistake.

I strongly favour the third party maintaining a URL.
I also strongly feel it is not the job of COMCIFS to maintain everybody's "extensions" to the ontology - much of which may be unusable and nonsensical. However third parties who wish to communicate their "improvements is undertaken, is not a non issue for a governing body like the IUCr who has responsibility for maintaining an ontology.

Hester:
I prefer embedded dictionaries because I don't think many third parties can hope to maintain such a URL for the expected lifetime of the datafile. A counterargument is that, if the data items are so valuable that a definition has been written for them then presumably a local URL can be maintained at least long enough that the third party can go through a process of getting the datanames vetted by the IUCr. This is presumably the status quo at present.

Spadaccini:
This is the argument I made before. What value do these extensions have if their use and definitions can't survive the simple maintenance of a URL? If they are worthy of retention then a body of people will support it. If it is just one person adding in whatever in to a data file and then not being able to retain its meaning in a widely accessible fashion - what value is there in those extensions?

Hester:
An institutional URL can be beyond the control of a given group, no matter how worthy their particular definition is and how strong the support for it is. This has been particularly true with my current and past employers (ANBF/ASRP vanishes Dec 31, ANSTO keep shuffling the deck).

Brown:
Introducing methods extends the flexibility of CIF in a way that conflicts with the maintenance of the strict ontology required by journals and databases. For example as part of a research project someone may wish to define a new item with a somewhat idiosyncratic algorithm. DDLm clearly allows this, but the official public archives would not normally be interested in retaining such an item because it would not be part of the official crystallographic ontology. If this new item is kept in-house there is no problem, but if it is shared with others working on the same project, the appropriate local dictionary must be accessible to all of them. It could be circulated separately, but it would be more convenient if DDLm had a protocol that allowed the local dictionary to be embedded so the recipient could immediately assess the new algorithm. In turn the recipient might reply with a

different algorithm for the same item, and in each case the correct dictionary would be available within the CIF. If such a CIF were later submitted to a public archive this item would be ignored as not fitting the profile of its archived information.

B. Description of SGML

McMahon:

It may be useful to consider the relationship between SGML document type definitions (DTDs) and CIF virtual dictionaries.

In SGML, which is an electronic publishing markup language, a document must refer to the DTD which describes its structure. The DTD is written itself in SGML. It may contain descriptions of the document structure and/or references to other DTDs. An SGML parser reads an SGML file, dereferences the calls to DTDs, and assembles a document structural model against which the remaining contents of the file are validated. (The "document structure" comprises valid tags, order in which they may appear or be nested, allowed data types, and perhaps discrete permitted values. There are many analogies with CIF/DDL.)

The SGML standard allows a DTD to be an external file (or set of files imported and assembled recursively), or an integral part of the SGML document. Again, the parallels with the DDLm dictionary being external to, or integral with, the CIF are clear.

In practice, organizations managing SGML documents maintain a central repository of standard DTDs, and the individual documents reference them rather than import them physically. This allows the physical size of the documents to be more compact. (Compare CIF, where a typical data file for a structure is a few tens of kB, while the core dictionary, even without methods, is nearly half a megabyte; the PDB exchange dictionary is over 3 MB.)

For SGML, the location of a DTD is usually specified through a registry of local directories and files rather than by URL (the SGML standard predates the Web). This reduces the portability of SGML, although it is considered adequate to allow, say, a typesetter and a publisher to interact; both organisations invest so much in their activities that the effort to create matching registries is considered acceptable.

If one wishes to transfer an SGML document to someone who does not have access to the same registry, or if one wishes to ensure that a single file can be archived without dependency on external resources, then the full DTD can be inserted into the document instance.

Technically, then, we should permit both possibilities for DDLm

dictionaries. However, we might take the view that stable and reliable URLs make it easier to provide a common "registry" accessible by all users, and so implement URL-based references to DDLm dictionaries as standard working practice. The IUCr would be reasonably happy to host copies of specialist or private dictionaries, so long as they met certain quality criteria.

Of course, the IUCr site might one day cease to operate, but if all the referenced dictionaries could be found at one location, it might be easier to ensure that they are transferred to another authority, or that other arrangements could be made to restore the integrity of the data files that referenced that one site.

* * * *

It's also interesting to consider the history of SGML. The purpose of a DTD is to impose a particular document structure and to validate a document against that structure. The computational requirements of document validation are high, and the management of multiple DTDs is also a complex programming task. Very few completely independent SGML systems have ever been written. Most software packages that handle SGML depend ultimately on James Clark's SP parser. Fully competent SGML software is complex, mostly available as expensive commercial packages, and generally implemented within an organisation that invests heavily in publishing or document management. [SGML is, however, a powerful software system, and for many such organisations the investment is fully repaid.]

SGML never made much head way in the outside world, but XML changed that. XML may be considered a subset of SGML, but it offers a number of simplifications. One was that DTDs were no longer mandatory. An XML file could be considered valid so long as it was 'well formed' (i.e. adhered to syntax and some simple structural standards). Of course one could do less with a document that was not specified by a DTD, but for many practical purposes a community could adopt certain procedural rules or conventions, and gain the benefit they required without needing to design and implement a full DTD. Semantic content could be carried through schemas; presentation could be externalised in style sheets. XML took off. It does still support DTDs (and I suspect most XML DTD parsers are still built on Clark's SP), but few applications use these.

So there is a historical inversion of what we are now proposing for CIF. XML is analogous to the older flavours of CIF ('well-formed' corresponding to syntactic correctness, schemas mapping to some extent onto DDL1 and

DDL2 dictionaries).

By analogy with SGML, I suspect that DDLm will only ever attract one or two fully functional parser/validators, so these must be robust and capable of being integrated into other application packages. Given that, the complexity of handling multiple dictionaries will probably be delegated to just one or two public domain libraries. It is likely that only a few organisations will be able to, or want to, handle the full complexity of CIF3 validation and processing (the IUCr, PDB, CCDC etc). I do not foresee Jmol, for example, implementing a full DDLm compiler (though with Bob Hanson at the helm, Jmol might just be the application to do that...).

So it might be that functions such as "return missing values by DDLm methods evaluation of the other content" move into the realm of web services provided by the IUCr or other service providers, rather than compiled-in functions of a crystallographic software suite. If that's the case, then the problem of exposing intermediate data items/values also becomes less acute.

Bernstein:
I, for one, find Brian's comments on SGML and XML to be on point. Whether the IUCr wishes it, or not, there is going to be significant use of CIFs by programs that will not validate those CIFs against any dictionaries. The vocabulary control will be in the programs, and unknown features will simply be ignored (think HTML).

C. Technical issues

Bernstein:
With respect to the use of \; to allow embedded text fields within text fields, we need to deal with the long-standing use of \; as ogonek, and other existing uses of \. Rather than continue flagging special treatment of text fields in context, I would suggest adding the syntax and semantics of a dictionary text field as a DDLm data type.

Hester:
I would fiddle with [Bernstein's] suggestion here of defining a (presumably single) DDLm data type for the text field. In general, a domain dictionary should be free to define the content of text fields e.g. LaTeX, or MIME, or a set of possible escape characters, etc. What DDLm can provide is a few templates for inclusion in dictionary definitions of those data items which have plain text values. This/these template(s) would simply consist of a description of the allowed escape characters and any other special conventions. As I see

it, the role of such definitions would not be to help machine interpretation, but as a definition for human readers to help them in producing conformant text that will be understood by downstream interpreters of text delivered by a CIF parser. Note that I am not proposing that the contents of such text fields could or would be validated by CIF software, but downstream recipients of the content are free to do so.

For example, the definition text for the IUCr publication_ item names would contain a potentially long description (or URI reference) describing all of the escapes available which would be understood by the Chester software. The DDLm dictionary itself could define some conventions for writing domain dictionary text, which would allow automated typesetting.

In the present case (embedded dictionaries) the downstream application is the CIF parser itself. Assuming we don't plan to apply a dREL method to the escaped text to produce the pure text, the human authors of a given CIF parser will be the source of the de-escaping code. Therefore, it is sufficient to include a statement in the descriptive text of the _audit domain dictionary stating the method of escaping <EOL><semicolon> digraphs. Note that it is not desirable to add this <EOL><semicolon> escape to our general text template described in the previous paragraphs and then just include that template in the _audit dictionary, because including this general template will also potentially include all sorts of other escapes which we don't want to use in this special case (e.g. we don't want to re-escape already escaped text in the embedded definitions). In this particular case, we want this one escape only.

The "\;" digraph occurs 319 times in the 32377 files in the IUCr CIF archive as of June 08, overwhelmingly inside author's names. There are currently no files containing <EOL><backslash><semicolon>.

The technical issues boil down to being able to escape the "<EOL><semicolon>" digraphs in the embedded dictionary text, which would otherwise prematurely end the datavalue.

Krahn:
This is definitely a problem. A single text datablock should be able to hold an entire CIF dictionary without having to indent the text.

Hester:
Some suggestions:

(1) Define "<backslash><semicolon>" as being an escaped semicolon (and this would require defining "<backslash><backslash>" as an escaped backslash in order to cope with those situations in which you actually want the text that comes out to be "<backslash><semicolon>"). Obviously the escaping character doesn't have to be backslash.

Krahn
I made a similar suggestion over a year ago. Nobody wants to define special escape sequences, do to interference with the set codes for special characters.

Hester:
(1a) Define "<EOL><backslash><semicolon>" to be an escaped "<EOL><semicolon>".

(2) Substitute <EOL><hash> for <EOL> in the entire text field. This immediately signals to the human reader that the entire text field is a single block, rather than bits of a CIF file (same as quoting in emails), and is easily reversible. And nestable, but let's not go there...

Krahn:
Another alternative is to just use MIME encapsulation, already defined for Binary CIF. As it is implemented there, the data block is all base-64 encoded, which contains no semicolons. But, MIME can also encapsulate unencoded text, by defining the end-of-data marker. This requires the low-level parser to understand the MIME format.

Hester:
This latter is the killer - this would require understanding of a whole other standard at the parsing level, change the CIF standard significantly at the syntactical level, and break all current parsers (see comments on save frames [below]).

Krahn:
I should also mention that the newer STAR format has a bracketed list format that includes backslash escape sequences to allow contained items to have a bracket. If this is adopted by CIF, that may be enough of a precedent to support backslash-escapes for beginning-of-line semicolons as well.

Hester:
I hadn't heard about the <backslash><bracket> escape sequence for STAR - any idea where information about that might be found? In general, enclosing any element of the bracketed list inside inverted commas or

apostrophes should be sufficient to prevent the STAR parser from interpreting the bracket, so the backslash notation would appear to be unnecessary. In general I don't think that COMCIFS would be too critical of a new escape sequence if there was a perceived benefit.

Krahn:
Here is a snippet of my idea for storing dictionary data within a datablock, using a SAVE frame, so that it is part of the actual CIF syntax rather than using a text block, which means the data has to be double-parsed.

By using save frames, it is easy to avoid conflicts, because CIF currently restricts their use to dictionaries. I would also support nested SAVE frames so that the whole dictionary syntax can be fit inside of one parent SAVE frame, rather than having to split it.

```
data_cns_mtf
save__dictionary
  _item.name        '_cns_mtf.title'
  _item.category_id cns_mtf
  _item_type.code   text
save_

_cns_mtf.title
;
FILENAME="10.mtf"
Written by O version 9.0.3
Thu Feb 22 16:54:17 2007
DATE:23-Feb-07  10:46:23       created by user: krahn
VERSION:1.1
```

Hester:
I agree that save frames are an excellent solution to the problem...except that most current CIF parsers are not able to parse save frames in data files, so by allowing data files to contain save frames you will immediately render most CIF-reading programs broken (at least in spirit). The rarity of user-defined dictionaries coupled with the rarity that that dictionary will be useful to a given program anyway means that the cost of breaking all those programs is not worth it.

Spadaccini:
I have been thinking about how to
wrap the dictionary between [] and the parsing of the internals would be seriously complicated.

Hester:
I agree that this is not the most practical solution. What about the other alternatives? (e.g. inserting some character like '>' after every embedded <EOL>, or defining <backslash><semicolon> to mean <semicolon> and <backslash><backslash> to mean <backslash>)

Spadaccini:
The problem is you are changing the contents of the data. There will be cases where <backslash><semicolon> is part of the method. You then have to parse the contents to create <backslash><backslash><semicolon> in those instances otherwise the un-wrapper will suppress the original <backslash>. To be honest the embedded solution is not good, but if you have to, then the mime solution suppresses all parsing problems. The <eol>; never appears in the encoding if I am not mistaken. These days there is no real argument to people not knowing how to handle mime types. There are a swath of routines in most languages top auto-handle them.

Hester:
I contend that e.g. mindlessly inserting a '>' character after every <eol> in an embedded definition is an easily reversed operation for which you need to know nothing at all about the data. MIME seems like an unnecessarily large hammer for such a simply-solved problem.

2. Hierarchy of methods

Methods are used to calculate the value of an item from other items in the CIF. Items for which *methods* are defined are necessarily derivative since by definition experimental measurements cannot be calculated in this way. There is therefore a natural hierarchy with measured quantities forming the foundation.

There are, however, some quantities that can be calculated in more than one way. Fobs^2 for example can be calculated either from Fobs or from I and the calculations are different (though they should give the same result if the CIF is self consistent). There are two approaches that can be taken and these represent two different views of how CIF should be developed.

A. ***The methods are seen as hierarchical. For example, Fobs^2 is always calculated from Fobs, but if Fobs is not present, it is first calculated from the observation, I. This approach implies an hierarchy of items in which a given item is always calculated from more fundamental items.***

Advantages: It creates a unique definition for each derivative item. As this is only a problem for dictionary writers it can be monitored by COMCIFS.

Problems: However, local dictionaries that include methods would not be monitored by COMCIFS. The hierarchy would have to be explicit so that

the writers of dictionaries would know that Fobs^2 should be calculated from Fobs rather than I. Most calculations involve several items and it is easy to imagine that there may be more convoluted cases where it might not be possible to maintain the hierarchy, or in which one inadvertently ends up with circular definitions (see for example the problem described in item 3 below).

B. ***Several looped methods are allowed.***
Advantages: A derived item could be calculated in different ways. The loop key would be a number that defines the order in which the *methods* would be attempted. Method 1 would be the default (and primary definition), method 2 would be used if method 1 could not be used because the required items were not present and could not be calculated. If required the loop key could be specified by the user at run time to indicate which *method* should be used.
Problems: The presence of multiple definitions could result in confusion about the primary definition even though the default would be regarded as primary.

Discussion
Hester:
I would cautiously advocate allowing a looped list of methods with a priority indicator. Dictionary validation tools should attempt to test both methods and make sure that the resulting values are identical to within some precision. In general I wouldn't expect this situation to crop up all that often.

du Boulay:
Actually, I am not sure about the logic of the example in A.
The intensity I should be "fundamental" since it's a measurement and therefor it shouldn't have an evaluation method.
Fobs2 is derived from I (via scale, lorentz, polarisation, absorption), if it exists, or falls back to Sqrt(|Fobs|) if that exists.
Should be programmatically as simple as if else, or try except?
|Fobs| either exists as data or is derived from Fobs2
Regarding the problems of corrupted local dictionaries, I am not sure why COMCIFs should even care about that.

Also possibly, in the event of circular dependencies - software like Python often has recursion limit checks. Presumably it would be possible to check if a top-level method has called back on itself programmatically? I'm sure Syd mentioned something like this once.

Spadaccini:
We are building a prototype as we speak and hitting several

blocks - it is not until you actually implement that you see the short fallings of what you have specified. APIs for several support languages? John Westbrook insists on having methods that are his C++ routines. How can these be actually implemented sensibly?

Hester:
Is there something worth changing in the dREL/DDLm spec? Anything I should consider for my review? What are these methods of which you speak? Are you suggesting that C/C++ might become an alternative to dREL? Or that the dREL engine would have some C/C++ linkage under the covers?

Spadaccini:
At the moment because most people don't have a clue what is involved in generating methods that actually execute, there are requests for their favourite Fortran (etc) algorithm to be included. For now, we believe the solution is to provide a link in the alternative method which can be de-referenced at runtime. It however precludes how it would ever be implemented - which is a huge problem (for them not me, because dREL is my solution). That being said at least there is a mechanism to identify a reference to the module.

Hester:
I didn't realise that there were plans afoot for alternative methods. This immediately raises the question of how to verify and vet these methods, and make sure that the dREL method is the canonical one. My expectation was that a tool could be written which would translate dREL into the language of choice (with suitable assumptions about execution context) if the standard Java engine was not suitable.

Spadaccini
What are we going to build? Will it be under a OpenSource licence? Who will maintain and control distributions?

3. Intermediate items used in computation

Since *methods* break the calculations into small steps, any given computation may involve the generation of values for several intermediate items. Some of these will be items already defined in the current dictionaries, for example the generation of Fobs when calculating Fobs^2 from I, but others may be items that are not currently defined (often for good reasons). This is well illustrated by an example taken from the current draft DDLm CIF dictionary: the use of the beta matrix format to simplify calculations involving the atomic displacement parameters.

The two CIF principles described above are at stake here. For example the atomic displacement parameters are given in the physically natural U form (though a

concession was reluctantly made of allowing protein structure reports to use the B form because of its ubiquitous use in that important field). The generation of a beta matrix from the U matrix during a routine calculation violates both principles: the availability of two different formats for the same information (atomic displacement parameters), and the presence of the same information twice in the same CIF (as U and as beta). Further, we cannot assume that everyone will use the U format in generating a CIF. If the beta format is defined in the dictionary eventually someone will create a CIF with atomic displacement parameters given only in the beta format.

This example raises a number of red flags. In the crystallographic literature, beta is defined in two different ways, either with or without an explicit 2 in the cross terms. Rarely do the original papers that report betas state which convention was used. The definition of beta in the draft dictionary is, of course, unambiguous, but how many people would make sure that the beta they report conforms to that definition? Further, there is no easy way to check whether the correct convention was used. This is no hypothetical danger; we have already received a request for a CIF definition of the beta format so that a database could use it to output entries containing beta parameters copied from the original paper. This request was turned down on two grounds, firstly as being contrary to the CIF principle requiring all items to be reported in the same format, and secondly on the grounds that most published betas are ambiguous and CIF should avoid any ambiguity. Possible solutions to this problem are

A. ***Prohibit the use of intermediate values that would not be welcome in an archival CIF.***
Advantages: This ensures that we adhere to current policies that are important principles in defining our dictionaries.
Problems: It limits the ways in which calculations could be made and may prevent certain shortcuts from being used.

B. ***Flag items that should not be included in an archive.***
Advantage: Even though the intermediates are included internally in the CIF, they can be removed when a CIF is saved. A program can easily check if temporary intermediates are present, i.e., atomic displacement parameters reported as betas would be flagged as non-conforming and removed or a warning issued.
Problems: It creates two classes of items in the dictionary, archival and computational (but this may be the price of the exploiting the multifunctionality of CIF)

C. ***Allow the intermediate values to remain in archived CIFs and abandon the two CIF principles described above, namely that an item of information be given only in one format and that the same information appear only once in the same CIF.***

Advantages: Avoids having two different types of item.
Problems: Two important CIF principles would be lost with serious implications for the future of CIF. Either it places an additional burden on software producers who must be prepared to read items in a variety of formats, or the CIF dictionary would need to contain a *method* for generating U from beta. Since beta is currently defined in terms of U, this would create a circular definition and play havoc with any hierarchical scheme in the computation of missing items (see the discussion in point 2 above). If two forms of the same information do not agree, we need a flag to indicate which form is the default. This solution raises many specters and would put the CIF project in peril.

Discussion

Bernstein:
What is nice about DDLm is that it now allows us to put the necessary cross-check information into the dictionaries, and that would seem to me the best way to address David's third issue of dealing with intermediate values used in a computation -- allow them in, but only with the methods necessary to validate them given. We only get ourselves in trouble if we use the same data item name for two different or computationally inconsistent meanings, not if we have unique names for related items.

Hester:
I go with option A, to prohibit the definition of intermediate values. An item defined in an official CIF has an important status and we shouldn't water that down or introduce unnecessary complexity. I believe that the correct way to deal with the intermediate result issue is to add a new function to the function category (e.g. BetaFromUIJ). *Implementations* of dREL are then free to attempt caching of results (eg by defining for themselves an internal dataitem attached to that function) for efficiency and the dictionary retains its purity.

Du Boulay:
A (non voting) vote for B. (flag for removal) since in the calculation:
|Fcal| = Sqrt(cFcal2) where cFcal = Fa+ i Fb,
you really won't care about the complex cFcal, or its real and imaginary components in any derived CIF, despite their importance in the calculation of |Fcal|

Brown:
Another possibility is to use a name (for the beta form of ADPs) such as _atom_site_aniso_nonsense or _atom_site_ansio_temp which should discourage people from being tempted to use it.

Spadaccini:
Finally having accepted responsibility for maintaining these extensions is the IUCr really wanting to deal with the very real scenario of say, Syd Hall, author of Xtal doing the following. Here are all the data items for the internals of Xtal at runtime, with the dictionary, _xtal_001, _xtal_002, ... _xtal_12345678912345 There are bucket loads of them. Yes there is some dictionary equivalent of most of them but I don't want to use those, I want to use mine. And yes _xtal_001 is the tan2(arcsin(cos(theta))) a meaningless quantity but heck it was there at runtime so I want to archive it as well. Now multiply that by the authors of Jmol, ShellX, Xplor ....

Pretty soon we will be back in 1987 when we were complaining about unusable exchange formats.

Hester:
If we assume that the IUCr would not take responsibility for whatever adhoc definitions might appear in submitted data files, do you still see the above scenario as likely? After all, it has always been possible for people to include their own private datanames in a CIF file, and to come up with a DDL definition for those names if they so wished. What does not have to change is the IUCr's attitude to such extraneous datanames (ie completely ignored). That is, I don't see why esteemed Prof Syd Hall would suddenly feel the need to include all the _xtal_ datanames in a CIF file just because he could now include DDL definitions in the very same file.

Spadaccini:
So? Those definitions never went anywhere did they? MY attitude has been for formal archive they be ignored, and certainly not the archivist having to retain dictionary definitions for them.

Because they have in the past been ignored by decree of the organisation taking responsibility for maintaining the ontology. Now because I can use a dictionary facility which I can include, I have the very real expectation that you WILL NOT ignore the contents. Syd can talk to the history of what might have been in the dictionary if all the players got their way, back in the early days of CIF.

Hester:
Sounds worth hearing about. I don't see why people's expectations would necessarily be changed if the IUCr/COMCIFS makes clear that any such embedded definitions have no special status.

Brown:
The fact remains that the current draft CIF dictionaries written in DDLm do

contain a number of intermediate quantities which we would not like to see in the archives but which are used in some of the methods. The alternatives are either to ban them and insist on using more complex method algorithms that don't require intermediates, or to flag them in some way as not being part of the official oncology of CIF. An example from the current draft are orthogonal Cartesian coordinates of the atoms. These can be useful in geometry calculations but are no substitute for crystal-based coordinates in the archive. On the other hand the geometry could be calculated directly from the (nonorthogonal) crystal coordinates (which is both elegant and not difficult to do with modern computers).

################## End of Summary of Comments ################