

In Jon Doyle, Piero Torasso, & Erik Sandewall, Eds., *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, Morgan Kaufmann, 1994.

## An Ontology for Engineering Mathematics

**Thomas R. Gruber and Gregory R. Olsen**

*Knowledge Systems Laboratory*

*Stanford University*

*701 Welch Road, Building C, Palo Alto, CA 94304*

*gruber@ksl.stanford.edu*

### Abstract

We describe an ontology for mathematical modeling in engineering. The ontology includes conceptual foundations for scalar, vector, and tensor quantities, physical dimensions, units of measure, functions of quantities, and dimensionless quantities. The conceptualization builds on abstract algebra and measurement theory, but is designed explicitly for knowledge sharing purposes. The ontology is being used as a communication language among cooperating engineering agents, and as a foundation for other engineering ontologies. In this paper we describe the conceptualization of the ontology, and show selected axioms from definitions. We describe the design of the ontology and justify the important representation choices. We offer evaluation criteria for such ontologies and demonstrate design techniques for achieving them.

### 1. Introduction

Engineers use mathematical models, such as sets of equations, to analyze the behavior of physical systems. The conventional notations for formatting mathematical expressions in textbooks and in the engineering literature usually leave implicit many of the details required to understand the equations. For instance, it is not clear from the expression  $f = kx + c$  which symbols are variables or constants; whether they represent numbers or physical quantities (e.g., forces, lengths); whether the magnitudes are reals, vectors, or higher-order tensors; whether the quantities are static values, functions of time, or functions of time and space; and how units of measure are treated. The reader must *interpret* these notations using background knowledge and context. This is error-prone for humans and beyond the capability of today's computer agents.

To enable the sharing and reuse of engineering models among engineering tools and their users, it is important to specify a conceptual foundation that makes these distinctions explicit and provides a context- and reader-independent semantics. Toward this end, we have developed a formal ontology for mathematical modeling in engineering, called EngMath. The ontology builds on abstract algebra and measurement theory, adapted to meet the expressive needs of engineering modeling. The specification includes a first-order axiomatization of representational vocabulary that is machine and human readable.

This paper is about the EngMath ontology, and how it exemplifies the design and use of such ontologies in support of agent communication and knowledge reuse. Such an ontology differs from what is found in engineering textbooks and philosophy books in that it is designed as a specification for these knowledge sharing purposes. We begin in Section 2 by describing the role of ontologies as formal specification and the uses of the EngMath ontology. In Section 3, we give define the basic concepts and relations in the ontology. In Section 4, we discuss a series of design decisions and their rationale. In Section 5, we offer design criteria--minimizing ontological commitment and maximizing monotonic extendibility--and demonstrate techniques used to achieve them. In Section 6, we discuss the relationship of the EngMath ontologies to relevant work in philosophy and AI

## 2. The Purpose of the Ontology

### 2.1 Ontology as Formal Specification

A body of formally represented knowledge is based on a *conceptualization*: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them [17]. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

For the purpose of knowledge sharing, formal ontologies serve as specifications of common conceptualizations [20] among agents. In the philosophy literature, ontology is the systematic account of Existence--aiming to account for all forms and modes of being [5]. For AI systems, what can exist in a conceptualized world is determined by what can be represented.[Note 1] If agents are to communicate in a shared language or if a body of formally represented knowledge is to be reused, then there must be some agreement about a universe of discourse. Furthermore, if the shared language includes vocabulary denoting entities and relationships in the conceptualization, there must be some way to specify what can be meaningfully stated in this vocabulary. Ontologies, in the context of knowledge sharing, are a means for making such content-specific agreements.

If we assume a common syntax and semantics for a core representation language, then we can specify conceptualizations by writing definitions of shared vocabulary. That is the strategy proposed by the ARPA Knowledge Sharing Effort [33,35], and is the tack we are taking. A Knowledge Interchange Format (KIF) [16] serves as the language for making assertions and definitions, and ontologies provide axiomatic and textual definitions of relations, functions, and objects. By 'definitions' we mean specifications of the well formed use of the vocabulary. Definitions include axioms that constrain the interpretation.[Note 2] Such an axiomatization specifies a logical theory, but is not intended as a knowledge base. Instead, the ontology serves as a domain-specific representation language in which knowledge is shared and communicated.

In practice, our ontologies define the vocabulary with which queries and assertions are exchanged among interoperating agents, some of which may be passive (e.g., deductive databases). The agents conform to ontological commitments [19, 20] which are agreements to use the shared vocabulary in a coherent and consistent manner. An ontological commitment is a guarantee of consistency, but not completeness, with respect to queries and assertions using the vocabulary defined in the ontology (c.f. [23]). Committed agents may "know" things not implied by the shared ontologies, and may not be able to answer queries that follow from the shared ontologies. Furthermore, the "shared knowledge" of these agents can be viewed at the Knowledge Level, as attributed and independent of symbol-level encoding [34]. Thus, the agents may operate on any internal representation desired, as long as they use the shared vocabulary consistently in communication. This model of agent collaboration is being pursued by several groups [9, 15, 22, 32].

### 2.2 Uses of the EngMath Ontology

In designing the EngMath ontology, we anticipate three kinds of use, and accept them as requirements. First, the ontology should provide a machine- and human-readable notation for representing the models and domain theories found in the engineering literature. Second, it should provide a formal specification of a shared conceptualization and vocabulary for a community of interoperating software agents in engineering domains. Third, it should provide a foundation for other formalization efforts, including more comprehensive ontologies for engineering and domain-specific languages. In this section we will give examples of each application.

#### EngMath as a Shared Notation

Engineers use mathematical expressions, such as constraint equations, to describe, analyze, and communicate models of physical devices and their behavior. The quantities represented in these expressions are different from purely numerical values, and the algebra for operating over them must

account for extra-numerical considerations such as dimensional consistency, units of measure, and vector and tensor operations. Some form of this 'physical algebra' [31] is taught in nearly every introductory physics or engineering course, and the subject is prominent in the standard texts [26, 37]. Students are taught to use dimensional consistency to check for modeling and equation solving errors. A technique called Dimensional Analysis [31] is used in design and to assist in the interpretation of experiments, and is an important area of study itself.

Textbook notations for physical quantities vary by author and leave much implicit--relying on context and background knowledge of the reader for proper interpretation. The problem of implicit notation is revealed when students try to encode engineering models using mathematical support software. Human expertise is required to map expressions about physical quantities to the purely mathematical constructs of current commercial math tools (e.g., Matlab, Mathematica, Maple).

The EngMath ontology is intended to provide a formal language sufficient to express the models in engineering textbooks and to map them to mathematical software tools. We view the latter application as an instance of agent communication, which is the subject of the next section.

### EngMath as a Vocabulary for Agent Communication

By providing a declarative, machine readable representation, the EngMath ontologies can enable unambiguous communication between software agents that would otherwise be difficult or impossible.

To illustrate a use of the ontology, consider a simple example of agents exchanging symbolic representations of spring behavior. Agent A is a specialist in the design of springs, and agent B is a specialist in quantity algebra. Agent A needs a solution to a set of equations relating spring and material properties that include the following:

$$k = (d^4G) / (8D^3N), \quad G = 11,5000\text{kpsi}$$

where  $k$  is the spring rate,  $d$  is wire diameter,  $D$  is spring diameter,  $N$  is number of turns, and  $G$  is the shear modulus of elasticity.

Agent A can send Agent B these equations as a set of KIF sentences, using the vocabulary of the EngMath ontology:

```
(scalar-quantity k)
  (= (physical.dimension k)
     (/ force-dimension length-dimension))
(scalar-quantity d)
  (= (physical.dimension d) length-dimension)
(scalar-quantity dm)
  (= (physical.dimension Dm) length-dimension)
(scalar-quantity N)
  (= (physical.dimension N) identity-dimension)
(scalar-quantity G)
  (= (physical.dimension G)
     (* force-dimension (expt length-dimension -2)))
  (= k (/ (* (expt d 4) G) (* 8 (expt Dm 3) N)))
  (= G (* 11.5 (expt 10 6) psi))
```

After receiving the equations in this form, agent B can answer questions about the values of the terms such as the diameter ( $d$ ). The vocabulary used in this interaction, such as the function constant `physical.dimension`, is independent of a domain theory for springs. The sentence simply states an algebraic relationship between quantities and the dimensional characteristics of those quantities. This type of information allows Agent B to perform algebraic manipulations such as solutions of simultaneous equations or numerical evaluations of individual parameters. Because dimensional information is included, the consistency of equations can be checked by the agent and agents are freed from committing to an explicit set of units. The spring example is typical of problems in introductory

engineering textbooks. More complex interactions are required to coordinate commercial design tools on industrial problems.

In the SHADE project [22, 32], we have constructed a set of software agents that interact to support collaboration on industrial problems like satellite system design. One SHADE agent is a specialist in rigid body dynamics (RBD), and another is responsible for the geometric layout of satellite components (Layout). Both commit to the EngMath ontology. The RBD agent queries the Layout agent about the inertial characteristics of a particular component. These characteristics include the mass whose value is a scalar quantity and the inertia tensor whose value is a second order tensor. In reply, the Layout agent specifies the inertia tensor with respect to a global reference frame and point. The reference frame is part of the shared domain theory for the two agents; it is not implicit in the representation of the tensor. This allows the RBD agent to translate the inertia into a different reference frame convenient for dynamic analysis.

Most SHADE agents are commercial tools wrapped so that they conform to ontological commitments and communication protocols. These agents are designed to be conformant at the interface, but are not required to represent the ontologies internally. Some agents can assimilate an ontology and use it as input. The Unit Conversion Agent is an example. Its contract is specified entirely by the EngMath ontology. This agent takes KIF expressions over quantities, and performs services such as symbolic simplification, unit conversion, and dimensional consistency verification. It can *read* ontologies that specify of other unit systems, and determine whether the system is complete for the dimensions specified.

### **EngMath as a Conceptual Foundation**

The EngMath ontology was also designed as a conceptual foundation for other ontologies; in its design we needed to anticipate how it would be used in other theories. For example, we are constructing a family of ontologies for representing components and relations among them (e.g., part-subpart relations, connections, association of component features and constraints). One ontology of mechanical components, for instance, is for models in which components have mass properties and associated reference frames and points, but lack complete geometric representations. This theory combines an abstract component ontology, a constraint expression ontology, parts of the EngMath ontology, and a simple geometry theory (that also includes EngMath).

The Compositional Modeling Language (CML)[ 12] is another example of building on the EngMath ontologies. CML is a modeling language that is intended to synthesize and redesign the various formulations of Compositional Modeling [ 8, 13, 14, 29 ] to enable model sharing among research groups. Part of the language design of CML is an ontology about time, continuity, object properties, etc. The semantics of the language are specified axiomatically, using the vocabulary of the CML ontology. The CML ontology builds on the EngMath ontology as a foundation.

## **3. Overview of the Conceptualization**

In this section, we describe the key concepts of the conceptualization specified by EngMath. The ideas will be familiar to many readers. However, since there are multiple ways to formulate the various concepts, it is important to clarify how their synthesis results in a coherent theory.

The entire ontology is too large and complex to present in static, linear form (about 2000 lines of definitions). The complete specification is available on-line on the World Wide Web in cross-indexed, machine-formatted hypertext [ 21]. To give a flavor for the details, we have included a few axioms from the actual ontologies in this section.

### **3.1 Physical Quantities**

A mathematical model of a physical system, of the sort we are interested here, consists of a set of constraints on the values of variables. These variables represent physical quantities. A **physical quantity** is a measure of some quantifiable aspect of the modeled world. Quantities "admit of degrees"

[11] in contrast to qualities, which are all-or-none (e.g., being pregnant). Physical quantities come in several types, such as the mass of a body (a scalar quantity), the displacement of a point on the body (a vector quantity), the altitude of the particle as a function of time (a unary scalar function quantity), and the stress at a particular point in a deformed body (a second order tensor quantity). For our purposes, what makes quantities "quantifiable" is the ability to combine them with algebraic operations. Physical quantities can be meaningfully added, multiplied, and raised to real-valued exponents. The types of quantities determine the conditions under which operations are allowed and the types of the results. For example, it does not make sense to add a mass quantity and a displacement quantity, and the result of multiplying a length and a length is a third type of quantity--an area. This ontology specifies in detail the conditions under which various algebraic operations on quantities make sense.

Although we use the term "physical quantity" for this generalized notion of quantitative measure, the definition allows for nonphysical quantities such as amounts of money or rates of inflation. However, it excludes values associated with nominal scales, such as Boolean state and part number, because they are not amenable to these algebraic operations.

```
(defrelation PHYSICAL-QUANTITY
  (=> (physical-quantity ?x)
      (and (defined (quantity.dimension ?x))
           (physical-dimension(quantity.dimension ?x))
           (or (constant-quantity ?x)(function-quantity ?x))))))
```

### 3.2 Physical Dimensions

The central difference between a physical quantity and a purely numeric entity like a real number is that a quantity is characterized by a physical dimension. The **physical dimension** of a quantity distinguishes it from other types of quantities. The physical dimension of a mass of a body is mass; the physical dimension of a position of a body is length, and the physical dimension of a stress quantity is

```
(* mass (* (expt length -1) (expt time -2)))
```

where `*` is multiplication and `expt` is exponentiation. Nonphysical dimensions are also possible, such as amount of money. Dimensions tell us something intrinsic about the quantity that is invariant over models and measurement. For example, there is no intrinsic difference between a quantity used to describe an altitude and a quantity used to describe a width; both are quantities of the length dimension. A length of three feet and a length of one yard are equal, although we specified them in different units of measure.

Physical dimensions can be composed from other dimensions using multiplication and exponentiation to a real power. It is important for Dimensional Analysis [31] that dimensions have certain algebraic properties. The product of any two physical dimensions is also a physical dimension, and the multiplication operator `*` is associative, commutative, and invertible with an identity element called the identity dimension (i.e., it forms an abelian group with `*`).

```
(defrelation PHYSICAL-DIMENSION
  (abelian-group physical-dimension * identity-dimension))
```

Constant quantities whose physical dimension is the identity dimension are called, paradoxically, **dimensionless quantities**. In this ontology, dimensionless quantities include the real numbers and numeric tensors.

```
(defrelation DIMENSIONLESS-QUANTITY
  (<=> (dimensionless-quantity ?x)
      (and (constant-quantity ?x)
           (= (quantity.dimension ?x)identity-
              dimension)))
  (=> (real-number ?x)(dimensionless-quantity ?x)))
```

Dimensional homogeneity is a prerequisite to unit conversion and other algebraic operations on quantities. Consider the simplest type of physical quantities, scalar quantities. **Scalar quantities** are

constant quantities with real-valued magnitudes, distinguished from higher-order tensors. (We will define the *magnitude* function precisely below.) A physical dimension defines a class of scalars with important algebraic properties. For example, the sum of any two scalars of the same dimension is a scalar of the same dimension. Physical dimensions also provide necessary conditions for *comparing* quantities; two quantities are comparable only if they are of the same physical dimension. It makes sense to quantitatively compare two masses but not a mass and a length.

### 3.3 Comparability and Order

Comparability is one way to ground the otherwise algebraic definitions of quantities. Quantities are quantitative measures; a meaningful measure is one that reflects order in the measured (or we would say, modeled) world. Adapting the definition by Ellis [11], we say that the elements of a class  $Q$  of (scalar) quantities of the same physical dimension must be *comparable*. According to Ellis, comparability for a quantity type holds if there is as a *linear ordering relationship* in the world given by an equivalence relation  $R=$ , and a binary relation  $R<$  that is asymmetric and transitive over  $Q$ , such that for any two quantities  $q_1, q_2$  in  $Q$ , exactly one of the following must hold:  $q_1 R= q_2$ ,  $q_1 R< q_2$ , or  $q_2 R< q_1$ . Using this definition, we can ask whether something we want to call a quantity type or physical dimension should be classified as such. Mass, for instance, is comparable by this definition because one can always order masses. The property of comparability is independent of measurement unit, measurement procedure, scales, or the types of physical objects that are being modeled.

Ellis defines a quantity [type] as exactly that which can be linearly ordered. We needed to depart on two fronts, to accommodate the sorts of quantities we find in engineering models. First, comparability is different for higher-order tensors (see Section 3.5); the tensor order and spatial dimensions of the quantities must be compatible to be able to compare them, and the ordering need not be total. Second, for scalars we insist that the order be dense: one can multiply any scalar quantities of a given physical dimension by a real number and obtain another scalar quantity of that physical dimension. This property also holds for mass, and illustrates that calling something a quantity is a modeling decision. That mass is densely ordered in this way is an assumption of continuum mechanics. It also was a consequence of including the reals as a species of physical quantity. Nonetheless, we depart from writers like Ellis primarily because our goals are slightly different. Our primary responsibility is to explicate a coherent framework that is adequate for expressing the content of engineering models.

The notion of physical dimension is intimately tied up with the notion of physical quantity, and both are primitive concepts ultimately grounded in the comparability of quantities in the world. Thus, from our definitions alone a computer program cannot infer that some entity is a physical quantity unless it defined in terms of other quantities. The practical consequence of including such primitives in a formal ontology is that the types of such entities must be declared.

### 3.4 Function Quantities

A physical quantity is either a constant quantity or a function quantity. The mass in our example model is a constant quantity, like 50kg. A function quantity is a function from constant quantities to constant quantities. It is not a function from physical objects to quantities. The altitude of a particle over time is a function quantity, mapping quantities of time to quantities of length. Function quantities can take any finite number of arguments (although in engineering they usually take 1, 3, or 4). For example, the quantities in ordinary differential equation (ODE) models are unary functions mapping scalar quantities (e.g., of time) to scalar quantities. Partial differentials involve functions of several quantities, such as three length quantities and a time quantity. Like all physical quantities, each function quantity has a physical dimension--the dimension of all the elements of the range (a function that maps to quantities of differing ranges is not a function quantity).

### 3.5 Tensor Quantities

In the conceptualization, physical quantities include not only scalars but also higher order tensors such as vectors and dyads. Vectors (first order tensors) are distinct from scalars, in that complete specification of a vector constant requires a statement of direction or orientation. A velocity vector, for instance, can be decomposed into a 3-tuple of scalars for a particular choice of reference frame.

Mechanical stress is represented by dyad (second order tensor) and instances of it can be mapped to a 3x3 matrix for a given reference frame. Tensors are a useful abstraction, because they possess properties that are invariant across reference frames. Though three dimensional reference frames (or vector spaces) and tensors of order two or less are most common in physical modelling, the concepts generalize to n dimensions and n-orders. Tensors, then, are characterized both in terms of order and spatial dimension, and these distinctions, in turn, imply a set of algebraic restrictions. The EngMath ontology integrates the algebraic properties of tensors with dimensional properties of all physical quantities.

### 3.6 Units of Measure

The identity of quantities does not depend on the process or nature of measurement, or units of measure. A quantity of mass like 50kg is the same thing whether it is measured with a balance beam or a spring, and it is comparable in every way with other mass quantities independently of whether they are specified in kilograms or pounds. However, units are not irrelevant; for example, one cannot specify a constant in an equation without making reference to units of measure.

In our conceptualization, units of measure are quantities themselves (positive, scalar, constant quantities). A unit of measure is an absolute amount of something that can be used as a standard reference quantity. Like all quantities, units have dimensions, and units can be defined as any other scalar quantity. For example, the kilogram is a unit of measure for the mass dimension. The unit called "pound" can be defined as a mass quantity equal to the kilogram times some constant, just as the quantity 50kg is equal to the product of the unit called "kilogram" and the real number 50. What makes the pound special, compared with quantities like 50kg, is a matter of convention. (We will return to the issue of standard units in Section 3.8.) To provide for unit conversion over all physical dimensions, every product and real-valued exponentiation of a unit is also a unit of measure.

```
(defrelation UNIT-Of-MEASURE
;; units are scalar quantities
  (=> (unit-of-measure ?u)(scalar-quantity ?u))
;; units are positive
  (=> (unit-of-measure ?u)
      (forall ?u2
        (=> (and (unit-of-measure ?u2)
                  (= (quantity.dimension ?u)
                     (quantity.dimension ?u2)))
            (positive (magnitude ?u ?u2))))))
;; units can be combined using *
  (abelian-group unit-of-measure * identity-unit)
;; units can be combined using expt
  (=> (and (unit-of-measure ?u)
           (real-number ?r))
      (unit-of-measure (expt ?u ?r)))
;; * is commutative for units and other qs
  (=> (and (unit-of-measure ?u)
           (constant-quantity ?q))
      (= (* ?u ?q) (* ?q ?u))))
```

### 3.7 Magnitudes

The magnitude of a physical quantity is not a property of the quantity, but is given by a binary function that maps a quantity and unit of measure to a numeric value (a dimensionless quantity). Once it was decided that units were just scalar quantities, it became apparent that the magnitude function is simply a restricted form of scalar division (it is only defined when its first argument is a constant quantity and its second argument is a unit of the same dimension). It is also total for all constant quantities: a constant quantity can be expressed in any unit of the same physical dimension, and the magnitude of a quantity in one unit can be converted to its magnitude in any other comparable unit.

The requirement for dimensional consistency fits our intuition. The magnitude of 50kg in kilograms is 50, but the magnitude of 50kg in meters is undefined. For higher-order tensor quantities, the value of the magnitude function is an ordinary dimensionless tensor. Since units of measure are *scalar* quantities, one can think of the magnitude function as factoring out the physical dimension of a quantity (returning a dimensionless quantity) and producing a value normalized on a scale corresponding to the unit.

Although a unit of measure implicitly determines a measurement scale, units of measure are not the same thing as scales in this conceptualization. Measurement scales are a more general way to map quantities to numeric values, and are described in Section 4.7.

```
(deffunction MAGNITUDE
  (<=> (and (defined (magnitude ?q ?unit))
            (= (magnitude ?q ?unit) ?mag))
    (and (constant-quantity ?q)
         (unit-of-measure ?unit)
         (dimensionless-quantity ?mag)
         (= (quantity.dimension ?q)
            (quantity.dimension ?unit))
         (defined (* ?mag ?unit))
         (= (* ?mag ?unit) ?q)))
;; dimensionless magnitudes can be factored
(forall (?q ?unit ?mag)
  (=> (and (constant-quantity ?q)
           (unit-of-measure ?unit)
           (dimensionless-quantity ?mag)
           (defined (* ?mag ?q)))
      (= (magnitude (* ?mag ?q) ?unit)
         (* ?mag (magnitude ?q ?unit))))))
```

### 3.8 Standard Systems of Units

Although we do not want to *fix* a set of standard units for the shared ontology, we want to provide the vocabulary with which to define sets of standard units so that agents can share them. For this, the concept of system of units is used. A **system of units** is a class of units defined by composition from a base set of units, such that every instance of the class is "standard" unit for a physical dimension and every physical dimension has an associated unit.

This is an interesting representation problem, because both the set of units and the space of physical dimensions are conventions, and both are constrained (but not determined) by the background domain theory assumed in a model. The set of dimensions and their mutual relationships are determined by a physical theory, while the choice of units for each dimension is a measurement convention. The relationship between force, mass, length, and time is given by physics. The theory does not need to give fundamental status to any one physical dimension, but it does say that the force dimension is equal to  $(* (* \text{length mass}) (\text{expt time } -2))$ . One system of measurement may take mass, length, and time to be primitive and derive force; another could take force as primitive and derive mass. The same physical laws could be expressed in either system.

The concept of system of units is defined so that commitments to physical theories, sets of fundamental dimensions, and standard units are independent. To define a system of units, the model builder chooses a set of fundamental dimensions that are orthogonal (i.e., not composable from each other). According to this physical theory, mass and time are orthogonal, but force and mass are not. The set of fundamental dimensions determines the space of possible quantities that can be described in this system--those whose physical dimensions are some algebraic combination of the fundamental dimensions. For each of the fundamental dimensions, the model builder chooses a standard unit of that dimension; these are called the base-units of the system. Then every other standard unit in the system is

a composition (using \* and expt) of units from the base set. For example, the Systeme International (SI) is a system of units that defines a set of seven fundamental dimensions with the base-units meter, kilogram, second, ampere, Kelvin, mole, and candela.

```
(defrelation SYSTEM-OF-UNITS(<=> (system-of-units ?s)
  (and (class ?s)
        (subclass-of ?s unit-of-measure)
;; The base-units of the system are with fundamental dimensions
(defined (base-units ?s))
  (=> (member ?unit (base-units ?s))
        (instance-of ?unit ?s))
  (orthogonal-dimension-set
    (setofall ?dim
      (exists ?unit
        (and (member ?unit
              (base-units ?s))
              (= ?dim
                 (quantity.dimension ?unit)))))))
;; Every unit in the system is the standard unit for its dimension.
  (=> (instance-of ?unit ?s)
        (= (standard-unit ?s
            (quantity.dimension ?unit))
           ?unit))))))
(defrelation ORTHOGONAL-DIMENSION-SET
  (<=> (orthogonal-dimension-set ?s)
    (and (set ?s)
          (=> (member ?d ?s)
                (and
                 (physical-dimension ?d)
                 (not (dimension-composable-from ?d
                    (difference ?s (setof ?d))))))))))
(defrelation DIMENSION-COMPOSABLE-FROM
  (<=> (dimension-composable-from ?d ?s)
    (or (member ?d ?s)
        (exists (?d1 ?d2)
          (and (dimension-composable-from ?d1 ?s)
                (dimension-composable-from ?d2 ?s)
                (= ?d (* ?d1 ?d2))))
        (exists (?d1 ?real)
          (and (dimension-composable-from ?d1 ?s)
                (real-number ?real)
                (= ?d (expt ?d1 ?real)))))))
```

### 3.9 Algebraic Properties of Quantities

We can borrow the well-established theories of algebra for the reals, vectors, and higher-order tensors. To adapt them to physical quantities we must consider physical dimensions in describing the domain and range of operators. We have already seen how dimensional homogeneity is a prerequisite for adding quantities, and physical dimensions establish classes of quantities that are comparable. For each physical dimension, the class of constant scalar quantities of a physical dimension forms an abelian group with the addition operator + and a zero identity element for that dimension (zeros of each dimension are different). The class of all scalars of any dimension, after removing the zero scalars, forms an abelian group with respect to multiplication.

For vector quantities, the sum of the quantities is only defined where the sum of the dimensionless versions of the vectors would be defined (i.e., the spatial dimensions must align). For higher-order tensors, tensor order and spatial dimensions, as well as the physical dimension, must be homogeneous. Analogous restrictions apply for the multiplication of tensors.

For function quantities, the sum or product of two function quantities is another function quantity that is only defined where the domains of the functions are equal. For unary scalar function quantities, the addition and multiplication operators are defined to handle a mix of function quantities and constant quantities. The sum of a constant  $k$  and a time-dependent function  $f$ , for example, is a function defined everywhere  $f(t)$  is defined and is equal to  $f(t)+k$ . Continuous time-dependent quantities can also be defined from others using a time-derivative function.

Most of the axiomatization in the specialized theories for functions, vectors, and tensors is concerned with specifying the conditions under which algebraic operators apply. This is essential for building agents with guarantees of completeness for some class of quantities.

## 4. Rationale for Important Distinctions

There are many possible ways to axiomatize this domain, and the axiomatization makes explicit many things that are implicit in the engineering literature. In this section, we discuss important distinctions that the formalization process forces one to clarify. For each distinction we offer a rationale for the choices made in terms of the purpose of the ontology.

### 4.1 Quantity Types Versus Instances

Some authors define "a quantity" as a set of values or a property having some order [11]. However, they offer no name for instances of this set, or for the values of this property. Following the AI convention of defining concepts as classes (or equivalently, sets, types, monadic predicates), we define the class physical quantity, and define species of quantities as subclasses of physical quantity. Where some would talk of the unitary concept "the mass quantity," we would say that there is a quantity type whose elements are constant scalars of dimension mass. This allows us to support the common usage in engineering modeling, where specific values such as the length of a beam are called quantities. It also allows us to modularly state properties of quantities, function quantities, scalar quantities, scalar-mass quantities, etc., and have the general properties inherit to the specializations.

### 4.2 Quantities are not variables

Physical quantities and physical dimensions are objects in the universe of discourse, and are not linguistic elements (e.g., "variables" of constraint expressions). Constraints over quantities that are found in engineering models are typically algebraic equations or inequalities, sometimes including the operations of differential calculus. Quantities are very different from linguistic elements. We have built ontologies (for the specification of a configuration design task for elevators--the VT experiment) in which constraint expressions and their constituents (variables and arithmetic operators) are part of the domain of discourse. In that formalization of constraints, constraint expressions are logical sentences with nonlogical constants (not logical variables) denoting physical quantities. The ontological distinction between variables and quantities allowed us to write an ontology in which both the form and denotation of constraint expressions were specified. This is necessary when the committing parties need restrictions on the form of expressions to guarantee completeness.

### 4.3 Quantities are not properties of objects

The identity of quantities is independent of physical objects that might be modeled. For example, let us say that the mass of a body B is the quantity 50kg. 50kg is just a measure of mass; there is nothing intrinsic about 50kg that says it is the mass of B (there is no total function from masses to bodies). Making quantities independent of objects also allows one to state that the mass of B is *equal* to the mass of a different body. It also supports pure parametric models, in which there are no physical objects and only quantities are described.

Of course, the model builder is free to define a function from physical objects to quantities, but this function is not the same thing as a mass quantity. Our formulation of quantities does not preclude such object-to-quantity functions; it provides the language in which to describe the range of those functions. In CML [12], for example, there are functions from objects (e.g. pipes) to time-dependent function

quantities. This distinction is central to the semantics of CML, which allows both time-varying and static relations. The object-to quantity functions are time-independent (e.g., the pipe always has a flow rate), but the quantities are functions of time (e.g., the flow rate is a function quantity that has different values throughout a scenario and can be undefined on some values of time).

#### 4.4 Quantities are not tuples

In our ontology, quantities exist as values that are related to, but independent of units and numbers. Quantities can be compared, multiplied, etc., without ever converting to reals or considering units of measure. This independence allows the model builder the flexibility needed to build measurement systems and physical theory independently. Furthermore, although quantities have algebraic properties, they are not purely abstract entities; we have grounded them in the modeled world with the condition of comparability. In contrast, an alternative formulation one often sees is to treat quantities as tuples of numbers and units [2,36] or as simply numbers. Besides the assumption about units that such a formulation makes, we find that it contradicts the conceptualization of quantities in the world made by physicists and philosophers. To reuse a simple example: 3 feet and 1 yard are equal, yet the tuples  $\langle 3, \text{ft} \rangle$  and  $\langle 1, \text{yd} \rangle$  are not.

#### 4.5 There are no fundamental quantities

The identity of a physical quantity is also independent of any fundamental quantities. We make no ontological distinction between base and derived quantities. For any pair of comparable physical quantities, their sum and product (if defined) are also physical quantities (the sum will be comparable with the original two; the product may not). This is also independent of scales, which are discussed below.

The rationale for this decision is again to provide generality and flexibility. We know from physics that there is no physical basis for giving some quantities primacy. Measurement theories make a distinction between quantities amenable to fundamental and associative measurement [6]. Again, we have made this distinction irrelevant for engineering models by avoiding the temptation to define quantities in terms of measurement. The analogous argument holds for not giving special status to some units of measure or physical dimensions. Even though it is possible to *compose* units and dimensions, there is no need to stipulate in the shared theory exactly which units and dimensions are fundamental.

#### 4.6 Nonphysical vs. Dimensionless Quantities

The conceptualization leaves it to the model builder to define the fundamental dimensions for a domain. The definition of physical dimension does not preclude one from defining new dimensions not mentioned in physics texts. We cited the example of amount of money as a possible physical dimension. The ontology also allows for dimensionless quantities, such as real numbers, whose physical dimension is the identity dimension. How does one decide whether to define a new physical dimension for a nonphysical quantity or to make it a dimensionless quantity?

We say that a physical dimension distinguishes a type or class of quantities that can be meaningfully combined with algebraic operations, can undergo unit conversion, and are comparable. Amount of money is a meaningful dimension because one can accumulate sums of money, do currency conversion, and compare relative wealth. Amount of money can be meaningfully combined with other dimensions. A rate of inflation, for example, is computed by dividing an amount of money (a change in price) by a different amount of money (the base price) and dividing the result by a unit of time (a year). The rate is a quantity of dimension (expt time -1). Money is something to be tagged as different in type from other quantities; that's why we see dollar signs (or other units) carried with the numbers in formulae.

As a negative example, consider quantities like number of politicians and number of constituents. We might write a formulae

```
(= Nconstituents (* Npoliticians 1000000))
```

In this model, we are making an abstraction; these quantities of humans are being compared as numbers. The = sign implies that the quantities must be of the same physical dimension, which would be, in this case, the identity dimension. Suppose, however, that we wanted to describe the number of molecules in those politicians. There is an international unit for number of molecules, the Mole.[Note 3] In the modern SI system of units, this is not a dimensionless quantity, but a quantity of the dimension amount of substance. Why does amount of substance get special status over number of politicians? Because chemical models need to distinguish amount of substance quantities from other measures. There is something homogeneous about molecules in that it makes sense to measure stuff by counting them. The formula for the average amount of gas in those politicians would use an amount of substance quantity for the amount of gas, and a dimensionless quantity for the number of politicians. This makes sense in the chemist's abstraction of politicians: that they can be viewed as volumes of gas.

## 4.7 Measurement Scales

A measurement scale is a determinative, non-degenerative assignment of numeric values to physical quantities [11]. Determinative means that the same quantities are consistently assigned the same numeric values, and non-degenerative means that different quantities get different values (ignoring issues of precision). Examples of measurement scales include Mohs scale for the hardness of minerals and the Celsius and Kelvin scales for thermodynamic temperature. Using the Coombs [7] classification, the Mohs scale is *ordinal* because it only provides information for inequality comparisons among points on the scale. The Celsius scale is classified as *ordinal-interval* because it supports inequalities between intervals on the scale, and Kelvin is a *ratio* scale because it supports comparisons of the ratio of any two values on the scale. For instance, it makes sense to say that 6 degrees Kelvin is twice as much as 3 degrees Kelvin.

By this mathematical definition, the units of measure in our conceptualization correspond to ratio scales.[Note 4] Each value on the scale is the ratio of the measured quantity to the degree-Kelvin unit. Thus the Kelvin scale can be defined from the degree-Kelvin using the magnitude function:

$$(\lambda \text{ (?q) (magnitude ?q degree-kelvin)})$$

We don't call Mohs ratings or degrees-Celsius units of measure, because they aren't quantities against which to compare other quantities. Of course one can write a function that does measure any temperature on the Celsius scale, such as

$$(\lambda \text{ (?q) (- (magnitude ?q degree-kelvin) 273.15)})$$

Since there is no principled constraint on the form of such function, we leave it to the model builder to define scales appropriate to a domain.

## 5. Design and Evaluation Issues

We view ontologies as designed artifacts, formulated for specific purposes and evaluated against design criteria. In a separate paper [19], we propose a set of ontology evaluation criteria and show examples of their application to this and other domains.

Two of the criteria will be illustrated here: to minimize *ontological commitment* while allowing for *monotonic extendibility*. Minimizing ontological commitment means making as few claims as possible about the world being modeled, allowing the parties committed to the ontology freedom to specialize and instantiate the ontology as needed. Extendibility means an ontology should be crafted so that one can extend and specialize the ontology *monotonically*. In other words, one should be able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions. Both of these criteria hold a natural tension with the goal of supporting the knowledge sharing needs of a range of agents with differing abilities and assumptions. Adding vocabulary to handle a broad range of representation needs will increase ontological commitment by

adding more constraints on the interpretation, and will make it more likely that some definitions will be incompatible with future representation needs.

In this section, we will discuss two techniques in the design of the EngMath ontology that help us meet these two criteria. One is the decomposition of a large ontology into modules. The second is another familiar design technique--parameterization--applied to the problem of representing conventions.

## 5.1 Decomposing the Ontology

The first technique is to decompose a monolithic ontology into a set of loosely coupled sub-ontologies. The EngMath ontology is decomposed into an inclusion lattice of individual ontologies, where each ontology is a set of definitions, and ontology inclusion is set inclusion. If an ontology B *includes* ontology A, then ontology B is the union of the definitions in A with those specific to B. More sophisticated methods for partitioning knowledge bases into modular theories are being explored [25], but set inclusion is sufficient for mutually consistent ontologies in a uniform namespace.

Figure 1 shows the inclusion lattice of theories for the EngMath family. The core ontology of physical quantities includes abstract algebra (evolved from the example in the KIF 3.0 specification [16]) and a theory of objects and relations called the Frame Ontology [20].

The EngMath family includes ontologies for Scalar Quantities, Vector Quantities, and Unary Scalar Functions. The Standard Units ontology defines many of the most common physical dimensions and units, and includes the SI system. Other engineering ontologies that build on the EngMath family--for describing component structure, design tasks, discrete events, and specific analysis domains such as kinematics--are being developed.



Figure 1: Inclusion lattice of ontologies

Decomposing into loosely coupled ontologies helps minimize ontological commitment by allowing one to commit to a coherent subset of the axioms of the entire ontology. For example, one can commit to scalars but not vectors; unary functions (for ODE models) but not n-ary functions (for PDE models); and static models (no functions). Even an agent that does not include physical dimensions in its conceptualization can be accommodated, since all the real number operations are also defined for quantities, and the reals are quantities.

Decomposition also supports the specialization of monotonically extendible ontologies. For example, the CML ontology inherits the basic commitments of the Unary Scalar Functions ontology and adds notions specific to its needs. Since the definitions in Unary Scalar Functions anticipate the general class of time-dependent functions, the CML extensions were possible without redefining the originals.

Designing a family of coherent ontologies is more difficult than designing a monolithic one, since the designer must anticipate intermodule interactions, such as axioms in several ontologies constraining the same vocabulary. To help manage the extension of vocabulary across ontologies, we borrow a technique from software engineering: *polymorphism*.

Polymorphism allows the same function constant to be defined in several places, where each definition adds axioms about the use of the constant. The `+`, `*`, and `exp` functions, for example, are polymorphically extended to each type of quantity and to physical dimensions. The form of a polymorphic definition of a function F is

```
(=> (and (p ?x) (q ?x))
      (=> (= (F ?x ?y) ?z) (r ?x ?y ?z)))
```

where  $x$  and  $y$  are the arguments of  $F$ ,  $p$  and  $q$  are the domains over which this definition applies (e.g., quantity types), and  $r$  is the condition that holds for  $F$ 's arguments and its value,  $z$ .

Due to theory inclusion, the definition of a function is the union of axioms contributed by each theory. For example, the definition of  $+$  for the Vector Quantities ontology is the union of the axioms for  $+$  for vector quantities,  $+$  for scalar quantities,  $+$  for physical quantities in general, and  $+$  for the reals.

## 5.2 Parameterizing Conventions

A second technique to facilitate extendibility and to minimize commitment is the parameterization of conventions. A parameter of a black box system is a representation of assumptions about how the environment interacting with the system will vary. In a computer program, the environment interacts with the program through formal arguments that are bound on invocation (and through the human interface, which is not as neatly parameterized). In the domain of the EngMath ontologies, the environment is the instantiation of the shared vocabulary for particular domain models. We have described several ways in which the domain models vary *by convention*. The choice of fundamental physical dimensions and standard units for a system, and the choice of particular units for individual quantities are conventions.

To minimize ontological commitment, we formulate these choices as parameters of the engineering model, rather than global constants of the shared ontology. To support extendibility, we provide an expressive representational vocabulary for specifying values of parameters. For example, to allow the model builder to specify the choices of fundamental dimensions and standard units, we provide the machinery to define systems of units.

It is in this sense that ontologies are a "coupling mechanism" [18] for knowledge bases and knowledge based agents. Parameters such as the system of units for a domain model are like abstract data types in software, except that the latter are ground and parameters of a domain model can be *theories*. We mentioned that the Unit Conversion agent can take, as inputs, ontologies specifying systems of units. When the other SHADE tools exchange sets of equations they are also exchanging theories. When agents pass around theories, the constraints on what they can pass around is specified in a shared ontology. In this sense, ontologies play a similar role as database schemata, except that ontologies may require a more expressive language for constraints than is typical in database specifications.

## 6. Related Work

### 6.1 Philosophy

The mathematics underlying the EngMath ontologies is drawn from standard textbooks used to teach engineeringmath [26, 31, 37], and from the philosophy literature on measurement [6, 11]. We had to choose distinctions from the conceptualizations, and modify them to produce the modular, internally coherent ontologies in the EngMath family. The engineering texts assume a conceptualization motivated by abstract algebraic operations, and the philosophy texts explore the grounding of the abstractions.

The aim of the philosophy texts is to describe The World as it Is in its Entirety, and to relate the results to prior writings. For the philosopher, it is important to relate the nature of quantities to the process of measurement and observation, and more generally, to the question of scientific knowledge. For instance, even the very notion that quantities such as mass can be meaningfully expressed as linear ratios of a standard turns out to be a convention, albeit a very useful and familiar one. Ellis[11] argues that the notion of a unit is incomplete without a choice of how such units are combined (which is related to, but not the same as, measurement procedure). We are assuming that there is a shared interpretation to the result of multiplying the meter times the real number 1000.

For our purposes--the sharing and communication of engineering models in machine and human readable forms--it is an *advantage* to be able to isolate the meaning of quantities from the process of measurement. We make no apologies: this is not a sweeping-under-the-rug of relevant issues, but a

strategic decoupling of issues. In building ontologies we are writing social contracts. We are free to invent the conceptualization as long as its meaning can be effectively communicated (which is why we use standard terminology and logic). By accepting the KIF language as a foundation, for example, we already commit to including abstract things like sets and relations in the universe of discourse. We add to that an ontology of abstract algebra, and extend it to physical quantities. The philosophical ontologist draws a heavy line when the objects cross from timeless mathematical entities to physical concepts like mass and length. Our agents need no strong boundary; if our engineering model states that a quantity of mass exists and is related to a length quantity by some algebraic expression, it is so for the agent. According to our evaluation criteria [19], clarity and coherence in the specification are paramount, and faithfulness with The World is not an issue.

Nonetheless, we can "share and reuse" the analysis found in philosophy writing, for very pragmatic ends. For example, one of the differences between a casual writing on quantities and a careful one is the treatment of property association. One often sees it stated that quantities are "properties" of "objects" -- like qualities but quantitative. However, the careful writer will point out that quantities as we use them are *relational*--they are about comparing objects in some respect, or about relationships among them (e.g., distance between reference points). This informs our design. Departing from the conventional "object oriented" approach, we give independent status to quantities and leave it as a modeling decision to map from objects to quantities. This is, in essence, a decoupling of theories about quantities from theories about model formulation.

Similarly, an understanding of the nature of physical theory and measurement guides us away from the impulse to oversimplify for the sake of computational elegance. For example, it would be simpler from a computational point of view to fix a single basis set of "fundamental" physical dimensions or units of measure, and recursively derive the rest. However, the laws of physics tell us that there are no inherently privileged physical dimensions, and the study of measurement tells us that the choice of basis sets for dimensions and units is a convention. The fact that engineers must deal with at least two systems of units, each of which chooses a different basis set (and which has changed over historical time), motivates us to provide the model builder with the representational machinery to define a system of units as part of the domain model.

## 6.2 Other Work on Ontologies

Work in Formal Ontology in the philosophy literature that is relevant to knowledge sharing ontologies are found in [4, 5], and (combined with papers from AI) in [24].

There is a growing body of ontologies appearing in the literature seen by the knowledge representation community, including as a sample [3, 10, 28, 30, 39].

The most closely related on engineering ontology is the thesis by Alberts [2]. Alberts describes a formal ontology intended as the basis for building interoperable and reusable knowledge systems for design. His ontology provides a vocabulary for modeling the structure and behavior of systems, based on systems theory [38] and finite element modeling. While the formalization of systems is exemplary, the treatment of quantities in that ontology is simplistic. First, quantities have no status other than as the values of variables with symbolic 'quantity types.' There is no provision for defining new dimensions or describing complex dimensions; the quantity types appear to be those that are anticipated by systems theory (analogs of effort and flow). Second, the values are always unary scalar functions of time, where time is "exceptional" (i.e., outside the range of the model). This prevents vector and tensor models, PDE models, phase-space models where time is not the independent variable, etc. Third, the values of these functions are tuples of numbers and fixed units.

More recent work by Akkermans and Top [1] develops the systems theory methodology to maturity. It proposes engineering ontologies at four levels of description: functional components, physical processes, mathematical relations, and model data. Work on model formulation using the CML language [12] and SHADE engineering agents [32] aims at a suite of ontologies not based on system theory that have similar coverage. Perhaps the ontology presented in this paper can provide a foundation for the mathematical and data level of models in these comprehensive engineering ontologies.

## Acknowledgments

The work is supported by ARPA prime contract DAAA15-91-C-0104 through Lockheed subcontract SQ70A3030R, and NASA Grants NCC 2-537 and NAG 2-581 (under ARPA Order 6822). Ideas on knowledge sharing have been shaped by rewarding conversations with Richard Fikes, Mike Genesereth, Pat Hayes, Doug Lenat, Bob Neches, and Marty Tenenbaum. We thank the anonymous KR'94 reviewers, Adam Farquhar, Pat Hayes, and Dan Russell for thoughtful reviews.

## References

- [1] H. Akkermans & J. Top. Tasks and ontologies in engineering modeling. *Proceedings of the 8th Knowledge Acquisition for Knowledge Based Systems Workshop*, Banff, Canada, 1994.
- [2] L. K. Alberts. *YMIR: an ontology for engineering design*. Doctoral dissertation, University of Twente, 1993.
- [3] J. A. Bateman, R. T. Kasper, J. D. Moore, & R. A. Whitney. A General Organization of Knowledge for Natural Language Processing: The Penman Upper Model. USC/Information Sciences Institute, Marina del Rey, CA, Technical report 1990.
- [4] M. Bunge. *Treatise on Basic Philosophy, Volumes 3-4: Ontology*. D. Reidel Publishing Company, Dordrecht, Holland, 1979.
- [5] H. Burkhardt & B. Smith (Eds.). *Handbook of Metaphysics and Ontology*. Philosophia Verlag, Munich, 1991.
- [6] N. R. Campbell. *An Account of the Principles of Measurement and Calculations*. Longmans, Green, London, 1928.
- [7] C. H. Coombs. The theory and methods of social measurement. In L. Festinger & D. Katz, Ed., *Research Methods in the Behavioral Sciences*, Dryden Press, New York, 1952.
- [8] J. Crawford, A. Farquhar, & B. Kuipers. QPC: A Compiler from Physical Models into Qualitative Differential Equations. *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, pages 365-371. AAAI Press/The MIT Press, 1990.
- [9] M. Cutkosky, R. S. Englemore, R. E. Fikes, T. R. Gruber, M. R. Genesereth, W. S. Mark, J. M. Tenenbaum, & J. C. Weber. PACT: An experiment in integrating concurrent engineering systems. *IEEE Computer*, **26**(1):28-37, 1993.
- [10] E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, 1990.
- [11] B. Ellis. *Basic Concepts of Measurement*. Cambridge University Press, London, 1966.
- [12] B. Falkenhainer, A. Farquhar, D. Bobrow, R. Fikes, K. Forbus, T. Gruber, Y. Iwasaki, & B. Kuipers. CML: A compositional modeling language. Stanford Knowledge Systems Laboratory, Technical Report KSL-94-16, January 1994.
- [13] B. Falkenhainer & K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, **51**:95-143, 1991.
- [14] K. D. Forbus. Qualitative Process Theory. *Artificial Intelligence*, **24**:85-168, 1984.
- [15] M. R. Genesereth. An Agent-Based Framework for Software Interoperability. *Proceedings of the DARPA Software Technology Conference*, Meridian Corporation, Arlington VA, pages 359-366. 1992.

- [16] M. R. Genesereth & R. E. Fikes. Knowledge Interchange Format, Version 3.0 Reference Manual. Computer Science Department, Stanford University, Technical Report Logic-92-1, March 1992.
- [17] R. Genesereth & N. J. Nilsson. Logical Foundations of Artificial Intelligence. Morgan Kaufmann Publishers, San Mateo, CA, 1987.
- [18] T. R. Gruber. The Role of Common Ontology in Achieving Sharable, Reusable Knowledge Bases. In James A. Allen, Richard Fikes, & Erik Sandewall, Ed., *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, Cambridge, MA, pages 601-602. Morgan Kaufmann, 1991.
- [19] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Nicola Guarino, Ed., International Workshop on Formal Ontology, Padova, Italy, 1992. Revised August 1993, to appear in *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Guarino & Poli (Eds), Kluwer, in preparation.
- [20] T. R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 5(2):199-220, 1993.
- [21] T. R. Gruber & G. R. Olsen. The Engineering Math Ontologies. World Wide Web, URL "<http://www-ksl.stanford.edu/knowledge-sharing/README.html>", 1994.
- [22] T. R. Gruber, J. M. Tenenbaum, & J. C. Weber. Toward a knowledge medium for collaborative product development. In John S. Gero, Ed., *Artificial Intelligence in Design '92*, pages 413-432. Kluwer Academic Publishers, Boston, 1992.
- [23] N. Guarino. An ontology of meta-level categories. KR'94.
- [24] N. Guarino & R. Poli (Eds.). *Formal Ontology in Conceptual Analysis and Knowledge Representation*. Kluwer, in preparation.
- [25] R. V. Guha. *Contexts: A formalization and some applications*. doctoral dissertation, Stanford University, 1991.
- [26] D. Halliday & R. Resnick. *Physics*. John Wiley and Sons, New York, 1978.
- [27] G. Hirst. Ontological assumptions in knowledge representation. KR'89, 1989.
- [28] J. R. Hobbs & R. C. Moore (Eds.). *Formal Theories of the Common Sense World*. Ablex, Norwood, NJ, 1985.
- [29] Y. Iwasaki & C. M. Low. Model Generation and Simulation of Device Behavior with Continuous and Discrete Changes. *Intelligent Systems Engineering*, 1(2)1993.
- [30] D. B. Lenat & R. V. Guha. *Building Large Knowledge-based Systems: Representation and Inference in the Cyc Project*. Addison-Wesley, Menlo Park, CA, 1990.
- [31] B. S. Massey. *Measures in Science and Engineering: Their Expression, Relation, and Interpretation*. Ellis Horwood Limited, 1986.
- [32] J. G. McGuire, D. R. Kuokka, J. C. Weber, J. M. Tenenbaum, T. R. Gruber, & G. R. Olsen. SHADE: Technology for knowledge-based collaborative engineering. *Journal of Concurrent Engineering: Applications and Research (CERA)*, 1(2)1993.
- [33] R. Neches, R. E. Fikes, T. Finin, T. R. Gruber, R. Patil, T. Senator, & W. R. Swartout. Enabling technology for knowledge sharing. *AI Magazine*, 12(3):16-36, 1991.

- [34] A. Newell. The knowledge level. *Artificial Intelligence*, 18(1):87-127, 1982.
- [35] R. S. Patil, R. E. Fikes, P. F. Patel-Schneider, D. McKay, T. Finin, T. R. Gruber, & R. Neches. The DARPA Knowledge Sharing Effort: Progress report. *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, Cambridge, MA, Morgan Kaufmann, 1992.
- [36] D. A. Randall & A. G. Cohn. Modelling topological and metrical properties in physical processes. *KR'89*, Morgan Kaufmann, 1989.
- [37] W. C. Reynolds & H. C. Perkins. *Engineering Thermodynamics*. McGraw-Hill, 1977.
- [38] R. C. Rosenberg & D. C. Karnopp. *Introduction to physical system dynamics*. McGraw-Hill, New York, 1983.
- [39] Y. Shoham. Temporal logics in AI: Semantical and ontological considerations. *Artificial Intelligence*, **33**:89-104, 1987.
- 

## Notes

[Note 1]By "what can exist" we mean "anything that can be spoken of," including all of the varieties of existence identified by Hirst [27]. The purpose of our specifications are not to give ontological status to various modes of being, but to offer a way of representing things in a shared conceptualization.

[Note 2]For the purpose of specifying common conceptualizations, we see no justification for restricting the form of the definitions e.g. to necessary and sufficient conditions, or to require that they be conservative definitions (which make no claims about the world).

[Note 3]It's actually number of particles, but if you understand that distinction you understand our point here.

[Note 4]Ellis (1969) gives no special status to units, saying that they are merely the names for the associated scales. We wanted to allow for agents to commit to unit conversion without committing to a theory of measurement scales.