Using Units of Measurement in Formal Specifications

Ian J. Hayes and Brendan P. Mahony

Department of Computer Science, University of Queensland, Brisbane, Australia

Abstract. In the physical sciences and engineering, units of measurement provide a valuable aid to both the exposition and comprehension of physical systems. In addition, they provide an error checking facility comparable to static type checking commonly found with programming languages. It is argued that units of measurement can provide similar benefits in the specification and design of software and computer systems.

To demonstrate this, we present an extension of the Z specification notation with support for the incorporation of units in specifications and demonstrate the feasibility of static dimensional analysis of the resulting language.

1. Introduction

It is common place for scientists and engineers to use units of measurement in describing models of physical systems because

- for such descriptions to be complete, the units of quantities need to be specified — a building plan is useless if the units of the dimensions are not known;
- having units explicit in a description aids the reader to understand it; and
- dimensional analysis [Bri31] can be used as both a consistency check on and guide in the development of a description.

In the specification of software systems, especially those interfacing with physical systems, one would like to enjoy the benefits of a notation that incorporates units. Units can easily be incorporated into specifications informally, but for specifications written in formal specification languages, such as VDM [Jon90] and Z [Spi92, Hay93], it is preferable that the language provide support for units. The aims of the current paper are

---

Correspondence and offprint requests to: Dr I. J. Hayes, Department of Computer Science, University of Queensland, Brisbane, Queensland 4072, Australia.

- to argue that incorporation of units into a formal specification language is worthwhile in general and especially where the language is to be used to describe components involving physical systems (Section 2);

- to provide a model of units in terms of standard Z (Section 3);

- to provide a suitable syntax for extending the Z notation to incorporate units (Section 4);

- to provide a toolkit for writing specifications using units in Z (Section 5); and

- to provide rules that extend type checking in Z to include dimensional analysis (Appendix A).

We assume the reader is familiar with the general concept of units as used in the physical sciences [ISO79]. Our intentions are to demonstrate that units are useful in writing formal specifications and adapt units to formal specification in the Z notation.

1.1. Units and Dimensions

The theory of measurement deals with how best to structure measurements systems in support of scientific theories. The interested reader is referred to [Ell66] as a useful introduction to the theory of measurement in general and the use of dimensions of measurements in the physical sciences.

Units and (more properly) dimensions of measurement fulfill a function in the physical sciences comparable to the function of strong typing in the computing sciences. Associating units, and hence dimensions, with measurements serves to categorise measurements in terms of what was measured and how measurement was effected, and more importantly, for our purposes, it provides a guiding context for the algebraic and arithmetic manipulation of measurements.

A *measurement* is an assignment of a value to some aspect of a system called a *quantity.* A measurement method (or a class of such methods) under which equal values are assigned to equal quantities under equal conditions is called a *scale* or a *unit.* Two scales are said to be *similar* if there is a strict ratio conversion factor between them. A class of similar scales which are thought to measure the same quantity are called a *dimension.*

When performing arithmetic on measurements, a consistent system for annotating measurement values with unit symbols provides an important error checking device which serves to prevent such foolishnesses as adding $12\,\mathrm{ft}$ to $3\,\mathrm{m}$ to get an answer of 15. An example of such a system is the mks (or SI) system of mechanical units [ISO79], based around the metre, the kilogram, and the second. Within the mks system each unit symbol is formed as a simple polynomial in these three base units, for example m^2 for area measurements, $\mathsf{m}\cdot\mathsf{s}^{-1}$ for speed, and $\mathsf{m}\cdot\mathsf{kg}\cdot\mathsf{s}^{-2}$ for force. Every measurement is assigned a unit symbol of this form, regardless of the actual measurement method used (a scale conversion is applied if necessary). The unit symbol serves as a simple guide to the arithmetic operations that may be legally performed. For example, measurements may only be compared, added, or subtracted if they have identical units and if measurements are multiplied their unit polynomials are added.

When performing algebra with real-valued variables representing measurable quantities the exact scales or units used become less important. It is customary to instead use a system of dimensions to provide consistency checks on algebraic manipulations. Corresponding to the mks system of units is the LMT system of dimensions for mechanical systems. Each dimension symbol is formed as a simple polynomial in the base dimensions of length (L), mass (M), and time (T), for example L^2 for area, $\mathsf{L}\cdot\mathsf{T}^{-1}$ for speed, and $\mathsf{L}\cdot\mathsf{M}\cdot\mathsf{T}^{-2}$ for force. Here the particulars of scales are abstracted, each dimensional formula is a schema for determining the appropriate scale for a quantity from a given selection of base scales. The rules for algebraic manipulation of dimensioned variables have the same form as those for arithmetic operations on measurements with units. The type-checking of equations according to the dimensions of the various algebraic components is called *dimensional analysis.*

Dimensions place a useful typing structure on the otherwise homogeneous field of real numbers, one that both serves to classify real-valued variables according to their use and to reflect the algebraic properties of the field of reals. This typing structure is so restrictive on the allowed forms of equations in physical quantities that it is often possible to predict the correct law for describing a physical system purely from the physical dimensions of the system observables [Bri31].

2. Motivation

In [DG90], Delisle and Garlan give a formal specification of an oscilloscope.[1] In extending this work to a specification of an oscilloscope that incorporated physical limitations [Hay90], we quickly discovered that it was advantageous to include the units of variables in the specification. Units were added to the specification informally at first, and then included formally by modelling units in standard Z. Although this allowed the units to be included in the specification, the notation was limited and cumbersome, and it did not provide for automatic dimensional analysis of the specification. By way of motivation we present the initial stages of the oscilloscope specification from [Hay90] using Z extended with notation for units, dimensions, and quantities.

2.1. Oscilloscope Specification

For an ideal oscilloscope we assume perfect resolution both in terms of time and signal levels; we model signal voltage measurements as real-valued[2] ($\mathbb{R}$) quantities of electric potential and time measurements as non-negative, real-valued ($\mathbb{R}_+$) quantities of time. In order to distinguish these two classes of quantity, we introduce a dimension name for electric potential, EP, which refers to the class of scales similar to the volt, and a dimension name for time, T, which refers to the class of scales similar to the second. We use an explicit operator '$\odot$' to combine a specification of the range of measurement values and the dimensions of measurement of a quantity.

$$Voltage == \mathbb{R} \odot \mathsf{EP}$$
$$Time == \mathbb{R}_+ \odot \mathsf{T}$$

The value thus stored for quantities of each dimension is their measure on the respective characteristic or *standard* scales, being the volt for electric potential and the second for time. We assume that all the standard arithmetic operators and relations are extended to work on compatible, real-valued quantities (as described in Section 5 below).

An input signal can be modelled as a function of time; we define the type *Signal* by

$$Signal == Time \nrightarrow Voltage$$

Elements of the domain of this function are quantities of time and elements of the range are quantities of electric potential. A segment to be displayed corresponds to a contiguous time interval selected from a signal. We define *Segment* as the set of all signals with non-empty, contiguous domains.

$$Segment == \{s : Signal \mid$$
$$(\exists\, t1, t2 : Time \bullet t1 < t2 \land \mathrm{dom}\, s = \{t : Time \mid t1 < t < t2\})\}$$

A screen coordinate can be represented by a real-valued quantity of length. We introduce a new dimension name L for length with standard scale the metre.

$$Coord == \mathbb{R} \odot \mathsf{L}$$

A point on the screen can be represented by a pair of coordinates.

$$Point == Coord \times Coord$$

A trace is a mapping from time to points.

$$Trace == Time \nrightarrow Point$$

Scaling. A segment of a signal must be scaled, both horizontally and vertically, for display on a screen. The vertical scale converts from a voltage to a length; a typical scaling factor might be 2 volts per centimetre. The vertical scale is thus a quantity of electric potential per length, which is to say it has dimensions $\mathsf{EP} \cdot (^{-1}\mathsf{L})$. Similarly, the horizontal scale factor converts from time to length giving it dimensions $\mathsf{T} \cdot (^{-1}\mathsf{L})$; a typical

---

[1] Modern oscilloscopes and other electronic instruments contain significant software components to provide, amongst other things, sophisticated user interfaces.
[2] For a definition of reals in Z see [Val93].

value might be 5 milliseconds per centimetre. Scaling also incorporates a translation so that the start of the segment corresponds to a horizontal trace offset of zero.

```
┌─ Scale ──────────────────────────────────────────────────────────────
│  segment : Segment
│  HScale : ℝ₊ ⊙ T · (⁻¹L)
│  VScale : ℝ ⊙ EP · (⁻¹L)
│  scaled : Trace
├──────────────────────────────────────────────────────────────────────
│  scaled = (λ t : dom segment •
│       ((t − inf(dom segment))/HScale, segment(t)/VScale))
└──────────────────────────────────────────────────────────────────────
```

The inclusion of dimensional information in the the typing system makes it possible to perform dimension-based consistency checks on the equation for *scaled*.

The function *inf* gives the greatest lower bound of a non-empty, contiguous time period, so that both t and $inf(\mathrm{dom}\,segment)$ represent quantities of time. Hence it is valid to subtract these quantities — only quantities with the same dimensions of measurement may be added or subtracted — and the result of the subtraction is also a quantity of time. This is then divided by *HScale*, a quantity with dimensions of time over length, making

$(t − inf(\mathrm{dom}\,segment))/HScale$

a quantity of length as required.

Since t is a quantity of time in the domain of *segment*, *segment*(t) is a well-defined quantity of electric potential. Hence, as *VScale* has dimensions of electric potential over length,

$segment(t)/VScale$

represents a quantity of length as required.

Benefits of Dimensions. The primary advantage of including dimensions explicitly in the oscilloscope specification is that it would not be possible to make it complete without them. For example, specifying a value for *HScale* would have no meaning except with respect to a particular scale such as seconds per metre. In specifying a dimension for *HScale* it becomes necessary to specify an associated scale when specifying a value for *HScale*. A secondary advantage is that the dimensions of the various variables give a clear indication to the reader of the way they are intended to be used. That the horizontal scale has dimensions of time over length clearly indicates that it is the horizontal component of the trace that represents the passage of time. A further advantage is that it is now feasible to subject the specification to dimensional analysis, as described in the above example, as part of the process of type checking it.

Further work on specifying real-time systems [MH92, MMH93] has emphasised the utility of specifying the dimensions of quantities in specifications and made us realise that there is perhaps a wider application for the use of dimensions in specification in general. Whenever conceptually different components of a specification or program are modeled by the same data structure, a system of dimensions will provide some or all of the advantages described above.

2.2. Type Checking and Dimensional Analysis

Before detailing our extension to Z to incorporate dimensions, it is appropriate to examine possible ways of providing dimensional information.

Separate Z Types. One possibility is to provide a new type for each different variety of quantity. In Z we can use the free type mechanism to introduce a type as the image of its constructor function. For example, a type representing time can be defined as a free type by $Time ::= seconds \langle\!\langle \mathbb{R} \rangle\!\rangle$. For the oscilloscope example this not only means providing new types for time, length and voltage, but also new types for quantities measured on the conversion scales volts per metre and seconds per metre. However, even after going to the effort of defining these new types, we need to define operations such as addition, subtraction, and ordering on them. These operations are not automatically provided by the Z free type mechanism. These problems can be overcome (to some extent) by the next alternative.

Similar Types. Another possibility (available in abstract data type languages such as OBJ3 [GW88]) is to provide new types for quantities that are essentially renamed copies of the value type along with copies of the operators on values. This has the advantage that it can automatically provide appropriate arithmetic operators on values within a new type. However, we still have a problem defining operations that use arithmetic expressions involving a mixture of types.

A comparable facility in Z is the ability to define generic types, since this also allows for distinguished copies of a single data type. However, in addition to the problem of treating mixed dimensions, the set generic facility is not a suitable approach for the treatment of dimensions since it allows arbitrary sets to act as dimension symbols and it is not algebraically convenient to treat dimensions as unstructured sets.

Explicitly Modelling Dimensions. Quantities may be explicitly modelled in Z by replacing each value with a magnitude-dimension pair and defining a (single) set of operators to act on these pairs [Hay90]. This approach is considered in detail in Section 3 and works reasonably well in that it allows quantities with different dimensions to be mixed. However, under the standard Z typing mechanism, this approach does not facilitate automatic static checking of specifications for consistent use of dimensions because the dimensional information is semantic in nature and cannot be statically determined using the standard Z typing mechanism.

To allow for static dimensional analysis of specifications making use of this model, it is necessary to introduce syntactic conventions for distinguishing the dimensional information in a specification, such as the base dimensions and dimensions constructors and operators, from the conventional parts of the specification. Our approach is to incorporate dimensions information in the static typing mechanisms of the notation. Each quantity is given a static or generic type which can include a dimension. We enable this by extending the Z generics system to allow the use of dimension generics as well as set generics, including the ability to define base dimensions as given dimensions in like manner to given sets. Since static dimensional information is treated in exactly the manner of standard static typing information, this represents a conservative extension to the Z static typing system.

3. A Model of Units and Dimensions in Standard Z

In this section we introduce a model for units, dimensions, and quantities in standard Z as a precursor to defining an extended syntax able to support the use of systems of units and dimensions similar to the mks system discussed in Section 1. In this model we identify each quantity with a measurement of the quantity on a standard scale associated with its dimension of measurement.

3.1. Unit and Dimensional Formulae

Recall from Section 1 that each unit formula is represented by a simple polynomial in some collection of standard base units or scales.

[BASE]

For example, in the physical sciences, the SI standard base units [ISO79] are

$\quad\mid\quad$ *Metre*, *Kilogram*, *Second*, *Ampere*, *Kelvin*, *Candela*, *Mole* : BASE

$\quad\mid\quad$ distinct⟨*Metre*, *Kilogram*, *Second*, *Ampere*, *Kelvin*, *Candela*, *Mole*⟩

The standard scales for other quantities are *derived* from the base units according to some simple formula for combining measurements on the base scales.

For example, the area of a rectangle may be measured by multiplying its breadth in metres by its length in metres. The area of more complex shapes may be determined by summing the areas of a collection of rectangles that form a covering of the shape. The area scale thus determined is called the square metre.

In like manner a standard, derived scale is determined for each physical quantity as a product of measurements on the base scales. Each derived scale may then be characterised by a simple polynomial in the base scales. We model such polynomials as a function returning the index of each base unit with a nonzero

index [Hay89]. For example, the index functions of length and time units are

$$\{Metre \mapsto 1\}$$
$$\{Second \mapsto 1\}$$

and of the acceleration unit is

$$\{Metre \mapsto 1, Second \mapsto -2\}.$$

The set of all units contains all such representations of polynomials.

$$\mathsf{UNIT} == \mathsf{BASE} \nrightarrow (\mathbb{Z} \setminus \{0\})$$

The use of partial functions simplifies some aspects of the model, base units not in the domain of a unit are considered to have zero frequency. The index function of a unitless quantity is

$$\begin{array}{|l|} \hline \perp : \mathsf{UNIT} \\ \hline \perp = \varnothing \\ \end{array}$$

For a given standard unit, the number of instances of any of its base units can be extracted from its index function with the operator '$(\_ \sharp \_)$':

$$\begin{array}{|l} \hline \_ \sharp \_ : \mathsf{UNIT} \times \mathsf{BASE} \to \mathbb{Z} \\ \hline \forall\, U : \mathsf{UNIT};\ b : \mathsf{BASE} \bullet \\ \quad (b \in \operatorname{dom} U \Rightarrow U \sharp b = U(b)) \wedge \\ \quad (b \notin \operatorname{dom} U \Rightarrow U \sharp b = 0) \\ \end{array}$$

Recall now that dimension symbols are also formed as polynomials over a collection of base symbols. Each dimension symbol represents the class of scales used to measure a given quantity. Dimensional formulae form an abstraction of the system used to determine the appropriate standard unit for a quantity from a particular collection of standard base units. Given a dimensional formulae for a quantity, its standard unit with respect a given collection of base units is given by replacing each dimension symbol with its associated base unit. Thus the index function of a standard unit is also a characterisation of the associated dimensional formula and we also adopt the index function as our model for dimensions of measurement.

$$\mathsf{DIM} == \mathsf{UNIT}$$

Thus, the SI dimensions of length and time are represented by the respective index functions of their standard units.

$$\mathsf{L} == \{Metre \mapsto 1\}$$
$$\mathsf{T} == \{Second \mapsto 1\}$$

In order to return to the standard representation of dimensions as polynomial formulae we introduce the binary operators '$(\_ \cdot \_)$' and '$(\_ \Uparrow \_)$'. The operator '$(\_ \cdot \_)$' is used to pointwise add the characteristic indices of two dimensions.

$$\begin{array}{|l} \hline \_ \cdot \_ : \mathsf{DIM} \times \mathsf{DIM} \to \mathsf{DIM} \\ \hline \forall\, U, V : \mathsf{DIM};\ b : \mathsf{BASE} \bullet \\ \quad (U \cdot V) \sharp b = U \sharp b + V \sharp b \\ \end{array}$$

The dimensions of magnitude $\perp$ is the identity of the $(\_ \cdot \_)$ operator. That is, for any dimensions of measurement, U and V,

$$U \cdot \perp = \perp \cdot U = U$$

and if $U \neq \perp$, then there is some b such that $U \sharp b \neq 0$ and hence $V \cdot U \neq V$. The operator $(\_ \cdot \_)$ is commutative and associative. This follows immediately from the commutativity and associativity of integer addition.

The '$(\_ \Uparrow \_)$' operator is used to multiply the characteristic indices by an integer factor.

$$\begin{array}{|l}
\hline
\_ \Uparrow \_ : \mathsf{DIM} \times \mathbb{Z} \rightarrow \mathsf{DIM} \\
\hline
\forall\, U : \mathsf{DIM};\ n : \mathbb{Z};\ b : \mathsf{BASE} \bullet \\
\quad (U \Uparrow n) \sharp b = (U \sharp b) * n \\
\end{array}$$

The operator $(\_ \Uparrow \_)$ satisfies the following laws.

$$U \Uparrow 0 = \bot$$
$$U \Uparrow 1 = U$$
$$(U \Uparrow n) \Uparrow m = U \Uparrow (n * m)$$
$$(U_1 \cdot U_2) \Uparrow n = (U_1 \Uparrow n) \cdot (U_2 \Uparrow n)$$

The standard method of writing dimensions of measurement is achieved by defining dimension symbols corresponding to each of the base units, such as L and T above, and combining these base dimensions, using $(\_ \cdot \_)$ and $(\_ \Uparrow \_)$, to construct a compound dimensional formula. For example, we write $\mathsf{L} \cdot (\mathsf{T} \Uparrow -2)$ as the dimensions of measurements of acceleration. As a convenient shorthand we make use of superscript notation for writing exponents of dimensions, for example, the dimensions of acceleration may be written $\mathsf{L} \cdot \mathsf{T}^{-2}$.

3.2. Quantities

A quantity consists of a value of some type M together with an associated dimension of measurement.

$$\mathsf{MEAS}[M] == M \times \mathsf{DIM}$$

The value associated with a measurement is that associated with the standard scale of the associated dimension. This does not preclude the expression of measurements on other scales, but does require the inclusion of a conversion factor in such cases.

Primarily this paper considers quantities with numeric values, i.e., $\mathsf{MEAS}[\mathbb{N}]$, $\mathsf{MEAS}[\mathbb{Z}]$, and $\mathsf{MEAS}[\mathbb{R}]$, and the dimensions system discussed above is designed to reflect and complement the algebraic properties of the reals. For example, the usefulness of dimensional analysis relies heavily on the ability to express a smooth, real-valued function as a Taylor series of polynomials [Bri31, Ell66]. Other measurement domains are possible, e.g.,

$$\mathsf{MEAS}[\{red, amber, green\}]$$

for traffic lights or

$$\mathsf{MEAS}[\{long, medium, short\}]$$

for qualitative length measurements, but in such domains it may not be possible to apply the full power of dimensional analysis, either because of the lack of algebraic structure or because of incompatible structures.

Two basic functions extract the value (val) and dimensions (dim) of a measurement.

$$\begin{array}{|l}
\hline
[M] \\
\hline
\mathsf{val} : \mathsf{MEAS}[M] \rightarrow M \\
\mathsf{dim} : \mathsf{MEAS}[M] \rightarrow \mathsf{DIM} \\
\hline
\forall\, m : M;\ U : \mathsf{DIM} \bullet \\
\quad \mathsf{val}(m, U) = m\ \wedge \\
\quad \mathsf{dim}(m, U) = U \\
\hline
\end{array}$$

We would like to declare a variable in a specification to have a fixed dimension of measurement as well as a fixed domain of values. To accommodate this we provide notation to construct subsets of MEAS in which all elements have the same dimension of measurement. We call such sets quantities.

$$\mathsf{QUANTITY}[M] == \{Q : \mathbb{P}\,\mathsf{MEAS}[M] \mid \exists\, d : \mathsf{DIM} \bullet \forall\, q : Q \bullet \mathsf{dim}(q) = d\}$$

The simplest quantities are the dimensionless quantities or *magnitudes*. With each standard Z type M we associate a domain of magnitudes.

$$M_{\bot} == M \times \{\bot\}$$

For example, the domain of real magnitudes is $\mathbb{R}_\perp$.

We write general quantities by combining a domain of magnitudes (or any existing quantity) with a dimension using the operator '$(\_ \odot \_)$'.

$$
\begin{array}{l}
\underline{[M]}\\
\hline
\_ \odot \_ : \mathsf{QUANTITY}[M] \times \mathsf{DIM} \to \mathsf{QUANTITY}[M]\\
\hline
\forall\, Q : \mathsf{QUANTITY}[M];\ U : \mathsf{DIM} \bullet\\
\quad Q \odot U = \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q) \cdot U)\,\}\\
\hline
\end{array}
$$

The dimension of magnitudes, $\perp$, acts as a right identity for $(\_ \odot \_)$. For any quantity, Q,

$$
\begin{aligned}
Q \odot \perp &= \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q) \cdot \perp)\,\}\\
&= \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q))\,\}\\
&= Q
\end{aligned}
$$

and for any dimension $U \neq \perp$,

$$
\begin{aligned}
Q \odot U &= \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q) \cdot U)\,\}\\
&\neq \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q))\,\}\\
&= Q.
\end{aligned}
$$

It interacts with $(\_ \cdot \_)$ as follows,

$$
Q \odot U \odot V = \{\, q : Q \bullet (\mathsf{val}(q), \mathsf{dim}(q) \cdot U \cdot V)\,\} = Q \odot (U \cdot V).
$$

Although the standard Z model described above adequately captures the necessary information about the dimensions of an expression, the standard Z type checking mechanism applied to specifications written using this model does not do any dimensional analysis of such specifications. For example, in the oscilloscope example of Section 2, the EP symbol is a shorthand for SI dimensional formula $\mathsf{L}^2 \cdot \mathsf{M} \cdot \mathsf{T}^{-3} \cdot \mathsf{I}$, but the standard Z type checking does not verify that a quantity from $\mathbb{R} \odot \mathsf{EP}$ is also a quantity from $\mathbb{R} \odot \mathsf{L}^2 \cdot \mathsf{M} \cdot \mathsf{T}^{-3} \cdot \mathsf{I}$. Without adding to the typing mechanism, this can only be determined by appeal to the semantics of the specification.

4. A Syntax for Dimensions in Z

So far we have developed a model of dimensions in standard Z. This can be used to define variables along with their dimensions of measurement in standard Z. But we would like to take dimensions of measurement one step further and integrate them into Z in a way that allows static dimensional analysis of specifications.

4.1. Introducing Dimensions

Z already allows the introduction of new *basic* sets, as well as allowing definitions that are generic with respect to sets. The primary change to the Z notation is to allow the introduction of new *basic* dimension symbols, as well as allowing definitions that are generic with respect to dimensions. In order to distinguish the two classes of generics we introduce a limited typing mechanism for generic parameters. To distinguish basic dimensions from basic sets we write

$$[S : \mathsf{SET}]$$

to introduce a basic set and

$$[U : \mathsf{DIM}]$$

to introduce a base dimension symbol. For definitions with a generic set parameter, S, we write

$$
\begin{array}{l}
\underline{[S : \mathsf{SET}]}\\
\hline
\quad \vdots\\
\hline
\end{array}
$$

and for definitions with a generic dimensional parameter, U, we write

$$
\begin{array}{l}
\rule[0.5ex]{1.2em}{0.4pt}[\,U : \mathsf{SET}\,]\rule[0.5ex]{22em}{0.4pt} \\
\left|\quad \vdots \right. \\
\rule{1em}{0.4pt}
\end{array}
$$

In order to maintain compatibility with existing Z specifications we allow the ": SET" syntax to be optional when introducing sets.

In support of the use of dimensions we treat the three operators, $(\_\cdot\_)$, $(\_\Uparrow\_)$ and $(\_\odot\_)$ introduced in Section 3 as primitive type constructor symbols. We also use the symbol $\perp$ as the dimensions of a unitless quantity.

The advantage of this system over the use of explicit value-dimension pairs is that the collection of distinct basic dimensions is declared statically, allowing the dimensions of all objects in a specification to be resolved statically in just the same way that normal typing is resolved. An algorithm for performing static dimensional analysis of dimensioned Z specifications is presented in Appendix A.

Quantities. Each dimensioned type or quantity has both a magnitude type and a dimension type. A special class of quantities are those with the dimensions of magnitude ($\perp$ in the model presented in Section 3) which we refer to as domains of magnitude. $\mathbb{R}$ is the domain of real magnitudes, $\mathbb{N}$ is the domain of natural number magnitudes, etc. In the extended syntax for units only the domains of magnitudes are visible to the user, since each quantity has an associated dimension. The underlying types are hidden below the dimensioned typing system. Thus when we write $\mathbb{R}$ in the extended syntax we actually refer to $\mathbb{R}_{\perp}$ from the standard Z model in Section 3.

The operator, $(\_\odot\_)$, constructs a new quantity from an existing quantity and a dimension of measurement. We identify each dimension symbol and dimensional formula with a standard scale of measurement so that a type expression $M \odot U$ represents quantities with dimensions U as measurements on a standard scale associated with the dimensional formula U which take values in the domain of magnitudes M. One implication of this is that when a base dimension symbol is introduced there is an onus on the specifier to also introduce the associated standard unit. The extractor function val, which returns the value or magnitude of a measurement, is particularly useful in defining standard scales and conversions between scales.

5. A Toolkit of SI Units

As an example of the use of the extended Z syntax we present a toolkit for incorporating the SI standard dimensions and units into Z specifications. The base dimensions of measurement of the SI standard are [ISO79]

$$[\mathsf{L}, \mathsf{M}, \mathsf{T}, \mathsf{I}, \Theta, \mathsf{J}, \mathsf{N} : \mathsf{DIM}],$$

corresponding to the physical quantities

$$
\begin{aligned}
LENGTH &== \mathbb{R} \odot \mathsf{L} \\
MASS &== \mathbb{R} \odot \mathsf{M} \\
TIME &== \mathbb{R} \odot \mathsf{T} \\
CURRENT &== \mathbb{R} \odot \mathsf{I} \\
TEMPERATURE &== \mathbb{R} \odot \Theta \\
LIGHT\_INTENSITY &== \mathbb{R} \odot \mathsf{J} \\
AMOUNT &== \mathbb{R} \odot \mathsf{N}.
\end{aligned}
$$

5.1. Arithmetic Operators

In order to perform arithmetic on SI quantities we extend the real arithmetic operators to form an algebra of real-valued quantities. We assume that the real magnitudes and the operators on them have been defined in the usual way ([Val93] contains such a development in Z).

We start with unary negation. The dimensions of a negated quantity are the same as those of the quantity.

In the following definition this is specified by making the definition generic in dimensions U and requiring that the types of the argument and result of the negation both have dimensions U.

$$
\begin{array}{|l}
\hline\!\![\,U : \mathsf{DIM}\,]\!\!\rule{0pt}{0pt} \\
\;\;-\_ : (\mathbb{R} \odot U) \to (\mathbb{R} \odot U) \\
\hline
\;\;\forall\, a : \mathbb{R} \odot U\, \bullet \\
\qquad \mathsf{val}(-\,a) = -(\mathsf{val}(a)) \\
\hline
\end{array}
$$

Addition, subtraction, and comparison may only be performed on arguments with identical dimensions. Again the generic parameter U stands for the dimensions of the arguments and results.

$$
\begin{array}{|l}
\hline\!\![\,U : \mathsf{DIM}\,]\!\! \\
\;\;\_ < \_,\, \_ \le \_ : (\mathbb{R} \odot U) \leftrightarrow (\mathbb{R} \odot U) \\
\;\;\_ + \_,\, \_ - \_ : (\mathbb{R} \odot U) \times (\mathbb{R} \odot U) \to (\mathbb{R} \odot U) \\
\hline
\;\;\forall\, a, b : \mathbb{R} \odot U\, \bullet \\
\qquad a < b \Leftrightarrow \mathsf{val}(a) < \mathsf{val}(b)\, \wedge \\
\qquad a \le b \Leftrightarrow \mathsf{val}(a) \le \mathsf{val}(b)\, \wedge \\
\qquad \mathsf{val}(a + b) = \mathsf{val}(a) + \mathsf{val}(b)\, \wedge \\
\qquad \mathsf{val}(a - b) = \mathsf{val}(a) - \mathsf{val}(b) \\
\hline
\end{array}
$$

Multiplication and division compose the dimensions of their arguments. Here we introduce two generic dimensions, U and V, to stand for the dimensions of the first and second arguments, respectively. The dimensions of the result of a multiplication are $U \cdot V$, and the dimensions of the result of a division are $U \cdot V^{-1}$.

$$
\begin{array}{|l}
\hline\!\![\,U, V : \mathsf{DIM}\,]\!\! \\
\;\;\_ * \_ : (\mathbb{R} \odot U) \times (\mathbb{R} \odot V) \to (\mathbb{R} \odot U \cdot V) \\
\;\;\_ / \_ : (\mathbb{R} \odot U) \times ((\mathbb{R} \setminus \{0\}) \odot V) \to (\mathbb{R} \odot U \cdot V^{-1} \\
\hline
\;\;\forall\, a : \mathbb{R} \odot U;\ b : \mathbb{R} \odot V\, \bullet \\
\qquad \mathsf{val}(a * b) = \mathsf{val}(a) * \mathsf{val}(b)\, \wedge \\
\qquad \mathsf{val}(b) \ne 0 \Rightarrow \mathsf{val}(a\,/\,b) = \mathsf{val}(a)/\,\mathsf{val}(b) \\
\hline
\end{array}
$$

Other arithmetic operators may be introduced as required through similar definitions.

It should be noted that some arithmetic operators, such as exponentiation, cannot be easily extended to measurements in their full generality if we are to support static dimensions checking. Using the model given in Section 3 an exponentiation operator $(\_ \uparrow \_)$ can be defined as follows.

$$
\begin{array}{|l}
\hline
\;\;\_ \uparrow \_ : MEAS[\mathbb{R}] \times MEAS[\mathbb{R}] \to MEAS[\mathbb{R}] \\
\hline
\;\;\forall\, x : \mathbb{R};\ n : \mathbb{Z}\, \bullet \\
\qquad \mathsf{val}(x \uparrow n) = \mathsf{val}(x) \uparrow n\, \wedge \\
\qquad \mathsf{dim}(x \uparrow n) = (\mathsf{dim}\ x) \Uparrow n \\
\hline
\end{array}
$$

Unfortunately, there is no way of statically determining the dimensions of the result in general because the exponent may be an arbitrary expression. For the same reason we cannot give a definition using generic dimensions similar to those for addition and multiplication above. For a given constant exponent it is possible to define the dimensions using the generic mechanism. For example,

$$
\begin{array}{|l}
\hline\!\![\,U : \mathsf{DIM}\,]\!\! \\
\;\;square : \mathbb{R} \odot U \to \mathbb{R} \odot U \odot U \\
\hline
\;\;\forall\, x : \mathbb{R} \odot U\, \bullet \\
\qquad \mathsf{val}(square(x)) = \mathsf{val}(x) * \mathsf{val}(x) \\
\hline
\end{array}
$$

While it is possible to define functions corresponding to particular constant exponents, a better approach would be to recognise exponentiation as a special case when the exponent is a constant and handle it explicitly in type checking.

5.2. Scales and Unit Symbols

In the physical sciences, the standard practice for expressing measurements made on a particular scale is to annotate the recorded value with a special symbol associated with that scale, the *unit symbol*. The importance of the unit symbol is to tell us what has been measured and how it has been measured, that we may understand the measurement in terms of the quantity being measured. When something is reported to be $15\,\mathsf{cm}$ long we are able to spread out our hand and say, "It was about this long." In an abstract sense the unit symbol maps a measurement to the quantity that was measured. One way of modelling a unit symbol is thus as a function from a value to a magnitude-dimension pair, ie

$$\mathbb{R} \to \mathbb{R} \odot U,$$

where U is the dimensions of the quantity being measured. For example, assuming the postfix operator $\_\mathsf{K}$ (defined below) takes a real and returns a temperature in kelvins, the Celsius and Fahrenheit scales can be introduced as follows.

$$
\begin{array}{|l}
\_^{\circ}\mathsf{C} : \mathbb{R} \rightarrowtail \mathbb{R} \odot \Theta \\
\_^{\circ}\mathsf{F} : \mathbb{R} \rightarrowtail \mathbb{R} \odot \Theta \\
\hline
\forall\, x : \mathbb{R} \bullet \\
\qquad x^{\circ}\mathsf{C} = (x + 273.15)\,\mathsf{K} \,\wedge \\
\qquad x^{\circ}\mathsf{F} = ((x - 32) * 5/9)^{\circ}\mathsf{C}
\end{array}
$$

A Generalisation. In order to conveniently support the usual conventions for expressing derived quantities and measurement scales, we adopt a generalisation of this model. Consider a particle moving through space. We would like to be able to construct an expression of the particle's velocity in three ways:

- $x\,\mathsf{m}\_\mathsf{s}^{-1}$ where x is a variable of type $\mathbb{R}$,

- y_s^{-1} where y is a variable of type $\mathbb{R} \odot \mathsf{L}$, and

- v where v is a variable of type $\mathbb{R} \odot \mathsf{L} \odot \mathsf{T}^{-1}$.

Using the above model of unit symbols only the first and last would be possible.

Instead, we can model a unit symbol as a family of functions generic in a dimensional parameter. For example, the standard length unit m is modeled by a family of functions, one for each dimension symbol U, each of type

$$\mathbb{R} \odot U \to \mathbb{R} \odot U \odot \mathsf{L}.$$

The basic structure we introduce in support of the various scales in the SI standard is the *similarity* function. The similarity functions are those functions which may be associated with real-valued scales in some constant ratio to the standard scale. Recall that scales in ratio relationship are called similar scales.

$$
\begin{aligned}
\mathbb{R} \odot U \overset{\mathsf{s}}{\rightarrowtail} \mathbb{R} \odot V ==& \\
\{f : (\mathbb{R} \odot U) &\rightarrowtail (\mathbb{R} \odot V) \mid \\
(\exists\, \alpha : \mathbb{R} &\bullet \alpha \neq 0 \,\wedge \\
(\forall\, m : \mathbb{R} &\odot U \bullet \mathsf{val}(f(m)) = \alpha * \mathsf{val}(m)))\}
\end{aligned}
$$

Thus $\mathbb{R} \odot U \overset{\mathsf{s}}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{L}$ is the set of ratio conversions between $\mathbb{R} \odot U$ and $\mathbb{R} \odot (U \cdot \mathsf{L})$ and it is the appropriate (generic) type for length scales and their associated unit symbols. We insist that α is non-zero to ensure each similarity function is one-to-one and hence has an inverse.

Each unit symbol associated with a scale similar to the standard scale has an associated similarity function. Quantity constants are constructed by applying unit symbols (similarity functions) to magnitude constants. In order to fully define a unit symbol we need to assign it a type using $\mathbb{R} \odot U \overset{\mathsf{s}}{\rightarrowtail} \mathbb{R} \odot V$ and specify the scale function that it represents. We begin by introducing the SI standard unit symbols as postfix operators.

$$\begin{array}{|l}
\hline [U : \mathsf{DIM}] \\
\hline
\_\mathsf{m} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{L} \\
\_\mathsf{kg} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{M} \\
\_\mathsf{s} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{T} \\
\_\mathsf{A} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{I} \\
\_\mathsf{K} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \Theta \\
\_\mathsf{cd} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{J} \\
\_\mathsf{mol} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{N} \\
\hline
\forall x : \mathbb{R} \odot U \bullet \\
\qquad \mathsf{val}(x\,\mathsf{m}) = \mathsf{val}(x) \wedge \qquad \mathsf{val}(x\,\mathsf{kg}) = \mathsf{val}(x) \wedge \\
\qquad \mathsf{val}(x\,\mathsf{s}) = \mathsf{val}(x) \wedge \qquad \mathsf{val}(x\,\mathsf{A}) = \mathsf{val}(x) \wedge \\
\qquad \mathsf{val}(x\,\mathsf{K}) = \mathsf{val}(x) \wedge \qquad \mathsf{val}(x\,\mathsf{cd}) = \mathsf{val}(x) \wedge \\
\qquad \mathsf{val}(x\,\mathsf{mol}) = \mathsf{val}(x) \\
\hline
\end{array}$$

For example, the unit symbol $\_\mathsf{m}$ adds a length dimension to a measurement without changing its magnitude, since it is the standard unit of length measure and we have identified the quantities with their measurements on the standard scale. The syntactic sugar $\_\mathsf{m}$ indicates that the symbol is to be used as a postfix function symbol, allowing us to write length measurements in the standard manner, eg $2\,\mathsf{m}$ indicates 2 unit lengths on the metre scale.

Other units of measure, such as the imperial yard and foot, can then be specified by describing the relationship between the scale they represent and the standard scale or other defined scales.

$$\begin{array}{|l}
\hline [U : \mathsf{DIM}] \\
\hline
\_\mathsf{yd}, \_\mathsf{ft} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{L} \\
\hline
\forall x : \mathbb{R} \odot U \bullet \\
\qquad x\,\mathsf{yd} = 0.9144 * x\,\mathsf{m} \wedge \\
\qquad x\,\mathsf{ft} = x\,\mathsf{yd}\,/3 \\
\hline
\end{array}$$

Since each unit symbol represents a linear mapping between quantities they each have an inverse. Hence we can raise such a symbol to any integer power to construct a new unit symbol. The raising to a power is the standard Z iteration of a function. Thus each unit symbol, u, has a corresponding inverse unit symbol, u^{-1}, which adds the inverse dimension of measurement to the original symbol. For example, using the inverse of the $\_\mathsf{s}$ constructor we can write a velocity constant such as $3\,\mathsf{m}\,\mathsf{s}^{-1}$. In order to use s^{-1} as a postfix symbol we assume that an iterated symbol inherits the syntactic class of the original symbol, so that an iterated postfix symbol remains a postfix symbol. As in the case of the exponential operator it is necessary for the typing system to give special treatment of iterations of unit symbols to constant powers.

The SI units standard allows for the prefixing of unit symbols with scaling factors such as *kilo* and *micro*. We model these as operators on unit constructors. Because we want the result of one of these operators applied to a postfix symbol to remain a postfix symbol we treat such symbols as a special syntactic class with this property.

$$\begin{array}{|l}
\hline [U, V : \mathsf{DIM}] \\
\hline
\mathsf{k}, \mu : (\mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot V) \rightarrowtail (\mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot V) \\
\hline
\forall \_\mathsf{u} : \mathbb{R} \odot U \stackrel{s}{\rightarrowtail} \mathbb{R} \odot V;\ x : \mathbb{R} \odot U \bullet \\
\qquad \mathsf{val}(x\,\mathsf{ku}) = 1000 * \mathsf{val}(x\,\mathsf{u}) \wedge \\
\qquad \mathsf{val}(x\,\mu\,\mathsf{u}) = \mathsf{val}(x\,\mathsf{u})\,/\,1000000 \\
\hline
\end{array}$$

By using the basic unit symbol and the inverse and scaling operators we are able to build up a library of derived units symbols, such as force (in newtons).

$$\begin{array}{|l}
\hline [U : \mathsf{DIM}] \\
\hline (\_\mathsf{N}) : \mathbb{R} \odot U \overset{s}{\rightarrowtail} \mathbb{R} \odot U \cdot \mathsf{L} \cdot \mathsf{M} \cdot \mathsf{T}^{-2} \\
\hline \forall\, x : \mathbb{R} \odot U \bullet \\
\qquad x\,\mathsf{N} = x\,\mathsf{m}\,\mathsf{kg}\,\mathsf{s}^{-2} \\
\hline
\end{array}$$

6. Conclusion

As in the exposition and analysis of physical systems, a system of dimensions can be of great value in the specification and analysis of computational systems. The use of units and dimensions adds expressive power to a specification language and also offers another avenue for the application of automated redundancy checks on a specification.

In this paper we have introduced a language for dealing with units, dimensions and quantities which is suitable for use in conjunction with the specification language Z. This language includes notations for the declaration of variables with dimensions, the construction of constants of quantity through the use of unit symbols, and for the introduction of user-defined quantity constructors and operators. In Appendix A we have demonstrated that this system of dimensions is amenable to automated redundancy checking by an extension of the standard type checking technology.

In the approach we have presented, aspects of measurement theory such as non-integral dimensions and logarithmic scales of measurement are not handled. Further extension would be required to include such features.

We lay no claim as to the originality of the idea of dimensions of measurement. The aim of this paper has been to demonstrate that dimensions of measurement can provide a useful adjunct to static typing mechanisms in computing as they do in the physical sciences and engineering. Our own use of dimensions has convinced us that they are a worthy notational device as they

- allow and encourage the expression of additional information in the specification that can be crucial to its interpretation;
- make specifications more readable;
- allow the detection of more errors mechanically than a simple typing mechanism; and
- offer some insight into the structure of specifications through the application of dimensional analysis.

We foresee no reason why dimensions of measurement could not be added to other specification or programming languages.

Acknowledgements

References

[Bri31]   P. W. Bridgman. *Dimensional Analysis*. Yale University Press, revised edition, 1931.

[DG90]   Norman Delisle and David Garlan. A formal specification of an oscilloscope. *IEEE Software*, 7(5):29–36, September 1990.

[Ell66]   B. Ellis. *Basic Concepts in Measurement*. Cambridge University Press, 1966.

[GW88]   J. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-88-9, SRI International, Computer Science Lab, August 1988.

[Hay89]   I. J. Hayes. A generalisation of bags in Z. In J. E. Nicholls, editor, *Proceedings of the Z User Meeting*, Workshops in Computing, pages 113–127. Springer-Verlag, December 1989.

[Hay90]   I. J. Hayes. Specifying physical limitations: A case study of an oscilloscope. Technical report 167, Department of Computer Science, University of Queensland, July 1990.

[Hay93]   I. J. Hayes, editor. *Specification Case Studies*. Prentice Hall International, second edition, 1993.

[ISO79]   International Organization for Standardization, Geneva. *Units of measurement: handbook on international standards for units of measurement*, 1979.

[Jon90] C. B. Jones. *Systematic Software Development Using VDM.* Prentice Hall International, second edition, 1990.

[MH92] B. P. Mahony and I. J. Hayes. A case-study in timed refinement: A mine pump. *IEEE Transactions on Software Engineering*, 18(9), September 1992.

[MMH93] B. P. Mahony, C. Millerchip, and I. J. Hayes. A boiler control system: An overview. In *International Invitational Workshop - Design and Review of Software Controlled Safety-Related Systems*, Ottawa, June 1993.

[Spi88] J. M. Spivey. *Understanding Z: A Specification Language and its Formal Semantics*, volume 3 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1988.

[Spi92] J. M. Spivey. *The Z Notation: A Reference Manual.* Prentice Hall International, second edition, 1992.

[Val93] S. Valentine. Putting numbers into the mathematical toolkit. In J. P. Bowen and J. E. Nicholls, editors, *Z User Workshop: Proceedings of the Seventh Annual Z User Meeting, London, December 1992*, Workshops in Computing, pages 9–36. Springer-Verlag, 1993.

A. Static Dimensional Analysis

In order to facilitate the static checking of dimensions of measurement we need only augment the standard Z type checking mechanism with dimensions information. As a starting point we take the standard Z type-checking mechanism described in [Spi88, p27]. Each object

$$[NAME]$$

is assigned a typing tag from the set

$$
\begin{aligned}
TYPE ::= \ & givenT \langle\!\langle NAME \rangle\!\rangle \\
& |\ powerT \langle\!\langle TYPE \rangle\!\rangle \\
& |\ tupleT \langle\!\langle \text{seq } TYPE \rangle\!\rangle \\
& |\ schemaT \langle\!\langle NAME \nrightarrow TYPE \rangle\!\rangle.
\end{aligned}
$$

In our dimensioned Z, each standard object may have a dimension as well as a type structure. The base dimensions are represented by identifiers.

$$BASE == NAME$$

and, as in our model in Section 3, the dimensions are represented as a mapping from base dimensions (names) to non-zero integers.

$$DIM == BASE \nrightarrow (\mathbb{Z} \setminus \{0\}).$$

For static dimensional analysis we must store this dimensional information with the typing information. We add types representing quantities and dimensions.

$$
\begin{aligned}
DTYPE ::= \ & givenDT \langle\!\langle NAME \rangle\!\rangle \\
& |\ powerDT \langle\!\langle DTYPE \rangle\!\rangle \\
& |\ tupleDT \langle\!\langle \text{seq } DTYPE \rangle\!\rangle \\
& |\ schemaDT \langle\!\langle NAME \nrightarrow DTYPE \rangle\!\rangle \\
& |\ quantDT \langle\!\langle DTYPE \times DIM \rangle\!\rangle \\
& |\ dimDT \langle\!\langle DIM \rangle\!\rangle
\end{aligned}
$$

Standard objects have a type from one of given, power, tuple, schema, or measurement. We do not include dimensions as a standard type. Dimensions may only appear as a part of a quantity type expression or in the definition of a dimension symbol. This rules out such objects as sets of dimensions.

$$
\begin{aligned}
& standardT : \mathbb{P}\, DTYPE \\
& \rule{4cm}{0.4pt} \\
& standardT = givenDT (\!| \ NAME \ |\!) \cup \\
& \qquad\qquad powerDT (\!| \ standardT \ |\!) \cup \\
& \qquad\qquad tupleDT (\!| \ \text{seq } standardT \ |\!) \cup \\
& \qquad\qquad schemaDT (\!| \ NAME \nrightarrow standardT \ |\!) \\
& \qquad\qquad quantDT (\!| \ standardT \times DIM \ |\!)
\end{aligned}
$$

The process of type-checking a specification consists of building up a type environment

$$ENV == NAME \nrightarrow (standardT \cup dimDT (\!| \ DIM \ |\!))$$

which determines the type of all the names introduced in the specification and using this environment to determine the types of the more complex expressions which appear in the specification. We write, $\rho \vdash exp{::}DT$, to indicate that the expression exp has dimensioned type DT in the environment ρ.

The dimensioned type of each expression is built up recursively. As a representative sample we consider variables, tuples, sets, function applications, power sets, cross products, and the addition of dimensions. These rules are essentially the same as those given by Spivey [Spi88] with additional rules to handle dimensions.

Given sets

If a name N is introduced as a given set or a set parameter to a generic definition then $\rho \vdash N ::$ $powerDT(givenDT(N))$.

Base units

If a name B is introduced as a base unit or a unit parameter to a generic definition then $\rho \vdash B ::$ $dimDT(\{B \mapsto 1\})$.

Variables

If a variable v is declared of type T and $\rho \vdash T :: powerDT(DT)$ then
then $\rho \vdash v :: DT$.

Tuples

If $\rho \vdash x_1 :: DT_1, \ldots, x_n :: DT_n$,
then $\rho \vdash (x_1, \ldots, x_n) :: tupleDT(\langle DT_1, \ldots, DT_n \rangle)$.

Schema comprehension

If $\rho \vdash SCHEMA :: schemaDT(\{v_1 \mapsto DT_1, \ldots, v_n \mapsto DT_n\})$
and $\rho \cup \{v_1 \mapsto DT_1, \ldots, v_n \mapsto DT_n\} \vdash x :: DT$,
then $\rho \vdash \{SCHEMA \bullet x\} :: powerDT(DT)$.

Function application

If $\rho \vdash f :: powerDT(tupleDT(DT_1, DT_2))$
and $\rho \vdash x :: DT_1$,
then $\rho \vdash f(x) :: DT_2$.

Power sets

If $\rho \vdash S :: powerDT(DT)$,
then $\rho \vdash \mathbb{P}\, S :: powerDT(powerDT(DT))$.

Cross products

If $\rho \vdash S_1 :: powerDT(DT_1), S_2 :: powerDT(DT_2)$,
then $\rho \vdash S_1 \times S_2 :: powerDT(tupleDT(\langle DT_1, DT_2 \rangle))$.

Composing units

If $\rho \vdash D_1 :: dimDT(u_1), D_2 :: dimDT(u_2)$
then $\rho \vdash D_1 \cdot D_2 :: dimDT(u_1 \cdot u_2)$.

Units to a power

If $\rho \vdash D :: dimDT(u)$ and n is an integer constant
then $\rho \vdash D \Uparrow n :: dimDT(u \Uparrow n)$.

Adding units

If $\rho \vdash S :: powerDT(DT), D :: dimDT(u)$
then $\rho \vdash S \odot D :: powerDT(canonical(quantDT(DT, u)))$

where the function $canonical$ reduces quantity types to a (flattened) canonical form. For types which are not of the form $quantDT(DT, u)$, $canonical$ acts as the identity function, but for quantity types $canonical$ acts as follows.

$$canonical(quantDT(quantDT(DT, u_1), u_2))$$
$$= \left\{ \begin{array}{ll} canonical(DT), & \text{if } u_1 \cdot u_2 = \bot \\ canonical(quantDT(DT, u_1 \cdot u_2)), & \text{otherwise} \end{array} \right.$$