A Proposed Semantics for the Sampled Values and Metadata of Scientific Values

Joseph Phillips

De Paul University
School of Computing and Digital Media
243 S. Wabash Ave
Chicago, IL 60604, USA
jphillips@cdm.depaul.edu

Abstract

We propose semantics for the manipulation of sample values and metadata for scientific data. Our approach uses domain knowledge from an ontology; computational knowledge about dimensions, units, coordinate systems, *etc*; and cultural and linguistic norms. It recombines annotated values with specialized operators. This semantics does several types of dynamic error checking and results in "self-documented computation".

1. Introduction

"Garbage-in, garbage-out." Professional computer scientists know this and craft software accordingly. Beginning computer scientists learn (and if needed, re-learn) this principle on the path towards professionalism. Some lay-public have caught on, and occasionally mock software engineering for this.

What, however, is "garbage"? On its ways both in and out it is the mismatch between how users and systems believe the Universe works.

This paper presents a principled approach to extend the knowledge of automated reasoning systems in scientific domains. It makes sample values and metadata an integral part of any value. These metadata tell:

- the domain of values (including dimensions, limits and units)
- the axis
- the subject being described
- the attribute of the described subject
- when the data hold
- where they hold
- who is responsible for it
- why the value was obtained
- any instruments used
- any methodology employed

Our values are annotated far more richly than is common in computing systems. That fact, however, serves as a severe indictment of contemporary computing systems because practicing scientists use these metadata to decide when (and *if*) to use scientific values.

Our paper is organized as follows. The next section motivates our work with examples of metadata. We follow with prior work, and discuss how we handle values, metadata, and operators. We discuss the ramifications of particular design choices. Then we conclude.

2. Motivation

Computing systems ought to use metadata because scientists place significance in reporting it. Take this passage from Bertozzi and Bednarski, 1991.

> The solution . . . afford(ed) 17.0 g (73% . .) of a colorless oil: mp 210 degrees C; IR (thin film) 2861, 834, . . . 661 cm^{-1}; ^{1}H NMR (400 MHz, $CDCl_3$) δ 0.04 (s, 6 H), 0.87 (s, 9 H) . . . (t, 2 H, *J* = 5.3); ^{13}C NMR ($CDCL_3$) δ -5.31, 18.33, . . . 72.62; mass spectrum (GC-MS) 305.2 (M -28(N_2), 43), . . , Anal. Calcd for $C_{14}H_{31}O_4N_3Si$: C, 51.04; H, 8.26; N, 12.75. Found: C, 51.56; H, 9.64; N, 12.16.

The melting point (210) has its units (Celsius) given with the implicit dimension temperature. The infrared spectroscopy peaks were given along with the technique (thin-film) and units (cm^{-1} is the wavenumber, or the inverse of the wavelength, which is another way of specifying the frequency of light). Two forms of nuclear magnetic resonance were used: "proton" (^{1}H) and "carbon-13" (^{13}C). The ^{1}H was tuned to radio frequency 400 MHz. Both had peaks whose offsets ("δ") are relative to the implicitly understood tetramethylsilane. Further, for proton NMR, the peaks have multiplicity (s = singlet, d = doublet, t = triplet, etc.), area (the number of H's) and separation between/among the multiple peaks (given by the *J* value). For NMRs whose frequencies differ from 400 MHz the δ-values are constant but the *J*-values will

change. The mass spectrometer technique was listed along with the method of separation: GC or "gas chromatography" using molecular Nitrogen (N_2). This gave peaks including the "molecular ion" one at 305.2 atomic weights. Finally, the percent by mass of each type of atom was computed theoretically and by chemical analysis.

This example shows metadata's importance in several regards. First, had other chemists analyzed the same compound they would expect similar numbers *except* for the *J*-values, which depend on the frequency of their NMR. Second, the units are not uniform: for NMR spectroscopy the frequency is in MHz while for infrared spectroscopy it is wavenumbers. Third, a clear distinction is made between the theoretical and the found mass percentages.

Neglecting metadata has had expensive and catastrophic outcomes. One example was NASA's Mars Climate Orbiter (NASA 1999), which was destroyed (in part) because angular momentum impulse data was in English (pound-seconds) rather than metric (Newtons). Another example, also from rocketry, was the first attempted launch of the European Space Agency's Ariane 5 rocket. Software from the Ariane 4 was reused, but the Ariane 5's more powerful engines triggered a bug in the conversion from a 64-bit floating point value to a 16 bit integer (Lions, 1996).

Clearly one lesson from these incidents is that metadata ought to be an integral part of a value rather than merely a label in some comment, buried in the user manual, or (worse) left in the minds of the programmers. Further, computing systems should properly carry this metadata through operations. Doing so can prevent or catch a host of errors related to dimensional analysis, unit conversion, range checking and general semantic integrity.

Proper metadata usage requires more knowledge. Some of this knowledge can be "built-in" to the definitions of our operators, but we will also require an annotated ontology and linguistic and cultural information.

A further complication about measurements is that they are often repeated. Several contemporary computing systems require scientists to substitute a rich distribution with a single value like a mean, weighted mean, median or mode. This, however, loses the variation (or the confidence of knowing the uniformity) of the distribution.

We address this issue by explicitly representing measurements as distribution vectors of floating point values, rationals or symbols. Values derived from measurements become (in principle) N-dimensional distribution matrices that have one dimension for each measurement that went into their formation.

The carrying forward and manipulation of distribution vectors and matrices will necessarily take more time and space. The growth in size becomes a significant concern and we outline a dynamic sampling technique to address it.

For the metadata apart from the distribution we believe the space is also the primary concern, but for a different reason. Although one important usage for the metadata is automatic error-checking, another is as "self-documenting computation" for the benefit of the user. The amount of computer-generated metadata annotation should be minimized (consistent with the mandate to sufficiently document) to lessen the burden on those who read it. The computation may be internally consistent but mismatch with some aspect of the world of which the system was not informed. Users will be relied upon to catch some external inconsistencies, and identifying the most salient metadata will help hold their attention.

3. Prior Work

This work is relevant to many areas but perhaps the closest is Computational Scientific Discovery (CSD). From the beginning CSD researchers have incorporated scientific knowledge into their programs. Classic examples include mass spectroscopy knowledge in *meta-Dendral* (Buchanan 1978), medical knowledge in *Mycin* (Buchanan 1984), and chemical reaction knowledge in *Mechem* (Valdés-Pérez 1992, 1993). Since these early programs researchers have either considered adding meta-data to their programs like *Inductive Process Modeling* (Bridewell *et al*, 2008), or have actually added basic knowledge of dimensions, units and limits in *Scienceomatic V* (Phillips 2003).

The situation for mathematical packages and simulation languages is similarly spotty. Mathematica, for example, can be made aware of dimensions and their units (Wolfram Research 2010). Matlab, however, needs an added package and has a quirky syntax (deCarvalho 2006).

With the rise of collaborative tools such as the World Wide Web, grid computing and other online services, many researchers have studied how to work with metadata at the level of simulations and whole experiments (*e.g*. Yang *et al* 2002).

The existence of these metadata integration schemes complement our efforts. After a computing system translates metadata from an XML schema into its own internal format the metadata must be accurately carried through its calculations. Only this will allow us to be more certain that the data coming out of the system is as accurate as the data going in.

4. Values

We distinguish four different types of values.

(1) **Naturally important values** are believed to have *a priori* importance because of their mathematical or other properties. Examples include the non-negative integers, π and e. Integers and rationals can be represented with arbitrary precision limited only by the availability of memory. Irrational values can be represented symbolically with full precision.

(2) **Culturally important values** are defined to be important. One example is conversions, *e.g*. 100 centimeters in 1 meter. A special case concerns when

phenomena are believed to interrelate dimensions in a fundamental fashion. For example the speed of light is *defined* to be 299,792,458 meters/second. Given that 1 second has a physically-based definition, this serves to define the length of a meter. Culturally-defined values are generally integers and can be stored with arbitrary precision.

(3) **Measurements** are readings of some aspect of the Universe. In principle they are limited only by the set of Turing-computable numbers. However, in practice they come from machines that read and store them in common IEEE floating point formats.

(4) **Derived values** result from computations. Values derived from the naturally and culturally important values may be expressed with arbitrary precision as symbolic operations upon rationals and symbols (*e.g.* sqrt(2)). However, as soon as a floating point value is used in the calculation such arbitrary precision is wasted due to the uncertainty inherent in the measurement.

We make two assumptions about the values in the distribution vector of a measurement value.

(1) Although measurements are repeated, there is the expectation that the values will fall within a relatively narrow range. This is the case when, for example, a presumably "static" thing is measured multiple times, or when a "dynamic" thing is measured simultaneously with several measurers. *We explicitly assume that the variance among values in the same distribution vector will be "small."* When this expectation does not hold it is proper to split the value up into distribution vectors small enough that the expectation does hold.

(2) Further, we expect that values in measurement value distribution vectors are "representative" and not systematically skewed somehow.

5. Value Metadata

Metadata require a language. We use symbols as unique nominal values to represent concepts which have properties. One such property is the standard isA/instanceOf relationship among sets and their members used to build ontologies. Symbols may be organized in lists.

Sets have two additional properties. First, there exists a symbol unique to each set that means "*unspecified member*" of that set. If symbol *set* represents some set then *unspecMbr(set)* represents its unspecified member. Second, sets are either labeled as having all of their members be functionally interchangeable with each other (*e.g.* as with pieces of equipment off the same assembly line that functionally differ only by serial number), or not. This is given with the boolean property over set symbols *areMembersInterchangeable()*.

Members also have a binary relation. The function *commonOverlap(member1,member2)* returns a member symbol that represents the overlap that members *member1* and *member2* share. For example, if *member1* is *twentiethCentury* and *member2* is *year1967CE* then *commonOverlap()* would return *year1967CE*.

We also define three functions that dynamically create new symbols or map to existing ones based upon cultural and linguistic norms.

The function *collection(memberList)* takes a list of symbols representing members and returns a member symbol representing a collection of those members. The collection is not a formal ontological set for purposes of organization, but just a bunch of things that happen to be lumped together for some reason or another.

The function *composite(memberList)* also takes a list of symbols representing members. It returns a member symbol that represents a new object with structural and/or functional integrity[1] that results from their grouping. For example, attaching a "horse" to a "buggy" conceptually yields a new object with its own integrity, even while being compositionally created.

The function *resultingAttr(op,member)* (and *resultingAttr(*$member_L$*,op,*$member_R$*)*) return the symbol for the attribute that results from applying unary (or binary) operator *op* on the given member(s). As with *composite(memberList)* it is expected that there will be significant cultural and linguistic content in the definition of these functions. For example, in English we call length times length "area", length times area "volume", *etc*.

The metadata associated with each value is inspired by the basic questions in journalism: *who*, *what*, *when*, *where*, *why* and *how*. To this list we add *domain*, *axis*, *which aspect*, and *using what*.

(1) **Domains** are composite objects that may have three properties: dimensions, units and limits. **Dimensions** are represented as a list of dimension symbols raised to integer powers[2]. Associated with each numeric dimension is one **primary unit** symbol. Secondary units of the dimension have a linear conversion[3] to the dimension's main units.

There are six notions of limits associated with each numeric domain.

(a) *loRangeLimit* and *hiRangeLimit* define hard logical endpoints for a domain. For example, latitude is defined to be between -90 to +90 degrees.

(b) *loSystemLimit* and *hiSystemLimit* define the endpoints of the system being measured. For

---

[1] But not necessarily touching.

[2] For non-complex numbers there are some special cases of expressions that have non-integer powers, like the exponents of reagent concentrations in chemical reaction rate equations. These, however, tend to be an anomaly and may be approximations. Concentrations, after all, are based on volumes which are lengths raised to the third power.

[3] In *y=mx+b* format, in most cases *b* will be 0. One important exception is temperature: *C=1*K + 273.15*.

example, a certain swimming pool can contain between 0 to 25,000 liters.

(c) *loDetectLimit* and *hiDetectLimit* define the range outside which phenomena cannot be detected.

(d) *loSaturateLimit* and *hiSaturateLimit* define the range inside which phenomena empirically lie, even though in principle it could go more extreme.

(e) *loReliableLimit* and *hiReliableLimit* define a softer limit on detection than *loDetectLimit* and *hiDetectLimit*. Outside that range values are not reliably detected.

(f) *loObservedLimit* and *hiObservedLimit* define the most extreme range inside of which phenomena actually have been observed.

The following conditions (and their high-limit equivalents) hold among the limits:

(a) *loRangeLimit* <= *loSystemLimit* <= *loSaturateLimit;*

(b) *loRangeLimit* <= *loDetectLimit* <= *loReliableLimit;*

(c1) *loSystemLimit* <= *loObservedLimit*;

(c2) *loDetectLimit* <= *loObservedLimit;*

Condition (a) is the *system constraint*. Condition (b) is the *instrument constraint*. Conditions (c1) and (c2) are the *data constraints*.

(2) **Axes** allow conversion between values whose origins have different translations and rotations in some coordinate system.

(3) **Subject** answers "*What?*" with a list of symbols. The value 299,792,458 meters/second for the speed of light has symbol *unspecMbr(photonSet)* in its subject list.

(4) **Attribute** answers "*Which aspect?*" with a list of symbols. The value 299,792,458 meters/second for the speed of light has symbol *speed* in its attribute list.

(5) **State** answers "*When?*" with a list of symbols. Associated with each is (a) a duration, (b) a midpoint time, and (c) a time axis (*e.g*. for conversion between different calendars).

(6) **Location** answers "*Where?*" with a list of symbols representing a place.

(7) **Authorship** answers "*Who?*" with a list of symbols. Each represents a person or organization deemed responsible for the value's measurement, computation, *etc*.

(8) **Reason** answers "*Why?*" with a list of symbols. For culturally important and measurement values these represent the (presumed) motivations of the members of the authorship in obtaining the value. For derived values these represent the "union" of the reasons that went into the calculation of the derived value. (The algorithm for computing unions is given below.)

(9) **Equipment** answers "*Using what?*" with a list of symbols. Each represents a significant piece of equipment used to obtain the value.

(10) **Method** answers "*How?*" with a list of symbols. Each represents a computational or experimental technique for used to obtain the value.

One could ask "*How much metadata should be given?*" We defer to the norms that scientists already follow, but suggest that now because metadata can be tracked automatically, more can be stated.

One could also ask "*Why use lists of symbols as opposed to sets?*" Although our algorithm makes minimal use of the ordering information, scientists might place importance on the order in which they give symbols. For example, the author who is listed first on a paper is generally considered the one who put the most work into it.

Rules for combining sample values

The rules for combining the sample portion of values rely on **vector sources**. A vector source is a symbol that identifies and gives information on a vector of values, *e.g*. by identifying the trials used in a measurement. A total ordering exists over vector sources so that lists of them may be sorted.

Consider a simple experiment in which a weight is dropped three times. Each time it has its mass taken (it is "weighed") and its velocity upon impact is measured. This yields two measurement vectors: one of masses $<m_1, m_2, m_3>$, and one of velocities $<v_1, v_2, v_3>$. Both measurement vectors came from the same trials thus both have the same vector source.

Having the same vector source lessens the multitude of values by preventing unobserved combinations. Say we want to multiply the mass by the velocity to compute the momentum. That both vectors have the same source means that we get the 3-numbered vector $<m_1v_1, m_2v_2, m_3v_3>$ (also with the same vector source) instead of the 9-numbered cross product matrix.

A unary operation applied to a vector or matrix gives a vector or matrix of equal dimensions and the same vector source(s) as the operand.

Binary operators are more complicated. Let the first operand be an M-dimensional matrix with an ordered list L_1 of M distinct vector sources. Let the second operand be an N-dimensional matrix with an ordered list L_2 of N distinct vector sources. Let L_1 and L_2 have O vector sources in common listed in L_C. The resulting matrix has M+N-O dimensions whose vector sources are the ordered merging of L_1 and L_2 with the duplicates given in L_C removed.

Even when duplicates are removed dimensionality and the size of the matrices is expected to grow. There are two approaches for dealing with this. One is when it becomes "too big" (where "too big" is defined by users and/or the implementing system) the matrix may be sampled to yield a reasonably-sized vector. This sampled vector gets a unique vector source, and associated with this source may be details associated with the sampling.

The other approach is to forgo the creation of matrices in favor of demand-driven sampling.

One purpose of the metadata discussed in the next section is to trap illegal values (*e.g*. divide-by-zero, square root of a negative number, *etc*.). If some values in a sample vector or matrix are illegal then they are

represented by the *null* symbol. Subsequent operations on that value will carry the null symbol forward. When it is believed that the value is inherently illegal in and of itself (rather than just the victim of an inappropriate operator) then that particular value can be noted as illegal within its vector source. This information can then be pushed forward to all values that directly and indirectly use the vector source, or can be dynamically checked by new operators as they are done.

Rules for combining metadata

Domains and axes have their own combination rules, and attributes try the *resultingAttr()* functions first, but the default algorithm for the other seven metadata fields is the same. We believe this uniformity follows from the nature of the questions, all eight of which ask for some description answerable by listing an individual that result from intersection, union, or a generalizing union.

(We emphasize that this is the *default* algorithm for computing metadata. The next chapter will show how specific operators override this algorithm in specific circumstances.)

All three combining approaches are required. Intersection is often needed for the state and location metadata. For example, consider a skinny antenna mounted atop a tall building. In one sense the whole building is now taller. However, more specifically it is the area on which the antenna stands, the *intersection* of the cross-sections of both building and antenna, that is now taller. Other cross-sectional area of the building remain the same height. The authorship metadata show the need for union. Bertozzi and Bednarski are unique individuals, both of whom created the compound data above, so their names should be listed individually. Lastly, the equipment metadata shows the need for a more generalizing notion of union. All 400 MHz NMRs are expected to yield very similar δ- and *J*-values. Indeed, specifying the make, model and serial number of the NMR may be seen as obfuscating detail, so all 400 MHz NMRs should be identified by their set.

An algorithm and subroutine are listed below.

```
SymbolList symbolListUnion
 (SymbolList symList1, SymbolList symList2)

SymbolList retList := [];
Symbol commonSym := null;
foreach symbol syml in symList1:
   if (matchSymbol(syml,retList,f) <> null)
      continue;
   commonSym := matchSymbol(syml,symList2,t);
   if  (commonSym = null)
      retList.append(syml);
   else
      retList.append(commonSym);
   endif
endforeach;

foreach symbol syml in symList2:
   commonSym := matchSymbol(syml,retList,f);
   if (commonSym = null)
      retList.append(syml);
endforeach;

return retList;
```

```
Symbol matchSymbol  (Symbol matchSym, SymbolList
symList, boolean shouldRemove)

int bestDist := -1;
int setDist := -1;
Symbol bestSym := null;
Symbol set := null;
Symbol removeSym := null;

foreach symbol symL in symList:
   if  (combineType() = intersection )
      bestSym := commonOverlap(matchSym,symL)
      if  (bestSym <> null)
         removeSym := symL;
         breakforeach;
      endif
   else if  (combineType() = union )
      if  (matchSym = symL)
         removeSym := bestSym := symL;
         breakforeach;
      endif
   else if  (combineType() = generalize )
      set := mostSpecificCommonSet(matchSym,symL);

      if(set<>null AND areMembersInterchangeable(set))
         setDist :=    ontologyDist(set, matchSym)
                     + ontologyDist(set, symL);

         if  (bestDist = -1 OR setDist < bestDist)
            bestSym := unspecMbr(set);
            bestDist := setDist;
            removeSym := symL;
         endif
      endif
   endif
endforeach

if  (shouldRemove AND removeSym <> null)
   symList.remove(removeSym);
endif

return bestSym;
```

The functions *unspecMbr(set)*, *commonOverlap(mem1,mem2)*, and *areMembersInterchangeable(set)* are described above. The ontology function *mostSpecificCommonSet(mbr1, mbr2)* takes two symbols representing set members and

returns a most specific set that includes both, or *null* if no such set exists. The ontology function *ontologyDist(set,mbr)* returns the smallest distance in the ontology graph between *set* and *member*. The function *combineType()* tells the type of combining to do based on the operator and metadata type.

For extension-addition (described below) *matchSymbol()* will return the time and space for when and where an antenna mounted on top of a building exists. However, it does other combinations for other metadata. For example, when told that all 400 MHz NMRs are interchangeable *symbolListUnion([varian400MHzNMR],[bruker400MHzNMR])* returns *unspecMbr(n400MHzNMRset)*. However, when told that people are unique *symbolListUnion([bertozziCarolyn], [bednarskiMark])* returns *[bertozziCarolyn, bednarskiMark]*.

Combining domains entails combining their three properties: dimensions, units and limits. Dimensions are combined by conventional dimensional analysis. Units are similarly combined with unit conversion. Limits are combined by applying the specified operator on the given limit, and using the more general limits to further constrain the result. For example, doubling an angle value representing latitude with *hiReliableLimit* = 89.5 degrees and *hiRangeLimit* = 90 degrees yields a value with *hiReliableLimit* = 90 degrees (instead of 2*89.5 = 179 degrees) so as not to violate the *instrument* constraint.

Combining axes is done in accordance to the coordinate system of which the axes are a part. This may involve translation of origins, rotations, and switching coordinate systems (*e.g*. among Cartesian, polar, spherical, *etc*.).[4]

6. Operators

The combination of values has been considered generically, but we require that our operators also preserve the desired metadata semantics. In this section we look at operator-specific metadata transformations.

Our analysis will be simplified by considering a small basis set of operator types, namely: addition, multiplication, exponentiation to rational powers, and metadatacasting. We follow the computer algebra tradition of defining subtraction as addition where the subtrahend has been multiplied by -1 (Cohen 2002). Similarly, we define division as multiplication where the divisor has been raised to the power of -1. Common functions that are mathematically smooth[5] (*e.g*. sine, inverse cosine, *etc*.) may be defined in terms of these operators by techniques like the Taylor Series.

There are four addition operators: grouping-addition, delta-grouping-addition, extension-addition and delta-extension-addition. All four "add" identically to yield the same digits; they differ in how they treat their metadata.

Grouping-addition represents the increase (or decrease for subtraction or addition of negative numbers) in the size of some set. The resulting sum has subject metadata being *collection()* applied to the subjects of both addends. Grouping-addition creates the state and location of the sum by generalizing that of its addends.

Delta-grouping-addition is like grouping-addition in that it represents the increase (or decrease) in the size of some set. In this case, however, the first addend is taken as the dominant contributor both numerically and semantically, and the sum's subject becomes the first addend's subject. For example, increasing the mass of water in a large, nearly full swimming pool by pouring 0.5 liters of water in it gives you the same body of water, slightly modified.

Extension-addition represents the further lengthening (or shortening for subtraction or addition of negative values) along some axis. The resulting sum has subject metadata equal to the *composite()* of the subjects of both addends. Extension-addition creates the state and location of the sum by intersecting that of its addends. Only the portions held in common get extended.

Last is delta-extension-addition which is like extension-addition in that it also represents the further lengthening (or shortening) along some axis. In this case, however, the first addend is taken as the dominant contributor both numerically and semantically, and the sum has the same subject as it. Going back to the antenna atop tall building example, you get the same building, slightly modified.

Grouping-addition and delta-grouping-addition are appropriate when the addition represents increasing (or decreasing) the size of a set-like collection of items that may or may not have any relation to each other, other than all belonging to the same collection. Thus, grouping-addition should be used for averages and standard deviations. Grouping-adding the masses of ten distinct weights and dividing by ten gives an idea how much mass a "typical" weight has.

By contrast, extension-addition and delta-extension-addition are appropriate when the addition represents the creation (or removal from) a composite object. Extension-adding the masses of our weights creates (if only in our minds and our computers' memories) a "superweight" that is the collection of all of them acting together[6].

Delta-grouping-addition and delta-extension-addition

---

[4] Of course in an N-dimensional space it is common to convert from *N* values in the old system to *N* values in the new as a group rather than individually. Such a conversion could be done with operators that worked on vectors of values (where the term "vector" now groups what we have considered sample vectors and matrices) instead of individual scalars. This is considered an implementation issue that is beyond the scope of this work.

[5] The function itself and all of its derivatives are continuous.

[6] The subjects do not necessarily have to touch. To other planets the Earth and Moon have the same gravitational effect as a "superplanet" whose mass is their sum.

implicitly consider the first addend to be the larger. The sum may be intolerably inaccurate when it is not because the second ("delta") addend may itself be an approximation. Making the "delta"-additions distinct operators gives us the opportunity to apply domain knowledge to see if the delta addend is "too big" to be trusted.

Unlike addition there is only one multiplication operator. It relies upon the default rules given in the previous chapter and uses intersection for state and location.

Similarly, there is one exponentiation operator. It is more restrictive than exponentiation in most computing systems because (1) the power must be a sample with a single value, and (2) that single value must be a dimensionless rational number. (Thus 0.3333333 is illegal; one must use 1/3.) The resulting value gets the domain's dimension and attribute from specific domain information like *resultingAttr()* (so that length^3 -> volume, and volume$^{1/3}$ -> length) but otherwise, because the power is dimensionless, the base will provide the majority of the metadata to the result. It also uses intersection.

The last basic operator is metadatacasting. Metadatacasting is like typecasting in that it allows the user to explicitly tell the computing system that they think it is okay to change the metadata in some specific fashion. For example, Bertozzi and Bednarski obtained their ^{1}H NMR data one day in 1990 from an NMR in the basement of Latimer Hall on the campus of the University of California Berkeley on planet Earth. However, they (and most chemists) probably believe that their spectrograph was not an artifact of that particular place and time. They may change the state of their measurements from *unspec(dayInYear1990CE)* to *allTime* and their location from *basementOfLatimerHall* to *allSpace* by explicitly casting.

Other operations can be defined using these. For example, trigonometric and inverse trigonometric functions can be defined that numerically compute with Taylor series and that check and change metadata like dimensions, units and attributes with metadatacasting. Trigonometric functions would take values with dimension *angle* and return values with dimension *dimensionless*. Inverse trigonometric ones would do the opposite. In both cases units (*radians* or *degrees*) become a non-issue because they are an inherent aspect of the value itself.

7. Discussion

We believe this computational paradigm to be distinctive enough that there are many issues to discuss.

(1) We want as many operators as needed to properly manipulate metadata and no more. Addition ballooned to four operators (or perhaps eight if you make the floating point vs. integer distinction). However, we believe they are all needed. "Delta"-addition vs. non-delta-addition lets users state when the right addend is semantically negligible. Grouping-addition vs. extension-addition lets users state whether a semantically new entity has been created as opposed to just another collection.

Similar distinctions exist for multiplication. Multiplication could mean "repeated addition" (*e.g.* 4 boxes of candy x 10 candies per box = 40 candies). It could mean "rescaling" where the right-hand-side has more semantic content and the left-hand-side is just a scale factor (*e.g.* $E=h\nu$; where E is the energy of a specific photon, ν is the frequency of a specific photon and h is a frequency-to-energy conversion factor common to all photons)[7]. It could also mean "rescaling" where the left-hand-side has more semantic content and the right-hand-side is just a scale factor (*e.g.* $(m_1 + m_2) / 2 = (m_1 + m_2) * 2^{-1} = m_{ave}$; m_{ave} clearly comes from m_1 and m_2, the 2 is just a normalization factor).

Rather than define three or more different multiplication operators we rely on the default metadata combining algorithm to compute the metadata appropriately.

(2) We can do more than just distinguish the theoretical and observed mass ratios in the Bertozzi and Bednarski data, we can say *how* they were calculated or were observed with the method metadata. The same "value" may be computed by the same simulator with different parameters, by different simulators of the same family, or by different families (*e.g. ab initio*, empirical-fitting or mixed approaches). Having this distinguishing metadata around is necessary if we want to keep all of it in the same knowledge base. We face a similar issue when storing multiple measurements taken with different resolutions or at different times. Scientists and philosophers of science tell us that all data assume some theory.

(3) If the average amount of metadata in a value is M and if we do N operations on N+1 values (as in $v_1 + v_2 + \ldots + v_{N+1}$) then the metadata can grow to O(MN). This is intolerably large but not expected to be the case for most scientific applications. The O(MN) size assumes that each new operand adds more unique semantic content. This would be the case if we were computing with semantically unrelated values. In real life planetary data is used with other planetary data, evolutionary biological data with other evolutionary biological data, *etc*.

(4) If the distribution matrices become intolerably large then instead of computing them an expression could return a stochastic lambda

[7] Reciprocal inversions are another example of left-scaling multiplications.

function whose structure mirrors the expression's parse tree. Running the function will stochastically return a value as in a Monte Carlo simulation. This function could be called many times to build a distribution.

(5) We have introduced several types of error checking. To this add checking the variance of values in distribution matrices. Because we assume a small variance in raw measurements, large variances, bimodal distributions, *etc*., in calculations may indicate the numeric instability of the expression. We may also add checks where the same (or overlapping) symbols appear both in the subject and authorship pieces of metadata. This signifies that a subject has reported on his-/her-/it-self. Depending on the attribute there may be questions about how objective this reporting is.

(6) We acknowledge that this approach will be several times slower than simply encoding numbers as floating point values and integers. We hope that this is CPU time well spent to create a documented computation.

(7) One problem with this approach is the hard distinction between the "delta"-addition operators and the non-"delta"-addition operators. For example consider filling an empty swimming pool a liter at time. Initially this appears to be an extension-addition operation, with the mass of the water increasing by relatively large amounts with each new liter. Towards the end, however, the pool is nearly full and just one more liter makes very little difference making this a delta-extension-addition. When exactly did extension-addition morph into delta-extension-addition?

Perhaps a better way of thinking about this is that there is only grouping-addition and extension-addition, and that they both may be more or less "delta"-like depending on the circumstances. What those circumstances are, and what their concrete ramifications for metadata manipulation are, are beyond the scope of this paper.

8. Conclusion

We have introduced a manner for handling vectors of distributions, metadata, and operators to apply to them appropriate for scientific data. This work was motivated by ongoing scientific representation and reasoning that is a continuation of Phillips 2009. Applications for this work, however, go beyond scientific reasoning to include constructive induction, data cleaning, and educational software for science and engineering.

Unfinished work includes how to handle comparisons such as less-than, less-than-or-equal-to, equal-to, *etc* and how to handle symbolic values (as needed in rules and decision trees). We could also consider handling complex numbers (used in some physics and engineering domains), and the simultaneous manipulation of related values (*e.g*. x, y and z coordinates).

References

[1] Bertozzi, C. R. and Bednarski, M. D. 1991. "The Synthesis of Heterobifunctional Linkers for the Conjugation of Ligands to Molecular Probes." *J. Org. Chem. 56*, 4326-4329.

[2] Bridewell, Will; Langley, Pat; Todorovski, Ljupčo; Džeroski, Sašo. 2008. "Inductive process modeling." *Machine Learning*. 71. p 1-32.

[3] Buchanan, B.G. & Feigenbaum, E. A. 1978. "DENDRAL and Meta-DENDRAL: Their Applications Dimension," *Artificial Intelligence*. 11:5-24.

[4] Buchanan, Bruce G. and Shortliffe, E. H. (eds.) 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley.

[5] Cohen, Joel S. 2002. *Computer Algebra and Symbolic Computation: Elementary Algorithms*. A. K. Peters. Natick, Massachusetts.

[6] deCarvalho, Rob. 2006. "Simple Units and Dimensions for Matlab"http://www.mathworks.com/matlabcentral/fileexchange /9873. Originally appeared Feb 2, updated Mar 3.

[7] Garfinkel, Simson. 2005. "History's Worst Software Bugs" *Wired Magazine*. Originally appeared November 8. Lions, J. L. 1996. *Ariane 5 Flight 501 Failure, Report by the Inquiry Board Report*. Paris. July 19.

[8] National Aeronautics and Space Administration. 1999. *Mars Climate Orbiter Mishap Investigation Board Phase I Report*. November 10.

[9] Phillips, Joseph. 2003. "Scilog: A Language for Scientific Processes and Scales." *Discovery Science*. Sapporo, Hokkaido, Japan, November. p 442-451.

[10] Phillips, Joseph. 2009. "Integrated Scientific Reasoning in the Scienceomatic 7B". *20th Midwest Artificial Intelligence and Cognitive Science Conference*. Indiana University-Purdue University Fort Wayne. p 146-152.

[11] Wolfram Research. 2010. "How Do I Use Units in Mathematica?" http://library.wolfram.com/howtos/units/. 2010 March 1.

[12] Valdés-Pérez, Raúl. 1992. "Algorithm to generate reaction pathways for computer-assisted elucidation". *J. Computational Chemistry*. Vol 13, No. 9, p 1079-1088.

Valdés-Pérez, Raúl. 1993. "Algorithm to Test the Structural Plausibility of a Proposed Elementary Reaction". *J. Computational Chemistry*. Vol 14, No. 12, p 1454-1459.

[13] Yang, Ruixin. Kafatos, Menas. Wang, X. Sean. 2002. "Managing Scientific Metadata Using XML" *IEEE Internet Computing*. Vol 6., Issue 4. p 52-59. July.