DRAFT — May 12, 2003 03:32 pm

Data Models for Spectra: Abstractions for Quantities and Coordinates

Steve Lowe and Jonathan McDowell
SAO
{slowe, jcm}@head-cfa.harvard.edu
May 8, 2003

1. Embedding the Spectral Data Model in an Abstract Data Model

The purpose of this document is to complement *Spectral Data Models, Draft May 7, 2003,* by McDowell and Lowe (http://hea-www.harvard.edu/~jcm/vo/vospec.ps) by describing how we envision the embedding components of the spectral data model in a general data model framework. In the present document, we focus no the aspects of quantities and coordinate systems.

A theme that we wish to stress in the design of Data Models is viewing the detailed requirements of each data regime as specializations of abstract model concepts. Moreover, we propose that this structure be explicit in the specification. That is to say, the Data Model consists of high-level objects for representing of data and data relationships, and the objects representing spectral data or position data are defined as subclasses derived from these high-level ones.

The reason for this orientation is provide, in so far as is possible, uniformity in the way in which physical properties are manipulated in software. Of course, there will be instances in which some property will have idiosyncratic attributes that fit poorly into the general framework, and these attributes may need to be handled as special cases. We do not recommend a Procrustean policy of force-fitting familiar physical data types into unnatural representations. Nevertheless, we will strive to identify common structure and to use this commonality to design a more concise conceptual framework.

At the same time, the design needs to be concrete, something that can be implemented in the near term as a functioning system. We think that by carefully choosing the elements to generalize we can achieve a model which is open to future expansion, yet is the same time simple in concept.

2. Quantities

The word “quantity” is often used in two senses, one conceptually to refer to something that we want to measure or specify a value for, and secondly for the value itself. Is the “the amount of milk in this glass,” which I don't know and plan to measure, the quantity; or is it “250 ml of milk.” Using “quantity” for the first and “value” for the second might make sense, but this may conflict with another usage under development. The names used are not important in the short run, but probably should be chosen more carefully as

the model is developed. There is an additional consideration of how to accomodate values that have uncertainties, such as “250 ml plus or minus 25 ml of milk.”

For the present purposes, we will use the terms *quantity concept*, *quantity value* and *quantity measurement* for these three usages. We have no objection at all to revising these names.

3. Quantity Values, Conversions and Representations

To specify a quantity value the pure numerical value must be augmented with units and a coordinate system. Different combinations of number, units and system can correspond to the same value for the quanitity. We all had this idea driven into our brains in high school! In addition, in some cases two different quantity concepts may be treated as equivalent in a context in which there is a constitutive relationship between them; an example is that the quantity concepts wavelength, frequency and photon energy of light are treated as interchangeable because in that context the wavelength and frequency are linked by the constant speed of light and the photon energy is linked to frequency by Plank's law.

Another way to say this is that in many cases a given quantity concept has several alternative ways in which they can be represented. An automatic process working with data needs to be able to determine *whether* and *how* conversions can be done between two representations.

In the context of specifying a value for a quantity, we are proposing to unify the ideas of unit, coordinate system and constitutive relationships into a single idea of a *representation* (or *quantity representation*). The complete representation for a value will be implemented as a chain of representation units which correspond to the units, coordinate system and so on.

To specify a quantity value will require a numerical value and a complete representation. See Figure 1. The Quantity Value object links to a Representation Object, which in turn contains a list of parts of the representation. Each part of the representation is a reference to a Quantity Concept object. Each Quantity Concept object has several attributes:

a) A name for the quantity, e.g., “wavelength”

b) References to other quantities that can serve as representations for this one

c) Conversion functions between the representations.

Thus, in the figure, the conversion functions between Angstroms, meters and so on are maintained in the Length object. Given a pointer to the Quantity Value object shown, a program can determine that the value, 6600, specifies a “Wavelength” represented in “Angstroms.” By examining the representation chain, it can find alternative

representations for the this “Wavelength,” e.g., the quantity is also a “Length” that has the alternate representation of “meter.” Finally, it can locate the conversion function (in the Length object) that will take in the Quantity Value object and return a new one with the wavelength specified in meters. Note that search for alternatives *can only move in the direction of the linking arrows.*

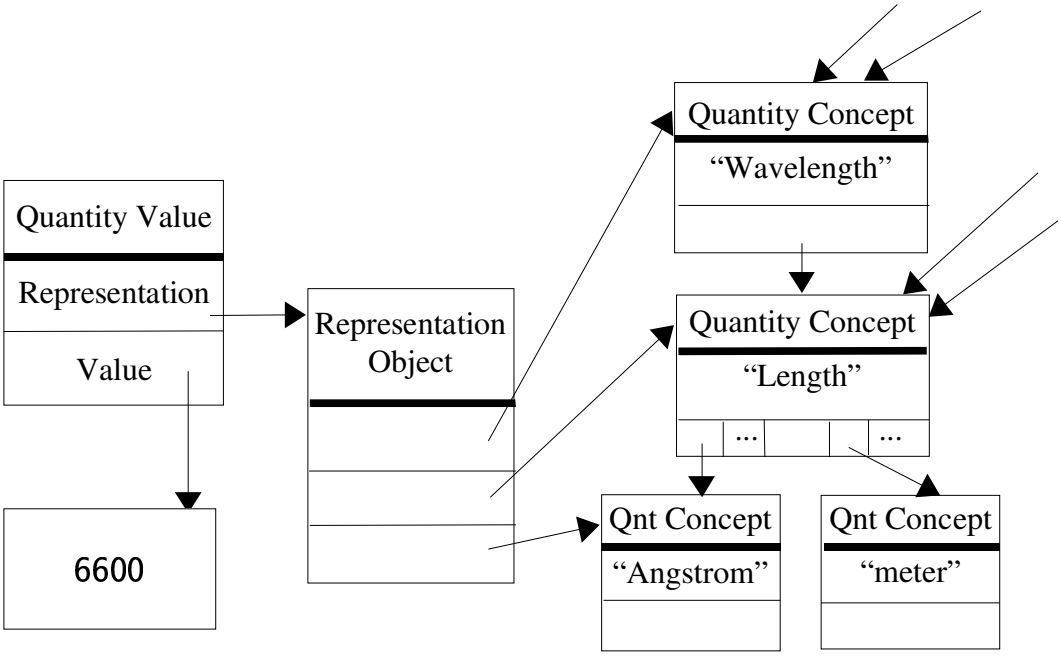

Figure 1: The description of a wavelength of 6600 Angstroms. The Length object contains the information for converting 6600 Angstroms to meters. The incoming arrows on the right indicate connections with a larger network of quantity objects.

The objects on the right in the diagram are part of a larger network of Quantity Concept objects that represent the relationships between physical quantities, coordinate systems, reference frames, units and so on. There may be many things that can be represented as a Length, and these can all share the same Quantity Concept “Length” object. However, a program processing the Quantity Value shown here can only follow the links in the directions shown; hence, it can find the “meter” object, but cannot find other objects that are representable as “wavelength” or “length.” If

One thing to note is that there are just three object classes here: Quantity Concept, Quantity Value and Representation. That is, we don't define a class for every quantity, just a class instance (i.e., a data object). Thus, the Quantity Concept hierarchy can be represented as data rather than as program code.

4. Quantity Conversion Object

There is an important fourth class of objects that was not shown explicitly in Figure 1. This is the class that implements conversion functions. In some cases, there will be no recourse but to implement a specialized conversion function in program code. However, many cases can probably be grouped into a small number of parameterized calculations. In particular, unit conversions will correspond to simple tables of numbers, and many coordinate transformations may be posed as matrix operations or polynomial evaluations. A class (or class hierarchy) will be needed to implement these functions.

In some cases, the generic conversion object will need to store a parameter, such as the *c* or *h*.

5. The Spectral Parameter for Electromagnetic Radiation

For spectra of electromagnetic radiation, we form a high-level concept that, to our knowledge, doesn't have a name: *the spectral parameter for electromagnetic radiation*. This is the "thing" that we think of as variously represented as wavelength, frequency or photon energy. The hierarchy of representations under this quantity is:

Spectral Parameter for Electromagnetic Radiation

- Spectral Parameter for 1-D Wave (with speed *c*)
  - Wavelength
    - Length
      - Angstrom
      - Meter
      - ...
    - Frequency
      - Hz
      - MHz
  - Photon Energy
    - Energy
      - eV
      - erg