NIST Technical Note 1943

Architecture for Software-Assisted Quantity Calculus

David Flater

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1943

National Institute of
Standards and Technology
U.S. Department of Commerce

NIST Technical Note 1943

Architecture for Software-Assisted Quantity Calculus

David Flater
Software and Systems Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1943

December 2016

U.S. Department of Commerce
Penny Pritzker, Secretary

National Institute of Standards and Technology
Willie May, Under Secretary of Commerce for Standards and Technology and Director

National Institute of Standards and Technology Technical Note 1943
Natl. Inst. Stand. Technol. Tech. Note 1943, 14 pages (December 2016)
CODEN: NTNOEF

This publication is available free of charge from:
https://doi.org/10.6028/NIST.TN.1943

Architecture for Software-Assisted Quantity Calculus

David Flater

2016-12-01

Abstract

A quantity value, such as 5 kg, consists of a number and a reference (often an International System of Units (SI) unit) that together express the magnitude of a quantity. Many software libraries, packages, and ontologies that implement "quantities and units" functions are available. Although all of them begin with SI and associated practices, they differ in how they address issues such as *ad hoc* counting units, ratios of two quantities of the same kind, and uncertainty. This short technical note describes an architecture that addresses the complete set of functions in a simple and consistent fashion. Its goal is to encourage more convergent thinking about the functions and the underlying concepts so that the many disparate implementations, present and future, will become more consistent with one another.

1 Introduction

Originally, mathematical operations applied only to numbers, and units of measure were just information that described what the numbers represented. This belief gave way to the practice of including units of measure within the scope of the mathematical operations, thereby formalizing the method of working with combinations of units. The resulting *quantity calculus* methodically determines the units of derived quantities and protects us from the error of computing nonsensical combinations of quantities that have different dimensions [1]. For example, if a property line is moved by 3 m, we are allowed to add 3 m to the width of the lot, but we cannot add 3 m to the lot size that is measured in m^2. We must multiply 3 m by the depth of the lot (x m) to obtain the change to the lot size ($3x$ m^2).

Many software libraries and packages that implement "quantities and units" functions are available; for example:

- Mathematica, quantities and units in the Wolfram language [2];
- Maxima package ezunits [3];
- Modelica, physical variables and SIunits library [4];
- LabVIEW unit labels [5];
- MathCad units [6];
- GNU Units [7];
- Ruby gems for units of measure (e.g., ruby-units [8], phys-units [9], and unitwise [10]; there are at least 10);
- C++ libraries for units of measure (e.g., Boost.Units [11]);
- F# units of measure (a built-in language feature) [12];
- Python package numericalunits [13]; and
- Many others.

A comparable number of formal models and ontologies related to quantities and units exist; for example:

- Conceptual model of the International Vocabulary of Metrology (VIM) [14, Annex A];
- Dybkaer's Ontology on Property [15];
- Unified Code for Units of Measure (UCUM) [16];
- Quantities and Units of Measure Ontology Standard (QUOMOS) [17];

- Units Markup Language (UnitsML) [18];
- Quantities, Units, Dimensions, and Data Types Ontologies (QUDT) [19];
- Quantities, Units, Dimensions, Values (QUDV) in Systems Modeling Language (SysML) [20];
- Ontology of units of Measure and related concepts (OM) [21]; and
- Many others.

Although all of the above libraries, packages, models, and ontologies begin with the International System of Units (SI) [22] and associated practices, they differ in how they address issues such as *ad hoc* counting units, ratios of two quantities of the same kind, and uncertainty. These issues and others that have undermined the functionality and consistency of software for metrology have been discussed in related work [6, 23].

This technical note describes an architecture that addresses the complete set of functions for "quantities and units" software in a simple and consistent fashion. The goal is not to produce *yet another* implementation or framework to compete in an already overcrowded arena, but to identify the major architectural features that have proven to be important or sorely needed in this author's experience, and thereby encourage more convergent thinking and improved compatibility among different implementations in the future.

The building blocks of the architecture are:

- A catalog of recognized units and numerical prefixes;
- Values (in the measurement sense), including derived values and compound values;
- Probability distributions to characterize uncertain values;
- Automated propagation of uncertainty to derived values; and
- An extensible type system for specializations of the SI unit 1.

Section 2 through Section 6 address each of these in turn. Additional recommendations concerning numeric data types are given in Section 7. Section 8 summarizes. Discussion of alternative approaches is provided in the appendices.

2 Units

Quite simply, one cannot begin to implement quantity calculus without first having a catalog of units that the software will recognize and refer to. We need not belabor the point, but all "quantities and units" software must at least recognize the standard base units, derived units, and numerical prefixes (k, M, etc.) that are identified in the SI brochure [22] and be capable of converting "non-coherent" units to their standard equivalents.

3 Values

The following terms are defined by the 3rd edition of the VIM [14].

> **quantity**: property of a phenomenon, body, or substance, where the property has a magnitude that can be expressed as a number and a reference.

[The "reference" is typically an expression in terms of SI units.]

> **quantity value**: number and reference together expressing magnitude of a **quantity**.
>
> **measurement result**: set of **quantity values** being attributed to a **measurand** together with any other available relevant information.

[The "set of quantity values" is intended to accommodate uncertainty, given that a single true quantity value generally cannot be determined.]

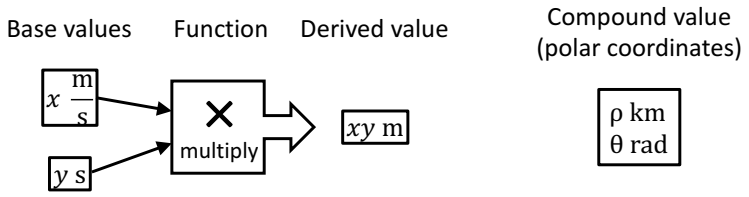

Figure 1: *Values* in a software implementation of quantity calculus.

A *value* in a software implementation of quantity calculus represents one or more measurement results with one or more **quantity value** variables. Those with only one such variable (the usual case) are *simple values*; the rest are *compound values*.

A *base value* represents immediate results of a measurement. A *derived value* is the result (output) of a function of other values (the inputs).

Figure 1 illustrates the concepts. On the left side, two base values, a speed expressed in meters per second and a duration expressed in seconds, are input to a function that multiplies them together, producing a derived value that is expressed in meters. On the right side, polar coordinates are given as an example of a compound value, because in most applications it is necessary to keep the radial coordinate and the angular coordinate together.

Like the catalog of SI units, simple values and value functions are essential to all software implementations of quantity calculus. Compound values, on the other hand, are not universally supported.

In the architecture described by this technical note, values are represented by distributions, as explained in the next section.

4 Distributions

To express the result of a measurement, it is common practice to state a single estimate with an interval expressed in the form $y \pm U$ [24, §6.2.1].[1] This approach is adequate much of the time, but not when the preferred interval is asymmetric around the estimate, is discontinuous, or otherwise fails to conform to the assumptions of the typical case.

A more general approach is to use probability distributions. Probability distributions take the place of both the point estimate and its conventional uncertainty. They may be continuous (for dimensional quantities and ratios) or discrete (for counted quantities). They may be univariate (for simple values) or multivariate (for compound values). They may be unimodal (for simple estimates) or multimodal (for when a single estimate would be misleading).

A quantity value that has zero uncertainty, such as a trivial count or a defined constant, is assigned a degenerate (deterministic) distribution that has only one possible value. A single estimate with expanded uncertainty U is typically translated as either a Gaussian distribution or a uniform (rectangular) distribution, depending on the state of knowledge. However, the distribution need not be Gaussian or uniform. Additional common cases are described in Supplement 1 of the Guide to the expression of Uncertainty in Measurement (GUM) [25, §6.4].

At present, most "quantities and units" software represents a value with a single estimate, with uncertainty being optional. Replacement of point estimates with distributions occurs only in software with a focus on uncertainty propagation, as described in the next section.

Note: Probability theory is itself a special case of more general theories that avoid the need to assign a probability distribution when the true distribution is unknown. Details and rationale for these options are provided in the appendices.

[1] The interval is usually described more specifically as a confidence interval, coverage interval, or credible interval, but each of these terms has statistical implications that are out of scope for the present discussion.

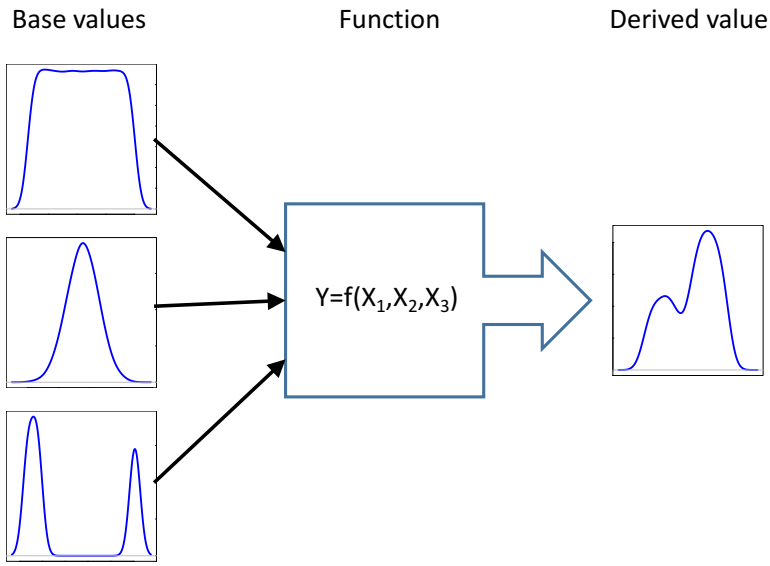

Figure 2: Propagation of uncertainty from the distributions of base values, through a function, to the distribution of a derived value.

5 Propagation of uncertainty

Distributions and the propagation of uncertainty can be implemented in two distinct ways:

1. A distribution is a black box software function that exposes only the interface to get sampled quantity values. A quantity value is characterized by a histogram or kernel density plot derived from a sufficiently large sample, and an interval is produced using statistical methods. When a derived value is sampled, its value is calculated using samples from other values as inputs to its function.

2. A distribution is a symbolic mathematical function that is either represented directly by the software or suitably approximated (e.g., with Chebfun [26]). Sampling is not required; instead, a quantity value is plotted directly, and intervals and derived values are derived by operating algebraically on the functions themselves.

A given system could implement both approaches. For example, the black box approach could be used as a fallback when it becomes infeasible to represent or derive a distribution algebraically.

Figure 2 illustrates the propagation of uncertainty using distributions. This is the same concept described in GUM Supplement 1 [25]. Implementations in software include Uncertain$\langle T \rangle$ [27], the NIST Uncertainty Machine [28], and many others [29]. Unfortunately, implementations are seldom pre-integrated with "quantities and units" software.

6 Subtyping unit 1

6.1 Background on SI unit 1

The following background on SI unit 1 and related concepts is necessary to explain the context of the proposed type system.

A quantity in SI can be stated as a mathematical expression—the product of a numerical value and a unit of measurement. The magnitude of a quantity can be expressed in terms of the seven SI base quantities length (m), mass (kg), time (s), electric current (A), thermodynamic temperature (K), amount of substance

(mol), and luminous intensity (cd), either individually or in combinations. The base quantities correspond to physical dimensions as used in dimensional analysis.

However, many kinds of quantities have no extent in any of the seven standard dimensions. For example, a *counted quantity* is a number of some distinguishable kind of thing, such as 23 neutrons. Subdividing a counted quantity consists of dividing a set of things into smaller sets which contain *fewer* of the kind of thing being counted, rather than *less* of something that would be measured on a continuous scale. What unit should such a quantity refer to, and what is the dimension corresponding to that unit? These questions are addressed in the VIM [14] and the SI brochure [22].

Under the current VIM, all counted quantities and all ratios of two quantities of the same kind are called *quantities of dimension one*, or alternately *dimensionless quantities*. The definition is "quantity for which all the exponents of the factors corresponding to the base quantities in its quantity dimension are zero." The unit of measure for such quantities is 1, the algebraic result of setting all of the exponents to zero.[2]

The current SI brochure unambiguously specifies that the coherent derived unit for any quantity that is defined as the ratio of two quantities of the same kind is always the number one, 1. It addresses counted quantities similarly but with some ambiguity, saying first that they are "*usually* regarded as dimensionless quantities, or quantities of dimension one, with the unit one, 1" (emphasis added) [22, §1.3], and later that they "are taken to have the SI unit one, although the unit of counting quantities cannot be described as a derived unit expressed in terms of the base units of the SI. For such quantities, the unit one may instead be regarded as a further base unit" [22, §2.2.3]. The extra words are explained by Mills: "Those dimensionless quantities defined as a ratio of two other quantities of the same kind (e.g., angles, mole fractions, etc.) suggest we should think of 1 as a derived unit, whereas counting suggests the concept of 1 as a base unit" [31].

Use of the unit 1 for all kinds of counts and all possible ratios of two quantities of the same kind has led to confusion and inconsistencies. Most prominently, the *de facto* standard practice of identifying the types of entities counted (e.g., neutrons or bits) as if they were units is inconsistent with official guidance [32, §7.5]. In addition, Mohr and Phillips noted that the absence of an explicit specification of cycles in the definition of hertz gives rise to errors of a factor of 2π and misuse of the unit hertz for rates of non-periodic events [33].

Software libraries and packages that implement "quantities and units" functions need to work with angles measured in radians, amounts of data measured in bits or bytes, and other quantities of SI dimension 1 in some way that is not astonishing to users [34]. As a result, they apply workarounds such as adding an explicit base unit for 1, treating radians as a special case, and allowing users to introduce arbitrary irreducible units. Different software has applied different workarounds, creating subtle problems for transfer of scientific data.

6.2 Proposed model

Here, again, we seek to encourage more convergent thinking by introducing a sufficiently general model. This aspect of the architecture has not yet, to the author's knowledge, appeared in any published software. However, it is based on the approach that was initiated by Mohr and Phillips, and is related to the conceptual analysis of quantities of dimension 1 that was done by Josef Kogan [35, §1.3].

The idea behind this model is to replace simple dimensional analysis using the seven SI dimensions with a version of quantity calculus that supports subtyping of the special unit 1. This enables traceability to SI to occur not only through direct reference to SI units (the identity relationship), but also through subtyping (the generalization/specialization relationship). For example, the radian becomes a specialization of the SI unit 1. Torque expressed in J/rad cannot then accidentally reduce to energy as it can when rad is just a "special name for the number 1" [22, Table 3, footnote b]. However, the reduction that is possible in SI can be obtained by generalizing rad up to the unit 1, which discards the "per *what*" information contained in the J/rad expression, leaving only J/1 = J.

[2] Krystek suggested that it would be better to refer to *dimension number*, with Z as its symbol and 1 as its coherent unit of measure [30].

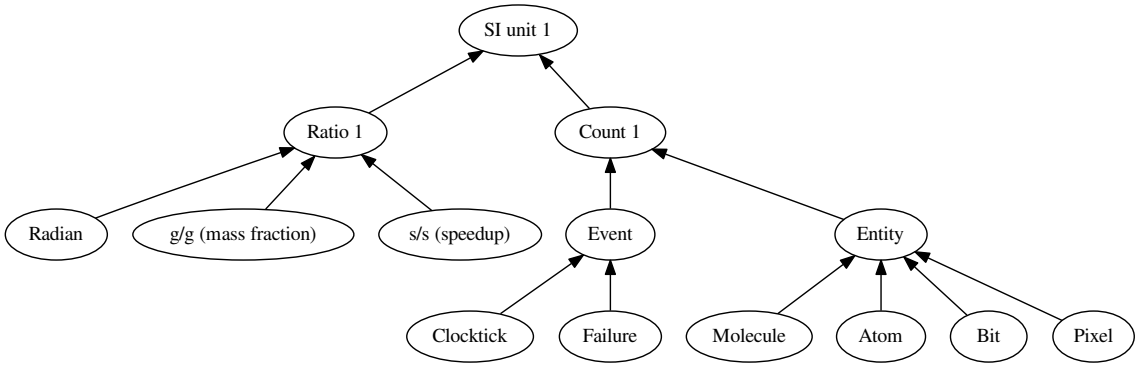

Figure 3: Types of non-dimensional units.

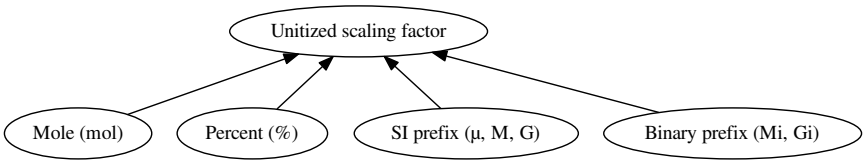

Figure 4: Types of "unitized" scaling factors, which represent pure numbers.

In the proposed model, the practice of using the types of entities counted as if they were units is reconciled with SI by making all "counting units" specializations of the SI unit 1. Consequently, the hertz that is interpreted as clockticks per second and the becquerel that is interpreted as nuclear decays per second will be distinguishable by the software, even though both of them reduce to the SI definition of s^{-1}. Moreover, the latter interpretation of hertz will be distinguishable from the one that is 2π rad/s.

The top levels of a type system that interprets dimensionless units, as well as numerical factors that have been "unitized" by convention, are shown in Figure 3 and Figure 4. All identified types are merely prominent examples chosen from a set that can be extended without limit. The system must be extensible, i.e., it must allow users to add new types, since it is plainly impossible to catalog every possible counting unit.

Scaling factors that are included in the unit expression can accurately be called "dimensionless" as they are simply numeric multipliers that were given designated symbols and then shifted from the numerical value to the unit expression ("unitized"). Once atoms and molecules are understood to be countable entities, the mole is reduced to a scaling factor that functions the same way as the existing SI prefixes k, M, etc., and percent. That is, when ^{12}C is treated as a counting unit (being a specialization of atom, which is a specialization of entity), the mol in mol ^{12}C functions the same way as the k in km.[3]

Scaling factors that have been included within unit expressions are not quantities of dimension 1, as modelled above, because they truly carry no information about the kind of quantity. Heretofore the mole was used only for amounts of substances, but in principle one could have a mole of any countable type of thing.

The type system does not need to be constrained to a taxonomic or tree structure (where each type has at most one direct supertype) but may instead be a lattice (where a type may have multiple direct supertypes). A rigorous application of lattice theory and set theory to deduce a type system based on attributes of objects studied can be found in Formal Concept Analysis (FCA) [36].

Introducing a type system of units for dimensionless quantities enables stronger integrity checking, but it is not a complete solution; rather, it is a compromise. Appendix C discusses a solution that is more complete but more onerous to implement.

[3] A minor issue is that the numerical value, Avogadro's number, that is specified indirectly in the 8th edition of the SI brochure is not known exactly and is not necessarily an integer. The redefinition of the mole that has appeared in drafts of the 9th edition eliminates this issue.

7 Numeric data types

It is unfortunately common in scientific applications for all numeric values to be represented using floating-point numbers as the catch-all data type. While additional numeric types and arbitrary-precision arithmetic are widely implemented, their integration with scientific applications in general and quantity calculus in particular is inconsistent.

In addition to the basic types of floating point, signed integer, and unsigned integer, the following constructed types are frequently useful:

- Fixed-point numbers, specifically integers that are scaled by powers of 10, for exact representation of amounts of money and other quantities where the decimal places are inflexible;
- Rational numbers, represented as the fraction of two integers, for exact representation of values like 1/3 that floating-point numbers can only approximate; and
- Complex numbers ($a + bi$), represented using two numbers of any of the preceding types.

Basic types should be available in both the fixed sizes that are natively supported by the CPU and in arbitrary-precision form. The constructed types, in turn, should be usable with either fixed-size or arbitrary-precision representations.

Finally, it should be possible to enable and handle exceptions for numeric overflow, underflow, etc. The user should not need to instrument every numeric calculation at the application level to try to detect or prevent these errors, which are properly detected and reported at the arithmetic-logic level.

8 Conclusion

This short technical note described an architecture that addresses the complete set of functions for quantities and units in a simple and consistent fashion. Hopefully, this will encourage more convergent thinking about the functions and the underlying concepts so that the many disparate software implementations, present and future, will become more consistent with one another.

Acknowledgments

Thanks to Raghu Kacker, Paul Black, and Barbara Guttman for helpful review comments and suggestions.

Appendix A Further generalized distributions

This appendix elaborates on the discussions of Section 4 and Section 5.

Beyond frequentist or Bayesian interpretations of probability, there are other related theories that may prove necessary or useful to accurately express states of knowledge (or ignorance) about quantity values. Dempster-Shafer theory, a.k.a. the theory of evidence [37], enables one to distinguish between lack of belief (due to lack of evidence) and disbelief (due to the existence of evidence that refutes the belief). The transferable belief model [38, 39] is a different branch off of the underlying theories that avoids introducing a closed-world assumption.

According to Salicone [40], epistemic uncertainty cannot validly be modelled with random variables. For example, although uncorrected and/or unknown systematic errors contribute to uncertainty in the general sense of the word, they cannot be controlled simply by taking larger samples; their expected value does not approach zero.

The GUM discusses valid modelling of uncorrected systematic effects only briefly [24, §F.2.4.5] because it is assumed that systematic effects will nearly always be corrected. Salicone instead models uncorrected systematic effects using the possibility (rather than probability) aspect of the theory of evidence.

Appendix B Limitations of principle of maximum entropy

This appendix explains a relevant motivation for more general distributions.

The principle of maximum entropy [41], which assigns a uniform distribution when only the range of possible values is known, can lead to inconsistencies when multiple variables or different scales are involved. This occurs, for example, when two variables, each of which should in principle be assigned a uniform distribution based on the state of knowledge, are related in such a way that a uniform distribution for one implies a nonuniform distribution for the other [42]. According to [43, §3.3], "such inconsistencies mean that maximum entropy cannot generally yield mathematically defensible selections."

When following the principle of maximum entropy, a variable whose value is believed to lie between x and y but for which there is no further information is modelled using a uniform random distribution over that interval. But the sum of a uniform distribution between x_1 and y_1 with a uniform distribution between x_2 and y_2 yields a trapezoidal distribution that peaks in the middle and goes to zero at the endpoints $x_1 + x_2$ and $y_1 + y_2$. The shape of the output distribution in this example is purely an artifact of the assumptions rather than an inference from knowledge. In truth, the shape of the output distribution is as unknown as the shapes of the input distributions. To comply with the principle of maximum entropy, the result should have been another uniform random distribution. This example can of course be answered correctly using interval math, but in most cases, distributions are more informative.

Using possibility theory instead of probability theory avoids the need to assume a probability distribution when the distribution of probability is unknown [40].

Appendix C Full tracking of kinds of quantities

This appendix discusses an integrity checking approach that is more complete than what is described in Section 6. To the author's knowledge, the implementations of this approach in software have been of very limited, application-specific scope. However, it is essentially a formalization and automation of the reasoning that all scientists currently are expected to do on their own.

For rigorous integrity checking, one must extend the implementation of quantity calculus to explicitly track **kinds of quantities** in addition to units. For example, in addition to performing the algebra on units to determine that $x\,\frac{\mathrm{J}}{\mathrm{rad}} \times y\,\mathrm{rad} = xy\,\mathrm{J}$, the system would explicitly understand that *torque* multiplied by *angular displacement* yields *work*.

To model all kinds of quantities and their functional interactions is a more ambitious undertaking than simply implementing a type system for units. When extending a type system for units, it is only necessary to identify the supertypes of a type being added; but when extending a system of quantity kinds, one must also identify meaningful functional interactions that the new kind has with previously-defined kinds. For example, consider the functional interactions that the kinds *distance* and *amount of rainfall* would have with *fuel consumption*. *Distance* and *amount of rainfall* may both be stated in meters, but while *fuel consumption* multiplied by *distance* yields *amount of fuel*, *fuel consumption* multiplied by *amount of rainfall* is an error.

Identifying the interactions for every possible combination of kinds of quantities known to science seems over-ambitious. Catalogs of kinds do already exist within ISO/IEC 80000 [44] and International Union of Pure and Applied Chemistry (IUPAC) recommendations [45], but these references only identify the most important kinds without identifying their interactions.

On the other hand, when the scope is limited to a particular application that uses only a handful of quantity kinds, it is both easy and productive to implement the corresponding application-specific system of kinds

and their interactions. In an object-oriented programming language that supports operator overloading, mapping quantity kinds onto classes results in concise and meaningful source code.

Full tracking of kinds of quantities can coexist with a type system for units, with the caveat that it is redundant for quantities to be distinguished both by specialized units and by kind.

References

[1] J. de Boer. On the history of quantity calculus and the international system. *Metrologia*, 31(6):405–429, 1995. https://doi.org/10.1088/0026-1394/31/6/001.

[2] Units—Wolfram language documentation, 2016. https://reference.wolfram.com/language/guide/Units.html.

[3] Ezunits, in Maxima 5.38.1 manual, 2016. http://maxima.sourceforge.net/docs/manual/maxima_56.html.

[4] Modelica SIunits library, 2009. http://goo.gl/0fEQ8N.

[5] Available units in LabVIEW—LabVIEW 2016 help, June 2016. http://zone.ni.com/reference/en-XX/help/371361N-01/lvhowto/available_units_in_labview.

[6] V. F. Ochkov, V. I. Lehenkyi, and E. A. Minaeva. Physical quantities, dimensions and units in mathematical packages. *Mathematical Machines and Systems*, 1:78–90, 2009. http://twt.mpei.ac.ru/ochkov/Units/QDUeng.pdf.

[7] GNU Units, 2016. https://www.gnu.org/software/units/.

[8] Kevin C. Olbrich. Ruby-units, 2016. https://github.com/olbrich/ruby-units.

[9] Masahiro Tanaka. Phys-units, 2016. https://github.com/masa16/phys-units.

[10] Josh W. Lewis. Unitwise, 2016. https://github.com/joshwlewis/unitwise.

[11] Matthias C. Schabel and Steven Watanabe. Boost.Units 1.1.0, 2016. http://www.boost.org/doc/libs/1_61_0/doc/html/boost_units.html.

[12] Den Delimarsky. Units of measure, in F# language reference, 2016. https://goo.gl/eOsNa5.

[13] Steven Byrnes. Numericalunits, 2016. https://pypi.python.org/pypi/numericalunits.

[14] Joint Committee for Guides in Metrology. *International vocabulary of metrology—Basic and general concepts and associated terms (VIM)*, 3rd edition. JCGM 200:2012, http://www.bipm.org/en/publications/guides/vim.html.

[15] René Dybkaer. *An ontology on property for physical, chemical, and biological systems*. IUPAC, 2009. https://doi.org/10.1351/978-87-990010-1-9.

[16] Unified Code for Units of Measure, 2016. http://unitsofmeasure.org/.

[17] OASIS. Quantities and Units of Measure Ontology Standard, 2016. https://oasis-open.org/committees/quomos/.

[18] Units Markup Language (UnitsML), 2011. http://unitsml.nist.gov/, https://oasis-open.org/committees/unitsml/.

[19] Ralph Hodgson, Steve Ray, Jack Hodges, Jack Spivak, and Paul J. Keller. QUDT—Quantities, Units, Dimensions and Data Types Ontologies, 2016. http://www.qudt.org/.

[20] Quantities, Units, Dimensions, Values (QUDV), 2009. http://goo.gl/bIXclX.

[21] H. Rijgersberg, M. Wigham, and J. L. Top. How semantics can improve engineering processes: A case of units of measure and quantities. *Advanced Engineering Informatics*, 25:276–287, 2011. https://doi.org/10.1016/j.aei.2010.07.008.

[22] BIPM. *The International System of Units (SI)*, 8th edition, 2006. http://www.bipm.org/en/publications/si-brochure/.

[23] Marcus P. Foster. Quantities, units and computing. *Computer Standards & Interfaces*, 35:529–535, 2013. https://doi.org/10.1016/j.csi.2013.02.001.

[24] Joint Committee for Guides in Metrology. *Evaluation of measurement data—Guide to the expression of uncertainty in measurement*. JCGM 100:2008, http://www.bipm.org/utils/common/documents/jcgm/JCGM_100_2008_E.pdf.

[25] Joint Committee for Guides in Metrology. *Evaluation of measurement data—Supplement 1 to the "Guide to the expression of uncertainty in measurement"—Propagation of distributions using a Monte Carlo method*. JCGM 101:2008, http://www.bipm.org/utils/common/documents/jcgm/JCGM_101_2008_E.pdf.

[26] Lloyd N. Trefethen. Computing numerically with functions instead of numbers. *Communications of the ACM*, 58(10):91–97, October 2015. https://doi.org/10.1145/2814847.

[27] James Bornholt, Todd Mytkowicz, and Kathryn S. McKinley. Uncertain⟨T⟩: A first-order type for uncertain data. In *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '14)*, pages 51–66, 2014. https://doi.org/10.1145/2541940.2541958.

[28] NIST Uncertainty Machine, 2016. http://uncertainty.nist.gov/.

[29] Wikipedia. List of uncertainty propagation software, 2016. https://en.wikipedia.org/wiki/List_of_uncertainty_propagation_software.

[30] M. P. Krystek. The term 'dimension' in the international system of units. *Metrologia*, 52(2):297–300, 2015. https://doi.org/10.1088/0026-1394/52/2/297.

[31] I. M. Mills. Unity as a unit. *Metrologia*, 31(6):537–541, 1995. https://doi.org/10.1088/0026-1394/31/6/013.

[32] Ambler Thompson and Barry N. Taylor. Guide for the use of the International System of Units (SI). NIST Special Publication 811, National Institute of Standards and Technology, 100 Bureau Drive, Gaithersburg, MD 20899, March 2008. http://www.nist.gov/pml/pubs/sp811/.

[33] Peter J. Mohr and William D. Phillips. Dimensionless units in the SI. *Metrologia*, 52(1):40–47, 2015. https://doi.org/10.1088/0026-1394/52/1/40.

[34] Wikipedia. Principle of least astonishment, 2016. https://en.wikipedia.org/wiki/Principle_of_least_astonishment.

[35] Josef Kogan. An alternative path to a new SI, Part 1: On quantities with dimension one. https://web.archive.org/web/20160912224601/http://metrologybytes.net/PapersUnpub/Kogan_2014.pdf, September 2014.

[36] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[37] Glenn Shafer. *A mathematical theory of evidence*. Princeton University Press, 1976.

[38] Philippe Smets and Robert Kennes. The transferable belief model. *Artificial Intelligence*, 66(2):191–234, April 1994.

[39] Philippe Smets. The combination of evidence in the transferable belief model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):447–458, May 1990.

[40] Simona Salicone. *Measurement Uncertainty: An Approach via the Mathematical Theory of Evidence*. Springer, 2007.

[41] Wikipedia. Principle of maximum entropy, 2015. https://en.wikipedia.org/wiki/Principle_of_maximum_entropy.

[42] Wikipedia. Principle of indifference (application to continuous variables), 2015. https://goo.gl/erXy5Y.

[43] Scott Ferson, Janos Hajagos, David S. Myers, and W. Troy Tucker. CONSTRUCTOR: Synthesizing information about uncertain variables. Report SAND2005-3769, Sandia National Laboratories, 2005. http://prod.sandia.gov/techlib/access-control.cgi/2005/053769.pdf.

[44] ISO/IEC. *Quantities and units*. ISO/IEC 80000, http://www.iso.org/.

[45] International Union of Pure and Applied Chemistry. Properties and units in the clinical laboratory sciences II. Kinds-of-property. *Pure and Applied Chemistry*, 69(5):1015–1042, 1997.