Relational Expressions in STAR File Dictionaries

Nick Spadaccini,*,† Sydney R. Hall,‡ and Ian R. Castleden†,‡

Department of Computer Science and Crystallography Centre, University of Western Australia, Nedlands, Perth, WA 6907, Australia

Received February 2, 2000

The STAR File (*J. Chem. Inf. Comput. Sci.* **1994**, *34*, 505−508) is used widely in structural chemistry for exchanging numerical and text information with scientific journals and databases. These exchanges are increasingly dependent on data dictionaries to facilitate automatic data validation and checking. Definitions in data dictionaries are constructed using attribute descriptors, and this paper describes a *method attribute* for specifying the relationships between data items as an executable script written in a new relational expression language called dREL. The addition of this attribute improves the precision and the semantic content of dictionaries by providing relational representations of data, as well as facilitating the direct evaluation of derivable data items. The capacity to evaluate derivative data directly from the combination of primitive data and dictionary expressions is expected to change future archival approaches. The design concepts of the relational expression language dREL parser, which are applicable to any discipline, are described.

RATIONALE

The main purpose of data dictionaries is to define data items at a precision that will enable their unambiguous identification. Broadly speaking, data can be classed as either *primitive* or *derivative*. Primitive data are either experimentally measured or theoretically predicted, and need to be defined in a dictionary in sufficient detail to be unique, so that dependencies on other experimental or theoretical parameters are well understood. Derivative data are determined from other data items, and their definition in a dictionary must specify exactly these relationships. In this paper we describe a dictionary attribute that enables the precise definition of derivative data items as relational expressions written in the script language dREL. These definitions are machine interpretable and can be used to calculate derivative data directly, and can be easily specified and modified by nonexperts.

We shall show that the existence of relational expressions in a text dictionary leads to data-handling approaches that seamlessly integrate data values into executable definitions. These facilities are akin to those of an *active* knowledge base.

The rationale for direct computation from text dictionaries is manyfold. The growing complexity of computers and conventional programming languages means that scientists are increasingly removed from the detail of calculation methods and are more dependent on “blackbox” software. Current programming practices focus on computer performance, and as a consequence considerable algorithmic knowledge is embedded in computer code that is difficult to understand or to document. This practice is detrimental to both the transfer and the retention of discipline knowledge. In particular, it limits easy access to this knowledge, and therefore to its extension and application. The problem has not diminished with recent computing developments. Modern languages and implementation tasks increasingly require formal training in computer science or software engineering. Certainly the era when major scientific packages were written by discipline experts for ready modification by other scientists has passed, and there is an real need to develop approaches that can provide simpler links between discipline knowledge and computational outcomes.

Data dictionaries containing relational expressions are one such approach. Used in conjunction with databases they provide electronic information that is extensible, interactive, and accessible. They are also of enormous value as a pedagogical resource. For example, scientists seeking specific advice and information within their discipline often use Internet newsgroups. Such an approach is not as reliable or extensive as accessing interactive data dictionaries that have been constructed by discipline experts.

In this paper we describe a dictionary relational expression language dREL that is functional and intuitive for humans, and amenable to machine interpretation. It is sufficiently symbolic to be easily translated into textbook information. It builds upon the previously reported dictionary definition languages[7,8] for the CIF dictionaries which form part of the STAR File exchange approach[1,2] adopted by the chemical structure community. The prototype version of the dictionary used as a proof-of-concept for the dREL relational expressions has been defined using structural science data, but exactly the same principles apply to data in other fields.

We shall also show that data dictionaries are simple text files that are made executable by translation and then compilation into Java byte-code. This compiled dictionary contains routines for parsing STAR data files. Data evaluation using the compiled dictionary is not fast by normal computational standards, but it is certain to benefit from

* Corresponding author. E-mail: nick@cs.uwa.edu.au. Telephone: +618 9380 3452. FAX: +618 9380 1089.
† Department of Computer Science.
‡ Crystallography Centre.

10.1021/ci000009k CCC: $19.00
Published on Web 00/00/0000 PAGE EST: 12.5

future advances in computer technology. More to the point, we maintain that the most important measure of "efficiency" is the speed with which users can specify new relational expressions and understand what has been previously defined.

BACKGROUND

A STAR File format[1,2] used in the dictionaries described in this paper is a *universal file* structure for exchanging and archiving data electronically. A discipline-specific application of this approach, the *crystallographic information file*[3] (CIF), is used widely in structural chemistry for journal and database purposes.[4,5] The STAR File and CIF were developed in the late 1980s in response to the rapid growth in measured and calculated scientific data and the need to interchange information electronically over the increasingly accessible Internet. The adoption of the CIF in 1990 by the International Union of Crystallography, as the recommended standard for the exchange of data, led to its extensive application to structural science for data submission to both journals and databases.

The development history of data dictionaries in structural science is relevant here as it illustrates the importance that some scientific disciplines are attributing to the careful organization and delivery of knowledge, largely in response to massive increases in data. In structural sciences the CIF development has, in addition to promoting open and flexible information exchange, stimulated the compilation of data dictionaries which have been internationally approved by the governing bodies in a number of subdisciplines.[6] CIF and STAR files use unique tags to identify exchanged data items, and these tags are defined in dictionaries written in a dictionary definition language[7,8] (DDL). Dictionary definitions are composed of *attributes* that ascribe particular properties to the item, e.g., its data type, its permitted values, and so on. These attributes represent the *vocabulary* of a STAR DDL, and determine the richness with which data items can be defined.

Data dictionaries support CIF exchange and archival processes by uniquely identifying and classifying data items. They also play a pivotal role in the automatic validation of exchanged data. Currently two dictionary languages are used in structural chemistry: DDL1.4[7] for *core* crystallographic and *powder* diffraction items[6] and DDL2.1[8] for the *macromolecular* structure items. Both DDL versions involve similar attributes; however, DDL2.1 contains a richer attribute set. The access and validation of data using dictionaries is strongly dependent on the scope and precision of the DDL attributes. The developments described in this paper have highlighted a need for at least three major additions to the DDL attributes: (1) stronger "typing" of data, i.e., the inclusion of "container" types such as lists, matrixes, vectors, etc.; (2) allowance for the hierarchical classification of data and for inheritance; and (3) an intuitive expression language that permits the concise representation of algorithmic relationships between data items.

In this paper we will mainly describe the purpose and properties of the DDL *method attribute* written in dREL. The detailed specification of the dREL syntax will be the subject of a later paper.

THE ROLE OF THE METHOD ATTRIBUTE

Method attributes serve to propagate the values of primitive data items into derivative items. As discussed earlier, primitive data result from measurement or theory and are irreducible. Derivative items can be calculated from other data by following a known algorithm. This derivation is normally achieved using customized computer software. We propose an alternative approach in which the method attribute of a derivative item provides this algorithm in the form of a machine-interpretable text expression.

Method attributes serve two important purposes. They contain data relationships that constitute *precise usable knowledge*, and they provide *executable algorithms* that may be used to evaluate the defined item in terms of others. Such attributes provide a simple, transparent, and direct approach to automating and generalizing processes which are currently either ad hoc or, at best, highly customized. Of longer-term importance is that precise relational information in data definitions is an essential requirement in extending STAR dictionaries into the realm of *active* knowledge bases.

A number of major and immediate benefits arise from this approach.

Enhancement of Knowledge. The use of analytical expressions in data definitions has the obvious pedagogical benefit of making this information more readily accessible to humans. Relational expressions written in dREL represent symbolic, self-descriptive, and mathematically logical descriptions that are easy to read and to understand. The underpinning objectives of the dREL language are to simplify this information for all dictionary users, and yet be sufficiently powerful to attract application experts.

Furthermore, the method attribute, along with all other dictionary attributes, is amenable to the creation of a "textbook" representation of the definitions. A prototype parser is already available that can create a PDF document from existing CIF dictionaries.[9] It is anticipated that future CIF definitions will be rendered automatically into a mathematical typeset (using $\mathrm{T_EX}$ or MathML), so that pedagogical documents of defined knowledge can be produced directly from a DDL dictionary.

Retention of Knowledge. The loss of discipline-specific technical knowledge, such as computing algorithms, is an increasing problem with the evolution of computer technology and programming languages. The erosion occurs in many ways: when computer software is unsupported by a new generation of hardware; when computing languages evolve and earlier versions are incompatible; when program authors are no longer accessible; or simply in those cases when knowledge is obscured by the programming language. The dictionary methods proposed herein offer a long-term solution because they promote the retention of computer algorithms in terms of a minimal-style language that enables relationships to closely resemble mathematical formulas. The dictionary method attribute contains the algorithms as transparent, machine-independent, yet executable expressions.

Reduction of Data Archives. A large proportion of data currently exchanged and archived is superfluous, in that they can be easily derived from primitive data if the interdependencies are known. The use of dictionary methods to directly evaluate derivative items will promote new and yet-un-

Chart 1. Example Definitions from a Prototype Dictionary Which Includes Method Attributes (Identified by the tag _method.expression)

a. Dictionary definition of **crystal density.**

```
_definition.id                  'crystal.density'
_description.text
;
     Crystal density calculated from crystal unit cell and atomic
content.
;
    _type.container             Single
    _type.value                 Real
    _enumeration.range          0.0:
    _units.code                 megagrams_per_metres_cubed
    _method.expression
;
    _crystal.density <- 1.6605 * _cell.mass / _cell.volume
;
```

b. Dictionary definition of **cell volume**.

```
_definition.id                  '_cell.volume'
_description.text
;
     Volume of the crystal unit cell.
;
    _type.container             Single
    _type.value                 Real
    _enumeration.range          0.0:
    _units.code                 angstroms_cubed
    _method.expression
;
     With v  as  cell_vector
           _cell.volume <-  v.a * ( v.b ^ v.c )
;
```

c. Dictionary definition of **cell mass.**

```
_definition.id                  '_cell.mass'
_description.text
;
Atomic mass of the contents of the unit cell. Calculated from the atom
sites present in the ATOM_TYPE list.
;
    _type.container             Single
    _type.value                 Real
    _enumeration.range          0.:
    _units.code                 daltons
    _method.expression
;
    _cell.mass <- 0.

    Loop t as atom_type
       _cell.mass +<- t.number_in_cell * t.mass
;
```

thought-of data-handling paradigms. It will reduce the need to exchange or archive derivative data that can be generated from dictionary methods. Additionally, these methods can be used to automatically verify and validate existing derivative data values. For example, this approach can replace customized software performing validation tasks for scientific publishers.[10]

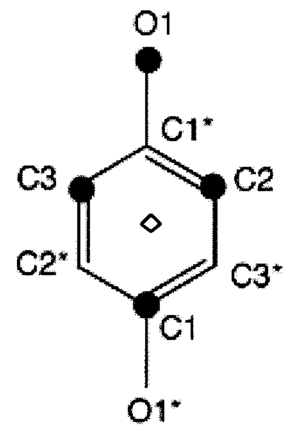
Evaluation of Data. The most important long-term advantage of the dREL methods is computational rather than pedagogical. Executable expressions in dREL avoid most of the obfuscation problems of existing imperative coding techniques and offer a programming paradigm empowering scientists to control and modify their computations in ways paralleled only by symbolic programming approaches within existing algebraic computing packages.

AN EXAMPLE OF THE METHOD ATTRIBUTE

Within the DDL formalism, each attribute is assigned a unique name or tag. A variety of attribute specifications are illustrated in three simple definitions of data items representing the cell characteristics of a crystal, as shown in Chart 1. For example, the attributes _type.container and _type.value specify the data TYPE for the data item _crystal.density (in Chart 1a) as a SINGLE, REAL number. Each attribute in these definitions serves to identify a particular property of the defined item.

Chart 2. An Example STAR Data File that Is Applied in Later Example Methods[a]

```
data_example_file

_cell_length.a                      6.100(12)
_cell_length.b                      5.900(11)
_cell_length.c                      4.920(5)
_cell_angle.alpha                   90.
_cell_angle.beta                    90.
_cell_angle.gamma                   90.8331(5)


 loop_
_symmetry_equiv.pos_as_xyz
  +x,+y,+z  -x,-y,+z

loop_
_atom_type.symbol
_atom_type.atomic_mass
  O     16.000
  C     12.011

loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
_atom_site.type_symbol

  o1  .880(1)  .280(2)  .100(3)  O
  c1  .280(1)  .620(2)  .150(3)  C
  c2  .720(1)  .620(2)  .100(3)  C
  c3  .500(1)  .250(2)  .100(3)  C

_crystal.density      ?
```



[a] This file contains a mixture of primitive and derivative data specifying the crystal structure of the molecule shown above. Note that there is a 2-fold symmetry rotor at the center of the ring, and the atom sites flagged with an asterisk are implied by this symmetry. The crystal density is included as an unknown value, i.e., "?".

The attribute _method.expression contains an expression for evaluating the defined item in terms of other defined items. The way that this is applied is best illustrated using a simple example. The data dictionary definition of the crystal density shown in Chart 1a has a method attribute _method.expression containing the simple dREL expression

```
_crystal.density<-1.6605*_cell.mass/_cell.volume
```

This expression defines the density *d* of a crystalline solid (in mg/m^3) as the ratio of the atomic mass within a crystal unit cell and the volume of that cell, with the mass in units of daltons and the volume in cubic angstroms. The _method.expression attribute enables the crystal density to be evaluated if both the mass and volume values are known. Otherwise, _cell.mass and _cell.volume are treated as *functions*[13] which need to be evaluated from their dictionary definitions, which are included as Chart 1b and 1c, respectively. This *reduction* approach to evaluation continues until either the density value is determined or all derivation pathways are exhausted. It is important to note that the relational expressions are kept as simple and as modular as possible. Second-order relationships, such as those needed to derive _cell.mass or _cell.volume, are not considered part of the _crystal.density method.

Chart 2 shows a small test data file in which the item _crystal.density is flagged as unknown with a "?". Parsing this file using dREL-compiled definitions shown in Chart 1 produces an output file shown in Chart 3. This contains a value of _crystal.density calculated from the method expressions, along with an appended standard uncertainty value derived from these expressions. Note that there are a number of other data values determined as a consequence of the derivation processes (shown in bold type).

One further lesson can be drawn from these simple example definitions. Note that the relational expression in the method of Chart 1a can be easily parsed to derive the formula

$$\mathrm{density}_{\mathrm{crystal}} = \frac{1.6605\mathrm{mass}_{\mathrm{cell}}}{\mathrm{volume}_{\mathrm{cell}}}$$

and is very similar to the textbook definition of the crystal density

$$d = \frac{1.6605M_{\mathrm{c}}}{V_{\mathrm{c}}}$$

Similarly, the expression for the volume of a crystal cell shown in the method of Chart 1b

$$\mathrm{volume}_{\mathrm{cell}} = \mathbf{a}_{\mathrm{cell\_vector}} \cdot (\mathbf{b}_{\mathrm{cell\_vector}} \times \mathbf{c}_{\mathrm{cell\_vector}})$$

is explicit because *vector* quantities can be used in the dictionary definitions. Although the expression for crystal

Chart 3. The output Data File from a dREL Run[a]

```
data_test_example

_cell_length.a                        6.100(12)
_cell_length.b                        5.900(11)
_cell_length.c                        4.920(5)
_cell_angle.alpha                     90.
_cell_angle.beta                      90.
_cell_angle.gamma                     90.8331(5)

_cell_reciprocal.gamma     89.1669(5)
_cell_vector.a    [6.099(12), -0.08869(18), 0.0]
_cell_vector.b    [0.0, 5.900(11), 0.0]
_cell_vector.c     [0.0, 0.0, 4.920(5)]
_cell.orthogonal_matrix
  [[6.099(12), 0, 0], [-0.08869(18), 5.900(11), 0], [0, 0, 4.920(5)]]

_cell.mass       104.066
_cell.volume      177.1(5)

loop_
_symmetry_equiv.pos_as_xyz
_symmetry_equiv.Seitz_matrix
  +x,+y,+z  [[ 1, 0, 0, 0], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]
  -x,-y,+z  [[-1, 0, 0, 0], [0,-1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]]

loop_
_atom_type.symbol
_atom_type.atomic_mass
_atom_type.number_in_cell
  O     16.000    2.0
  C     12.011    6.0

loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
_atom_site.type_symbol
_atom_site.symmetry_multiplicity
_atom_site.fract_xyz
  o1  0.8800(10)  0.280(2)  0.100(3) O  2  [0.8800(10), 0.280(2), 0.100(3)]
  c1  0.2800(10)  0.620(2)  0.150(3) C  2  [0.2800(10), 0.620(2), 0.150(3)]
  c2  0.7200(10)  0.620(2)  0.100(3) C  2  [0.7200(10), 0.620(2), 0.100(3)]
  c3  0.5000(10)  0.250(2)  0.100(3) C  2  [0.5000(10), 0.250(2), 0.100(3)]

_crystal.density           0.976(3)
```

[a] The input file is shown in Chart 2. The data items determined from dictionary methods are in bold type.

cell mass in the method of Chart 1c

$$\mathrm{mass}_{\mathrm{cell}} = \sum_{t}^{\mathrm{atomic\_type}} (\mathrm{number\_in\_cell}_t \times \mathrm{mass}_t)$$

is not as straightforward as for the density and volume, the syntax involving a Loop of products in the atom_type list matches the structure of the data file (see Chart 3). The in-place operator "+<-" in the method script signals a summation of these terms.

These introductory examples illustrate that dictionary attributes provide for more than just derivative evaluation. They restrict data quantities in terms of typing, enumeration (e.g., all values must be nonnegative real numbers), and expected units. These are used to validate existing data values.

THE REDUCTION APPROACH

As the crystal density definition has shown (see Chart 1), it is operationally advantageous to treat data items in dREL relational expressions as *functions*, and to employ a *reduction* approach to their evaluation. This simplifies the compilation and maintenance of dictionary definitions by enabling new method scripts to be added with little or no knowledge of data dependencies in the existing definitions. As a consequence, method scripts need only contain the most directly related items (e.g., in the density method these are the cell mass and volume items, not the cell vector items), with no prior knowledge of how these are to be evaluated. This type of modularity promotes simple, concise, and symbolic relational expressions.

Instead of placing the highest priority on execution efficiency, the dREL approach emphasizes the clarity of data relationships. This has important implications for the long-term retention of knowledge because algorithms are no longer obscured by the syntax of conventional programming languages. The philosophy underpinning the development of the dictionary relational expression language dREL is that *knowledge be retained in its most comprehensible human-readable form*, independent of current computational constraints. Because large improvements in computer performance are certain, human, rather than machine, efficiency has been made the main consideration.

The relative importance of expressing knowledge intuitively needs some elaboration. Most computer software is written in imperative procedural code that is largely incomprehensible to nonprogrammers. To illustrate this, we contrast the expression used for the volume definition in Chart 1b

$$V = \boldsymbol{a}\cdot(\boldsymbol{b} \times \boldsymbol{c})$$

where the volume V of a parallelepiped is expressed in terms of its cell *vectors* $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{c}$, with the conventional approach to coding a volume calculation in which the vectors are expressed in terms of the scalar cell lengths a, b, and c and angles α, β, and γ, as

```
s=(alpha+beta+gamma)/2.
V=2.*a*b*c*Sqrt(Sin(s)*Sin(s-alpha)*Sin(s-beta)*Sin(s-gamma))
```

This expanded code may be computationally efficient, but its intent is obscure, compared to the dREL expression shown in Chart 1b. The usual coding practices used in C, C++, and Fortran languages also complicate the *algebraic structure* of relationships, and this limits their use for noncomputational purposes, such as translation through algebraic manipulation software.

DESIGN CONCEPTS OF DREL

"*Conventional programming languages are growing evermore enormous, but not stronger. Inherent defects at the most basic level cause them to be both fat and weak: the primitive word-at-a-time style of programming inherited from their close ancestor—the von Neumann computer, their close coupling of semantics to state transitions, their division of programming into a world of expressions and a world of statements, their inability to effectively use powerful combining forms for building new programs from existing ones, and their lack of useful mathematical properties for reasoning about programs.*" Backus, 1978

Backus[11] expressed these widely shared concerns in his acceptance speech for the 1977 ACM Turing Award. Imperative programming languages are efficient because they mimic the von Neumann architecture, but in doing so they obscure the meaning of the computed algorithm. Despite this serious limitation, it has only been with the most recent improvements to computer performance that serious consideration has been given to alternative programming rationales, such as provided by *functional* languages. These have been successful with some numerical computing tasks,[12,13] but have been little used in scientific applications.

In the design of dREL, the benefits of the *functional*, *procedural*, and *object-oriented* programming paradigms have been considered. As emphasized earlier, we gave the execution efficiency of method scripts a lower priority compared to other dREL objectives, such as expression simplicity, conciseness, and clarity. It was rationalized that future improvements to computer performance would improve execution efficiency, but only a better language design would lead to similar efficiency gains for the average user. These considerations made the prospect of designing an intuitive language with good operational properties an achievable goal.

dREL Design Criteria. The principal design criteria for dREL were the following.

1. The relational expressions in a DDL method attribute of a dictionary definition must be linked operationally to all other DDL attributes within that definition, and to other related definitions in the dictionary. The language scope must encompass the entire contents of a dictionary in order that all data items and their attributes are accessible in the execution of a method script.

2. The execution of a method script can only produce a result if the values of dependent data items can be determined. Ultimately, each evaluation depends on the availability of one or more data values from the data file being parsed by the dictionary application. It follows that the dREL parser must be completely conversant with the STAR File syntax and structure, and must have access to particular known values in a data file in order that evaluation can occur.

3. Relational expressions should mimic corresponding mathematical formula as closely as possible. A comprehensive set of data "container" types, e.g., Complex, List, Tuple, Table, Vector, and Matrix must be supported by dREL and be manipulated without explicit iteration over the elements of the container type. *Operator overloading* based on operand types should be supported. An example of this is given in the penultimate statement of the code fragment in Chart 6 in which the argument $\boldsymbol{h}$*$\mathbf{S}$*$\boldsymbol{x}$ represents the multiplication of a matrix $\mathbf{S}$ by the row vector $\boldsymbol{h}$ and the column vector $\boldsymbol{x}$.

4. dREL must support a data type referred to as Measurement. These are data which are not exact; i.e., each number has an associated uncertainty value (e.g., the measurement value recorded as 10.42(6) means there is a 0.667 probability that the true value is in the range 10.36−10.48). The statistical uncertainties of measurement data in a relational expression must be handled automatically. That is, the evaluation of a method script involving any data of type Measurement must result in a value of this type and have an associated uncertainty value. Mathematically, the expression for calculating this uncertainty is a function of the partial derivatives of the original evaluation expression, and this can be generated using standard computer algebraic approaches. Accordingly, an expression language must promote the use of symbolic representations that can be differentiated algebraically.

Dictionary Attribute Requirements. The dREL design is dependent on the versatility and scope of the dictionary DDL syntax and attributes.[7,8] The most important of these considerations are as follows.

1. Each definition in a dictionary should be an encapsulation of the properties of a specific data item, and this entry be treated as a discrete *object*. During execution these dictionary objects are married to specific data values from a data file.

2. Data items should be classified into groups of related items, referred to in a STAR File as a *category*, and this classification is declared explicitly in the dictionary. In this context a data item may be considered an *attribute of the*

Chart 4. Definitions of Hierarchically Related Categories, Showing that the Category Key Items for the ATOM_SITE_ANISO Category Is Identically Equivalent to the key for the Parent Category ATOM_SITE[a]

```
save_ATOM_SITE

    _definition.id              atom_site
    _defintion_scope            CATEGORY
    _definition.update          1999-11-26
    _description.text
;
     The CATEGORY of data items used to describe atomic site information
     used in crystallograhic structure studies.
;
    _category.family_id         structure

    _category_key.item_id      '_atom_site.label'
save_
```

Parent category **Equivalent keys**

```
save_ATOM_SITE_ANISO

    _definition.id              atom_site_aniso
    _defintion_scope            CATEGORY
    _definition.update          1999-11-26
    _description.text
;
     The CATEGORY of data items that may be used to describe anisotropic
atomic
     site information in separate loop lists.
;
    _category.family_id         atom_site

    _category_key.item_id       '_atom_site_aniso.label'
save_
```

[a] See Chart 5 for an example of this sort of data.

category. For example, the data items _atom_site.label and _atom_site.fract_xyz (see Charts 4 and 5) are members of the ATOM_SITE category. Within this category they may be accessed through their data name extensions, label and fract_xyz.

3. A single data item may have a *list* of values. A category of such items may be thought of as a table in which the columns contain the values and the item data names form the column header. A category of list items must have at least one data item with unique values. This item is the *category key* for uniquely referencing a loop packet (i.e., a row in the table).

4. Data categories may be divided into three classes according to their roles. They are *major function*, *hierarchical*, and *derivation*. Identifying these roles assists in the correct definition of data classes and in how the method attributes will be parsed by dREL. This is best conceptualized as a *three-dimensional organizational model*. The *major function* categories each occupy one axis of organizational space. In structural sciences these would be the major data classes CRYSTAL, DIFFRACTION, STRUCTURE, and MODEL, representing the families of data separated by purpose, origin, or computational progression.

A second dimension of the organizational model separates *hierarchical* categories. Such categories form families of data that are potentially merged into a common parent category. For example, ATOM_SITE data is part of the STRUCTURE family, and ATOM_SITE_ANISO is a child member of the ATOM_SITE category. The knowledge of such relationships is important because dREL merges child categories containing list data into the lists of parent categories of the same family (see Charts 4 and 5).

The third organizational dimension separates *derivation* categories. These categories contain data that are derived from other data. They are separated in this dimension by the order in which they can be derived: primary, secondary, tertiary, and quaternary, such as with ATOM_SITE, MODEL_SITE, GEOMETRY, and ENTITY.

5. Data items in *list* categories related by *hierarchy* may be merged with "parent" categories in the same family because their *category key items are equivalent and interchangeable*. Chart 4 illustrates such a relationship for ATOM_SITE data, and Chart 5 shows how data in these list categories may be operationally "joined".

6. Data items belonging to *list* categories related by *derivation share common category keys*. These keys are explicitly specified in the dictionary so that lists related by derivation can be mapped onto each other.

dREL Operational Requirements. Operational requirements of dREL arise specifically from the need for assignments in the method scripts. These have had a significant bearing on the dREL design criteria.

1. Defined data *names* (i.e., tags) appearing on the right-hand side of relational expressions are treated by dREL as *functions*, in that they reference a specific value or they refer to a method script in the dictionary that can be evaluated.

Chart 5. Example of How List Category Data May Be Merged with Its Parent Category Data

a. Contents of a data file showing atom site data as separate list of coordinates and a list U values. As shown in CHART 4, the highlighted items are category keys for referencing packets of items in these lists.

```
loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
      o1    .5501  .6371  .1601
      o2    .4012  .5162  .2290
      o3    .2502  .5705  .6011

loop_
_atom_site_aniso.label
_atom_site_aniso.U_11
_atom_site_aniso.U_12
_atom_site_aniso.U_13
_atom_site_aniso.U_22
_atom_site_aniso.U_23
_atom_site_aniso.U_33
   o1 .035   .012   .003   .043   .001   .022
   o3 .048   .011   .021   .034   .009   .032
```

b. This data file contains the identical atom site items as in CHART 5a but now as a single list. Because the category `ATOM_SITE_ANISO` is linked hierarchically to `the` category `ATOM_SITE`, the single reference key `_atom_site.label` may be used in a merged list.

```
  loop_
  _atom_site.label
  _atom_site.fract_x
  _atom_site.fract_y
  _atom_site.fract_z
  _atom_site_aniso.U_11
  _atom_site_aniso.U_12
  _atom_site_aniso.U_13
  _atom_site_aniso.U_22
  _atom_site_aniso.U_23
  _atom_site_aniso.U_33
o1  .5501 .6371 .1601 .035  .012  .003 .043  .001.022
o2  .4012 .5162 .2290 ? ? ? ? ? ?
o3  .2502 .5705 .6011 .048  .011  .021 .034  .009 032
```

[a] As shown in Chart 4, the highlighted items are category keys for referencing packets of items in these lists. [b] Because the category ATOM_SITE_ANISO is linked hierarchically to the category ATOM_SITE, the single reference key _atom_site.label may be used in a merged list.

2. The evaluation of specific data items in a relational expression take place on an "as-needed" basis, just as in functional languages.

3. dREL statements involving an imperative syntax will be supported to control access of stored variables in certain cases.

DREL RUN-TIME SCOPING

Although the specification of the scoping rules for dREL relational expressions is the topic of a later paper, we give a summary of these requirements here to help explain the fundamental differences between dREL and other scripting languages. The run-time evaluation model employed by dREL has similarities to existing functional languages, but differs markedly in that it is based on the specific scoping requirements for accessing STAR File data. They are as follows.

1. A STAR data tag in a dREL script represents a *pointer* to either its value in an associated STAR data file or its method script from which a value can be derived. Once evaluated, the pointer is replaced by the value. In this way, evaluation occurs only once.

2. An evaluation in a method script is postponed if the process branches to another method script, and the new script assumes the same scope and context for data items from the previous process.

3. Only the data item within the current scope is evaluated. In the case of repeated (list) data items, only one *loop packet* of data is, by default, in scope. Examples of this scoping requirement are illustrated in Chart 6, where the _method.expression operates over the *single* reflection packet currently in scope.

4. For *list* categories, the *local-reference pointer* to the current loop packet (represented by the current value of the

Chart 6. The Dictionary Definition of the Complex Structure Factor

```
    _definition.id              '_refln.F_complex'
    _definition.scope            SINGLE
    _definition.class            ITEM
    _description.text
;
     The structure factor vector for the reflection calculated from
     the atom site data.
;
    _type.container              Single
    _type.value                  Complex

    _method.expression
;
With r  as  refln
      h   <-  r.hkl
      fc  <-  Complex (0., 0.)

   Loop a  as  atom_site  {
          x  <-  a.fract_xyz
          f  <-  r.form_factor_list [a.type_symbol]
          m  <-  a.symmetry_multiplicity * a.occupancy

      Loop s  as  symmetry_equiv  {
              S  <-  s.Seitz_matrix
              t  <-  _function.adp_factor

             fc +<-  m * f * t * ExpImag ( TwoPi * ( h * (S * x) ) )
      }
   }
      _refln.F_complex  <-  fc
;
```

defined category key) is the statement

With *local-reference* **as** *category-name*

Data items in the category *category-name* are addressed by appending their data name extension to the identifier *local-reference*. Hence, the statement

```
With a as atom_site
```

places the items in the current packet of the ATOM_SITE category in scope, and these can be accessed as, say, a.label, a.fract_xyz, etc. The scope of the "With" statement is unchanged from the prior evaluation sequence. For example, if the joined list shown in Chart 5b is being processed, and the current loop packet in scope has a category key value of _atom_site.label = o2, then the invocation of a new method containing the statement

```
With s as atom_site
```

will not affect the scope of a.

5. For *list* categories, packets of data are successively brought into scope by the statement

Loop *local-reference* **as** *category-name* {*Statement-block*}

For the example in 5 above, the statement

```
Loop a as atom_site {              }
```

implies that the pointer to all loop packets (identified in Chart 5b as the category key items _atom_site.label with the values o1 to o3) will be successively placed in scope.

FURTHER DREL APPLICATIONS

Some examples of complex evaluation methods are now needed to illustrate the full versatility of the dREL approach. The first example calculates the structure factor value for specific diffraction vectors, and this shows, in particular, the operation of the reduction process during data evaluation. The second example method determines the list of interatomic angles within a molecular structure, and this illustrates how ab initio lists of items are created from more primitive data in the derivation pathway.

Example 1. A Crystallographic Structure Factor Calculation. For crystallographic studies, the determination of molecular structure from diffraction data requires that the measured structure factors be compared with those values calculated from the proposed molecular model of individual atom sites. The calculated structure factor $F(\boldsymbol{h})$ for each diffraction point, $\boldsymbol{h} = h, k, l$

$$F(\boldsymbol{h}) = \sum_{j=1,N} \mathbf{f}_j(s)\mathrm{e}^{-\mathbf{B}j(s)}\mathrm{e}^{-2\pi i\boldsymbol{h}\cdot\boldsymbol{r}_j}$$

is a Fourier transform expressed as the summation of the N atom sites in the unit cell, where $\mathbf{f}_j$ is the atomic scattering as function of the scattering direction s and $\mathrm{e}^{-\mathbf{B}_j(s)}$ is the atomic displacement factor and is a function of the scattering direction s and the atomic site coordinate vectors $\boldsymbol{r}_j$ (= x_j/a, y_j/b, z_j/c). When the crystal unit cell contains atomic sites related by space group symmetry, the expression for the

Chart 7. A Typical Derivation Pathway for the Evaluation of the Structure Factor

Method A

```
With r  as  refln
  h   <-  r.hkl                                    # A.2
  fc  <-  Complex (0., 0.)

  loop a  as  atom_site  {          # A.3
    x <-  a.fract_xyz
     f <- r.form_factor_list[a.type_symbol]
     m <-  a.symmetry_multiplicity * a.occupancy

     loop s  as  symmetry_equiv  {
       S  <-  s.Seitz_matrix
       t   <-   _function.adp_factor
       fc +<-  m * f * t * ExpImag ( TwoPi *
                      ( h * (S * x) ) )
     }
   }
   _refln.F_complex  <-  fc                # A.5
```

Method B

```
With r  as  refln
  s <- r.sin_theta_over_lambda
  Loop t as atom_type  {
    if (_diffrn.radiation_probe ==
                              'neutron')
      f <- t.length_neutron
    else if (s < 2.0 ) {
      f <- (t.c
         + t.a1 * Exp (-t.b1 * s*s)
         + t.a2 * Exp (-t.b2 * s*s)
         + t.a3 * Exp (-t.b3 * s*s)
         + t.a4 * Exp (-t.b4 * s*s) )
    } else
      f <- Exp (t.c0 + t.c1*s
         + t.c2*s*s + t.c3*s*s*s)

   refln.form_factor_list[t.symbol] <- f
}
```

Method C

```
With a  as  atom_site
With r  as  refln
With s  as  symmetry_equiv
With l  as  cell_reciprocal_length

if (a.apd_type == 'Uani')   {
  h <- r.hkl
  S <- s.Seitz_matrix
  U <- S * a.matrix_U * Transpose ( S )
  C <- [-[l.a,0,0],[0,l.b,0],[0,0,l.c]-]

  u <- Exp (-2 * Pi **2 * ( h * ( C * ( U * ( C * ( h ))))))
 }
 else             {
  u <- Exp (-8*Pi**2* a.U_iso_or_equiv *
r.sin_theta_over_lambda**2)
 }
  _function.adp_factor <- u
```

structure factor becomes

$$F(\mathbf{h}) = \sum_{j=1,U} \mathbf{f}_j(s) \sum_{k=1,M} e^{-\mathbf{B}_j(s)} e^{-2\pi i \boldsymbol{h}\cdot\mathbf{S}_k\mathbf{r}_j}$$

where $\mathbf{S}_k$ is the Seitz matrix for the kth symmetry transformation, and the summation is partitioned into the number of unique atomic sites U and the number of symmetry matrixes $\mathbf{M}$ ($N = U\mathbf{M}$). The dictionary definition for a calculated structure factor is shown in Chart 6, and the attribute _method.expression contains a relational expression representing the above equation.

The different attributes in a dictionary definition contribute to the evaluation of _refln.F_complex, which is a *single complex* number belonging to the REFLN category. Its evaluation from _method.expression is dependent on the values of other data items, _atom_site.fract_xyz, _atom_site.type_symbol, etc., and these must be available from either an associated data file or through the method scripts in their definitions. The evaluation of the calculated structure factor involves the following basic steps. These steps are typical of all dREL evaluation processes.

Step 1. A STAR file containing list data in the category REFLN is parsed sequentially. Each loop packet can be uniquely identified by the value of its category key _refln.hkl. The evaluation of each _refln.F_complex in the REFLN list may be requested simply by the presence of a "?" in place of a value. The evaluation of _refln.F_complex is made for each loop packet independently.

Step 2. If the value _refln.F_complex for a specific _refln.hkl value is available from a *prior* evaluation, it will be accessed directly, and the _refln.F_complex method script is not processed.

Step 3. If the value of _refln.F_complex is unknown for the specific _refln.hkl value, the parser processes the _method.expression script inserting the values of the referenced data items from either the associated STAR data file or from processing the defined method scripts of these referenced data items.

Scheme 1

The list of atomic coordinate data (category ATOM_SITE) are present in the data file.
↓
The list of "connected" atom sites in a molecule (category MODEL_SITE) are calculated.
↓
The list of molecular angles (category GEOM_ANGLE) are calculated.

Chart 8. The Definition of the Data Category GEOM_ANGLE Showing the Method Attribute that Generates a Complete List of Molecular Geometry Angles from the MODEL_SITE List

```
    _definition.id               geom_angle
    _definition.scope            CATEGORY
    _definition.class            ITEM
    _definition.update           2000-05-26
    _description.text
;
     The CATEGORY of data items used to specify the geometry angles in the
     structural model as derived from the atomic sites.
;
    _description.compact        'BondAngleList'
    _category.family_id          geom
    _category_key.item_id      '_geom_angle.id'

    _method.expression
;
     dmin <- _geom.bond_distance_min

      Loop  m1  as  model_site  :i  {   # loop vertex model site

        rad1  <-  m1.radius_bond + _geom.bond_distance_incr

         Loop  m2  as  model_site  :j  {  # loop first target site

          If (i==j or m1.mole_index != m2.mole_index) Next
          v1 <- m2.Cartn_xyz - m1.Cartn_xyz
          d1 <- Norm (v1)

          If (d1<dmin or d1>(rad1+m2.radius_bond))  Next

          rad2  <-  m2.radius_bond + _geom.bond_distance_incr

           Loop  m3  as  model_site  :k>j  {  # loop second target site

            If (i==k or m1.mole_index != m3.mole_index) Next
            v2 <- m3.Cartn_xyz - m1.Cartn_xyz
            d2 <- Norm (v2)

            If (d2<dmin or d2>(rad2+m3.radius_bond)) Next

             angle <- Acosd ( v1*v2 / (d1*d2) )

            geom_angle( .id        <- Tuple ([m2.id, m1.id, m3.id]),
                        .distances <- Tuple ([d1, d2]),
                        .value     <- angle )
     }   }   }
;
```

A typical derivation pathway for Step 3 in this calculation is shown in Chart 7. Method A is the _method.expression script for _refln.F_complex. Methods B and C are applied only if the values of _refln.form_factor_list and _function.adp_factor are not available from the data file or from a prior evaluation. The arrows in Chart 7 represent the branches to the methods of the items _refln.form_factor_list and _function.adp_factor. The parser continues to "stack" incomplete evaluations until all the needed items have been determined or the possible derivation pathways have been exhausted.

Example 2. Creating ab Initio List Data. This example is intended to show how lists of derivative data are generated ab initio from the dictionary definitions in the event that a requested data item is not present in a data file. In previous examples the data name of the required data item was present in the data file, but with a missing value. i.e., "?". When there are no tags in the data file for a requested category of data, the _method.expression within the *category* definition is used to construct a list of data items for this category. This example also shows how the calculation sequence flows automatically to other categories referred to in the method

Chart 9. Examples of Ab Initio List Data Created by a Category Method

a. An excerpt from a data file output from a *dREL* run showing the generated MODEL_SITE data.

```
loop_
  _model_site.index
  _model_site.id
  _model_site.radius_bond
  _model_site.radius_contact
  _model_site.fract_xyz
  _model_site.Cartn_xyz
1  ['o1','1_555']  0.74 1.99 [0.8800(10),0.280(2),0.100(3)] [5.367(12),1.574(12),0.492(15)]
2  ['c1','2_665']  0.77 2.02 [0.7200(10),0.380(2),0.150(3)] [4.392(11),2.178(13),0.738(15)]
3  ['c2','1_555']  0.77 2.02 [0.7200(10),0.620(2),0.100(3)] [4.392(11),3.594(14),0.492(15)]
4  ['c3','1_555']  0.77 2.02 [0.5000(10),0.250(2),0.100(3)] [3.050(9), 1.431(12),0.492(15)]
5  ['c3','2_665']  0.77 2.02 [0.5000(10),0.750(2),0.100(3)] [3.050(9), 4.381(14),0.492(15)]
6  ['c2','2_665']  0.77 2.02 [0.2800(10),0.380(2),0.100(3)] [1.708(7), 2.217(13),0.492(15)]
7  ['c1','1_555']  0.77 2.02 [0.2800(10),0.620(2),0.150(3)] [1.708(7), 3.633(14),0.738(15)]
8  ['o1','2_665']  0.74 1.99 [0.1200(10),0.720(2),0.100(3)] [0.732(6), 4.237(14),0.492(15)]
```

b. An excerpt of a data file output from a *dREL* run showing the generated GEOM_ANGLE data.

```
loop_
  _geom_angle.value
  _geom_angle.distances
  _geom_angle.id
  118.1(13)   [1.174(12),1.437(13)]   [('o1','1_555'),('c1','2_665'),('c2','1_555')]
  115.9(11)   [1.174(12),1.556(10)]   [('o1','1_555'),('c1','2_665'),('c3','1_555')]
  116.5(11)   [1.437(13),1.556(10)]   [('c2','1_555'),('c1','2_665'),('c3','1_555')]
  119.9(12)   [1.437(13),1.555(11)]   [('c1','2_665'),('c2','1_555'),('c3','2_665')]
  120.1(9)    [1.556(10),1.555(9)]    [('c1','2_665'),('c3','1_555'),('c2','2_665')]
  120.1(9)    [1.555(11),1.556(10)]   [('c2','1_555'),('c3','2_665'),('c1','1_555')]
  119.9(10)   [1.555(9),1.437(13)]    [('c3','1_555'),('c2','2_665'),('c1','1_555')]
  116.5(10)   [1.556(10),1.437(13)]   [('c3','2_665'),('c1','1_555'),('c2','2_665')]
  115.9(10)   [1.556(10),1.174(10)]   [('c3','2_665'),('c1','1_555'),('o1','2_665')]
  118.1(13)   [1.437(13),1.174(10)]   [('c2','2_665'),('c1','1_555'),('o1','2_665')]
```

expression. In this instance the categories form a derivation pathway that link the *atom coordinate data* (available from the data file) to the methods that evaluate the *angles between bonded atomic sites in a molecule*. The derivative pathway is shown in Scheme 1. Note that this sequence of calculations requires the generation of an intermediate list of connected sites, a step that arises automatically from the relationships of the dictionary definitions.

With the request for molecular angles, the dREL evaluation process proceeds as follows.

Step 1. The request for either _geom_angle.value or the GEOM_ANGLE list cannot be satisfied from the data items present in the STAR data file shown in Chart 2. This file contains the crystallographic results for the hypothetical molecular structure shown.

Step 2. The parser then refers to the attribute _method.expression in the GEOM_ANGLE category definition, shown in Chart 8, for the expressions needed to calculate the interatomic angles from the MODEL_SITE list of connected atom sites. However, these sites are also not available from the STAR file.

Step 3. The parser then refers to the attribute _method.expression in the MODEL_SITE category definition for the expressions needed to calculate the connected atoms sites in the molecule from the ATOM_SITE list of symmetry-unique atom sites.

Step 4. The ATOM_SITE list of sites are present on the STAR file (see Chart 2), and the parser proceeds to generate the connected atom sites based on the algorithm in the MODEL_SITE category definition. This results in the MODEL_SITE list shown in Chart 9a.

Step 5. The parser process stack now reverts to the algorithm in the GEOM_ANGLE category definition. This shows that if the positions of three connected atoms in the MODEL_SITE list are known in terms of the vectors $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ in a Cartesian system, then the angle ϕ subtended at site 2 is

$$\mathrm{Cos}\ \phi = \mathbf{v}_{21}\cdot\mathbf{v}_{23}/d_{21}d_{23}$$

where $\mathbf{v}_{21} = \mathbf{r}_1 - \mathbf{r}_2$; $\mathbf{v}_{23} = \mathbf{r}_3 - \mathbf{r}_2$ are the interatomic vectors and $d_{21} = |\mathbf{v}_{21}|$; $d_{23} = |\mathbf{v}_{23}|$ are the interatomic distances. This step produces the GEOM_ANGLE list shown in Chart 9b.

CONCLUSION

We have shown that the addition of relational expressions as method attributes of STAR data dictionaries can promote their use as *active* knowledge bases. The rationale and the purpose of these attributes are described. These provide for much greater clarity and precision in the definition of data. The availability of method scripts for use in future discipline-specific dictionaries, and in laboratory dictionaries containing local information, will promote the scientific exchange and exploitation of data, well beyond that currently achievable.

Relational expressions in dREL are simple, concise, and algorithmic, and this greatly improves the human readability and interpretation of definition information. Additionally, because method attributes are executable, derivative data can be directly evaluated as needed. This means that much derivative data need not be stored in data files, reducing the volume of data exchanged and archived.

Access to executable method scripts in dictionaries offers the promise of new computing paradigms for future scientific applications. Calculations can be initiated simply by requesting the value of a specific data item. The calculation sequence and algorithms will be determined entirely by the method attributes encountered in the evaluation pathway.

A prototype parser for dREL may be viewed at the Web site www.cs.uwa.edu.au/dREL. The executable versions of the examples presented in the paper are available at that address, as is information on future versions of dREL and related publications. The source code of dREL will in the future be distributed as open system software entitling users to free use and code access, conditional that new developments can correctly parse supplied compliance scripts.

REFERENCES AND NOTES

(1) Hall, S. R. The STAR File: A New Format for Electronic Data Transfer and Archiving. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 326−333.

(2) Hall, S. R.; Spadaccini, N. The STAR File: Detailed Specifications. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 505−508.

(3) Hall, S. R.; Allen, F. H.; Brown, I. D. The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography. *Acta Crystallogr., Sect. A* **1991**, *A47*, 655−685.

(4) Electronic submission of material in the CIF format is recommended by American Chemical Society (pubs.acs.org:80/supmat/suppdoc.html), Royal Society of Chemistry (London) (www.rsc.org/is/journals/authrefs/submit.htm), International Union of Crystallography (www.iucr.org/iucr-top/cif/index.html), CSIRO/Australian Academy of Science (www.publish.csiro.au/journals/ajc/), and Oldenbourg Verlage (www.oldenbourg.de/verlag/zkristallogr/).

(5) Submission of data in the CIF format is requested by Nucleic Acid Database, Rutgers University (ndbsever.rutgers.edu), Protein Data Bank, Rutgers University (www.rcsb.org/pdb), Cambridge Structural Database (www.ccdc.cam.ac.uk), Powder Diffraction Database (www.icdd.com), Inorganic Crystal Structure Database (www.fiz-karlsruhe.de/stn/Databases/icsd.html), and BioMagResBank (www.bmrb.wisc.edu).

(6) IUCr-approved dictionaries are listed at www.iucr.org/iucr-top/cif/#dics.

(7) Hall, S. R.; Cook, A. P. F. STAR Data Definition Language: Initial Specification. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 819−825.

(8) Westbrook, J. D.; Hall, S. R. A Dictionary Description Language for Macromolecular Structure. Draft DDL Version 2.1.1; 1995 (available from ndbserver.rutgers.edu/mmcif/ddl).

(9) IUCr PDF verion of the 2.1 Core CIF Dictionary (available from ftp://ftp.iucr.org/cif_core_2.1.dic.pdf).

(10) *Acta Crystallographica* validation criteria (www.iucr.org/iucr-top/journals/acta/dv.html).

(11) Backus, J. Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs. *Commun. ACM* **1978**, *21*, 613−641.

(12) Halfant, M.; Sussman, G. J. Abstraction in Numerical Methods. *Lisp Funct. Program. ACM* **1988**, 1−7.

(13) Hughes, J. Why Functional Programming Matters. *Comput. J.* **1989**, *32*, 98−107.

CI000009K