Citation for published version:
Stratford, J & Davenport, J 2008, Unit Knowledge Management. in S Autexier, J Campbell, J Rubio, V Sorge, M Suzuki & F Wiedijk (eds), Intelligent Computer Mathematics: Proc. AISC/Calculemus/MKM 2008. Lecture Notes in Computer Science, no. 2, vol. 5144, Springer, Berlin, Heidelberg, pp. 382-397. https://doi.org/10.1007/978-3-540-85110-3

DOI:
10.1007/978-3-540-85110-3

Publication date:
2008

Unit Knowledge Management

Jonathan Stratford & James H. Davenport

Department of Computer Science
University of Bath, Bath BA2 7AY, United Kingdom
Jonathan.Stratford@alumni.bath.ac.uk
J.H.Davenport@bath.ac.uk

Abstract. In [9], various observations on the handling of (physical) units in OpenMath were made. In this paper, we update those observations, and make some comments based on a working unit converter [21] that, because of its OpenMath-based design, is modular, extensible and reflective. We also note that some of the issues in an effective converter, such as the rules governing abbreviations, being more linguistic than mathematical, do not lend themselves to easy expression in OpenMath.

1 Introduction

For the purposes of this paper, we define a **unit** of measurement as *any determinate quantity, dimension, or magnitude adopted as a basis or standard of measurement for other quantities of the same kind and in terms of which their magnitude is calculated or expressed* [19, unit].

Units are generally thought of as "fairly easy", but, as this paper shows, there are some subtleties. One of the early design goals for Java was that it should be 'unit safe' as well as 'type safe', but this was dropped due to the difficulties [12].

There have been many famous examples where unit conversion was not undertaken, or where it was incorrectly calculated. The Gimli Glider [18, 24], as it became known, was a (then) new Boeing 767 plane, which, during what should have been a routine flight in 1983, ran out of fuel just over halfway to its intended destination. The ensuing investigation established that an incorrect conversion had been performed, leading to a woefully insufficient fuel payload, because the aircraft was one of the first of its kind to use a metric measure of fuel, and the refuellers had used an imperial conversion instead of the correct metric one. In addition, although a second check was carried out between legs of the flight, the same incorrect conversion was used.

Large organisations such as NASA are not immune to such problems [17]. Software controlling the thrusters on the Mars Climate Orbiter was configured to use imperial units, while ground control, and the other parts of the space craft, interpreted values as if they were metric. This led to the orbiter entering an incorrect orbit too close to Mars, and ultimately to its being destroyed.

Again, even widespread systems such as Google can get this wrong — see the example in section 4 — as can attempts such as OntoWeb to "understand" MathML in terms of simple structures such as RDF [6] (section 4.3).

2 Prior Work on Semantics of Units

2.1 OpenMath

OpenMath [3] is a standard for representing mathematical semantics. It differs from the existing versions[1] of Content MathML [4, 5] in being *extensible*: new Content Dictionaries (CDs) can add new OpenMath symbols, known as OMS, and can prescribe their semantic, via Formal Mathematical Properties (FMPs). In contrast, OpenMath *variables*, known as OMVs, are purely names.

OpenMath is essentially agnostic with respect to type systems. However, one particular one, the Simple Type System [8] is used to provide arity and similar information that is mechanical, and also information that is human-readable, but not currently machine processable, such as stating that `<OMS name="plus" cd ="arith1"/>` takes its arguments from, and returns an answer in, the *same* Abelian semigroup, by having the following signature.

```
<OMA>
 <OMS name="mapsto" cd="sts"/>
 <OMA>
  <OMS name="nassoc" cd="sts"/>
  <OMV name="AbelianSemiGroup"/>
 </OMA>
 <OMV name="AbelianSemiGroup"/>
</OMA>
```

2.2 Prior Work on Units in OpenMath

The major previous work on the semantics of units on OpenMath is [9]. This proposes several Content Dictionaries of units: `units_metric1`, `units_imperial1` and `units_us1`. These contain definitions of many common units covering a variety of dimensions (the dimensions themselves are defined in the Content Dictionary `dimensions1`) — metric (SI)[2] units are contained in `units_metric1`, for

[1] Versions 1 and, to a lesser extent, 2. It is intended that OpenMath 3 and Content MathML 3 will have converged on this important point

[2] The system in [9] actually differs in one respect from the SI system in [13, 15]. [9] takes the fundamental unit of mass to be the gram, rather than the kilogram. This is necessary, as a slavish following of the general principles of [13] would lead to such absurdities as the millikilogram (see section 3.1 of this paper) rather than the gram. [13, section 3.2] explains the special rules for multiples of the kilogram, as follows.

> Names and symbols for decimal multiples and submultiples of the unit of mass are formed by attaching prefix names to the unit name "gram", and prefix

example. [9] suggests using the “usual” `times` operator (that stored in `arith1`) to represent a number in a particular unit — i.e. storing the value as the number multiplied by the unit, with the unit following the value to which it refers. The suggestions for unit “implementation” in OpenMath are stated as being based on those used by a complementary mathematics display language, MathML — although not blindly; where the authors believe MathML has some deficiencies, these have been corrected. This document also specifies a reasonable way of connecting a prefix to a unit (described in section 3.1), thus defining `kilo` as a separate concept, which can then be used to construct `kilogram`.

[9] uses STS in a novel (for OpenMath) manner. Rather than merely ‘human-readable’, as with `<OMV name="AbelianSemiGroup"/>` above, it uses formal OpenMath symbols as the type, thus the type of `gram` is

```
<OMS cd="dimensions1" name="mass"/>
```

More complicated dimensions can be expressed, e.g. `Newton`’s type is

```
<OMS cd="dimensions1" name="force"/>
```

which has the formal property

```
<OMA>
  <OMS cd="relation1" name="eq"/>
  <OMS cd="dimensions1" name="force"/>
  <OMA>
    <OMS cd="arith1" name="times"/>
    <OMS cd="dimensions1" name="mass"/>
    <OMA>
      <OMS cd="arith1" name="divide"/>
      <OMS cd="dimensions1" name="length"/>
      <OMA>
        <OMS cd="arith1" name="power"/>
        <OMS cd="dimensions1" name="time"/>
        <OMI> 2 </OMI>
      </OMA>
    </OMA>
  </OMA>
</OMA>
```

Hence this system supports “dimensional analysis” (which should properly be called “dimensional algebra”).

2.3 Unit Converters

There are a great many unit converters publicly available online. These have a range of units and features — see the analysis in [21, chapter 2]. However,

symbols to the unit symbol “g” (CIPM 1967, Recommendation 2; PV, 35, 29 and *Metrologia*, 1968, 4, 45).

in all cases, they are monolithic, in that new units cannot be added to them by the user. In some senses, this means that they go against modularity and incrementality, and are not reflective, in that they do not know that other units exist.

None of the converters surveyed seem to know about dimensions, and hence attitudes to the question "convert months into days", instead of being our (1), were generally (2), and surprisingly often (4).

1. They are both `time`, so the conversion is meaningful, but I don't have an exact conversion factor.
2. There are $30\frac{699}{1600} = 30.43687500$ days in a month, which is correct on average, but false for every month (for some reason, Google uses 30.4368499)!
3. There are 30 days in a month, which is "the nearest", but not the most common (e.g. `http://online.unitconverterpro.com/unit-conversion/convert-alpha/time.html`), and which leads to absurdities such as "1 decade $= 121\frac{2}{3}$ months".
4. I don't know about months.

3 Abbreviations and Prefixes

Units have a variety of abbreviations and, particularly in the metric system, a range of prefixes. It is possible, as apparent in [14, section 5.3.5], to regard prefixed units as units in their own right, and introduce a unit `centimetre` with a formal property relating it to the `metre`, but this way lies, if not actual madness, vast repetition and the scope for error or inconsistency (who would remember to define the `yottapascal`?).

3.1 Prefixes

OpenMath therefore defines prefixes in the `units_siprefix1` CD, with FMPs to define the semantics, e.g. the following one for `peta`.

```
<OMA>
  <OMS name="eq" cd="relation1"/>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMI> 1 </OMI>
    <OMA>
      <OMS name="prefix" cd="units_ops1"/>
      <OMS name="peta" cd="units_siprefix1"/>
      <OMV name="unit"/>
    </OMA>
  </OMA>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMA>
```

```
      <OMS name="power" cd="arith1"/>
      <OMI> 10 </OMI>
      <OMI> 15 </OMI>
    </OMA>
    <OMV name="unit"/>
  </OMA>
</OMA>
</OMOBJ>
```

OpenMath uses a `prefix` operation (described as option 4 of [9, section 4]) to apply prefixes to OpenMath units. Its signature is given as follows.

```
<Signature name="prefix" >
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS name="mapsto" cd="sts"/>
    <OMS cd="units_sts" name="prefix"/>
    <OMV name="dimension"/>
    <OMV name="dimension"/>
  </OMA>
</OMOBJ>
</Signature>
```

which can be seen as

$$\text{prefix} \times \text{unit} \to \text{unit}. \tag{1}$$

This has the slightly unfortunate property that it would allow, for example, 'millimicrometre', which is explicitly forbidden by [13, p. 122]. This could be solved by making the signature

$$\text{prefix} \times \text{unit} \to \text{prefixed unit}, \tag{2}$$

which should probably be done.

This construction also allows the use of prefixes with non-SI units, but this is in fact legitimate [13, p. 122].

3.2 Abbreviations

One issue not covered in [9] is that of abbreviations. Here we must confess to not having a completely worked-out and sensible solution yet. The following possibilities have been considered.

Alternative Definition in the same CD This would mean that, for example, as well as `units_metric1` having the symbol `metre`, it would also have `m`. These would be linked via a FMP saying that the two were equal. Similarly, we would have prefixes `k` as well as `kilo`.

Pro A small extension of [9].

Con Allows "mixed" units such as `kilom` or `kmetre`, which are (implicitly) forbidden in [13].
Con No built-in way of knowing which is the full name and which is the abbreviation.

Alternative Definition in different CDs This would mean that `units_metric1` would have the symbol `metre`, and a new CD, say `units_metricabbrev1`, would have the symbol `m`. Again, these would be linked via a FMP saying that the two were equal. We would also have a new CD, say `units_sipefixabbrev1`, containing the abbreviations for the prefixes, and a *different* operation for combining the two, say

```
<Signature name="prefixabbrev" >
<OMOBJ xmlns="http://www.openmath.org/OpenMath">
  <OMA>
    <OMS name="mapsto" cd="sts"/>
    <OMS cd="units_sts" name="prefixabbrev"/>
    <OMV name="dimensionabbrev"/>
    <OMV name="dimensionabbrev"/>
  </OMA>
</OMOBJ>
</Signature>
```

Pro Prevents 'hybrid' units.
Pro A converter such as [21] could output either full names or abbreviations ('symbols' in [13]) depending on which CDs were available on the output side.
Con Knowledge of which is the name and which is the abbreviation is still implicit — merely moved from the name of the symbol to the name of the CD. The linkage between the name of the CD and the fact that the symbol should be regarded as `<OMV name="dimensionabbrev"/>` would be outside the formal OpenMath system.

"This isn't an OpenMath problem" It could be argued that abbreviating units and prefixes isn't an OpenMath problem at all, but a presentation one. This is superficially tempting, but poses the question "Whose problem is it?" Do we need a new layer of software to deal with it? One interesting sub-question here is whether an ontology language such as OWL [7] would be better suited to expressing such concepts.

3.3 Non-SI (but metric) Units

The reader will have noticed that the CD is called `units_metric1` rather than `units_si1`. This is deliberate, as it includes the `litre`, which is explicitly *not* an SI unit [13, Table 6, note (f)]. What of the other units in [13, Tables 6, 8]?

bar This is 100kPa, and presumably is retained because of its convenience for atmospheric pressure. Prefixes are valid with it [13, p. 127], though the only common one is the millibar (which is also the mbar, since the bar, uniquely, is its own abbreviation).

tonne (alias 'metric ton') [13, Table 6] This is essentially an alias for the megagram, and as such does not take prefixes[3]. If the "different CDs" approach above were to be adopted, this could be in yet another CD, say `units_metricmisc1`, on which no prefixing[4] operated.

hectare As 10^4m^2, this is in a very similar category to the tonne, and again does not take prefixes. The only question might be whether we ought to start with the `are` instead, but it is possible to argue that the `are` is obsolete, and conveys no advantage over the square decameter. If `litre_pre1964` moves to a different CD, we could reasonably leave the `are` there as well.

ångström Similarly.

nautical mile (= 1852m) Similarly.

knot ($=\frac{1852}{3600}$ m/s) Similarly. This is also an excellent argument for the representaton of definitional conversions (section 5.1) as exact fractions.

4 Not All Dimensions are Monoids

[9] assumed, implicitly, that all physical dimensions could be regarded as (Abelian) monoids, in the sense that they could be added, and hence multiplied by integers. This is in fact not the case.

4.1 The Temperature Problem

One problem was not addressed in [9], but has been observed elsewhere [1, `Celsius#_note-10`], viz. that temperatures are not the same thing as temperature intervals. This confusion is widespread, as evidenced by the Google calculator's ability to produce computational absurdities such as

```
(1 degree Celsius) plus (1 degree Celsius) = 275.15 degrees Celsius
```

More subtly, the reader should compare the following Google outputs (generated from `(-1C) in F` and `-(1C) in F` respectively).

```
(-1) degree Celsius = 30.2 degrees Fahrenheit
```

with

```
-(1 degree Celsius) = -953.14 degrees Fahrenheit
```

[3] The reader may ask "what about the megaton(ne)?" This is, of course, the mega[ton of TNT equivalent], and is not a unit of mass at all, but rather of energy, and is in fact 4.184 petajoules [22, Appendix B.8], where the figure 4.184 is definitional in the sense of section 5.1.

[4] Except that, in Belgium, the megaton (Nederlands) or megatonne (Français), and *certain* other multiples (kilo- to exa- only) are legal [2, Chap II, §4].

Possibly the best explanation of the difference between relative and non-relative temperatures is in [22, Appendix B.9][5].

4.2 To Monoid or not to Monoid

We can ask whether this is a peculiarity of temperature. The answer is in fact that it is not.

Most units form (Abelian) monoids, i.e. they can be added: 2 tonnes + 3 tonnes = 5 tonnes etc. *Non-relative*[6] temperatures are one obvious counter-example: $2°\text{F} + 3°\text{F} \neq 5°\text{F}$ or indeed any other temperature. The point is that *relative* temperatures, as in "A is ten degrees hotter than B", *are* additive, in the sense that if "B is twenty degrees hotter than C", then indeed "A is thirty degrees hotter than C", *are* additive, as are non-relative plus relative, but two absolute temperatures are *not* additive.

The same problem manifests itself with other scales such as decibels. Strictly speaking, these are purely relative, but in practice are also used in an absolute way, as in "the sound level exceeded 85dB". Again, the relative units form a monoid, but the absolute units do not.

This forces us to rethink the concept of "dimension". Though not using the word here (it is used in section 1.3), these are defined in [13, section 1.2] as follows.

> The base quantities used in the SI are length, mass, time, electric current, thermodynamic temperature, amount of substance, and luminous intensity.

This implies that all masses, for example, have the same dimension, and can be treated algebraically in the same way. But, as we have seen, not all temperatures are the same, and indeed have different algebraic properties. Two relative temperatures can be added, as in the example of A, B and C above. A relative temperature can be added to an absolute temperature, as in the following examples.

$$X \text{ was heated by } 10°\text{C}, \text{ from } 20°\text{C to } 30°\text{C}. \tag{3}$$

$$X \text{ was heated by } 10\text{K}, \text{ from } 20°\text{C to } 30°\text{C}. \tag{4}$$

$$X \text{ was heated by } 10°\text{C}, \text{ from } 293.15\text{K to } 303.15\text{K}. \tag{5}$$

$$X \text{ was heated by } 10\text{K}, \text{ from } 293.15\text{K to } 303.15\text{K}. \tag{6}$$

Equations (3) and (4) mean precisely the same thing (similarly for equations (5) and (6)), and this is obvious because, in the Content Dictionary defining relative temperatures, we state that the two are equal[7].

[5] `http://physics.nist.gov/Pubs/SP811/appenB9.html\#TEMPERATUREinterval`.

[6] Referring to 'absolute' temperatures would be likely to cause confusion, though that is what we mean in sense 10 of [19, absolute].

[7] This is the standard OpenMath way of doing so for units. It might make more sense simply to declare that the two symbols were precisely equal — see the discussion

```
<OMA>
  <OMS name="eq" cd="relation1"/>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMI> 1 </OMI>
    <OMS name="relative_Kelvin" cd="units_metric1"/>
  </OMA>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMI> 1 </OMI>
    <OMS name="relative_Celsius" cd="units_metric1"/>
  </OMA>
</OMA>
```

Equations (3) and (5) also mean precisely the same thing, but this time we need to rely on the definitions of non-relative temperatures, as in the following[8],

```
<OMA>
  <OMS name="eq" cd="relation1"/>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMI> 1 </OMI>
    <OMS name="degree_Kelvin" cd="units_metric1"/>
  </OMA>
  <OMA>
    <OMS name="minus" cd="arith1"/>
    <OMA>
      <OMS name="times" cd="arith1"/>
      <OMI> 1 </OMI>
      <OMS name="degree_Celsius" cd="units_metric1"/>
    </OMA>
    <OMA>
      <OMS name="divide" cd="arith1"/>
      <OMI>  27315 </OMI>
      <OMI>    100 </OMI>
    </OMA>
  </OMA>
</OMA>
```

and need to do some actual arithmetic, as in [21].

in section 7.2. It could be argued that, since the two symbols are equal, we do not actually need to have both — a minimalist view. This is similar to the discussion about `<OMS name="Landauin" cd="asymp1"/>` in [11], and our conclusion would be the same — convenience of rendering outweights minimality.

[8] This differs from the currently-published `experimental` CD `units_metric1` in following the recommendation in section 5.1 that 273.15, as a defined number, should be represented as an element of **Q**.

The system of dimensions in [9] had, as the Simple Type System [8] signature of dimensions, the OpenMath `<OMV name="PhysicalDimension"/>`. As its format implies, this is a mere name with no formal semantic connotation. We therefore suggest that this be replaced by two objects: `MonoidDimension` for those cases where the dimension *does* represent an (additive) monoid, and `NonMonoidDimension`. While these *could* be OMVs as before, we now believe that it would make more sense for them to be OMSs, since there is a definite semantics being conveyed here *to software packages*, rather than to human beings. This also agrees with a growing feeling in the OpenMath community that STS could "do more".

4.3 The Confusion is Widespread

It should be noted that the confusion between temperatures and relative temperatures manifests itself elsewhere in "web semantics". Consider an excerpt[9] from ontoworld's "approach to rewrite Content MathML so that it is expressable (sic) as RDF."

```
<owl:Class rdf:about="&phml;Temperature">
  <rdfs:subClassOf rdf:resource="&phml;PhysicalDimension"/>
</owl:Class>

<owl:Class rdf:about="&phml;TemperatureDifference">
  <rdfs:subClassOf rdf:resource="&phml;Temperature"/>
</owl:Class>
```

This could be argued to illustrate the difficulties of using a general-purpose language such as RDF beyond the semantics it is capable of handling, or, more simply perhaps, as an illustration of the fact that, since the *presentation* of temperatures and temperature intervals are the same, it is hard to distinguish the *semantics*, different though these may be.

5 Precision

5.1 Accuracy in the OpenMath

Conversion factors between units can be divided broadly into three categories.

Architected There are those that, at least conceptually, arose when the unit(s) were defined. All the metric prefixes fall into this category, as do conversions such as "3 feet = 1 yard", or even "1 rod = $5\frac{1}{2}$ yards". These conversions, and their inverses, clearly ought to be stored as elements of $\mathbf{Q}$, i.e. as OpenMath integers (`OMI` objects) or fractions thereof.

[9] `http://ontoworld.org/wiki/Semantic_MathML/0.1\#Physical_Dimensions`.

Experimental These are those that are truly determined by an experiment, such as the measurement of a standard of length in one system in terms of another such. An obvious example is those units that involve g, as in "1 slug $\approx$ 32.17405 pounds". These are probably best represented by means of floating-point numbers (`OMF` objects).

The reader might object that, since these items are only approximate, we should represent them by intervals, which are well-handled by OpenMath, as in the CD `interval1`. This is a plausible point. We happen to disagree with it, for the reasons about to be given, but nevertheless it is fair to say that more usage of these factors is called for before a definitive decision can be made.

- Manipulation of intervals is not conservative unless it is done symbolically — [10]. Hence, if g were to be represented by an interval, say $[32, 33]$ (absurdly wide, but this makes the point better), one slug would be $[32, 33]$ pounds, which, on conversion back, would become $[\frac{32}{33} \approx 0.97, \frac{33}{32} \approx 1.03]$ slugs.
- Definitions in OpenMath are intended to be permanent, so an increase in precision would have to lead to a change in the formal definition.
- Experimentalists tend not to work in terms of intervals, but in terms of the standard accuracy [16]. It would be a fair argument, though, to say that there *ought to be* OpenMath interval types capable of representing these.

Definitional These are those that started life as experimental, but have since been adopted as architected definitions. An obvious example is "1 yard = 0.9144 metre", which was adopted as a formal definition, replacing the previous experimental result[10] of "1 metre $\approx$ 39.370147 inches" [20] in 1959.

Another example would be the value of "absolute zero", in the days of an independent celsius scale, which was about $-273.15°$C. Nowadays, this is fixed as precsiely this value, or, more accurately, the concept of °C is defined in terms of absolute zero and the number 273.15 [13, 2.1.1.5].

We now believe that all *definitional* numbers occurring in unit conversions, as well as those architected, should be expressed as elements of $\mathbf{Q}$, i.e. as (fractions of) OMI. Hence the 0.9144 mentioned above should in fact be encoded as

```
<OMA>
  <OMS name="divide" cd="arith1"/>
  <OMI>  9144 </OMI>
  <OMI> 10000 </OMI>
</OMA>
```

This suggestion is well-characterised by the foot. Thus U.S. survey foot is defined[11] as $\frac{1200}{3937} \approx .3048006096$m, whereas the 'international' foot is defined

[10] It is worth noting that [20] describes this as "1 yard = 0.91439841 metres", [1, Imperial_units] as 0.914398416 metres, and an accurate conversion of the headline figure in [20] is .9143984146. This illustrates the general point that a number and its reciprocal are unlikely both to be exact decimals.

[11] U.S. Metric Law of 1866.

as *precisely* 0.3048m. The difference is only just detectable in IEEE (single-precision) floating-point, and is best stored exactly.

5.2 Precision of Display

There is also the issue of how much precision to display in the result. In general terms, the result should not be more precise than the least precise value used in the calculation. [21] currently supplies the entire result from the calculation, with a user-controllable "significant figures" level as part of the system's front-end. This was chosen on the basis that several alternatives considered appeared nonsensical or unreasonably difficult to implement or make firm decisions about. For example, internally, fractions (which in OpenMath are comprised of two infinite precision integers) are stored as finite-precision floating point numbers. It is impossible to tell, when presented with such a floating point number, whether it was made as such (again, OpenMath Floats, in decimal, can be of arbitrary precision) or whether it came from a fraction; in both these cases rounding would not be required, or if it was the result of a calculation, in which case rounding would be necessary. A nonsensical answer would clearly result if values were rounded during the calculation, and due to the aforementioned unknowable fact of where the value came from, it would also be impossible to maintain an internal counter of to how many significant figures the end result would be reasonable. With the chosen approach, a currently unanswered question regards the number of significant figures to display in a result such as `10 metre is 1 rod 5 yard 1 foot 3 inch 0.700787401574787 mil`. Should the number of significant figures only cover the last part of the result?

6 The two meanings of "obsolete"

According to the OpenMath standard [3], a content dictionary can be declared to be `obsolete`. This facility is needed so that, when an area of OpenMath gets rewritten in a (hopefully) better way, the semantics of existing OpenMath objects are preserved. However, there has been no need to deploy it yet. It is a feature of OpenMath[12] that this takes place at the content dictionary level, rather than the symbol level.

However, when we say that

```
<OMS name="litre_pre1964" cd="units_metric1"/>
```

is "obsolete", we do *not* mean that it is obsolete as an OpenMath symbol, rather that it is a current OpenMath symbol denoting an obsolete unit of measurement, and therefore that it *should* be in an `official` CD. Does this matter? There are two views.

No This is the view of [9]. It is a unit, which may still be encountered as old texts/experiments etc. are analysed, so should be present.

[12] At least at version 2. This may change in version 3.

Yes In [21] we produced a unit converter that attempted to produce the "best" fit to a given input. Hence, as 175.98 `pint`s converts to 100.002614775 `litre`s, but also 99.99991478 `litre_pre1964`s, the latter conversion would, much to the user's surprise (and indeed ours on first encountering this issue), be preferred. In fact, we actually get

```
OpenMathConverter.exe --source_quantity 175.98 --source_unit
pint --destination_unit metric
0.999998147801862 hectolitre_pre1964
```

Similarly, as 10 `metre`s is 10.93613298 `yard`s, but 1.988387814 `rod`s, the latter will again be preferred[13].

From the point of view of 'user-friendliness', we are inclined to sympathise with [21], and state that obsolete units belong in separate CDs, in particular that `litre_pre1964` should be moved from `units_metric1` to, say, `units_metricobs` before `units_metric1` becomes `official`.

7 Conclusion

We conclude that it is possible to use the OpenMath unit system (or ontology, as one might call it) of [9] to produce a serious and, unlike others, *extensible* unit converter, as in [21].

7.1 Recommendations for OpenMath Unit/Dimension CDs

The most important recommendation is a recognition that some (in our sense of the word) dimensions are (additive) monoids, and some are not, as outlined in section 4.

1. Move `litre_pre1964` into a different CD, which is an `official` CD of "obsolete" units. Similar steps should be taken for "obsolete" imperial units.
2. Fix `dimensions1` so as to have a definition for power.
3. Delete `metre_squared` from the `units_metric1`. It is anomalous (why isn't there `metre_cubed`, and why doesn't `units_imperial1` have `foot_squared`?) and tempts a piece of software (such as earlier versions of [21]) into creating units such as

```
<OMA>
  <OMS name="prefix" cd="units_ops1"/>
  <OMS name="deci" cd="units_siprefix1"/>
  <OMs name="metre_squared" cd="units_metric1"/>
</OMA>
```

[13] which will in fact come out as `10 metre is 1 rod 5 yard 1 foot 3 inch 0.700787401574787 mil`.

which is a deci(metre^2), as opposed to a (decimetre)^2, and is illegal [13, p. 121].
4. `units_imperial1` is missing units such as `inch`, which need to be added.
5. Add a CD for U.S. units, where different (e.g. for volume). Move U.S. Survey units[14], currently in `units_imperial1` into this.
6. Add a CD for E.U. units, where different. The only case known to the authors is the `therm`, which comes in both U.S. and E.U. variants. [22, footnote 25] states the following.

   > Although the therm (EC), which is based on the International Table Btu, is frequently used by engineers in the United States, the therm (U.S.) is the legal unit used by the U.S. natural gas industry.

   The difference is about 0.02%.
7. Update all the semantics in the world of OpenMath units so as to adhere to the principles of section 5.1, in particular definitional numbers should be expressed as elements of $\mathbf{Q}$, i.e. as (fractions of) OMI.
8. Sort out electrical energy definitions and other suggestions in [9].
9. Modify the signature of `prefix`, as described in section 3.1, from (1) to (2).
10. Update the definition of `pascal` to include an FMP: currently missing.

7.2 Further Considerations

We saw, in section 4.3, that the sort of semantics of RDF [6] are inadequate to convey the relationship between, for example, relative temperature and non-relative temperature. However, the OpenMath required to state that

```
<OMS name="relative_Kelvin" cd="units_metric1"/>
```

and

```
<OMS name="relative_Celsius" cd="units_metric1"/>
```

mean *precisely* the same thing is clumsy, and requires OpenMath-capable reasoning whereas all that is needed *in this case* is RDF-like, or OWL-like, reasoning.

We can also ask whether OWL would not be better at solving the abbreviations problem than OpenMath (see section 3.2).

Some units (`calendar_year` is the notable example) have multiple FMPs, whereas most of the other secondary units have only one, which is essentially a *defining* mathematical property in the sense [23, I, p. 11] that the definiens can be completely replaced by the definiendum. Making the distinction clear, as has been proposed elsewhere in the OpenMath community, would be a step forward.

7.3 Unsolved Problems

We see two currently unsolved problems.

[14] See the discussion at the end of section 5.1 for the (small) difference.

1. The abbreviations issue — section 3.2, and the fact that legal prefixes can differ between countries, as in note [4]. This last might be a problem best solved outside OpenMath.
2. The `dimensions1` CD has symbols `length` and `displacement`, with the difference being explained (under `displacement`) as

   > This symbol represents the spatial difference between two points. The direction of the displacement is taken into account as well as the distance between the points.

   It would be possible to read this as a non-monoid version of `length`, but more thought is necessary.

Acknowledgements. The authors are grateful to Dr Naylor for his explanations of [9], to Dr Donner for useful comments, and to Dr De Vos for her help with Belgian metrology.

References

1. Anonymous. Wikipedia, English. `http://www.wikipedia.org`, 2008.
2. The Kingdom of Belgium. *Réglementation Métrologique*. `http://mineco.fgov.be/organization_market/metrology/showole_FR.asp?cParam=1150`. *Metrologische Reglementering*. `http://economie.fgov.be/organization_market/metrology/showole_nl.asp?cParam=1152`.
3. S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase. OpenMath Standard 2.0. `http://www.openmath.org/standard/om20-2004-06-30/omstd20.pdf`, 2004.
4. World-Wide Web Consortium. Mathematical Markup Language (MathML[tm]) 1.01 Specification. `http://www.w3.org/TR/MathML/`, 1999.
5. World-Wide Web Consortium. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). `http://www.w3.org/TR/MathML2/`, 2003.
6. World-Wide Web Consortium. RDF Semantics. `http://www.w3.org/TR/2004/REC-rdf-mt-20040210/`, 2004.
7. World-Wide Web Consortium. Web Ontology Language OWL. `http://www.w3.org/2004/OWL/`, 2004.
8. J.H. Davenport. A Small OpenMath Type System. *ACM SIGSAM Bulletin*, 34(2):16–21, 2000.
9. J.H. Davenport and W.A. Naylor. Units and Dimensions in OpenMath. `http://www.openmath.org/documents/Units.pdf`, 2003.
10. J.H. Davenport and H.-C. Fischer. Manipulation of Expressions. *Improving Floating-Point Programming*, pages 149–167, 1990.
11. J.H. Davenport and P. Libbrecht. The Freedom to Extend OpenMath and its Utility. *To appear in Mathematics in Computer Science*, 2008.
12. M. Donner. Private Communication, May 2008.
13. Organisation Intergouvernementale de la Convention du Mètre. The International System of Units (SI) 8th edition. `http://www.bipm.org/utils/common/pdf/si_brochure_8_en.pdf`.
14. World-Wide Web Consortium (ed. D.W. Harder and S. Devitt). Units in MathML. `http://www.w3.org/Math/Documents/Notes/units.xml`, 2003.
15. IEEE/ASTM. SI 10-1997 Standard for the Use of the International System of Units (SI): The Modern Metric System. *IEEE Inc.*, 1997.

16. International Standards Organisation. *Guide to the Expression of Uncertainty in Measurement*, 1995.
17. Mars Climate Orbiter Mishap Investigation Board. Mars climate orbiter: Phase i report. `ftp://ftp.hq.nasa.gov/pub/pao/reports/1999/MCO\_report.pdf`, November 1999.
18. Wade H. Nelson. The Gimli Glider. *Soaring Magazine*, 1997.
19. Oxford University Press. Oxford English Dictionary. `http://dictionary.oed.com/entrance.dtl`, 2008.
20. J. E. Sears, W. H. Johnson, and H. L. P. Jolly. A New Determination of the Ratio of the Imperial Standard Yard to the International Prototype Metre. *Phil. Trans. Roy. Soc. A*, 227:281–315, 1928.
21. Jonathan Stratford. An OpenMath-based Units Converter. Dissertation for B.Sc. in Computer Science, University of Bath, 2008. Technical Report CSBU–2008–02, `http://www.cs.bath.ac.uk/department/technical-report-series/technical-report-series/index.php`
22. B. N. Taylor. *Guide for the Use of the International System of Units (SI)*. `http://physics.nist.gov/Pubs/SP811`, may 2007.
23. A.N. Whitehead and B. Russell. *Principia Mathematica*. Cambridge University Press, 1910.
24. Merran Williams. The 156-tonne Gimli Glider. *Flight Safety Australia*, 2003.