A Compact Representation of Units

Bruce Hamilton
Measurement Systems Department
Hewlett-Packard Laboratories

Units are useful in understanding the meaning of data about physical quantities. For example, “6 liters” gives more information than “6.” Other properties, such as accuracy and precision, also give useful information, but we do not discuss those here. Most units used in the physical sciences are well-standardized [1,2,3,22], and there are standards for the use and spelling of unit names [2,4], but there are few standards for the representation of units within computing systems.

We present a representation which can accommodate unforeseen units, requires minimal agreement among the communicating parties, reveals the physical relationship among quantities, and requires only a few bytes, no matter how complex the unit.

1 Introduction

1.1 What Units Are

The physical sciences quantify a variety of properties of the things they study—their size, electrical resistance, thermal resistance, index of refraction, etc. Each distinct property represents what is commonly called a *dimension* [26].Within a given dimension, quantities are measured by comparing them to a quantity whose magnitude is defined to be unity. Such a quantity is called a *unit*. A sample with twice the mass of a unit sample is said to have a mass of 2. When more than one unit quantity is known, we state both the magnitude and the unit, as in “2 kilograms.” If there is no ambiguity, the name of the unit can be omitted: the population of Palo Alto is 56 000.

Choices of unit, and even of dimension, are matters of convention. Early scientists did not consider electrical properties to have the same status as length, mass and time. However, the SI system (see Section 3.1) provides a sufficiently stable agreement for this

work. This standard gives the preferred unit for each quantity (e.g. the second, rather than the fortnight, for time), and where necessary, the unit's definition.

Roughly, the calculus of units is as follows: let D_1 and D_2 be two dimensions, not necessarily distinct. Let U_1 be a unit associated with D_1 and U_2 a unit associated with D_2. Let P_1 be an expression representing a physical quantity expressed in units of U_1 (e.g. "6 feet"); similarly for P_2 and U_2.

- $P_1 \pm P_2$ is legal only if $U_1=U_2$. U_1 may be converted to U_2 (usually through multiplication by a constant) only if $D_1=D_2$. The result has dimension $D_1=D_2$ and unit $U_1=U_2$.
- $P_1 < P_2$, $P_1 = P_2$, $P_1 > P_2$, and $P_1 \leftarrow P_2$ (assignment) follow the same rules as addition and subtraction, except that the result is not a dimension, and has no unit.
- $P_1 \cdot P_2$ and $P_1 \div P_2$ are always legal. The result has dimension $D_1 \cdot D_2$ or $D_1 \div D_2$, respectively, and unit $U_1 \cdot U_2$ or $U_1 \div U_2$, respectively.

The exact rules are surprisingly subtle, and are explored in [24].

1.2 Interoperation

We are concerned with communication among computing systems. Within a single system it is feasible to legislate a single idiosyncratic unit, or for programmers to remember which values are in which units. This is infeasible between independently-developed systems. If systems are to interoperate, they must indicate a quantity's unit explicitly, in the same way that they indicate its magnitude.

1.3 Goals

We seek a representation which:

- Requires minimal agreement among the communicating parties. A 5-line header file or smaller is in the right range. One or two citations of standards would do. An enumeration of several hundred unit names is not acceptable.
- Is more compact than the ASCII name of most units. It should require about the same number of bytes as "meter," even for complex units such as "kilogram-meter^2/ampere-second."
- Settles (or better, obviates) questions like the spelling of "metre," the use of singular or plural, the symbols used to represent multiplication, division and exponentiation, and the ordering of terms (e.g. "meter-seconds" vs. "second-meters").
- Accommodates unforeseen units. These occur for several reasons:
  - A program developed in a domain that never deals with, say, molarity, might be placed into service in a chemistry lab.
  - Calibration programs often want to deal with the partial derivative of various quantities with respect to time, temperature, voltage, etc.
  - General-purpose plotting packages may be asked to label the slope of a line, or the area under a curve.
- Reveals the physical relationship among quantities. If a strip chart is asked to plot a quantity Q vs. time, it should be able to label the area under the curve, even if it hasn't been told explicitly about Q. For example, the representation should facilitate discovering that the time integral of meters/second is meters, and the time derivative is meters/second^2.

Of course, it is impossible to construct a compact representation which can accommodate *any* unit, since there are infinitely many units. We set a practical bound which we think is generous. This is discussed in Section 4.1.

1.4 Acknowledgments

Thanks to William Kent, Stephanie Janowski and Dan Hepner for valuable discussions.

2 Scope

2.1 Physical quantities

We consider only quantities of interest to the physical sciences. Quantities such as money, employee performance, IQ, algorithmic complexity, etc. present difficulties outside the scope of this paper [9,12]. Further, there is an unbounded number of such quantities. We limit ourselves to the SI base dimensions (see Section 3.1) and their multiplicative combinations.

2.2 Unit-Based

We consider only schemes which represent quantities as multiples of some reference quantity. We call such schemes *unit-based*. We exclude rankings and binning schemes, such as hardness ratings, high/med/low ratings, wire gauges, figure skating scores, etc. The touchstone for whether a measurement scheme is unit-based is the following:

- Choose a dimension, D, and consider two samples, S_1 and S_2 of D—two masses, two refractive indices, two speeds, etc.
- Their magnitudes stand in some ratio R. That is, S_1 is R times as massive, refractive, fast, etc. as S_2. In order to determine R it is necessary to tell whether two quantities are equal, but it is not necessary to "measure" anything.[1]
- Let M_D be a quantification scheme which measures D. That is, if X is a sample of dimension D, then $M_D(X)$ is a number which represents the amount of D in X.
- M_D is unit-based if and only if $M_D(S_1)/M_D(S_2) = R$, for all samples S_1 and S_2, and their ratio R.

Bridgman [10] calls this the "Absolute Significance of Relative Magnitude." For an example, consider two lengths of steel, one 14 cm long and one 3 cm long. The ratio of their lengths is 4.67. If we measure those lengths in meters, we get 0.14/0.03 = 4.67. If we measure them in inches we get 5.5/1.2 = 4.67. Similarly if we measure in paces, furlongs, parsecs, etc.

This principle excludes what are sometimes called "numerical-value equations." These are equations which are true only for particular units. One example is:

h = 0.7 * (220 - A)

1. If an object moving at constant speed reaches a destination and returns in the time it takes another to reach the same destination, then the first is moving at twice the speed of the second. We can say this without knowing the speed of either object, nor the distance to the destination nor the travel time. If we cannot tell, even in principle, whether two samples of a dimension D are equal, then D is outside the scope of this paper.

where h is a person's heart rate in beats/minute and A is the person's age in years. It can be shown [10] that the principle of Absolute Significance of Relative Magnitude forbids adding two quantities with different dimensions, or testing them for equality. Thus the equation above, and expressions like "h + 0.7*A" are illegal, since age ≠ frequency. Notice that there is no unit for "h + 0.7*A." Expressions and equations of this sort are outside our scope.

Some things which are commonly mapped to unit-based measurements are not themselves unit-based. Dates, for example, do not obey the calculus of physical quantities. There is no such thing as tomorrow squared, or yesterday+today. The elapsed time between two instants is a unit-based measurement, as is the duration of any particular day, but an instant itself is not the measure of anything.

2.3 Units Only

There are many other useful facts about the representation of a physical quantity: its precision, the time and location at which it is (or will be) true, an indication of which quantity it represents (e.g. width vs. thickness), etc. Each of these would make it easier for computing systems to cooperate with each other. We concern ourselves here only with units.

2.4 Scalars

Various collections of physical quantities occur in practice—vectors, tensors, waveforms, etc. This work applies to the scalar components of any such collection, but is not concerned with collections *per se*.

2.5 Communication Among Computing Systems

We deal with communication among computing systems. This work is useful for communication within a computing system, but there are other solutions to that problem.

We do not consider the issue of communicating with people. People require a system to deal with many units (such as degrees Fahrenheit) which would not otherwise be required. Converting to/from these units is trivial, but knowing what unit was presented on input, and what unit should be presented on output, can be tricky. Similarly, people sometimes use idiosyncratic notations, such as "6 feet, 4 inches." These problems are largely independent of the ones we consider in this paper.

3 Previous Work

3.1 The SI system

The SI[1] system of units [22] defines a set of base units, in terms of which other units may be derived. These are:

TABLE 1. The SI Base Units

Quantity	Unit	Symbol
Length	meter	m
Mass	kilogram	kg
Time	second	s
Electric current	ampere	A
Thermodynamic temperature	kelvin	K
Amount of substance	mole	mol
Luminous intensity	candela	cd

The “Quantity” field describes what we call dimensions. These quantities are “base” only by convention. One could say instead that charge is basic and current is derived, for example, but this standard is well established by now. Other dimensions, such as speed, are considered “derived,” in the sense that they are multiplicative combinations of the base quantities.

The standard specifies not only which quantities are to be considered the basis set, but also the reference unit for each quantity. Thus, the kilogram, rather than the gram, stone, etc., is the reference unit for mass. The standard also specifies reference units for derived quantities such as force, surface tension, etc. These units are derived via the same algebraic expression as the quantities themselves. That is, the unit for the quotient of quantities A and B is the quotient of the units for A and B. A few commonly-used units, such as volt, farad, and hertz, are given simple names.[2]

The standard permits alternative units in specific domains. The hour and day may be used, for example, in the context of “life customs and calendar cycles.” But these are secondary choices. For every quantity, the standard clearly specifies a single, preferred unit.

By narrowing the choice of unit, this standard goes a long way toward simplifying communication among independent systems. The level of agreement among the communicating parties is limited to the table above, the rule for constructing derived units, and (optionally) the list of special names.

Note, though, that the representation is oriented towards humans. It specifies units in terms of printable characters like “kg,” “Hz,” or “J/(mol·K).” Even if we convert this to ASCII (and solve the typographic difficulties), there are still two problems. First, the

1. Le Système International d’Unités.
2. The standard calls them “special” names.

representation can get long if there are many multiplicands. Second, it is not single-valued. “J/(mol·K),” “J·mol^{1}K^{-1},” and “J/(K·mol)” are all legal. We can use special names to shorten the representation, but they introduce further ambiguity: kg·m/sec^{3} can be either N/s (newton/second) or W/m (watt/meter).

The standard also specifies the radian and steradian as the units for plane and solid angle, respectively. These units are “supplementary,” or optional. We discuss these in Section 5.3.

3.2 Enumerations

A representation better suited to computer use is to enumerate a few thousand of the SI units, and transmit only an index into the enumeration. Such a representation is compact and unambiguous; it is used in some proprietary schemes. However, it is not yet what we seek.

- It does not accommodate unforeseen units.
- It does not reveal the units’ structure. The representation of meters/second, for example, is unrelated to the representations of meter and second.
- The list is large, and every communicating entity needs a copy.

3.3 Standard Variable Types

Some bodies developing communication standards for industrial use [5,14,15] have proposed industry-specific variable type or object definitions for commonly-used physical quantities, such as pressure, temperature, concentration, etc. These standards often specify a resolution as well as a unit, so that one type is pressure in psi, another is pressure in 0.1 psi, etc.

Processes declare variables to be of one of these types, and communicate with each other by sending variables of these types. The rules forbid storing a quantity of one type into a variable of a different type. Rather than misinterpreting the data, therefore, these systems can generate an error at run time.

This is a form of enumeration, but it deserves separate mention. Its benefit is that it is very compact. Since the resolution of a quantity is known, the magnitude can be stored as an integer, often in half the space of a float. If the variable type is used only at compile or link time, the space required for the enumeration index can be saved as well. However, the scheme has all the disadvantages of an enumeration, and more:

- It allows more than one unit for a given quantity (pressure in both pascals and psi, temperature in both °C and °F, etc.). It is thus a matter of chance whether two systems developed independently can communicate.
- Entities choosing different resolutions also cannot communicate.
- The more choices of unit and resolution available, the lower the probability that two systems will be able to communicate. Yet each standards body has a process for adding units and resolutions.

3.4 Proprietary Representations

Proprietary schemes have been developed, for use within a closed set of users. These are usually domain-specific (RF electronics, pharmaceutical, etc.). All the ones we have seen are combinations of the schemes previously discussed.

3.5 Programming Languages

The ATLAS language [21] was ahead of its time in support for units, but the set of units and the number of operations on them are small and not extensible.

As new programming languages are introduced, often with better support for user-defined types than their predecessors, there is often a movement to augment these languages with support for units. There were such movements after the introduction of Pascal [6,8,13,15,20,25], Ada [17,18,19,25], C++ [7,11], and ML [23,29].

If integrated into a language, the representation of units becomes part of the storage type, and inherits the language's existing support. Further, language integration has the opportunity to support the calculus of units, mentioned in Section 1.1.

None of these proposals has borne fruit. Languages typically allow collections of existing storage types, but not modifications of those types. Further, authors seem unaware of the subtleties of the units calculus. One scheme, for example, allows a user to state a constant to be used in converting from degrees Celsius to kilograms.

Novak [27] develops a representation for GLISP [28] which, like ours, is based on a product of powers. His work is concerned with conversion, and thus does not deal with the issues we raise in Section 5. It treats fractional powers, but as a special case; it accommodates money, but does not deal with the attendant time-varying unit conversions. Novak's work does deal with presentation to humans (which we do not), and yields a particularly clean way to simplify expressions.

4 A Compact Representation

4.1 The Power Product

A theorem due to Bridgman [10] shows that a sufficient representation for a unit-based quantity is a product of a set of basis units, each raised to some power. We can thus do the following:

- Choose the SI units as the basis set.
- Choose an ordering of them, such as the one in Section 3.1. This choice is arbitrary, but all must agree on it.[1]
- Omit the units themselves, and store just their powers, in the chosen order, as a vector. The quantity is then the product of the SI units, each to the given power. For example, the joule ($m^2 \cdot kg/s^2$) is [2,1,-2,0,0,0,0], the candela per square meter (cd/m^2) is [-2,0,0,0,0,0,1], and the volt ($(kg \cdot m^2/s^2) / (A \cdot s)$) is [1,2,-3,-1,0,0,0].

We will call this is the *canonical form* for a unit.

The most general representation should allow the powers to be rational numbers[2] of infinite size and either sign. A practical representation should allow them to be rationals with numerator in the range of a small signed integer, and denominator equal to 1 or 2.

1. This is almost the only agreement required among the cooperating parties. We will see an exception in Section 5.
2. Noise power is sometimes measured in $\frac{\text{watts}}{\sqrt{\text{hertz}}}$, for example.

For a compact representation, we can observe that the vast majority of actual exponents are integers in the range –4 to +4, inclusive.[1]

A vector of small numbers can be packed into 4-5 bytes:

- If we allocate four bits per exponent, we can represent integer powers in the range (-8 to +7, inclusive)[2] with 28 bits. For ease of equality testing, we might pad this to 32 bits.
- If we allocate five bits per exponent, we can represent integer multiples of $\frac{1}{2}$, in the range -8 to $+7\frac{1}{2}$, inclusive, with 40 bits (including 5 bits of padding).

Each of these representations is shorter than the ASCII name of most units, and is much more powerful. Since we know at least one practical unit involving a $\frac{1}{2}$ power, we propose the second representation. One possible bit layout is as follows:

TABLE 2. Sample Bit Layout

Bits	Meaning
0-4	Reserved
5-9	Exponent of base unit 1
10-14	Exponent of base unit 2
…	…
35-39	Exponent of base unit 7

4.2 Operations

Assume that U_1 and U_2 are two units. Here is how to determine the unit of the result of various operations on quantities represented this way. Remember that we do not have to convert between equivalent units (such as feet and inches) because our representation permits only a single unit for each dimension.

- Addition and Subtraction: legal only if $U_1=U_2$. The unit of the result is the same as U_1 and U_2.
- Multiplication: the representation of $U_1 \cdot U_2$ is the vector sum (using element-by-element addition) of the representations of U_1 and U_2. For example,
  meter/sec [1,0,-1,0,0,0,0] · sec [0,0,1,0,0,0,0] = meter [1,0,0,0,0,0,0].
- Division: the representation of U_1/U_2 is the vector difference of their representations.
- Equality: $U_1=U_2$ if and only if their representations are bitwise equal.

We mention these operations only to illustrate that they are efficient on most microprocessors. The actual operations should be hidden from programmers, in a library of operations on this data type.

1. Voltage slew rate involves time^{-4}, and capacitance involves time^{+4}.
2. We pick -8 to +7 rather than -7 to +8 arbitrarily. We don’t know of any quantities that require exponents of either -8 or +8.

4.3 Assigning and Extracting Units

Along with functions for +, -, ·, /, and =, the library must also provide a way to set and interrogate the unit of a quantity. Rather than enumerate all units, it is simpler to provide prototypes of the seven base units and let `set_unit` work by example. Suppose the prototypes are called METER, KG, etc., and the operators are called +, -, ·, /, and =. Then, to set the unit of a variable V to meter/sec^2, one would use something like `set_unit (V, METER/(SEC·SEC))`.

4.4 Performance

With this representation it is fast to test two units for equality, but slow to map between a unit and its string representation (though common mappings can be cached). We expect conversion to/from strings to be required less often than other uses, and only for operations that take place at human speed.

4.5 Portability

The sample bit layout given above is for concreteness only. This canonical form can be carried in a vector of seven small signed integers. Any modern communication protocol can represent such a construct portably.

5 Adequacy of the Representation

The canonical form just given accommodates the SI base units and all their multiplicative combinations, up to powers of $7\frac{1}{2}$. Is that enough? In this section we explore the expressiveness of the representation.

5.1 Quantities Not in Lowest Terms

Standard [22] does not say whether derived units are to be reduced to lowest terms. All of the examples in [22] are expressed in lowest terms, and common practice is to do that. For example, lengths are rarely expressed as meters^3/meter^2, or as meter-seconds/second. We want to offer only one way to express any given unit, and in any case, our canonical form offers no way to represent an unreduced expression, since each dimension gets only a single exponent. We therefore require derived units to be reduced to lowest terms.

We know of two types of useful quantity which this excludes. An example of the first is automobile fuel efficiency, commonly expressed in liters/kilometer or miles/gallon, equivalent to 1/area. All we can offer is the latter form. This is an area of further work.

The second type of quantity which is commonly not reduced to lowest terms is the dimensionless ratio. We discuss it next.

5.2 Dimensionless Quantities

Many so-called “dimensionless” quantities need representations in computing systems. These are quantities like strain, refractive index and concentration—ratios in which the numerator and denominator are the same dimension. Strain, for example, is length/

length. The calculus of dimensions states that the dimension of such a thing is null. Our canonical form says that it is [0,0,0,0,0,0,0]. We can certainly represent these things, but they have a few unusual properties, discussed in the following paragraphs.

5.2.1 Unit Name

Standard [1] specifies three choices of unit for dimensionless quantities, none suitable for our purposes.

1. It says that "terms such as percent, parts per thousand, and parts per million may … be used." These terms are in common use, and well-understood. However, we cannot permit them, since we restrict ourselves to a single unit per dimension. The unit for dimensionless quantities is (implicitly) "parts per part."
2. It says that the appropriate unit "… may be expressed by the number 1…" But the unit "1" is not intended to label values. Rather, it serves as a placeholder in expressions, such as:

$$\begin{aligned} \text{cal/mol} \cdot \text{mol/kg} \cdot \text{kg/m}^3 &= \text{cal} \cdot \text{mol/mol} \cdot \text{kg/kg} \cdot 1/\text{m}^3 \\ &= \text{cal} \cdot 1 \cdot 1 \cdot 1/\text{m}^3 \\ &= \text{cal/m}^3 \end{aligned}$$

3. It says that dimensionless quantities "are expressed by pure numbers" i.e. with no unit. Thus a refractive index of 3.0 should be written "3.0." But failing to provide a unit is not the way to label a quantity which has no unit. The right way is to *label it as having no unit*—that is, to give it the unit [0,0,0,0,0,0,0].

5.2.2 Unambiguous Value

Note that there is no ambiguity, either for humans or computing systems, in the slope of a hill expressed as "0.67." Although it is logically possible for that value to be feet/mile, inches/foot or meters/kilometer, etc., our canonical form forbids all of those; it requires that a ratio of lengths be expressed in meters/meter. A unitless number, known to be a ratio of lengths, is thus known to be a ratio of lengths in meters. The same is true for all other units, both basis and derived.

5.2.3 Implied Dimension

From Section 5.2.2, we can see that in a so-called unitless ratio U/U, it is still important to know what U is. But the canonical form maps all such ratios to the same unit—[0,0,0,0,0,0,0]—and thus discards important information. We must therefore use a different representation for unitless ratios. We need a boolean variable, to indicate that the intended unit is not the one given, but rather the quotient of the given unit and itself.

Our canonical form loses the information mentioned above because the calculus of units loses it. This is an instance of a more general problem with the calculus of units, which we touch briefly in Section 6.3. It is discussed fully in [24].

5.2.4 Nameless Dimension

Besides dimensionless ratios, there are also quantities which are dimensionless because the thing they quantify isn't a recognized dimension. Most of these are counts or amounts of some sort, which are sometimes combined with time to form rates or flows. Examples are:

- The output of a factory. The units might be cars, cookies, etc.

- The population of a city, committee, team, etc. The units are people.
- The population of a forest might be in trees, or mammals, or animals.
- Heart rate is beats per minute.
- The reading of a general-purpose counter. If it is in use and counting something, then its reading may be said to be that thing, but if we buy a counter, open the box and find that it says 721824, the units of that are just "counter units."

No general-purpose scheme can ever account for all of these. The SI standard leaves them nameless, and we follow its lead. It is for units of this sort that we reserve the null unit, [0,0,0,0,0,0,0], without the "dimensionless ratio" flag. We make no recommendation about how to present this unit on output or infer it on input. One obvious possibility is to leave it nameless by default.

5.3 Angles

Angles are problematical in several ways. For one, there are several different kinds of angle, distinguished by what it means for two of them to be equal. Thus, angles are sometimes compared modulo π, sometimes modulo 2π, and sometimes modulo ∞. This is discussed further in [24].

Angle is sometimes used to represent the notion of repeating events. In these cases a better measure is "turns" or "rotations." These are ordinary dimensionless quantities, as in Section 5.2.4, and their frequency of occurrence is represented in Hz.

The remaining sense of angle is a sort of "angular aperture." This is the case when we are concerned with the angular flux of some quantity (the fraction that is visible or effective through a given angle), or with the components of a vector. One question is whether this is a proper dimension (angle) or a dimensionless ratio (arc-length/radius).

Standards bodies themselves [1,22] have had trouble with this question. In 1960, plane and solid angles were given their own units (radian and steradian, respectively). These units were classified as "supplementary," and it was not stated whether angle was a base dimension or not. In 1980, the 1960 decision was interpreted to mean that angles are dimensionless, and that users have the option to use the supplementary units (in addition to other units for dimensionless quantities, such as percent, ppm, etc.).

We can not allow a choice of unit, and therefore cannot adopt the standard's interpretation. Here are the various considerations:

1. Angles and repetitions are separate concepts. We must allow electronic engineers to measure frequencies in cycles/sec. On the other hand, if torque·angle = energy, and torque is in newton-meters and energy is in joules, angle must be in radians. Similarly $\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots$ only if x is stated in radians. We thus need separate representations for angle and count.
2. We must be able to represent products and quotients of angles with regular dimensions, such as angular velocity and momentum. These should follow the calculus of units.
3. If its angular property is the way that we distinguish torque (newton-meters = $kg \cdot m^2/sec^2$) from energy (joules = $kg \cdot m^2/sec^2$), then torque must be energy/angle, so that torque·angle = energy. This means that we must be able to represent $angle^{-1}$.

4. We know of no need for angles with powers other than -1, 0 and 1.
5. We need to represent both plane and spherical angles.

These considerations lead us to make plane and spherical angles separate dimensions,[1] with units of radian and steradian, respectively. This departs from the current interpretation of the standard, but we see no choice. Note that, although we can now distinguish torque from energy, the units of torque are surprising: newton-meters/radian. Fortunately, this representation is for internal use only; applications are still free to present torque to humans in newton-meters, foot-pounds, etc.

5.4 Logarithmic Units

Some units, such as dB, are the logarithm of some quantity (usually a dimensionless ratio). Since the logarithm is isomorphic to the underlying quantity, we could require that quantities be linearized for representation, and not burden ourselves with another unit, nor with keeping track of whether a given quantity is linear or logarithmic.

Logarithms are useful in two ways. First, the functional form captures people's perception of the behavior of many quantities. Second, some quantities have a dynamic range too wide to fit in a single-precision floating-point number.

The first benefit is a presentation issue, and we can do without it in an internal representation. The second is a serious practical problem. We must therefore provide a way to represent logarithmic units. To represent log(U), it suffices to send U plus a boolean to indicate that the associated magnitude is the logarithm of that unit.

In practice, logarithmic units are computed in several different ways, such as $\log_{10}(x)$, $20 \log_{10}(x)$, $\log_e(x)$, etc. This presents a problem exactly parallel to the problem of multiple linear units separated by constant factors (such as inches, feet and yards). The solution is the same: we permit only unscaled natural logarithms in our representation. Applications are free to convert to/from other representations on input and output.

Some units used in practice represent $\log(U/U_0)$, where U_0 is a reference value. We make no attempt to represent those cases, nor what the reference value is. Computing systems must provide additional context if they wish to communicate that.

5.5 pH

pH is a measure of acidity, defined as minus the natural logarithm of the concentration, in moles/liter, of hydrogen ions in a solution. Our proposal already has a way to represent such concentrations. It specifies that we measure in moles/m^3, take the natural logarithm, and not invert the sign. Acidities will thus appear internally as negative numbers with small absolute values. This is, again, a presentation issue—applications can convert to and from pH on output and input. Let P be acidity in pH, and C be acidity in our canonical form. Then $P = -(C + 13.8)$.

pH has an interesting quirk. Moles/liter (or even moles/m^3) is not a dimensionless ratio. Taking the logarithm of such a quantity is like subtracting age from heart rate—it violates the calculus of units.[2] Strictly speaking, this means that pH is outside our scope. Less pedantically, it means that if we are willing to represent such a logarithm, then our

1. Though we can allocate fewer bits for them.

way of indicating that a quantity is logarithmic must not assume that it is a dimensionless ratio.

Thus we recommend that acidity be represented as $\log_e(\text{mol/m}^3)$ rather than as pH, and we note that other concentrations may be represented in that form as well.[1] In addition, we will ensure that the "log" and "dimensionless" flags in our representation are independent of each other.

5.6 Representation, revisited

With the additions mentioned above to accommodate corner cases, let us be sure the physical layout is still feasible and compact.

5.6.1 Fields Required

In addition to powers of the base dimensions, we must also capture the following:

- A boolean to indicate whether the intended unit is the given one, or the quotient of that unit and itself.
- A boolean to indicate whether the intended unit is the given one, or the log of that unit.

There are four ways we can add information:

- Add a dimension
- Use some of the bits we've been calling "reserved."
- Add a new field or fields.
- Reserve values, on the assumption that quantities like $\text{length}^{7.5} \cdot \text{mass}^{7.5} \cdot \text{time}^{7.5} \cdot \text{current}^{7.5} \cdot \text{temperature}^{7.5} \cdot \text{molarity}^{7.5} \cdot \text{intensity}^{7.5}$ rarely occur in practice.

2. To see this, represent the logarithm by a power series. Each term is raised to a different power. This is dimensionally consistent only if every term is itself dimensionless.

1. Though they are unlikely to be.

5.6.2 Representation

The choice of which distinctions to draw, and which methods to use to draw them, is a matter of taste. Here is one recommendation:

TABLE 3. Methods for representing dimensionless quantities

	Add a dimension	Reserve a bit	Add a field	Reserve a value
NULL unit				Reserve 1 value
Dimension-less ratio		Reserve 1 bit		
Log		Reserve 1 bit		
Angle	Add plane and spherical angle			
pH	<do not treat specially>			

(We note in passing that the unit we recommend for pH is so unusual that for practical purposes we are reserving a value.)

Again, we plan to hide the details of the representation from programmers, so the choice does not need to be simple or pretty. We don't even need to choose right now, as long as we are satisfied that there is room to put the information somewhere.

At the application level, programmers will need a way to set and interrogate the "type" of a dimensionless quantity. We recommend a pair of functions (called, e.g. "set_dimensionless_type" and "get_dimensionless_type") which do that.

One possible representation is then:

TABLE 4. Normal Unit Format

Field #	Meaning	# bits
1	Reserved	7
2	0: Unit is as described below. 1: Unit is U/U, where U is as described below. 2: Unit is $\log_e$(U), where U is as described below. 3: Unit is $\log_e$(U/U), where U is as described below.	2
3	<exponent of radians> + 1	2
4	<exponent of steradians> + 1	2
5	(2 × <exponent of meters>) + 16	5
6	(2 × <exponent of kilograms>) + 16	5
7	(2 × <exponent of seconds>) + 16	5
8	(2 × <exponent of amperes>) + 16	5
9	(2 × <exponent of kelvins>) + 16	5
10	(2 × <exponent of moles>) + 16	5
11	(2 × <exponent of candelas>) + 16	5

Again, although we present a bit layout here, a portable representation can regard this as an enumeration plus a vector of small, unsigned integers.

5.6.3 Operations, revisited

Operations on quantities in this form are as in Section 4.2, with the following additions:

1. If a unit is logarithmic, multiplication and division are not allowed.
2. Let R be any dimensionless ratio, let U, U_1, and U_2 be units which are neither logarithmic nor dimensionless, and let R_1 be U_1/U_1 and R_2 be U_2/U_2, where $U_1 \neq U_2$. Then:
   - $R \cdot U = U$
   - $U / R = U$
   - $R / U = U^{-1}$
   - $R_1 \cdot R_2 = (U_1 \cdot U_2) / (U_1 \cdot U_2)$.
   - R / R = [0,0,0,0,0,0,0,0,0]. The form cannot represent dimensionless ratios more than one level deep.
   - $R_1 \neq R_2$. If a developer wants two different dimensionless quantities to have “the same” unit, our recommendation is to override the “=” function.

6 Future Work

6.1 Quantities Not in Lowest Terms

It is not yet clear what other members belong in the family with miles/gallon, nor the best way to treat such things.

6.2 Ratios of Dimensionless Quantities

Our representation uses a unit U, together with the "dimensionless" flag, to indicate the unit U/U. This is adequate for many purposes (and is more powerful than any other representation we know), but it cannot represent $(U_1/U_1)\div(U_2/U_2)$, or deeper quotients. It also cannot represent a quotient of logarithms. There are infinitely many such quotients, even for a finite set of units. We know of no compact way to represent these.

6.3 Specialized Dimensions

We noted in Section 5.2 that dimensionless ratios are a case in which two quantities have the same canonical form even though people regard them as distinct kinds of thing. This can also happen with ordinary dimensions. In some contexts, for example, people distinguish among various kinds of length, such as width and height (width·height=area, but $\text{height}^2 \neq \text{area}$) or radius/axis/circumference. Torque and energy are both $\text{length}^2\cdot\text{mass}/\text{time}^2$, but it is useful to distinguish between them. Some transducers measure length, while others measure altitude; their vendors would not say that they measure the same thing. We are still working on a theory for this [24].

One difficulty is that it is application-dependent whether a given distinction is useful. It seems nonsense, for example, to assert that the slope of my driveway plus the concentration of sugar in my coffee equals the amount by which my watch is slow. In an instrument, however, it makes sense to apportion the error budget among the mechanical, electrical and optical subassemblies, and to require that their sum be less than 0.1%.

Another difficulty lies in multiplicative combinations of these things: if width and height are both distinct from length, what is width·length? What is $\text{length}^2/\text{width}$?

Until the theory is developed, our only tools are the same as for dimensionless quantities: tags of the sort described in 5.6.1, and a consistent interpretation of them. Since a dimension may in general be partitioned into more than two distinct categories (as in length, above), a single-bit flag will not be sufficient. A separate field will probably be necessary.

6.4 Collections

Structured collections of physical quantities, such as vectors, time series and spectra, need special treatment. This work is adequate for representing the scalar components of a collection, but additional issues arise in determining whether it is legal to form the product of a vector and a scalar, the inner and outer product of two vectors, the difference of two time series, etc., and the unit of the result. These are the subject of further work.

6.5 Expansion

Eventually, users will want to add new dimensions. New combinations of existing dimensions, such as temperature·time/length, are automatically accommodated. We mean truly new dimensions, such as money[1] or "comfort," or "interestingness."[2] It's easy to add a space in the vector of dimensions; the hard part is assuring interoperability. How do we standardize the meaning of, say, the 12th position? What are its standard units? To keep interoperability, some sort of standardization is needed.

Even with standardization, a simple vector representation doesn't lend itself to the sort of expansion we describe, since every new dimension increases the vector length, and all users bear the burden. An escape, pointing to a more flexible representation, may be needed. Here is an example of how expansion might be managed:

- One of the "reserved" bits indicates whether this is a normal or extended representation.
- If it is extended, then eight of the "dimension" bits are used to indicate an application domain. Domains might be such things as Transportation, HVAC, Medical, etc. Some central organization, such as NIST, allocates domain IDs.
- Space is then left for 6 basis dimensions with 5-bit exponents (fractions between -8 and $+7\frac{1}{2}$), or possibly 7 basis dimensions with 4-bit exponents. The assignment of base dimensions (and possibly the width of exponents) is standardized within the application domain, by whatever committee is appropriate.

Alternatively, we could decide that packets using the extended format are larger (64 bits, for example). This would allow them to implement a superset of the standard units. By the time they are needed, 64-bit quantities may fit in the registers of a low-cost microprocessor.

1. Money raises an additional issue with unit conversion: currencies are the units of money and their values are well-standardized, but they vary with time.
2. Never mind how users plan to measure (or actuate!) these.

It is premature to decide on bit layouts, but just to be careful not to spend bits we don't have, here is one possible assignment, keeping the packet size unchanged:

TABLE 5. Normal Unit Format

Field #	Meaning	# bits
1	Version number (allows format to be changed later)	6
2	=0 (Normal Format)	1
3	0: Unit is as described below. 1: Unit is U/U, where U is as described below. 2: Unit is $\log_e$(U), where U is as described below. 3: Unit is $\log_e$(U/U), where U is as described below.	2
4	<exponent of radians> + 1	2
5	<exponent of steradians> + 1	2
6	(2 × <exponent of meters>) + 16	5
7	(2 × <exponent of kilograms>) + 16	5
8	(2 × <exponent of seconds>) + 16	5
9	(2 × <exponent of amperes>) + 16	5
10	(2 × <exponent of kelvins>) + 16	5
11	(2 × <exponent of moles>) + 16	5
12	(2 × <exponent of candelas>) + 16	5

TABLE 6. Sample Extended Format

Field #	Meaning	# bits
1	Version number	6
2	=1 (Extended Format)	1
3	Application Domain ID	8
4	Exponent of base unit 1	4
5	Exponent of base unit 2	4
…	…	
12	Exponent of base unit 9	4

7 Conclusions

We have presented a compact representation of units which can accommodate unforeseen units, requires minimal agreement among the communicating parties, reveals the physical relationship among quantities, and can deal with dimensionless and logarithmic units. We have shown how to determine the legality of operations on two quantities with arbitrary units, and how to determine the unit of the result.

We have compared this representation with other forms, discussed its advantages, described the contexts in which the representation is applicable, and shown its limitations.

8 References

1. American Society for Testing and Materials (ASTM), "Standard Practice for Use of the International System of Units," E380-92, PCN 03-543-092-34, 1992.
2. ANSI/IEEE Std. 280-1985, "IEEE Standard Letter Symbols for Quantities Used in Electrical Science and Electrical Engineering.
3. ANSI/IEEE Std. 945-1984, "IEEE Recommended Practice for Preferred Metric Units for Use in Electrical and Electronics Science and Technology."
4. ANSI/IEEE Std. 260-1978, "IEEE Standard Letter Symbols for Units of Measurement."
5. American Society of Heating, Refrigerating and Air-Conditioning Engineers, Inc., "BACnet™—A Data Communication Protocol for Building Automation and Control Networks," Atlanta, GA, March 1995.
6. Baldwin, Geoff, "Implementation of Physical Units," SIGPLAN Notices Vol. 22, No. 8, August 1987, pp.45-50.
7. Barton, J., and Nackman, L., "Dimensional Analysis," C++ Report, January 1995.
8. Biedl, Albrecht, "An Extension of Programming Languages for Clerical Computation in Science and Engineering with Special Reference to Pascal," SIGPLAN Notices, April 1977.
9. Boring, Edwin G., "The Beginning and Growth of Measurement in Psychology," in *Quantification, A History of the Meaning of Measurement in the Natural and Social Sciences*, Harry Wolf, Ed., Bobbs-Merrill, 1961.
10. Bridgman, P.W. "Dimensional Analysis," Yale University Press, 1922, revised 1963.
11. Cmelik, Robert F. and Narain H. Gehani, "Dimensional Analysis with C++," IEEE Software, 5/88, pp. 21-27.
12. Coombs, Clyde H., "A Theory of Data," Psychological Review, Vol 67, No. 3, pp. 143-159, May 1960.
13. Dreiheller, A., Moerschbacher, M. & Mohr, B., "Programming Pascal with Physical Units," SIGPLAN Notices, Vol. 21, No. 12, pp.114-123, December 1986.
14. Echelon Corporation, "The SNVT Master List and Programmer's Guide," March, 1992.
15. EIA IS-60 CEBus Specification, October 1992.

16. Gehani, Narain H., “Units of Measure as a Data Attribute,” Computer Languages, Vol. 2, pp.93-111, 1977.
17. Gehani, Narain H., “Ada’s Derived Types and Units of Measure,” Software—Practice and Experience, Vol. 15 No. 6 pp. 555-569, June 1985.
18. Grosberg, J., “Object-Oriented Dimensional Units,” Dr. Dobb’s Journal, September 1988.
19. Hilfinger, Paul N., “An Ada Package for Dimensional Analysis,” ACM Transactions on Programming Languages and Systems, Vol. 10, No.2, 4/88, pp. 198-203.
20. House, R.T., “A Proposal for an Extended Form of Type Checking of Expressions,” The Computer Journal, Vol. 26, No. 4, 1983.
21. IEEE Standard C/ATLAS, IEEE Standard 716-1982.
22. ISO 31-0:1992(E), “General Introduction to ISO 31—General Principles Concerning Quantities, Units and Symbols,” International Organization for Standardization, Geneva, Switzerland, 1974.
23. Kennedy, A., “Dimension Types,” European Symposium on Programming ‘94, LNCS Volume 788.
24. Kent, W., Janowski, S., Hamilton, B., and Hepner, D., “Dimensioned Data,” in progress.
25. Männer, R., “Strong Typing and Physical Units,” SIGPLAN Notices, Vol. 21, No. 2, pp.11-20, March 1986.
26. Maxwell, J. Clerk, “On the Mathematical Classification of Physical Quantities,” Proceedings of the London Mathematical Society, March 1871, pp 224 et seq.
27. Novak, G. S., Jr., “Conversion of Units of Measurement,” IEEE Transactions on Software Engineering, Vol. 21, no. 8, pp 651-661, August 1995.
28. Novak, G. S., Jr., “GLISP: A LISP-based programming system with data abstraction,” AI Magazine, Vol. 4, no. 3, pp 37-47, Fall 1991.
29. Wand, Mitchell & O’Keefe, Patrick, “Automatic Dimensional Inference,” in *Computational Logic: Essays in Honor of Alan Robinson*, Jean-Louis Lassez and Gordon Plotkin, Eds. pp. 479-483, MIT Press, 1991.