

Dealing with Mathematical Relations in Web-Ontologies

Muthukkaruppan Annamalai
Department of Computer Science & Software
Engineering
The University of Melbourne
Victoria 3010, Australia
mkppan@cs.mu.oz.au

Leon Sterling
Department of Computer Science & Software
Engineering
The University of Melbourne
Victoria 3010, Australia
leon@cs.mu.oz.au

ABSTRACT

The growing use of agent systems and the widespread penetration of the Internet has opened up new possibilities for scientific collaboration. We have been investigating the role for agent systems to aid with collaboration among Experimental High-Energy Physics (EHEP) physicists. A necessary component is an agreed ontology, which must include complex mathematical relations involving such quantities as the energy and momentum of elementary physics particles. We claim that the current web-ontology specification languages are not sufficiently expressive to be useful for explicit representation of mathematical expressions. We adapt some previous work on representing mathematical expressions to produce a set of mathematical representational primitives and supporting definitions that will allow knowledge sharing in agent systems. The paper sketches out a scheme for dealing with mathematical relations in scientific domain web-ontologies, illustrated with examples arising from our interactions with the EHEP physicists.

1. INTRODUCTION

The growing use of agent systems and the widespread penetration of the Internet has opened up new possibilities and created new challenges for scientific collaboration. On one hand, it is possible to make large amounts of scientific analyses of Experimental High-Energy Physics (EHEP) experiments available to scientists around the world [3, 4, 7]. On the other hand different scientific groups, even within a single collaboration, utilise different calculation methods, and it is sometimes difficult to know how to interpret particular analyses. It is assumed that practitioners in this domain possess the necessary background knowledge to interpret the intended meaning of the appropriated jargon in the domain of discourse. Unfortunately, application developers, newcomers to this field, and software agents lacking in relevant expertise are not capable of making a similar kind of interpretation. Knowledge models, or ontologies built to express specific facts about a domain can serve as the basis for understanding the discourse in that domain [5].

The notion of ontology as specification of a partial account of shared conceptualisation [13, 16] is adopted in this paper, that is an ontology defines a set of representational vocabulary for specific classes of objects and the describable relationships that exist among them in the modelled world of a shared domain.

In 2002, we were involved in a project [1, 8] supported by

the Victorian Partnership for Advanced Computing [21] to investigate whether ontologies would be useful for EHEP collaboration, in particular in the Belle [4] collaboration. The research is founded on the idea that suitable web-ontologies be developed and reused to facilitate this scientific community to produce and share information effectively on the semantic web. While the final verdict has not been reached, it is clear that our project extends existing capabilities of web-ontology specification languages and tools.

One concerning issue is as to how to define ‘function-concepts’ in the EHEP ontology, that is concepts equated with mathematical expressions involving such quantities as the energy and momentum of elementary physics particles. A function-concept is a manifestation of an n-ary mathematical relation or function binding a set of quantifiable terms defined in the ontology. The current web-ontology specification languages, such as DAML [17] and OIL [10] are founded on Description Logic [2] and can only express unary and binary relations. They offer no representational features for expressing functions. We ask, “How do we facilitate the semantic recognition of function-concepts in web-ontology?” The answer to this question led us to cognise the extensions required for additional expressivity in web-ontology. This paper shows the approach we have taken in order to ensure that we could describe mathematical relation underlying a function-concept in the ontology.

This paper is organised as follows. In Section 2, we provide a glimpse of the function-concepts in the EHEP domain. In Section 3, we describe the principles of dealing with mathematical relations in web-ontology and our approach to the problem is elaborated in Section 4. Finally, Section 5 concludes the contribution of this paper.

2. EVENT-VARIABLES IN THE EHEP ONTOLOGY

Mathematics plays a significant role in the EHEP experimental analysis – from the time sensor data is captured up to statistical analysis and systematic error calculation. In this paper however, we limit our scope of discussion to the treatment of mathematics in the vital event selection phase.

The superfluous event data captured by detectors is systematically filtered to suppress much of the background events, while preserving the vital signal events. Parametric restrictions on the event selection variables or ‘cuts’ are utilised to sift the signal events from background events. Loose cuts (or

skimming), followed by more decisive topological, kinematic and geometric cuts is aimed to produce a set of desired event data, fit for justification of the empirical findings.

A category of event selection variables is defined in the EHEP ontology. These event-variables are identified based on their use and need to specify the event selection or background suppression cuts. A set of competency questions [15] drawn-up while the ontology being built guides the conceptualisation of these required event-variables. A typical competency question is: “What are the kinematic selection criteria applied in an analysis?” The ensuing answer would list the selection criteria enforced, such as: “Beam constrained mass $M_{bc} > 5.2 \text{ GeV}/c^2$, Track transverse momentum $P_T > 100 \text{ MeV}/c$, Energy difference $\Delta E < 0.2 \text{ GeV}$, Likelihood of electron over kaon $L_{e/K} > 0.95$ ”. The ontology must define the necessary vocabulary to represent the competency questions that arose and the answers that were generated.

Some event-variables are constants, while others are functions. In the above illustrative example, $L_{e/K}$ is a constant; whereas, M_{bc} is a function event-variable. $M_{bc} = \sqrt{E_{beam}^2 - P_{3B}^2}$, where E_{beam} is the energy of the beam and P_{3B} is the 3-Momentum of B particle. E_{beam} is a constant. The 3-Momentum $P_3 = \sqrt{P_x^2 + P_y^2 + P_z^2}$ is a function. In here, P_x , P_y and P_z are constant event-variables that denote the individualised track momentum along the x, y and z axes, respectively.

Note that a function event-variable is expressed algebraically in terms of constant event-variables. A simple function event-variable accepts only constants as parameters. Examples are 3-Momentum, P_3 and Transverse Momentum, P_T that describe the momentum of a track. The function P_3 is given above, while $P_T = \sqrt{P_x^2 + P_y^2}$.

On the other hand, a higher-order function event-variable also admits other functions as parameters. The M_{bc} is a higher-order function. One of its parameter is P_3 , a function event-variable. Another example is the Fox-Wolfram Moment-0 event-variable, $H_0 = \sum_j \sum_k (P_{3j} \times P_{3k})$, whose parameter is a set of 3-Momentum of tracks. The indices j and k enumerate the tracks in an event.

Sometimes, a function requires weighted parameters. An example is the Fisher Discriminant event-variable $F = \sum_i (\alpha_i \times R_i)$, which combines a set of correlated event-variables R_1 , R_2 , etceteras to form a single variable. As in the previous case, the index i enumerates the considered event-variables. The coefficient α_i denotes the weight for the parameter R_i .

3. THE MAIN PRINCIPLES OF OUR APPROACH

We are concerned about explicating mathematical relations that bind a set of the scientific domain concepts in web-ontology. We seek to find a formal way to represent the mathematical relationships among domain concepts. The idea is to make clear this factual knowledge to humans and software agents during event analyses.¹ Our development

¹Note that our interest is quite different from that pursued by the mathematical-knowledge representation community.

drew heavily from the EngMath ontology [14], an extensive past attempt to capture the semantics associated with mathematical expressions in engineering models. EngMath is a declarative first order KIF [11] axiomatisation, which is supported by sets of theories for describing physical quantities, mathematical object such as scalar, vector, tensor and, functions and operations associated with them.

To a lesser degree, general-purpose ontologies like CYC [9] and SUMO [20] also attempt to declaratively capture the semantics of ‘evaluatable’ function. However, such function is categorised as unary, binary, ternary, quaternary and continuous types; thereby placing a limit on the number of its argument. Perhaps, this restraint is imposed in order to refer to a function’s argument based on its order in the ‘argument list’. In our case, a function argument must be able to denote a collection of class instances, which SUMO does not allow. In EngMath, such collection is represented as tensor or vector.

Although, the EngMath approach of providing rigorous descriptions appears to faithfully represent mathematical expressions in instantiated models, it is difficult to understand and apply to working systems. The theories represented in declarative style, as axioms (KIF sentences) are hard to read and tedious to specify, more so by physicists who are not well versed with ontology. Furthermore, it is not feasible to port this ontology on the web because web-ontology languages do not provide for definitions of arbitrary n-ary relations and functions, and axioms that make up EngMath.

Since we aim to build suitable EHEP ontologies for the semantic web, we parted from the EngMath approach from the outset. The contrast between EngMath and our modelling principles are described below.

- I. Web-ontologies are serialised in XML [22], a mark up language intended to encode metadata concerning web document.
- II. The domain concepts are assembled in a hierarchy and their distinguishing properties and relationships among them are specified, using a frame-like syntax. In light of this fact, we have conceived the concepts using notions like class, subclass, range-relation and cardinality. (See the examples in next section)
- III. A mathematical relation is closed on a function-concept in the ontology. It explicates the algebraic expression used to derive the function-concept. The knowledge associated with this derivation is representationally attached to the function-concept.
- IV. The quantifiable terms in the ontology are modelled upon mathematical data types such as scalar, boolean, record (cartesian product), set and function. The types specify how to interpret the values of these terms and restrict the set of operations that can be applied on them.

They are experimenting with different formalisms for representing mathematical theories, definitions, axioms, proofs, etceteras in the domain of mathematics.

- V. We also recognise the need to extend the algebra of primitive data type operators to constant quantities.
- VI. We have introduced the notion of a parameter, which is associated with function quantity. We have classified the parameters of a function quantity as *individual*, *collection* and *weighted* parameters to distinguish ones role from the other.
- VII. The arithmetic-logical expression denoting the intension of a function quantity will be encoded in a suitable format (not KIF expression). A possible candidate is OpenMath [19], an XML standard for exchanging mathematical objects on the web.

4. EXPLICATING MATHEMATICAL RELATIONS IN ONTOLOGY

The constant and function event-variables mentioned in Section 2 are physical quantities. We need to first define suitable concepts for representing these quantities in the ontology. Then, we will propound a scheme for providing an abstract description of mathematical relations involving these quantities.

4.1 Representation of Quantity

The need to deal with physical quantities in scientific ontologies is obvious. Our conceptualisation of physical quantity is different from that in EngMath. The definition of physical quantity is based upon our viewpoint on how quantities, dimension and units are treated in the EHEP domain.

For this, we defined a metaclass called *PhysicalQuantity*, having *magnitude* and *dimension* as its attributes (or properties). The concept-oriented definition shown in Figure 1 is an abstract idea of a physical quantity described in a Protege [12] -like representation.² A physical quantity has a scalar (integer or real) magnitude and its unit of measure is associated with a particular physical dimension specified in the ontology.

class PhysicalQuantity			
Property	Range	Relation	Cardinality
magnitude	Scalar		= 1
dimension	DimensionUnit		= 1

Figure 1: Definition of PhysicalQuantity

A constant quantity is a representation of a quantifiable object in the domain, which holds a single constant value of measure. The metaclass *ConstantQuantity* is defined as a ‘concrete’ subclass of *PhysicalQuantity* and a constant quantity can be directly instantiated from it. The constant event-variables defined in the ontology are constant quantities. For example, the fundamental track momentum, *Momentum-X* defined in Figure 2, is a constant quantity. A *Momentum-X* object’s magnitude is a real value and its unit of measure is given in terms of units of momentum dimension. Note however that *magnitude* and *dimension* properties are inherited from *PhysicalQuantity*.

²The semi-formal ontology developed using this frame-based ontology modelling tool, will be eventually formalised as EHEP web-ontology.

class Momentum-X : ConstantQuantity		
subClassOf Momentum		
Property	Range	Value
magnitude	Real	
dimension	MomentumUnit	

Figure 2: Definition of Momentum-X

The function event-variables defined in the ontology are function quantities. A function quantity is distinguished from constant quantity by the fact that the value of measure of the denoted quantifiable object is derived from other constant quantities. In other words, a function quantity can be seen as a function that maps one or more constant quantities to a constant quantity. Consequently, a grounded function quantity can be casted into constant quantity by anchoring it in the domain ontology. The function quantity is elaborated in Section 4.3.

Each quantity belongs to exactly one dimension. A unit is a measure of quantity in some dimension. There is a range of units associated with each dimension in the EHEP ontology. For example, the units for Energy dimension are *eV*, *KeV*, *MeV*, *GeV* and *TeV*.³ Relativistic physics asserts that *Energy = Momentum × SpeedOfLight*. Accordingly, the Momentum units such as *eV/c*, *KeV/c*, *MeV/c* etceteras are derived from Energy units. Likewise, *Momentum = Mass × SpeedOfLight*. So, Mass units such as *eV/c²*, *KeV/c²*, *MeV/c²* etceteras are derived from Momentum units.⁴

Note that in this system of units, the symbol *c*, which denotes the speed of light is featured as part of the unit. This rather unconventional way of describing the unit of a physical quantity in terms of abstract dimension symbol allows the physicists to work with a limited set of dimension unit symbols. It also facilitates the dimensional analysis of algebraic expressions involving related quantities. It would not be necessary, in our case to compose the dimension of a derived quantity from a larger set of fundamental dimensions as prescribed in EngMath and SUMO sub-model on ‘Unit of Measure’. The onus of dimensional consistency in algebra over the quantities rests with the modeller. However, the ontology is needed to support the unit analysis involving those quantities.

Based on the above standpoint, we have assigned a set of canonical units to each of the dimensions that appears in the ontology. In order to facilitate unit analysis over a dimension, a base unit is selected for each dimension. The other units of the dimension are then specified as conversion factor from a unit to the base unit. For example, the base unit for Energy dimension is *eV*. Its succeeding units, *KeV*, *MeV*, *GeV* and *TeV* are 10^3 , 10^6 , 10^9 and 10^{12} multiples of the base unit, respectively. Quantities to be manipulated must be same dimension and unit. For instance, the difference in their conversion factor can be used to reconcile the

³*eV* stands for electronvolt, is a non-standard unit for energy. The SI unit for energy is *J* (joule).

⁴The SI units for mass is *kg* (kilogram), and momentum is *Ns* (newton second) or *kg.m/s*, depending on the dimensions that are employed in its derivation.

quantities before they are operated upon.

4.2 Quantity and Data Types

Data type identifies the type of values that may be assumed by quantifiable objects and expressions in the ontology. We will make use of mathematical data types [6] to model these sets of values. The primitive data types are scalar and boolean, while the composite data types are record and set.

4.2.1 Primitive Data Type

The simplest data types are scalar and boolean types. The scalar types are Real and Integer. The arithmetic operators in Figure 3 are required to operate on scalar data types.⁵

plus	minus	times	divide	pow	mod	max	min
uminus	abs	log	ln	cos	sin	tan	acos
asin	atan						

Figure 3: Arithmetic Operators

Binary operators such as *plus*, *minus* and *times* have signature: $Scalar \times Scalar \rightarrow Scalar$, while a unary operators like *uminus*, *abs* and *log* have signature: $Scalar \rightarrow Scalar$. Arithmetic expressions are constructed using these operators.

A boolean type can have *False* or *True* constant values. The boolean operators associated with this type are *and*, *or* and *not*. The *and* and *or* operators have signature: $Boolean \times Boolean \rightarrow Boolean$. The *not* operator's signature is: $Boolean \rightarrow Boolean$.

and	or	not	less	more	equal
-----	----	-----	------	------	-------

Figure 4: Logical Operators

We also need the relational operators *less*, *more* and *equal* that maps a pair of Scalars to Boolean. Logical expressions constructed using the boolean and relational operators listed in Figure 4 will be used to specify conditions and constraints involving the quantifiable terms in the ontology.

4.2.2 Composite Data Type

The composite data types are built upon other data types. We have identified the need for two kinds of composite data types in the ontology, namely record and set data types. A record is a cartesian product of its constructed element types, while a set is a mathematical abstraction of a collection of elements.

Record

A record type is composed of a fixed number of data types. The set of values represented by a record is a cartesian product of its data types. A record is analogous to a class construct defined in ontology specification language.

We utilise record data type to model a structured set of values associated with an aggregate of disparate quantities

⁵Other scalar types such as Rational and Complex would be included, when their need arise. Likewise, the existing set of operators for will be expanded accordingly.

class Track : RecordQuantity		
Property	Range	Value
mom-x		Momentum-X
mom-y		Momentum-Y
mom-z		Momentum-Z
energy		Energy
class Momentum-X : ConstantQuantity		
subClassOf Momentum		
class Momentum-Y : ConstantQuantity		
subClassOf Momentum		
class Momentum-Z : ConstantQuantity		
subClassOf Momentum		
class Energy : ConstantQuantity		
subClassOf EventVariable		

Figure 5: Partial Definition of Track and its Range Relations

such as *Track* (Figure 5). Track information is typically denoted by an n-tuple of distinct quantities, partially described as $\langle P_x, P_y, P_z, E \rangle$. We choose to refer to such grouped quantities in ontology as record quantity because we are able to provide explicit names for the individual quantities in the structure, rather than simply rely upon the ordinal the quantities in the n-tuple.

An access operator called *select* is required to select individual elements of a record. The property names serve as the identifier of a record element. For example, $select(T, mom-x)$ accesses the *mom-x* component of track *T*.

Set

A set is a data type representing a collection of individuals with common characteristics. In the context of our work, we use set to represent a collection of homogenous individuals (class instances of same type). Set operators are specifically defined to be applied on an entire collection of individuals. The operators named in Figure 6 are required for constructing a set, determining its size, checking set membership, telling the maximum and minimum individual, and summing up all the individuals in a set.

setOf	union	intersect	oproduct	filter
size	member	maximum	minimum	summation

Figure 6: Set Operators

The operators on the first row are set constructors. Notably, the *oproduct* operator constructs the outer product of a pair of sets. For example, the Fox-Wolfram Moment-0 event-variable $H_0 = \sum_j \sum_k (P_{3_j} \times P_{3_k})$ can be expressed as follows: $summation(oproduct(setOf(P_3), setOf(P_3)))$. The summation operator repeatedly applies the plus operation on all the individuals in the constructed set of outer product of 3-Momentum.

The filter operator applies a condition (logical expression) on a set to filter out a subset of desired individuals, that is individuals that fulfil the specified condition. As an example, the following expression sums up the transverse momentum of tracks (P_T) that makes an angle of more than 45° with the Thrust axis (\hat{T}): $summation(filter(setOf(P_T),$

class FunctionQuantity		
subClassOf	PhysicalQuantity	
Property	Range Relation	Cardinality
parameter	Parameter	≥ 1
intension	ArithmeticLogicalExpression	
magnitude	Scalar	$= 1$
dimension	DimensionUnit	$= 1$

Figure 7: Definition of FunctionQuantity

class Parameter		
Property	Range Relation	Cardinality
argument	ConstantQuantity	≥ 1
coefficient	Scalar	≤ 1

Figure 8: Definition of Parameter

and(*more*(*angleBetween*(x, \hat{T}), $\pi/4$), *member*(x, P_T))). The function *angleBetween* is a geometric event-variable defined in the ontology. It represents the plane angle between any two axes in a particular frame of reference.

4.3 Mathematical Relations as Function Quantities

Mathematical relations in the ontology can be described abstractly in the form of grounded function quantities. A function quantity represents a physical quantity that is arithmetically derived from previously defined quantities. It is invoked with an explicit set of constant quantity parameters, which map to a constant quantity. In other words, a function quantity describes the numerical dependencies between its parameters, and the resulting quantity. Consequently, the intension of a function quantity is specified by an arithmetic-logical expression involving its parameters.

We give the definition of function quantity in Figure 7. It is conceptualised as a subclass of *PhysicalQuantity* with a set of properties, namely *parameter*, *intension*, *magnitude* and *dimension*. A function quantity inherits the *magnitude* and *dimension* properties from *PhysicalQuantity*, which together denote the characteristics of the resulting constant quantity.

4.3.1 Parameter of Function Quantity

The parameters of a function quantity are constant quantities, which includes existing grounded function quantities and record quantities. Recall that a grounded function quantity delivers a constant quantity as result. A record quantity constitutes a structured set of disparate constant quantities.

Each parameter of a function quantity is either an individual quantity or a collection of quantities. Alternatively, a weight is attached to a parameter that indicates its degree of influence on the resulting value. The metaclass definition in Figure 8 is an abstraction of the various types of parameters of a function quantity, whose *argument* and *coefficient* are constrained to *ConstantQuantity* and *Scalar* type, respectively.

Individual Parameter

An individual parameter is a subclass of *Parameter* whose *argument* cardinality is equal to 1. It denotes a specific

class TransverseMomentum : FunctionQuantity	
subClassOf	Momentum
Property	Range Value
parameter	P_x, P_y
intension	$abs(pow(sum(pow(P_x,2),pow(P_y,2)),0.5)$
magnitude	Real
dimension	MomentumUnit
class P_x : IndividualParameter	
Property	Range Value
argument	Momentum-X
class P_y : IndividualParameter	
Property	Range Value
argument	Momentum-Y

Figure 9: Partial Definition of Transverse Momentum and its Range Relations

quantity argument of a function quantity. Scalar and relational operators can be applied on the magnitude of this argument.

Figure 9 illustrates the definition of Transverse Momentum function quantity with two individual parameters. An instantiated *TransverseMomentum* maps instances of *Momentum-X* and *Momentum-Y* to an instance of momentum quantity, according to the specified intension.

Collection Parameter

A collection parameter is a subclass of *Parameter* whose *argument* cardinality is greater than 1. This type of argument is viewed as a collection of homogenous quantities and is typically applied to set operators as a whole. A case in point is the collection parameter (a set of 3-Momentum) of the Fox-Wolfram Moment-0 function quantity.

Weighted Parameter

A weighted parameter is a subclass of *Parameter* whose *coefficient* has a definite value, as in the case of Fisher Discriminant function quantity. This value will be dealt within the arithmetic-logical expression that specifies the intension of this function quantity.

4.3.2 Intension of Function Quantity

An arithmetic-logical expression (a constrained sequence of symbols) conveys the intension of a function quantity. The semantics is determined largely by the arithmetic, logical and set operations applied on the parameters of the function quantity. One possible way to encode this expression is using OpenMath [19].⁶ Figure 10 shows the OpenMath encoding of the arithmetic-logical expression that describes the intension of TransverseMomentum (see definition in Figure 9).

Expressions encoded in OpenMath are constrained by XML-syntax with implied semantics. An OpenMath object (OM-OBJ) is a sequence of one or more application objects (OMA).

⁶Another W3C data exchange standard is MathML [18]. While OpenMath focuses on the semantics in mathematical expression, MathML gives importance to the presentation (rendering) of the mathematical expression.

```

<OMOBJ>
  <OMA>
    <OMS cd="arith1" name="abs"/>
    <OMA>
      <OMS cd="arith1" name="pow"/>
      <OMA>
        <OMS cd="arith1" name="plus"/>
        <OMA>
          <OMS cd="arith1" name="pow"/>
          <OMV name="Px"/>
          <OMI> 2 </OMI>
        </OMA>
      </OMA>
    </OMA>
  </OMOBJ>

```

Figure 10: Intension of TransverseMomentum Encoded in OpenMath

OpenMath maintains reportative definitions of the mathematical-oriented metadata in a set of content dictionaries. In this example, the content dictionary (cd) named *arith1* holds the definition of the following symbols (OMS): *abs*, *pow* and *plus*. These symbols coincide with the data type operators identified in Section 4.2. The parameter or variable (OMV) is associated with constant quantity, in our case.

We appeal to an interpreter for manipulating this encoded expression. OpenMath only recognises scalar values such as integer (OMI) and real (OMF). The design decision will also have to consider on how to extend the scalar and boolean algebra to cover the constant quantities in the ontology. For instance, the interpreter could apply the operators on the scalar magnitude of dimensionally compatible quantities whose units have been reconciled. We do not wish to dwell further into this implementation issue at this stage.

4.3.3 Result of Function Quantity

The result of a grounded function quantity is a constant quantity, whose *magnitude* and *dimension* are same as that of the function quantity. Even a pure scalar result can be viewed as a constant quantity whose magnitude is scalar and its physical dimension is termed as ‘dimensionless’. The *dimension* of a function quantity is only defined generically. An instance of a function quantity will however use one of the prescribed units associated with its dimension type.

4.4 Dynamic Mathematical Relations in Ontologies

We want to deal with both static and dynamic mathematical relations in the ontology. A static relation is a permanent relationship that exist between conceptual terms in all instantiated model (read ‘experimental analysis description’) of the EHEP domain. An example is the relation describing Transverse Momentum function quantity, which is always composed from Momentum-X and Momentum-Y constant

quantities. In other words, the parameters of function quantity denoting a static relation are fixed.

On the other hand, a dynamic relationship between concepts is coerced specifically for a particular use or purpose. An example is the relation described by Fisher Discriminant function quantity, which combines a set of correlated event-variables. Different event selection analysis entails different sets of correlated event-variables. Therefore, the set of event-variables combined by Fisher Discriminant varies from one instantiated model to another. Although the intension of this function quantity is fixed, its parameters are not.

Function quantities to find the difference or the sum of any two quantities also fall under the category of dynamic mathematical relations. In here, the number of parameters is fixed, but their types can vary. For instance, the quantity sum of two mass quantities is not same as the quantity sum of two momentum quantities, as they have dissimilar dimensions.

This paper mainly discusses on the representation of the static relations. We think it is possible to extend the existing framework to handle dynamic relations in the ontology. An alternative is to define a dynamic mathematical relation as an ‘abstract’ function quantity, still ungrounded in the ontology. The intension of this function quantity will be described in terms of physical quantities (that is, the type of the argument is *PhysicalQuantity*, instead of *ConstantQuantity*). Subsequently, a specialised form of this function quantity can be derived from its abstract definition, to suit a specific need. The specialised function quantity will assign appropriate interpretation of the mathematical relation by specifying the exact parameters and characteristics of the resulting quantity. This idea is still under consolidation.

5. CONCLUSION

This EHEP ontology should provide the necessary representational vocabulary to facilitate the scientific community to collaborate effectively on the semantic web. One concern however is due to the limitations of web-ontology languages, which disallow the direct representation of mathematical relations in ontologies. The existing web-ontology languages falter when the need to specify metaclasses, n-ary relations and functions arises. They also lack the necessary epistemological and arithmetic primitives to explicitly represent algebraic expressions involving domain terms.

This paper highlights the additional vocabulary and language features required to deal with mathematical relations in web-ontology. Future web-ontology languages may offer a richer set of primitives to express complex relationships among entities. Until such time, our approach recommends the use of implicitly defined metadata to partly describe the meaning of the mathematical relations in web-ontology.

6. ACKNOWLEDGMENTS

We are grateful to Glenn Moloney and Lyle Winton of the Physics department for providing insights into EHEP experimental analysis. We also thank the members of the Intelligent Agent Laboratory at the University of Melbourne for their helpful comments.

7. REFERENCES

- [1] M. Annamalai, L. Sterling, and G. Moloney. A collaborative framework for distributed scientific groups. In S. Cranefield, S. Willmott, and T. Finin, editors, *Proceedings of AAMAS'02 Workshop on Ontologies in Agent Systems*, Bologna, Italy, 2002.
- [2] F. Baadar and W. Nutt. Basic description logics. In F. Baadar, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors, *Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2002.
- [3] The Babar Physics Collaboration. <http://www.slac.stanford.edu/BFROOT/>.
- [4] The Belle Physics Collaboration. <http://belle.kek.jp/belle/>.
- [5] B. Chandrasekaran and J. R. Josephson. What are ontologies, and why do we need them? *IEEE Intelligent Systems*, pages 20–26, January/February 1999.
- [6] J. C. Cleaveland. *An Introduction to Data Types*. Addison-Wesley, 1986.
- [7] The Cleo Physics Collaboration. <http://www.lns.cornell.edu/public/CLEO/>.
- [8] L. Cruz, M. Annamalai, and L. Sterling. Analysing high-energy physics experiments. In B. Burg, J. Dale, T. Finin, H. Nakashima, L. Padgham, C. Sierra, and S. Willmott, editors, *Proceedings of AAMAS'02 Workshop on AgentCities*, Bologna, Italy, 2002.
- [9] Cycorp. <http://www.cyc.com/>.
- [10] D. Fensel, I. Horrocks, F. vanHarmelen, D. McGuinness, and P. Patel-Schneider. Oil: Ontology infrastructure to enable the semantic web. *IEEE Intelligent Systems*, pages 38–45, March/April 2001.
- [11] M. Genesereth and R. Fikes. Knowledge interchange format. Technical Report Logic-92-1, Computer Science Department, Stanford University, 1992.
- [12] W. E. Grosso, H. Eriksson, R. W. Fergerson, J. H. Gennari, S. W. Tu, and M. A. Musen. Knowledge modeling at the millennium (the design and evolution of Protege-2000). In *Proceedings of KAW'99 Workshop on Knowledge Acquisition, Modelling and Management*, Banff, Alberta, 1999.
- [13] T. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
- [14] T. R. Gruber and G. R. Olsen. An ontology for engineering mathematics. In J. Doyle, P. Torasso, and E. Sandewall, editors, *Proceedings of International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufmann, 1994.
- [15] M. Gruninger and M. S. Fox. Methodology for the design and evaluation of ontologies. In *Proceedings of IJCAI'95 Workshop Basic Ontological Issues in Knowledge Sharing*. Montreal, Canada, 1995.
- [16] N. Guarino. Ontologies and knowledge base: Towards a terminological clarification. In N. Mars, editor, *Towards Very Large Knowledge: Knowledge Building and Knowledge Sharing*, pages 25–32. IOS Press, Amsterdam, 1995.
- [17] J. Hendler and D. McGuinness. The Darpa Agent Mark up Language. *IEEE Intelligent Systems*, pages 67–73, November/December 2000.
- [18] Mathematics Mark up Language. <http://www.w3.org/TR/MathML2/>.
- [19] Openmath Mark up Language. <http://monet.nag.co.uk/cocoon/openmath/index.html>.
- [20] Suggested Upper Merged Ontology. <http://ontology.teknowledge.com/>.
- [21] Victorian Partnership for Advanced Computing. <http://www.vpac.org/>.
- [22] EXtensible Mark up Language. <http://www.w3.org/XML/>.