RC22964 (W0311-021) November 4, 2003
Computer Science

IBM Research Report

The Informal Language of Measure Expressions on the Web

Alan D. Marwick
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

The Informal Language of Measure Expressions on the Web

Alan D. Marwick
IBM T.J. Watson Research Center
Hawthorne, NY 10532
marwick@us.ibm.com

November 3, 2003

Abstract

A study of measure expressions occurring in web pages is presented. It is shown that the expressions can be automatically parsed and evaluated with an error rate of $<$10%. Quantitative statements in web pages, embodied in measure expressions, are an important source of information for the semantic web and other applications. The web also provides a source of measure expressions that allows the idioms commonly used for such expressions to be identified. We call the language used in these expressions the informal language of measure expressions, and the extent to which it diverges from the forms of expression sanctioned by standards bodies is described using the quantitative results of the analysis. The data used in the study consists of 5,165 expressions divided into development and test sets. The syntax of the expressions is discussed, and elements of a grammar that describes it are presented. Heuristics for dealing with ambiguities in the expressions are described, as are means for dealing with embedded HTML markup and other features of the data.

Contents

1 Introduction **1**

2 A Standard Form for Measure Expressions **3**

3 Methods **5**

3.1 Test Collection 5

3.2 The Measure Expression Analyzer 8

4 The Language of Measure Expressions **9**

4.1 Simple Expressions in the Informal Language 10

4.1.1 Operands 10

4.1.2 Operators 11

4.2 Sequence expressions 14

4.3 Grammar 14

5 Analyzing the Informal Language **16**

5.1 Name Ambiguity 17

5.1.1 Differing unit systems 17

5.1.2 Case of abbreviations 18

5.1.3 Prefix+unit collisions 18

5.1.4 Plurals of abbreviations 19

5.1.5 Case of prefixes 20

5.2 The Symbols of the Language 20

5.2.1 Special characters 20

5.2.2 Unit names and abbreviations 20

5.2.3 Numbers 21

5.2.4 Proper fractions and mixed numbers 22

5.2.5 Operators 22

5.2.6 Polysemous symbols 23

5.2.7 Sequence expressions 24

5.3 Input Transformation Rules 24

5.4 Evaluation of a Measure Expression 25

5.5 Error Rate of the Analyzer 26

6 Characteristics of Measure Expressions **27**

7 Discussion and Conclusions **33**

8 Acknowledgments **35**

9 Appendix **35**

A Annotation Policies **35**

1 Introduction

Quantitative information about the physical world is expressed in measure expressions. In their simplest form, measure expressions can consist simply of a number and an associated unit of measurement, for example “6 feet”, but more complex expressions containing several units as well as numbers and operators are often encountered, for example “20 liters per 100 km”. Measure expressions play a particularly important role in technical domains such as science and engineering, where quantification is of fundamental importance. A number of general and domain-specific standards specify how measure expressions should be constructed and represented in typeset documents (examples are [1, 2, 3]) and in flat text [4, 5]. Taken together, these in effect define a standard language for measure expressions. The standards specify the symbols used in this language, including the allowable names and abbreviations of units and prefixes. They also specify capitalization conventions, typographic representations, and the allowable syntax of the expressions.

In the present study, we focussed on measure expressions discovered in web pages written in HTML, a source of measure expressions that use the idioms that are in common use. Information about these idioms was derived from analysis of the expressions. It was found that the language used in measure expressions from the web is significantly different from that defined in the applicable standards. We call this language that is actually in use on the web the *informal language* of measure expressions, and the goals of this study were to describe it, to determine with what accuracy it can be analyzed, and to assess how it differs from the standard language.

Information about the informal language is expected to be of use in designing applications that must deal with measure expressions in their input. These include data integration [6], IT systems monitoring [7], information extraction from text [8, 9], question answering [10] and the semantic web [11], as well as numerous applications used in all branches of the physical sciences and engineering. To facilitate these applications it is clearly desirable to be able to accurately interpret measure expressions through automatic analysis of them.

The automatic analysis of measure expressions presents some challenges. First and most obvious, the surface form specified by the standards is designed to be read and understood by people, not computers. As a consequence, the standards emphasize the typographic details of the representation, so as to ensure

comprehensible and unambiguous representation of measure information on the printed page, in a browser, or in a flat text file. The more relaxed conventions of the informal language increase the range of possible forms of expression, and make automatic analysis even more difficult. For semantic web applications, another challenge is dealing with the variety of markup that is used in HTML to approximate to the standard representation when the page is rendered in a browser. In addition, all measure expressions convey quantitative information, which the analysis must extract. As we have already seen from the examples given, they can contain numbers, unit names and other symbols that have magnitudes, as well as operators. This indicates that their analysis must include some kind of evaluation, similar to that of an arithmetic expression or an expression in a programming language. Thus, an analysis system that can deal with measure expressions should accept as input a string representing an expression and should output numerical data containing the reulsts the evaluation, and perhaps also other information about the expression.

Overall, then, measure expressions in the informal language combine some of the characteristics of a natural language (designed for human comprehension; variety of expression; and, as we will see, ambiguity), and of computer languages (syntax is constrained; can be evaluated quantitatively). Some of the issues involved in automatic analysis of measure expressions have already emerged in previous work on applications that address the problem of conversion from one set of measurement units to another, a tedious and error-prone task which is common in science and engineering. While the majority of such applications are menu-driven, some can parse and analyze free-form input: measure expressions that are entered by users [12, 13]. It is clearly desirable that this kind of conversion application should accept and be able to process measure expressions that are written in a natural way, rather than require users to learn a special syntax Such applications therefore are developed to support a broad range of input syntax. But, how many different forms of input should they support, and of what sort?

Previous work on information extraction from text has also dealt with measure expressions. Information extraction consists of three steps: recognition of a measure expression in text; analysis of the expression (the subject of this paper); and assignment of class labels and other annotations to its mention in the text. The latter step bridges to the ideas of the semantic web, for which the semantics of all kinds of information on web pages must be specified with respect to ontologies. In the work done to date on measure expressions the ontology has been fairly simple. In the Message Understanding Conference (MUC) named

entity task [14], the NUMEX class included only money amounts and percentages. the Tipster annotation architecture [15] recognized the following classes of Measure annotation: WEIGHT, LENGTH, COUNT, AREA, VOLUME, CURRENCY, TEMPERATURE, and PERCENT. In the PIQUANT question-answering system [16, 17], text is annotated with 95 semantic categories [18], among which are MONEY, POWER, LENGTH, AREA, VOLUME, TEMPER (temperature) and WEIGHT. Ultimately, information extraction systems, especially those contributing to the semantic web, will have to annotate with respect to much more complete ontologies of quantification and measure, such as the Ontology of Engineering Mathematics [19], or the revisions suggested by Pinto and Martins [20] to the draft IEEE Upper Ontology (the Standard Upper Ontology, SUO [21]). They suggested as examples 34 named subclasses of units of measure to replace the nine originally in SUO. Other named subclasses exist, and the number of subclasses is essentially unbounded. The difficulty of the classification step will be considerably increased in these more complex ontologies, which again highlights the desirability of accurate analysis.

The structure of this paper is as follows. First, we discuss the numerical representation of measure expressions that is the result of analysis, the test collections gathered from the web and the system used to analyze them. Next, we present a description of the informal language as it is used in the data, and how it differs from the standards, followed by a discussion of how it can be analyzed. We show that the analysis can be done with an error rate of $<$10%, and we describe the characteristics of the measure expressions in the data deduced from the analysis. Finally, we discuss the results.

2 A Standard Form for Measure Expressions

Evaluation of a measure expression produces a numerical result that includes information about the magnitude of the expression and also a specification of what it measures. We call this result the *value* of the expression. Following much previous work, for example [22, 23], the value of a measure expression is characterized by scalar magnitude (called here its *size*), and an associated ordered tuple of integers called here its *dimensions*:

$$E = \{s, D\}. \tag{1}$$

Here E is an expression, s is its size, and D its dimensions:

$$D = \{d_i\}. \tag{2}$$

The d_i are the exponents of a set of base quantities to which the expression can be reduced through analysis. It is a basic results of dimensional analysis (see for example [1, 24]) that any physical measurement can be expressed by Equations 1 and 2 with an appropriate choice of base quantities. In the SI system, the seven base quantities are mass, length, time, electric current, thermodynamic temperature, amount of substance, and luminous intensity. In the present work the nine base quantities used were the SI base quantities plus plane angle and solid angle, which were formerly known as supplementary quantities in the SI system. The SI units of these base quantities are kilogram, meter, second, ampere, kelvin, mole, candela, radian and steradian, and the d_i are therefore the exponents of these units when the measure expression E is expressed in terms of them, i.e.

$$\{d_i\} \equiv \{\text{kilogram}^{d_1}, \text{meter}^{d_2}, \text{second}^{d_3}, ..., \text{steradian}^{d_9}\}. \tag{3}$$

Thus, for example, the measure expression "25 meters per second" in terms of Equations 1 and 2 is:

$$25 \text{ meters per second} = 25 \, \frac{\text{meter}}{\text{second}} = \{25, \{0, 1, -1, 0, ..., 0\}\} \tag{4}$$

Here, the exponent of mass is zero, the exponent of *length* (whose unit here is meter) is 1, and the exponent of *time* (whose unit is second) is -1.

The choice of what size to use in Equation 1 is made as follows. It is convenient to use the convention that the size of a simple unit is the conversion factor from that unit to the equivalent SI unit. For example, the unit *inch* has the size 0.0254, since one inch equals 2.54 centimeters or 0.0254 meters. Thus by Equation 1 we have

$$E(\text{inch}) = \{0.0254, \{0, 1, 0, ..., 0\}\} \tag{5}$$

In this way a simple unit can be expressed as a measure expression. The size of a number is its magnitude, with zero dimensions, i.e. $d_i = 0 \, \forall i$. Evaluation of a measure expression determines the size and dimensions

of the value of the expression by applying the operations in Table 1 to the operands within the expression, each of which has the form of Equation 1.

Table 1: Operations defined for measure expressions

Name	Symbol	Definition
Multiplication	$*$	$U_1 * U_2 = \{s_1 s_2, \{d_{1i} + d_{2i}\}\}$
Division	/	$U_1 / U_2 = \{s_1 / s_2, \{d_{1i} - d_{2i}\}\}$
Exponentiation	^	$U^p = \{s^p, \{d_i + p\}\}$
Equality	=	$U_1 = U_2 \iff s_1 = s_2 \; and \; d_{1i} = d_{2i} \; \forall i$

3 Methods

For this investigation we used a test collection of several thousand measure expressions that were manually extracted from web pages, and an expression analyzer that was used to parse and evaluate the expressions.

3.1 Test Collection

Web pages likely to contain measure expressions were collected from the Internet by issuing queries to a set of Internet search engines[1] and downloading the pages returned as search results. The queries are shown in Table 2. They included the names of some units of measure that are likely to be common in technical domains, as well as some like gallon and pound that were expected to be fairly common in ordinary documents.

Table 2: Search queries used to retrieve test data

	Query	Pages returned
1	+coulomb +second	583
2	+pascal +newton	1121
3	+farad +ohm	1169
4	+rad +joule	1098
5	+watt +erg	1178
6	+gallon +pound	914
7	+dyne +gram	1135
8	+mole +millilitre	964

[1] In May 2002.

After some filtering to remove PDF, postscript and other formats, 4,207 HTML files were retrieved. From these, 100 documents were randomly selected and further divided into a development test collection and an unseen test collection, each of 50 documents. After some furthering filtering (for example to remove pages that were found to be listings of source code), these test collections had the characteristics summarized in Table 3.

Table 3: Test collections

Collection	Pages	Measure Expressions
Development	47	1731
Unseen	49	3434
Combined	96	5165

Measure expressions occurring in the pages in the test collections were manually identified and recorded as standoff annotations of the HTML text of the pages by using the Gate tool [25]. The annotation policies are described in the Appendix. The numbers of measure expressions in the test collections are shown in Table 3, and the distribution of their frequency in the web pages is plotted in Figure 1. The figure shows that about a third of the web pages in each collection contained no measure expressions. These included, for example, pages that were biographies of Blaise Pascal or Isaac Newton (c.f. Table 2). At the other extreme, some pages contained hundreds of measure expressions. Some of these pages were tables of conversion factors. It might be thought that such pages were produced by professional scientists or engineers who are trained in the standards of representation of measure expressions and that therefore these pages would not be representative of the most common idioms used by untrained authors. However inspection of the pages showed that many, perhaps most, contained many eccentricities, including errors in the conversion factors and non-standard syntax, that indicated that their authors were not professionally trained. From inspection, it appeared that such pages included many idioms characteristic of pages containing fewer expressions. An example of a page that contained many measure expressions but was not a table of conversion factors was a transcript of bulletin board postings by many different builders of models of medieval seige engines. In their postings they compared information about the ranges, throw weights and dimensions of their creations. Web pages of recipes also contained many measure expressions.

To facilitate evaluation of the measure expression analyzer that is described in the next section,

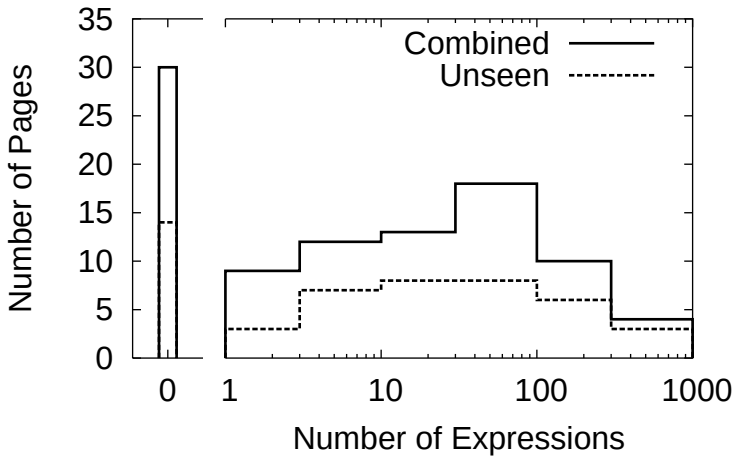

Figure 1: Logarithmic frequency distribution of measure expressions in web pages in the combined and unseen test collections. The measure expressions were found by manual annotation.

ground truth was added to a feature structure associated with each annotation. The truth was in the form of a text string containing an expression that was equivalent to the annotated expression, in that it used the same units and had the same syntactic structure, but which was known to be parsable, and contained more standard language. Thus for example, if the annotated expression was "`btu's/min.`" then the truth expression was "`BTU/minute`", i.e. without the plural, the abbreviation or the trailing period. In measuring the accuracy of the expression analyzer, the truth expression and the measure expression from the data were both analyzed, and the results compared. If either analysis failed, or if the comparison indicated that the results did not satisfy the equality operation of Table 1, then the analysis was judged to have been in error.

The truth expressions were entered manually, but to assist the human annotator an application was used to enter them that used the expression analyzer described in the next section. The analyzer first attempted to analyze the measure expression in the data. If this analysis was successful, the expression recreation feature of the analyzer was used to propose an equivalent expression to the annotator for use as the truth expression. If the annotator rejected the proposal, or if the analysis failed, the annotator entered the correct truth expression. To help to determine the intent of the author of the expression, the annotator was able to view it in its original context in the HTML of the web page. A normalized form of the original expression was also added to the annotation; the normalization consisted only of replacing new line characters

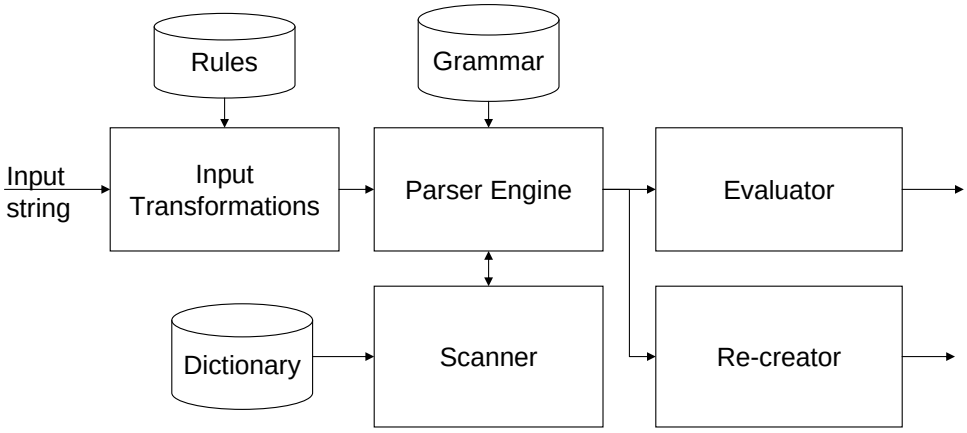

Figure 2: Block diagram of the Measure Expression Analyzer

with spaces. It was this normalized form that was analyzed during the evaluation of the expression analyzer.

3.2 The Measure Expression Analyzer

The measure expression analyzer accepts as input a string representing a measure expression, and returns the value of the expression that was described earlier in Equation 1. Optionally, the analyzer can return other information, such as the parse tree for the expression, or a string containing a recreated version of the expression.

The architecture of the measure expression analyzer, which is implemented in Java, is shown in Figure 2. Before parsing, the input string is subjected to a series of transformations, discussed below, according to transformation rules. The transformed string is then passed to the parsing engine, which is a simple table-driven top-down parser of conventional design that is driven by a grammar. As is usual, the grammar consists of a series of productions, each of which names a single syntactic category on the left, and specifies a sequence of constituent categories on the right. Alternative productions in the grammar for a given non-terminal syntactic category are attempted consecutively in the order of their specification. When a syntactic category that is a terminal is encountered, the parser calls the scanner with the identifier of the category and the current position in the input string. If the terminal is found, the scanner advances the current position to the next character beyond it, and returns information about it (such as the unit) to the parser. The scanner incorporates a number of heuristics in its terminal checking routines, as described later.

This allows the grammar to be considerably smaller than would otherwise be the case.

If the scanner fails to find the terminal in the input at the specified position, the current production fails and the parser resets the current position to the value it had at the beginning of the production. Since the grammar is not left-factored, the same terminal is often sought at the same position in consecutive productions. The intrinsic inefficiency of this design is alleviated by using a cache in the scanner. If all possible productions are exhausted without matching the input then the parse fails.

The dictionary contains lexical information about units – their names, abbreviations and permitted variant forms – as well as semantic information, in particular the size and dimensions of the units as in Equations 1 and 2. It contained 476 unit names, variant names and abbreviations. The dictionary can also be modified by the analyzer by adding alternative names for existing units.

If a parse is successful, the parser outputs a parse tree in which each terminal is assigned a value according to Equation 1 as well as a syntactic category. An example of a parse tree is shown in Figure 3. The names of many of the categories used in this example are discussed in later sections of this paper. The evaluation module uses the information in the parse tree to determine the value of the expression. A separate module is optionally used to recreate a valid measure expression from the parse tree. This expression is created simply by doing a pre-order traversal of the parse tree, and outputting a string representation of each terminal when it is encountered. The syntax of the re-created expression allows it to be accurately re-parsed by the analyzer. This is the feature that was used to propose truth expressions during annotation of the test collections.

4 The Language of Measure Expressions

In this section we describe the informal language as it was used in the test data. The description evolved through the use of the development test set to refine the grammar, the dictionary, and the heuristics embedded in the analyzer. These were all improved over time in the development phase in such a way as to iteratively reduce the proportion of measure expressions which were in error, until further improvement was considered impractical. At the same time, the description of the language on which the analysis was based was refined.

4.1 Simple Expressions in the Informal Language

The data contained both simple expressions and more complex expressions such as range expressions and other types. The latter, which we call sequence expressions, are described in the next section. Simple expressions, like "100 acre-feet per year" can be reduced to a single value. Most expressions were of this type. Here we focus on the operators and operands in the grammar of simple expressions. These are the terminal symbols of the grammar.

Table 4: Operands in the syntax

Operand	Example	Description
BASE_UNIT_NAME	meter, m	The name or abbreviation of a unit
PREFIXUNIT	millimeter, mm	A unit name or abbreviation prefixed by a prefix name or abbreviation
PREFIX_SEQUENCE	mega, millimicro	A contiguous sequence of one or more prefix names spelled out in full
REAL	127, 12.75, 1.6E-10, $\frac{1}{2}$, $1\frac{3}{4}$	Real number
WORD_INT	one hundred and three	A spelled out integer
EXPON_INT	cm3	An integer suffix denoting exponentiation
FRACTION	quarters, half,	Spelled-out fractional numbers
OFA	of a, an	Optional appendage to a fraction
LOG_RATIO	bel, dB	A dimensionless unit denoting the logarithm of a power or voltage ratio
EXP_WORD	squared, cubic	A word that denotes an integer in the context of exponentiation

4.1.1 Operands

The operands required to fully describe the informal language of measure expressions are listed in Table 4. We now present the rationale for the choice of this set of operands.

Occurrences of unprefixed unit names or their abbreviations were represented by the operand whose syntactic category was BASE_UNIT_NAME. A separate operand, PREFIXUNIT, was used to identify names or abbreviations with a contiguous prefix in order to allow the scanner to consider the prefix and the unit together. By doing so it could apply a number of heuristics used to resolve ambiguities involving prefix+unit combinations, as further discussed below. Prefixes in the test data were sometimes separated

from the unit name by a space. A different terminal category, PREFIX_SEQUENCE was used for these cases, for reasons described below when the operator PREFIX_MULT is discussed.

Real numbers occur in most measure expressions. The REAL operand is used both for numbers that are expressed numerically and for numbers which include proper fractions that are represented in text by special Unicode characters, and in a browser by glyphs such as "$\frac{1}{2}$". Spelled-out numbers are not allowed in the standard language, but were fairly common in the test data. Of these, spelled-out integers present no particular problem and are represented by the single operand WORD_INT. A sub-grammar for these numbers was not needed as they can be parsed by components in standard software libraries [26]. However, the data also contained spelled out numbers with a fractional part, for example "half a meter", "three and seven eights", "12-and-a-half", "meter and a half" and so forth. To describe the syntax of these forms in the grammar it was found necessary to define the operand FRACTION as well as the special operators ANDA and MULT_ANDA (see Table 5).

Exponentiation of unit names in the standard language is allowed though constructs like "meters per second squared" and "cubic meters", which motivate the definition of the EXP_WORD operand. A special category of integer, EXPON_INT, is used to distinguish those integers that occur in expressions like "`cm3`" where they signify exponentiation (this is allowed in the standard language for applications where a limited character set is available [4, 5]). Lastly, a special category of operand, LOG_RATIO, will be discussed when the POST_ANTILOG operator is introduced.

4.1.2 Operators

The operators used in measure expressions are listed in Table 5. Explicit multiplication and division signs are assigned the categories MULT_SIGN and DIV_SIGN. Note that the period is used as a multiplication symbol. Because it plays several other syntactic roles as well, it can be described as being a polysemous symbol, as further discussed later. Whitespace is another polysemous symbol. Its role as an implied multiplication operator is represented by the syntactic category MULT_WHITESPACE in the grammar. This operator is allowed in all contexts where an explicit multiplication symbol is allowed. Although a separate syntactic category is not strictly required, it serves to make the role of whitespace explicit in the syntax.

Table 5: Operators in the syntax.

Name	Examples[a,b]	Description
MULT_SIGN	newton_meter, N_m, amps×volts	Multiplication
DIV_SIGN	miles÷hour, gm/cm3	Division
EXP_OPER	in^3, cm:3, hour⁻¹	Exponentiation
MULT_WHITESPACE	12 _ meters, three _ quarters	Multiplication implied by whitespace.
NO_WHITESPACE_MULT	1.25:meters, (kg):(meter/second)	Absence of whitespace implies multiplication
PREFIX_MULT	milli _ meters, kilo-watt	Multiplication between a prefix and a unit (see text).
MULT_WORD	meters times megahertz	Multiplication
DIV_WORD	miles per hour	Division
ADJ_EXPON	square _ centimeters	Exponentiation implied by whitespace
EXP_WHITESPACE	meters _ cubed	Exponentiation implied by whitespace
ANDA	three and a quarter	Addition of numbers
MULT_ANDA	meter and a half	$x\langle mult\_anda\rangle y = x(1+y)$
POST_ANTILOG	3 _ dB, +10:dB	Special for bels and decibels; see text
LEFT_PAREN, RIGHT_PAREN	litres/(100 kilometers)	Parentheses

a. The character(s) representing the operator are underlined.

b. The symbol : denotes the position of an implied operator in a string of contiguous characters.

When separating a PREFIX_SEQUENCE and a unit, as in “milli meters”, whitespace also represents multiplication but in this role is assigned the separate category PREFIX_MULT so that if the expression is re-created this operator can be elided, so that the prefix and the unit are concatenated for a more pleasing result. Another role of whitespace is to separate EXP_WORDs from units. In this case the whitespace represents the special exponentiation operators ADJ_EXPON and EXP_WHITESPACE, which apply where the EXP_WORD precedes or follows the unit, respectively.

Whitespace plays yet another syntactic role when it occurs between a number and a LOG_RATIO operand, as in expressions like “10 dB”. This follows from the special nature of units like the bel (B) or its prefixed form decibel (dB), which refer to the logarithm of a ratio of electrical powers (the neper refers to a ratio of two voltages). For example, if the magnitudes of two powers have the ratio 100, this is expressed as 2 B, or 20 dB. Conversely, a power ratio described as “3 dB” corresponds to the numerical factor $10^{0.3}$. Thus the whitespace between the number and a LOG_RATIO unit represents a binary operator, POST_ANTILOG, which is defined by the equation

$$r \ \langle post\_antilog \rangle \ l = 10^{r|l|} \tag{6}$$

where r is a real number, l is a LOG_RATIO unit, and $|l|$ signifies the Size of the LOG_RATIO operand as in Equation 1.

Yet another role for whitespace in measure expressions is as a thousands separator in numbers. This is handled by the scanner as part of its recognition of real numbers, as described below. Finally, if whitespace does not play any of the syntactic roles described above, it is ignored.

There are many cases in the test data of an operator being implied by the *absence* of whitespace in a certain syntactic context. The use of an integer suffix to a unit to imply exponentiation, as in “`cm3`”, has already been mentioned; in this case the presence of an EXP_OPER is implied by the integer suffix on a unit or a prefixed unit. Another example arises when the space between a number and a unit is omitted, as in “`10mm`”. The omission of the space is deprecated in the formal language but was found to be common in the data. The operator NO_WHITESPACE_MULT was used to describe these situations. The same operator was used for cases where parenthesized sub-expressions were concatenated without an explicit operator.

Finally, although addition as a binary operator in general is not allowed in the standard language,

it occurs in the data embedded in spelled out real numbers such as “one and a half”. The operator ANDA was used to describe these situations. Addition is also implied in mixed numbers like “$1\frac{1}{2}$”, but these constructs could be handled in the scanner rather than being made explicit in the grammar, as further discussed later. An interesting hybrid occurs in expressions like “inch and a half”, meaning 1.5 inches. The operator MULT_ANDA was defined as in Table 5 to handle this construction.

4.2 Sequence expressions

Certain composite measure expressions are sanctioned by the standards and occurred in the data. We call these *sequence expressions*. They take the form of a sequence of simple expressions separated by special symbols that indicate the type of the expression. The syntax of sequence expressions is described by the regular expression

$$\langle sequence\ expression\rangle := \langle simple\ expression\rangle(\langle separator\rangle\ \langle simple\ expression\rangle)+ \qquad (7)$$

To give the dimensions[2] of a three-dimensional object using the standard language an expression such as “12.5 mm $\times$ 12.5 mm $\times$ 50 mm” is used. We call this a *box dimension* expression. These, and the two-dimension analogue which we call a *sheet dimension* expression (e.g. “4×6 feet”) are quite common in the test data. In such expressions, the units must be of length, and the $\langle separator\rangle$ symbol is typically “$\times$” or an equivalent symbol. *Range expressions* like “4 to 6 weeks” are also part of the standard language and occurred in the data. Here any units are permissable. The separator symbol in the standard language is the word “to”, but in the informal language variations on the hyphen are more common, as in “10-15 mm”.

4.3 Grammar

The grammar that describes the syntactic structure of simple measure expressions in the informal language incorporates the terminals already described. In addition, it includes productions that ensure that the different syntactic roles of certain symbols, notably whitespace, are correctly identified during parsing. The syntactic structure of measure expressions generally resembles that of expressions in computer languages,

[2]This usage of “dimensions” is different from that in the discussion of Equation 1.

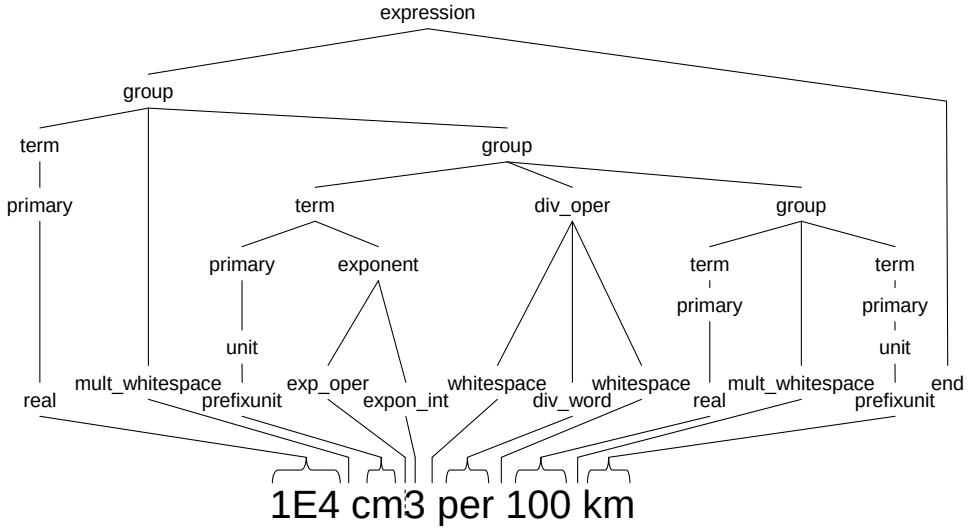

Figure 3: The parse tree of the expression “1E4 cm3 per 100 km”.

and the starting point for constructing the grammar followed standard approaches [27]. The final result is a context-free grammar (CFG) in the classification proposed by Chomsky. Currently the language it describes is non-regular. One reason is that it allows arbitrarily deep nesting of parentheses, though since the depth of nesting found in the data was limited, the grammar could be modified to avoid this. However there is the further question of whether the grammar could in principle be re-factored into a regular form given the ambiguity of many of the symbols used in the language. This is a topic for future research.

The grammar currently contains a total of 81 productions. Many of these are needed to constrain the syntactic context in which certain symbols are allowed to occur, in order that ambiguous symbols can be disambiguated. A much smaller grammar is required to describe the unambiguous language of measure units proposed by Schadow et al. [23]. Their grammar contains only 6 productions, involving 4 categories of terminal symbols. To make this possible, their language is even more constrained than the standard language.

An example of a production whose main role is to disambiguate a symbol, in this case whitespace, is

$$\langle term \rangle := \langle term \rangle \langle mult\_whitespace \rangle \langle term \rangle \tag{8}$$

which describes expressions like “5 inches”. Here, both “5” and “inches” parse as instances of the category $\langle term \rangle$, while the space between them is assigned the category $\langle mult\_whitespace \rangle$. Another example of a

production that disambiguates whitespace is

$$\langle term \rangle := \langle real \rangle \langle post\_antilog \rangle \langle log\_ratio \rangle \tag{9}$$

which matches input like "-3 dB". Here the "-3" parses to a $\langle real \rangle$, the "dB" parses to a $\langle log\_ratio \rangle$, and the space is recognized as the operator $\langle post\_antilog \rangle$.

Some productions of the grammar are in order to make an implied operator explicit. For example, consider the following fragment:

$$\begin{aligned} \langle term \rangle &:= \langle primary \rangle \langle exponent \rangle \\ \langle exponent \rangle &:= \langle exp\_oper \rangle \langle expon\_int \rangle \\ \langle exp\_oper \rangle &:= \wedge | \texttt{**} | \langle no\_whitespace \rangle. \end{aligned} \tag{10}$$

This ensures that an integer following a unit name without an intervening explicit operator can only signify exponentiation if there is no intervening whitespace, as in "`m3`" meaning cubic meters. If there is intervening whitespace, as in "`newton 12`", then the production in Equation 8 applies, in order that expressions like "`6 newton 12 meters`" should parse to be equivalent to "6 newton $\times$ 12 meters".

As a consequence of the large number of alternative productions in the grammar, back-tracking during parsing is common. In this respect the task of parsing measure expressions is similar to the task of parsing natural language. As already noted, the use of a cache in the scanner helps to overcome the potential inefficiency of the analysis.

5 Analyzing the Informal Language

While the grammar provides the basis for correctly parsing the syntactic structure of measure expressions, a full analysis that culminates in evaluation requires additional configuration of other components of the analyzer in order to deal with the surface representations of the expressions. Some relevant features of the dictionary, scanner and pre-processor will be described in this section, followed by a brief discussion of the

evaluation of the expressions. Finally, a measurement of the accuracy of the analyzer will be presented.

5.1 Name Ambiguity

Considerable ambiguity in the names of units and abbreviations, especially when combined with prefixes, was found in the test collection. Schadow et al. [23] point out that even in the standard language the names of units are ambiguous to some degree, and propose that the ambiguities be classified into five types. Additional ambiguities are introduced by the more relaxed conventions of the informal language. However, most ambiguities could be resolved by the application of certain rules in the scanner. These are summarized in Table 6, where the number of times the rule fired in parsing the combined test collection is also shown.

Table 6: Occurrence of name ambiguities in the combined test set.

Ambiguity	Count	Example
US/UK definition	322	gallon (US) vs. gallon (UK)
Case of abbreviations	867	S (siemens) vs. s (second)
Abbreviation collides with prefix+unit	1445	gm (gram) vs. Gm (gigameter)
Ambiguous plurals of certain abbreviations	145	ms (meters) vs. ms (millisecond)

5.1.1 Differing unit systems

Schadow et al.'s Type I name ambiguity arises when the same name symbol can refer to units in more than one system. A significant source of ambiguity of this kind, called *US/UK definition* in Table 6, is caused by the fact that the US and British systems of non-metric units include several, like "gallon", "pint", "ton" etc., whose magnitude in the two systems is different [28]. In a small number of cases the author provided a disambiguating qualifier, such as "gallons (UK)". However, in the great majority of cases this was not done, and the expression analyzer could not resolve the ambiguity from evidence in the expression alone. The strategy adopted to resolve ambiguities of this kind is to coerce the analyzer's interpretation of such unit names to be either the US or the British one if, for example, information that is extrinsic to the expression being analyzed is available. In the present work, evidence from outside the expression was not used, and it was assumed in annotating truth and in analysis that the US definition of such units was appropriate, unless

the author's intention was apparent and different.

A very similar situation arises for certain symbols used as abbreviations. For example "12′ " can mean 12 feet, or 12 minutes of time or 12 minutes of angle. Here again the ambiguity cannot be resolved with intrinsic information. In the analysis, such symbols were coerced to refer to units of length (foot and inch) by entering them as alternative names of these units in the dictionary.

5.1.2 Case of abbreviations

Another kind of Type I name ambiguity arises in the informal language when the abbreviations of two units are distinguished only by case, such as the standard abbreviation "a" (*annum*) for year, and the standard abbreviation "A" for ampere. These potentially collide in the informal language since case is usually not significant there. This ambiguity is called *case of abbreviations* in Table 6, and occurred frequently in the combined test collection. The most common example was "s" (second) which potentially collides with "S" (siemens). To resolve this ambiguity, the dictionary flags the abbreviations whose case must be respected in input.

5.1.3 Prefix+unit collisions

Several types of name ambiguities in Schadow et al.'s scheme arise when a prefix+unit combination collides with a unit name or abbreviation. In Type II ambiguities both units are metric, in Type III ambiguities the prefixed non-metric unit collides with the name of a non-metric unit, in Type IVa ambiguities it collides with a metric unit, and in Type IVb the name of a non-metric unit collides with the combination of a prefix with a metric unit (e.g. "nm" could be nautical mile or nanometer). In the standard language, the names of metric units and their standard abbreviations have been chosen so that Type II ambiguities can occur only when case-sensitivity is not enforced, but this is the case in the informal language. An additional source of Type II ambiguities in the informal language is that many authors use non-standard abbreviations for metric units. For example "gm" as an abbreviation for gram is common in the informal language, where in the absence of case enforcement is collides with "Gm" (gigameter). Furthermore, even in the standard language ambiguities arise when the names or abbreviations of non-metric units are involved, as is the case for the

"nm" example just quoted. Collisions of abbreviations with prefix+unit combinations were very common in the test collection, with the total of 1445 occurrences of all types (see Table 6).

A number of strategies were used to resolve prefix+unit name ambiguities. The first was to prefer an abbreviation of a unit to the colliding prefix+unit combination, i.e. to prefer a BASE_UNIT_NAME to a PREFIXUNIT. Thus "ft" was interpreted as foot rather than as femtotonne, and "min" as minute rather than milliinch. However, for some combinations a different strategy was used: the abbreviation was selected only if its case matched that of the corresponding entry in the dictionary. In these cases the dictionary contained a flag that indicated that the case of the entry had to be respected in order for it to be matched. This allowed "gm" (which was marked as case sensitive) to be interpreted as gram, but "Gm" to be interpreted as gigameter (the upper case "G" is the standard abbreviation of the prefix giga). The third strategy was simply to omit from the dictionary the unit abbreviation that collided with a common prefix+unit if the abbreviation occurred rarely. This was done in the case of "nm" as an abbreviation of nautical mile; this input was therefore interpreted by the analyzer as meaning nanometer. In another corpus nautical miles might be mentioned more often, and mentions of nanometers might be rare. In this case "nm" could be registered with the dictionary as a alternative name of nautical mile, with the result that this meaning would become the preferred one.

5.1.4 Plurals of abbreviations

Yet another source of naming ambiguity arises because in the plurals of abbreviations are common in the informal language, though they are not allowed in the standard language. It was found for example that many authors would write "`6 ins`" to mean six inches. Some plural abbreviations are ambiguous because they collide with prefix+unit combinations where the unit is "second", as in "ms" (meters or millisecond), "gs" (grams in the informal language, but gigasecond in the standard language without case enforcement), or "ts" (tons or terasecond). These mostly involve type II ambiguities since there are many metric units whose abbreviation is the same as that of a prefix, though a few of type III such as "hs" (hours in the informal language, but hectosecond in the standard language) are also possible. To avoid such cases, the abbreviations that cannot be suffixed by "s" to form plurals were flagged in the dictionary. The scanner's logic to respect this marking was invoked 145 times (see Table 6).

5.1.5 Case of prefixes

An additional potential source of ambiguity is introduced by the prefixes mega (M) and milli (m). In practice, authors successfully distinguished between the abbreviations by giving them the correct case. In annotating truth, the intention of the author was respected where it could be determined from the context. With this approach, only two expressions were found where case errors in these prefixes would have caused an equality error in the analysis, but it happened that they both occurred in expressions where parsing failed for other reasons.

Lastly, Type V ambiguities arise from the collision of prefix+non-metric unit combinations with prefix+metric unit combinations. There was no occurrence of a collision of this kind in the test data.

5.2 The Symbols of the Language

A striking feature of the informal language found in the test data is the range and variety of the surface forms of the terminals in the expressions. In many cases the authors of web pages were clearly trying to reproduce the typographic conventions recommended in the standards that define the standard language. In other cases, they used or invented new conventions. In analyzing the measure expressions in the data, a combination of approaches had to be used to deal with this.

5.2.1 Special characters

The authors used a number of approaches to represent special characters. In most cases they used HTML markup, including numeric and character entity references. These and other characters were all converted to Unicode in the analyzer's preprocessor.

5.2.2 Unit names and abbreviations

Multi-word unit names in the data were usually represented with embedded whitespace, as in "`foot pound`" or "`nautical mile`", but other variants, such as "`foot-pound`" and "`foot_pound`" were also found in the data. These were recognized during scanning for BASE_UNIT_NAME by appropriately

normalizing them in the scanner before looking them up in the dictionary. Specifically, if the string "`foot pounds seconds`" was encountered, this would first be normalized to "`foot_pounds_seconds`", but of course this would not be found in the dictionary, in which case the last token would be dropped, and "`foot_pound`" would be looked up, which would return a match. The same matching would be done for input such as "`foot-pound-seconds`". In this particular case it is important that the first hyphen should not be interpreted as multiplication, since "pound" represents the unit pounds (force), not a mass unit. The expression "`foot-pound-second`" must therefore be parsed as "`foot_pound` $\times$ `second`". By handling multi-word unit names as has been described, the correct parsing was achieved.

A number of non-standard prefix names occurred in the test data, such as "mk" for "micro", and "Meg" for "mega" (as in "`Meg. ohm`"). The prefix "micro" was also abbreviated as "u", as in "`um`" (micrometer), but this is part of the standard language that is allowed for applications using constrained character sets [4, 5]. Several special symbols and entities were used by authors to represent the Greek "μ" to mean the prefix "micro". All these variations were accommodated by adding them to the dictionary of prefixes used by the Analyzer.

5.2.3 Numbers

There was considerable variability in the way numbers in the data were expressed. As an example, the same number could be expressed in a number of ways, as "`1 234.567 89`", "`1 234,567 89`", "`1,234.56789`", "`1.23456789E3`", "`1,23456789x10<sup>3</sup>`" and so forth. These examples include the use of the space as a thousands separator (as recommended for the standard language), alternative uses of the comma as a thousands separator and as a decimal separator, and various approaches to writing powers of 10. The conventions for the use of the comma are locale-dependent, but in the development test data there were examples of numbers formatted according to different locales occurring in the same web page, for example where authors from different countries had contributed to a discussion database whose postings were rendered in a single HTML page. Where possible, therefore, the use of locale information was avoided. It was found possible in almost all cases to normalize numbers unambiguously into a single format in the preprocessor for subsequent parsing, eliminating nearly all need for the scanner to know the locale. The only exception was numbers like "`1,234`", where it cannot be determined by inspection whether the comma is

a decimal or a thousands separator. This form was interpreted according to the conventions of the locale specified to the Analyzer when it was initialized, which in this work meant that the comma in such numbers was interpreted as a thousands separator.

5.2.4 Proper fractions and mixed numbers

Proper fractions and mixed numbers were represented in a variety of ways. Certain Unicode characters represent proper fractions, and certain 8-bit ASCII characters also render in a browser as proper fractions. The test collection contained examples of both. The scanner for the REAL terminal used a list of these, and returned the corresponding real number in each case. Mixed numbers, consisting of an integer with a proper fraction, were written in a variety of ways in the data. For example, it was found that the number $1\frac{3}{8}$ might be written "`13/8`", "`1-3/8`" or "`1 3/8`". The first of these examples might be confused with "$\frac{13}{8}$", and given that the hyphen and the space in other contexts signify multiplication, the second and third might be confused with "$1 \times \frac{3}{8}$". The key to correct interpretation of the mixed numbers in the data was found to be the recognition that the authors expected certain combinations of digits and solidus to be interpreted as proper fractions whatever their context might be. Only certain combinations of numerator and denominator were used in this way, including "`1/4`", "`1/2`", "`3/4`", "`1/8`" and so forth. Most of these *canonical fractions* that were found in the data are in fact recognized in Unicode by being granted distinct character codes. They were handled in the Analyzer by being replaced by their corresponding Unicode characters, using the FRACTION rule described later. Then, the combination of the integer and the fraction character could be recognized by the REAL scanner, as already described. Other canonical fractions were subsequently discovered in the unseen test set; these caused evaluation errors since the scanner was not programmed to recognize them. Examples were "1/1000" and "1/3600".

5.2.5 Operators

The symbols for multiplication operators that are permitted in the standard language are the multiplication sign, the raised dot (as in "N·m"), and the period. In the test collection it was found that authors used a variety of markup to represent multiplication, including the HTML entity `&times;` and the letters "x" and

“X”. For the raised dot authors often used the HTML markup “[.]” or more rarely one of several HTML entities that render in a similar way. In addition to these explicit symbols, whitespace or its absence could also imply the presence of a multiplication operator as already discussed. Division operators in the standard language are the solidus “/” as in “5 m/s”, and the word “*per*”, as in “coulomb per kilogram”. In the informal language, authors also use the division sign “÷”, which they obtained from the HTML named entity ÷. The informal language also was found to include the use of the caret character “ ^ ” as an exponentiation operator. Superscript integers, which imply exponentiation, were represented in a number of ways. In many cases, HTML markup achieved the typographic effect, using the ^{...} element. The HTML entities ² and ³ also occurred, as did ASCII characters that also render as superscript 2 and 3 in a browser.

5.2.6 Polysemous symbols

A further feature of the informal language is the polysemy of certain symbols used in expressions. The varying roles of whitespace, and the parsing strategies to distinguish them, have already been discussed. Another polysemous character is the period, which was found to play several roles in the informal language in addition to those sanctioned by the formal language. These include its use as a multiplication symbol (sanctioned by the standard language only for output devices with limited capabilities, but common in the data) and as a decimal separator. The period is also used to terminate or to punctuate abbreviations, and to terminate expressions. In most cases, the scanner is able to distinguish between these roles by using the local context. Periods that terminate abbreviations or whole expressions, or which terminate the abbreviations of EXP_WORDs, as in “12 sq. feet”, were treated as whitespace by the scanner. It was found that periods and other punctuation occurred within the abbreviations of unit names in the unseen test data, as in “H/P” (horsepower), “h.p.” and even “h.p”. The scanner was not able to handle these forms, which as a consequence were wrongly parsed. Lastly, while periods can be used as thousands separators in some locales where the decimal separator is a comma, no occurrence of a period used in this way occurred in the test data.

The dash, or hyphen, is another polysemous symbol. In most cases, it represented multiplication, but as already noted it can also separate the words of multi-word unit names, or the components of a mixed

number. It is also used as a unary operator to indicate negation in numbers and in particular in exponents. The hyphen is also used in range expressions to separate the constituent simple expressions. Since no example of a binary subtraction operator occurred in the test data, the use of the hyphen in range expressions caused no ambiguity.

5.2.7 Sequence expressions

Sequence expressions were detected in the Analyzer and separated into their constituent simple expressions, on which the Analyzer was called recursively. Where the unit was given only after the last number (e.g. "`6x8ft`"), the results were adjusted to add the unit into all the sub-expressions in the sequence. Some box dimension expressions were non-standard in that they contained embedded letters to indicate height, width and depth, as in "`6'H x 4'W x 12"D`".

5.3 Input Transformation Rules

Another strategy used in the analysis was to transform the input before it was passed to the parser. Transformation was used to normalize away certain kinds of variation, as described in this section. A number of transformations were applied, as follows.

`SUP_INTEGER`: Transforms patterns of HTML that are used to represent exponents into with a standard form. Example: `cm<sup>-2</sup>` → `cm^-2` (`"^"` is an exponentiation operator recognized by the parser.)

`DEGREE_KELVIN`: Transforms HTML used to form a construct like "$^\circ K$" into a valid form. Example: `<sup>0</sup>K` → `K` (kelvin).

`DEGREE_CELSIUS`: Transforms HTML used to form a construct like "$^\circ C$" into a usable form. Example: `<sup>0</sup>C` → `degC` (which is understood by the parser).

`USUK_PERIOD`: Removes the periods from strings like "U.S." and "U.K." Example: `U.S.` → `US` . Used with `USUK_QUALIFIER` which it must precede.

`USUK_QUALIFIER:` Handles cases where the specifier of a US or UK unit follows the unit name. Only certain unit names are handled. They are: gallon (and "gal"), quart, pint, tablespoon (and "tbsp"), and hundredweight (also "hundred weight" and "cwt") Example: `gallon (US)` → `US gallon`. After normalization in this way, the unit name can be recognized as a multi-word name.

`BRIT_QUALIFIER:` Changes various specifiers of UK units in the forms "Brit.", "British" and "Imperial" when these words precede a unit name. Example: `Brit. gallons` → `UK gallons`.

`ALL_HTML:` Specifies that all HTML markup will be removed from the input string.

`SUP_DEGREE:` Handles superscript '0' (zero), and the letter 'o', either lower or uppercase, as degree signs. This rule must precede `SUP_INTEGER` if it is to be effective.

`FRACTION:` Handles patterns used to express common simple fractions. The fractional part of the expression is replaced by the equivalent Unicode character. For example, patterns like `"1 1/2"`, `"11/2"` and `"1-1/2"` are replaced with "`1\u00bd`" (`+U00bd` is the Unicode "half" character).

5.4 Evaluation of a Measure Expression

Evaluation of a measure expression calculates a value, of the form of Equation 1, for the parsed expression. Given a parse tree like the example shown in Figure 3, the operands and operators were extracted. The operands each have a value, and the operations shown in Table 1 were successively applied to produce a final evaluated value. The analyzer used arbitrary-precision arithmetic internally [29] to minimize any influence of rounding errors.

One notable aspect of the evaluation of measure expressions is the low precedence of the division operator in such expressions. This follows from the requirement of the standard language that an expression such as "`20 liters per 100 kilometers`" should be interpreted as "`(20 liter)/(100 kilometer)`", not "`(20 liters/100)`×`kilometer`", or that "`10 J/cm2.s`" should be interpreted as "`10 J/(cm2.s)`". This is of course different from the convention in most programming languages. The data showed that while many of the web page authors used the standard convention, others made their desired precedence explicit, in forms like "`J/cm2/s`", which are in fact disallowed in the standard

Table 7: Effect of configuration of the analyzer on the error rate

Configuration	Error rate
Basic	10.95%
+ Sequence expressions	9.84%
+ Input transformation	8.68%

language, or through the use of parentheses, as in "`J/(cm2.s)`".

5.5 Error Rate of the Analyzer

Once the analyzer had been adapted to handle the development test data, its accuracy was measured with respect to the 3,435 measure expressions in the held-out unseen test data (see Table 3). Each evaluated expression was compared with the evaluated truth expression using the equality operation in Table 1, and if the equality held this was counted as an instance of a successful analysis. For sequence expressions, both the data and truth expressions were required to be of the same type, and the corresponding sub-expressions in the two expressions were required to be equal. For all expressions, an analysis error was recorded if the equality was not satisfied or if the analysis of either the data expression or the truth expression failed. Table 7 shows the error rate measured in this way for different configurations of the analyzer. Adding to the base configuration the ability to recognize sequence expressions in the data resulted in a 10.1% reduction in the error rate, and turning on the input transformations described earlier reduced the error rate by a further 11.7%. The final error rate achieved on the unseen test data was 8.68%.

Table 8 shows the categories of errors that occurred in the analysis. All the errors that occurred in parsing the truth expressions were due to the presence in them of units that were not in the dictionary, which in turn were included in the truth expression because they were used in the corresponding data expression. Examples are lines of magnetic force, money units, the "circular mil", and "pounds of water".

The largest category of analysis failures, amounting to 59% of the total, were those in which the failure occurred in parsing the data measure expression after successful analysis of the truth expression. The largest single cause of these was coding errors that had not been previously detected. The next most common cause (22% of expression parse failures) was expression syntax that had not been found in the development test data and that defeated the parser. Examples are: "`sq.cm`", with no space between the

period and the unit; "`10^(-10)`", with parentheses in the exponent; and the word "a" indicating division, as in "`thirty pounds a month`". Variants of unit names or abbreviations that were not included in the dictionary caused 18% of parse failures. Examples are "`VDC`" for volt and "`htz`" for Hertz.

Table 8: Error rates for parsing and evaluation of measure expressions in the unseen test collection.

Expressions	3435
Parse failure in truth	34
Parse failure in expression	177
Equality errors	87
Total errors	298
Error rate	8.68%

The equality errors fell into several classes. The most frequent, accounting for 45% of these errors, were due to either a unit name or abbreviation that was not in the dictionary, or was not recognized for some other reason, but was interpreted by the scanner as a prefix+unit combination. For example, "`cms`" is illegal as the plural of "cm" but it was interpreted as "centimilliseconds" by the scanner). These errors could be eliminated by simply not allowing a PREFIXUNIT to contain more than one prefix abbreviation. The next most frequent cause of equality errors, accounting for 11% of them, was due to a coding error which caused the equality relation for some sequence expressions to be wrongly evaluated. Other less frequent causes of error (each occurring between 2 and 5 times) were misinterpretation of "`H.P`" as "henry $\times$ poise" instead of horsepower, misinterpretation of unrecognized canonical fractions, and unresolved ambiguity such as the use of "`g`" to mean standard acceleration rather than gram.

6 Characteristics of Measure Expressions

Some quantitative characteristics of measure expressions found on the web can be determined from the analysis of the test data. For this aspect of the study we used the combined test collection, for which the error rate was 6.8%. This error rate is adequate to allow the main features of the expressions to be determined.

The number of terminals in the parse tree provides a measure of the complexity of a measure expression. Figure 4 shows the frequency distribution of terminals (this data is for the unseen test collection).

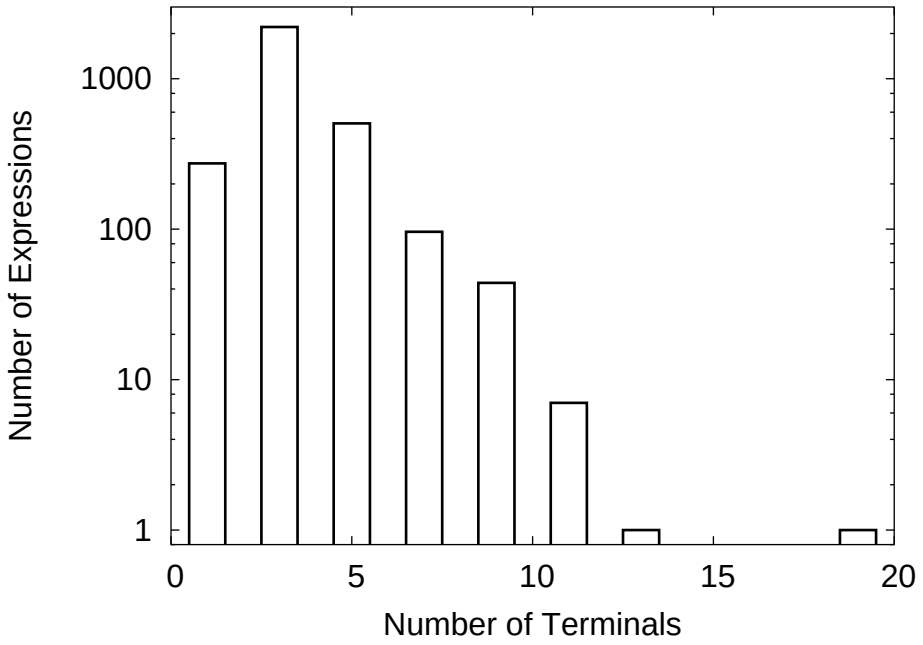

Figure 4: Frequency of terminals in measure expressions in the unseen test collection.

The number of terminals in a sequence expression was taken to that in the first expression in the sequence. The figure shows that all the expressions contained an odd number of terminals, reflecting the fact that no unary operators occurred in the data other than those preceding a real number, which were subsumed into the number and not counted as a separate terminal. Expressions with three terminals were the most common, amounting to 2209 of the 3137 expressions that were successfully parsed, or 70.4% of them. An example of an expression with three terminals is “152 cm”. Expressions with one terminal are those consisting of a single prefixed unit name, for example “centimeter”. Single unit names, such as “meter”, were not annotated as measure expressions unless they were preceded by a number.

The frequency of expressions with three or more terminals falls approximately exponentially with increasing terminal count. In all, 20.9% of the measure expressions contained more than three terminals. Only two expressions with more than 11 terminals were found in the unseen test data. They were

“`6.5x10^10/(cm^2-s)`” (13 terminals)

“`5.67 x 10<sup>-8</sup> joules/m<sup>2</sup>/s/`
`(<sup>0</sup>K)<sup>4</sup>`” (19 terminals)

Table 9: Frequency of operands in the combined test collection

Operand	Number of occurrences
BASE_UNIT_NAME	5534
REAL	2927
PREFIXUNIT	898
EXP_WORD	629
EXPON_INT	338
WORD_INT	173
FRACTION	23
LOG_RATIO	16
OFA[a]	10
PREFIX_SEQUENCE	6
INTEGER	3

a. Has no magnitude; occurs as a component of some instances of FRACTION

Table 10: Frequency of operators in the combined test collection

Operator	Number of occurrences
MULT_WHITESPACE	2595
DIV_SIGN	967
ADJ_EXPON	605
MULT_SIGN	514
NO_WHITESPACE_MULT	443
EXP_OPER	338
DIV_WORD	216
EXP_WHITESPACE	24
LEFT_PAREN, RIGHT_PAREN	23
POST_ANTILOG	16
PREFIX_MULT	6
ANDA	4
MULT_WORD	4
MULT_ANDA	1

The frequencies of operands and operators in the combined test collection are shown in Tables 9 and 10 respectively. While the counts of the low frequency types in these data are subject to potential systematic errors caused by analysis failures, some conclusions can be drawn from this data. First, Table 9 shows that prefixed units (PREFIXUNIT types) are relatively infrequent, occurring only 16% as often as simple unit names. This probably reflects the relative frequency of non-metric units, which are rarely prefixed, in the data (see below and Table 11). In expressions from an exclusively scientific or engineering domain, where metric units are more likely to be used, the relative frequency of prefixed units (and the corresponding potential ambiguities already discussed) would probably be higher. Table 10 shows that exponentiation of units was relatively common. About 2/3 of the time exponentiation was expressed as a word, as in "square feet", and was parsed as the EXP_WORD operand. A numeric exponent (EXPON_INT) appeared in the remaining cases.

Real numbers occurred with only about half the frequency of base unit names and prefix units combined. At first sight this is inconsistent with the conclusion that the most common form of measure expression is a quantifier together with a unit, but this apparent anomaly is accounted for the fact that the data contained a number of tables of conversion factors, which generally contain entries of the form "nanogram $= 10^{-9}$gram", in which the prefix+unit on the left has no quantifier. The pages with very large numbers of measure expressions (Figure 1) were mostly of this type. Spelled out numbers, predominantly integers, occurred in about 3.6% of the expressions. Lastly, occurrences of PREFIX_SEQUENCE, i.e. a prefix separated from a unit name, were quite rare, perhaps reflecting the fact that non-contiguous prefixes are forbidden by the standards.

The listing of the frequency of operators in Table 10 shows that multiplication implied by whitespace was by far the most common operator. Explicit multiplication (also permitted by the standards) was much less common, about as frequent in fact as the implied multiplication operator NO_WHITESPACE_MULT, which is forbidden by them. Explicit division operators, predominantly the solidus and the DIV_WORD "per", were more common than explicit multiplication. The two operators that correspond to exponentiation in words, ADJ_EXPON and EXP_WHITESPACE, were almost as common as division or multiplication. In fact, exponentiation was about as common in the data as prefixing of units. Some occurrences of EXP_OPER corresponded to exponentiation of numbers, as in "`10<sup>-10</sup> coulomb`".

Table 11: The number of occurrences of the most frequent instances of BASE_UNIT_NAME and PREFIXUNIT types in the combined test collection

BASE_UNIT_NAME		PREFIXUNIT	
foot	704	centimeter	277
meter[a]	487	kilometer	101
second	456	millimeter	63
pound	439	milliliter	57
inch	296	kilowatt	54
kilogram[b]	276	millisievert	26
hour	228	milligram	19
minute	195	megaelectron_volt	16
US_cup	159	megahertz	16
gram	125	millirem	15
US_gallon	125	decibel	12
year	112	kilocalorie	9
ampere	111	kilopascal	9
watt	99	microliter	9
Joule	96	nanometer	9
teaspoon	96	kiloJoule	8
mile	95	milliampere	8
yard	90	millibar	8
volt	84	millivolt	8
ounce	76	decimeter	7

a. Includes occurrences of both “meter” and “metre”.
b. Because of its status as one of the base units of the SI system, kilogram is treated as a base unit name.

Table 11 shows the 20 most frequent names of BASE_UNIT_NAME and PREFIXUNIT types in the data. Of the former, 12 of the 20 are non-metric units, while all of the latter are prefixed metric units. Of the top 20 BASE_UNIT_NAMES the proportions of units of mass, length, volume and time are roughly equal, with the remainder being various electrical units. No unit of area makes the top 20, perhaps because there is no named area unit of convenient size for everyday purposes, so that most areas are expressed as lengths raised to the second power, as in “square feet”. Units common in recipes, such as cup and teaspoon (interpreted by the analyzer as the US variants of these units) are among the most common ones. The list of the 20 most frequent prefix+unit combinations in the same table shows a wider range of units, though it seems to show some effect of the query words (Table 2) used to retrieve web pages, with millisievert and rem perhaps appearing because of the query word “rad” (all are units of radiation dose or radiation exposure).

Table 12: Frequency of sequence expressions in the combined test collection

Type	Occurrences
Range	16
Sheet dimensions	31
Box dimensions	8

Table 13: Frequency of certain language features in the combined test collection

Feature	Occurrences
Mixed number: an integer immediately followed by a fractional part	66
Period following an abbreviation	51
Commas used as thousands separator in a number	15
Blanks used as thousands separator in a number	11

Sequence expressions account for just 1.1% of expressions. The frequency of the different types is shown in Table 12. Inspection of the results shows that these frequencies, while low, are reliable in that the parser correctly identified all the sequence expressions in the data. All the range expressions in the data diverged from the standard [1] in that the unit was given only after the second number, as in "`13 to 20ghz`", whereas the standard mandates that this should be written "13 GHz to 20GHz". Furthermore, in the majority of the range expressions a hyphen was used as the range separator instead of the word "to" as mandated by the standard. In contrast, almost all the sheet and box dimension expressions conformed to the standard. Where they differed, it was because the unit was given only after the last number, or because embedded letters were used within the sequence to indicate width, depth and height, as was done in 2 of the eight box dimension statements in the data.

Some other features of the language used in the measure expressions that occurred relatively infrequently are shown in Table 13. Mixed numbers, which are deprecated by the standards, occurred in about 1% of the expressions; the importance of canonical fractions in such numbers has already been noted. Abbreviations terminated by a period, also deprecated, occurred with about the same frequency. The table also shows that on the rare occasions where thousands separators were used in numbers, they were slightly more likely to be commas (deprecated) than the spaces recommended by the standards.

7 Discussion and Conclusions

This study has presented a description of the informal language of measure expressions, as discovered in web pages. The description is embodied in a grammar, a dictionary, and a set of heuristics used by the expression analyzer that was applied to the expressions. The adequacy of the description is measured by the accuracy of the analysis against the unseen test set.

The results in preceding sections show that the accuracy of the analyzer is adequate to allow the main features of the informal language to be determined. Furthermore, it is clear that the accuracy can be further improved. The analysis of the parse failures that was presented indicates that the majority of the errors could be eliminated by fixing coding errors, and by modifying the dictionary to include other units and to add names and abbreviations of units that it already contains. Additional improvements could come from modifying the grammar and the scanner to handle new syntax that was discovered in the unseen data. It should also be noted that the accuracy of analysis in other domains may well be higher than it is for web data. Part of the motivation for using such data in this study is the hypothesis that this data includes a wider range of idioms than is found in other data sources. Edited content is likely to contain more standardized syntax, and the names and abbreviations used will probably be less ambiguous. It is likely, therefore, that it can be analyzed with a lower error rate. Additional improvements are also likely if content from a single domain, or a restricted set of domains, is considered, because then the conventions of those domains can be incorporated into the heuristics used by the analyzer. This is a topic for further research.

The study has provided an analysis of the ways in which measure expressions on web pages diverge from the recommendations promulgated by standards bodies. Among the most significant deviations are the incorporation of HTML markup to achieve the typographic effects called for by the standards, neglect of conventions of case that are designed to eliminate ambiguity in prefix+unit combinations when using metric units, and using non-standard abbreviations of the names of metric units that can be confused with prefix+unit combinations. Consequently, the informal language contains significantly more potential ambiguity than does the standard language. This probably limits the ultimate accuracy with which it can be analyzed, although the heuristics developed in the present work were quite successful in resolving ambiguity in almost all cases. Nevertheless, the prevalence of ambiguity suggests that a viable strategy to achieve

the highest possible accuracy would be to use additional information, such as might be obtained from the context of a measure expression in text, to resolve ambiguity. To do this, it might be useful to carry alternative parses through the analysis, instead of parsing deterministically as was done in the present work. Using techniques of natural language parsing, such a chart parsers, might be one way to achieve this.

Another way in which the informal language was found to depart from the standards was in the use of syntax that is deprecated by the standard. The most common deviations were found to be the omission of a space between a number and its unit, and the use of mixed numbers. Combinations of operators and operands that are deprecated by the standards were also common, such as using more than one division operator in an expression, or using operator symbols rather than words with the full forms of unit names. However, the study has also shown that it is possible to configure the analyzer to deal with all these cases.

Thus, from the perspective of information extraction, this study has indicated that analysis, which in the Introduction was identified as the second step of the three involved in information extraction of measure expressions, can be done adequately. Well established techniques of information extraction seem likely to be applicable to the first step, recognition. In fact the results presented here provide some insights into what features may be most suitable to use in an automatic recognizer, whether based on machine learning or on pattern-based recognition.

The third step in information extraction, classification, presents some challenges. One is related to the fact, noted by Thun [22] among others, that the dimensions of a measure expression as defined by Equation 2 do not uniquely determine the quantity that it measures. Take for example the expression “liters per 100 km”, which one might want to assign to the class “fuel_consumption”. The dimensions of this expression are $\text{length}^3/\text{length} = \text{length}^2$, which are the same as those of area. Thus a classifier would not be able to distinguish between the classes “fuel_consumption” and “area” on the basis of the dimensions alone. Also, many expressions that are dimensionless are not equivalent to each other. To go some way towards resolving this difficulty, it has been proposed [30] that the calculation of dimension should carry not just the exponent of the base quantity, but also a record of the exponents that occurred in the numerator and the denominator of the analyzed expression. The effectiveness of this approach remains to be determined. A further challenge is to determine whether accurate classification can be based only on intrinsic features of expressions, or must in addition take account of extrinsic features derived, for example, from the context in

which the measure expression occurs in a document. The resolution of these challenges remains for future research.

8 Acknowledgments

The author wishes to thank Keh-Shin Cheng for her help in gathering the web pages for the test collections, and Karen Setz and Anni Coden for their review of the draft.

9 Appendix

A Annotation Policies

The annotation used the following guidelines:

1. Simple unit names or their abbreviations were not annotated unless they were preceded by a number. That is, `meter` was not annotated as a measure expression, but "`5 meter`" was. Prefixed single names (e.g. "`millimeter`") were annotated.

2. Spelled-out numbers within unit expressions were annotated if they were specific "(`one thousand meters`") but not if they were non-specific ("`thousands of meters`").

3. Measure expressions within HTML element attributes were not annotated. These expressions are common in certain HTML-ized word processor output.

4. Temperatures were not annotated, but expressions involving a temperature unit were (e.g. "`deg. C per second`").

5. Measure expressions within `<script> ... </script>` elements (e.g. in JavaScript source code in web pages) were not annotated.

6. Pure money amounts were not annotated, however expressions involving measurement units along with money were annotated.

7. Expressions where a symbol was represented by an image were not annotated, but any textual sub-parts that were themselves measure expressions were annotated.

8. Expressions that included non-unit words, such as "`25 grams of sugar per 6 ounces`" were not annotated as a whole, but sub-parts that were measure expressions were annotated.

9. Purely numeric expressions, such as "`25`" or "`6.00 per dozen`" were not annotated.

References

[1] Barry N. Taylor. *Guide for the Use of the International System of Units (SI). NIST 811.*, 1995.

[2] E. Richard Cohen and Pierre Giacomo. Symbols, Units, Nomenclature and Fundamental Constants in Physics. *Physica*, 146A:1–68, 1987.

[3] Deutches Institut fur Normung (DIN). *Medical Informatics - Expression of Results of Measurements in Health Sciences, ENV 12435,* 2000.

[4] American National Standards Institute, 1430 Broadway, New York, NY 10018, USA. *Representations for U.S. Customary, SI, and Other Units to be Used in Systems with Limited Character Sets, ANSI X3.50,* 1986.

[5] International Standards Organization. *Representation of SI and Other Units in Systems with Limited Character Sets. ISO 2955,* 1983.

[6] Kalervo Järvelin and Timo Niemi. Data Conversion, Aggregation and Deduction for Advanced Retrieval from the Heterogeneous Fact Databases. In *Proceedings of the 14th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–182. ACM Press, 1991.

[7] Dan Gunter and Warren Smith. Schemas for Grid Performance Events. Technical Report GWD-PERF-1-1, Global Grid Forum, 2000.

[8] Jim Cowie and Wendy Lehnert. Information Extraction. *Communications of the ACM*, 39(1):80–91, January 1996.

[9] Maria Teresa Pazienza, editor. *Information Extraction*. Number 1299 in Lecture Notes in Computer Science. Springer, Heidelberg, DE, 1997.

[10] Mark T. Maybury, editor. *New Directions in Question Answering: Papers from the 2003 AAAI Spring Symposium*. AAAI Press, 2003.

[11] Ian Horrocks and James A. Hendler, editors. *The Semantic Web - ISWC 2002, First International Semantic Web Conference, Sardinia, Italy, June 9-12, 2002, Proceedings*, volume 2342 of *Lecture Notes in Computer Science*. Springer, 2002.

[12] Adrian Mariano. Gnu Units, 2000. See http://www.gnu.org/software/units/units.html.

[13] Convertit units conversion utility, 2000. See http://www.convertit.com.

[14] Proceedings of the Seventh Message Understanding Conference (MUC-7), 1999. See http://www.itl.nist.gov/iad/894.02/related_projects/muc/.

[15] Ralph Grishman. *TIPSTER Architecture Design Document version 3.1*. http://www.tipster.org, 1998.

[16] John Prager, Eric Brown, Anni Coden, and Dragomir Radev. Question-Answering by Predictive Annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, 2000.

[17] John M. Prager, Jennifer Chu-Carroll, and Krzysztof W. Czuba. Question Answering using Constraint Satisfaction: QA-by-Dossier. In *Proceedings of HLT-NAACL'03*. Association for Computational Linguistics.

[18] John Prager, 2003. Private Communication.

[19] Thomas R. Gruber and Gregory R. Olsen. An Ontology for Engineering Mathematics. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, 1994.

[20] Helena Sofia Pinto and João P. Martins. Revising and Extending the Units of Measure "Subontology". In *Workshop on the IEEE Standard Upper Ontology, IJCAI 2001*, 2001.

[21] Ian Niles and Adam Pease. Towards a Standard Upper Ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems*, pages 2–9. ACM Press, 2001.

[22] R.E. Thun. On Dimensional Analysis. *IBM Journal*, page 349, July 1960.

[23] Gunther Schadow, Clement J. McDonald, Jeffrey G. Suico, Ulrich Fohring, and Thomas Tolxdorff. Units of Measure in Clinical Information Systems. *J Am Med Inform Assoc*, 6(2):151–161, 1999.

[24] B. S. Massey. *Measures in Science and Engineering: Their Expression, Relation and Interpretation.* Ellis Horwood, 1986.

[25] Robert Gaizauskas, Hamish Cunningham, Yorick Wilks, Peter Rodgers, and Kevin Humphreys. Gate: An Environment to Support Research and Development in Natural Language Engineering. In *Proceedings of the 8th International Conference on Tools with Artificial Intelligence (ICTAI '96)*, page 58. IEEE Computer Society, 1996.

[26] Richard T. Gillam. A Rule-Based Approach to Number Spellout. In Unicode Consortium, editor, *Asia, Software + the Internet: Going Global with Unicode: The Twelfth International Unicode/ISO10646 Conference (IUC12) April 8–10, 1998, Tokyo, Japan*. The Unicode Consortium.

[27] A. V. Aho and J. D. Ullman. *Foundations of Computer Science*. Computer Science Press, W.H. Freeman and Co., New York, 1992.

[28] Specifications, Tolerances, and other Technical Requirements for Weighing and Measuring Devices. Technical Report NIST Handbook 44, National Institute of Standards and Technology, 2003.

[29] Michael F. Cowlishaw. Decimal Floating Point: Algorism for Computers. In Jean-Claude Bajard and Michael Schulte, editors, *Proceedings of the 16th Symposium on Computer Arithmetic*, pages 104–111, Santiago de Compostela, Spain, 2003.

[30] Frank Olken and John McCarthy. What XML Schema Designers Need to Know About Measurement Units, 2000. Presentation to XTECH 2000, San Jose, California, Feb. 29, 2000. See http://pueblo.lbl.gov/ olken/mendel/units/xtech_units_slides.ppt.