Contents lists available at ScienceDirect

Journal of Visual Languages and Computing

journal homepage: www.elsevier.com/locate/jvlc

# Automatic detection of dimension errors in spreadsheets

Chris Chambers, Martin Erwig *

*Oregon State University, USA*

ARTICLE INFO

*Keywords:*
Spreadsheet
Dimension
Unit of measurement
Static analysis
Inference rule
Error detection

ABSTRACT

We present a reasoning system for inferring dimension information in spreadsheets. This system can be used to check the consistency of spreadsheet formulas and thus is able to detect errors in spreadsheets.

Our approach is based on three static analysis components. First, the spatial structure of the spreadsheet is analyzed to infer a labeling relationship among cells. Second, cells that are used as labels are lexically analyzed and mapped to potential dimensions. Finally, dimension information is propagated through spreadsheet formulas. An important aspect of the rule system defining dimension inference is that it works bi-directionally, that is, not only "downstream" from referenced arguments to the current cell, but also "upstream" in the reverse direction. This flexibility makes the system robust and turns out to be particularly useful in cases when the initial dimension information that can be inferred from headers is incomplete or ambiguous.

We have implemented a prototype system as an add-in to Excel. In an evaluation of this implementation we were able to detect dimension errors in almost 50% of the investigated spreadsheets, which shows (i) that the system works reliably in practice and (ii) that dimension information can be well exploited to uncover errors in spreadsheets.

## 1. Introduction

End users engage in a variety of programming activities, including the creation and maintenance of spreadsheets [1]. However, it has been shown that spreadsheets contain many errors [2,3], and that these errors are often the cause of substantial negative impacts on society [4].

A variety of approaches have been investigated to prevent, detect, and remove errors from spreadsheets. Preventive approaches to improve the quality of spreadsheets include a variety of guidelines for spreadsheet design [5–8,2] and techniques for the automatic generation of spreadsheets [9,10] from visual [11] or object-oriented specification [12,13]. However, since preventive approaches, in principle, have to interfere with the spreadsheet creation process that makes spreadsheets so attractive to end users, much research has focused rather on the detection and removal of errors.

The detection of spreadsheet errors has been mainly approached from two different angles, auditing/testing and automatic checking.

Although a variety of effective strategies and principles for spreadsheet auditing have been proposed [14–16], a major limitation of these approaches is that they cannot provide guarantees or even measures for the (likelihood of) spreadsheet correctness. The situation is different in the case of testing where test-adequacy criteria can inform the testing strategies [17]. The only systematic testing approach for spreadsheets is the "What You See Is What You Test" approach [18,19] that uses data-flow adequacy and coverage criteria to give the user feedback on how well tested the spreadsheet is.

* Corresponding author.
*E-mail addresses:* chambech@eecs.oregonstate.edu (C. Chambers), erwig@eecs.oregonstate.edu (M. Erwig).

1045-926X/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jvlc.2009.04.002

A principal problem of the testing approach is for users to find test cases that cover enough computations and data flows. To support users in this effort, test-case generation systems [20–22] can generate test cases that improve the test coverage.

Another problem with testing approaches is that they suffer from oracle mistakes, that is, incorrect decisions made by users during testing [23] might introduce more errors into spreadsheets. Some of these problems can be alleviated by automating parts of the testing/debugging process [24,25].

Finally, testing approaches also present a serious motivational challenge, because they require substantial effort on part of the user. This aspect is particularly relevant in the case of spreadsheet users since many of them are end users who mainly want to get their job done; they are much less motivated than professional software developers to spend additional time on their spreadsheets for testing purposes.

The latter aspect makes automatic checking approaches very attractive since they promise error detection with minimal user input. Two obvious problems with traditional type checking systems are (1) that they are limited in the kinds of errors they find and (2) that abstract typing concepts may be difficult to communicate to end users. The limited scope of type checking simply means that type systems should not intend to replace testing, but to complement it. That this can work very well has been demonstrated, for example, in Lawrence et al. [26]. The usability concern has been addressed in two different (although related) ways.

First, based on the observation that spreadsheet users often place labels as comments into spreadsheets close to the relevant data, we can reason about the combination of these labels in formulas that refer to labeled data and thus detect inconsistencies [27,28]. In a recent study on the usability of a type system in spreadsheets we discovered that end users can effectively use such label-based type systems to debug a variety of errors in their spreadsheets [29].

Second, we can employ units of measurements as a concrete notion of types that is well known among end users [30]. Dimensions are used to characterize different kinds of values, much like traditional, more abstract, type systems used in general-purpose programming languages, but on a more fine-grained level. For example, a floating point number, which has just one type, can nevertheless represent different kinds of quantities, such as length or time values, which is captured through the concept of dimensions. For each dimension, such as length, there are several different units of measurements, such as cm, ft, m, or km, that describe values of that dimension at an even finer-grained level.

The incorrect combination of values of different dimensions has been a cause of major problems. One of the most famous dimension errors is a lacking value conversion in the $320 Million Mars Climate Orbiter where one software component sent thruster data in pounds, an English unit of measurement, whereas the Orbiter was expecting the metric unit Newtons (N) [31]. Since one pound is equal to 4.48 N, the craft slowly drifted off course. Over time the orbiter dropped 60 miles closer to the surface of Mars and was destroyed.

As we will demonstrate in this paper, dimension errors occur frequently in spreadsheets. Therefore, an approach to detect such errors can be an important part of a tool suite to improve the quality of spreadsheets. In this paper we describe *dimension inference*, a method to automatically find dimension errors in spreadsheets. Our work builds on previous approaches and extends them in several important ways. First, through incorporating header inference [32], the presented system does not have to rely on additional user annotations and provides therefore a high degree of *automation* ("one-click checking"). Second, in addition to checking whether dimensions of values are correctly dealt with in formulas, our approach can *infer* dimensions based on context provided by formulas. This feature is particularly helpful in cases when header inference does not provide a detailed enough account of the dimensions for all values in the spreadsheet. Dimension inference can then in many cases close the gap. Finally, the presented system can automatically infer *conversion factors* between different units of measurement (such as meters and feet) and can enforce the correct use of conversions in formulas.

In addition to the formal model of dimension inference, we describe a practical tool that has been implemented as an extension to Microsoft Excel. We also present an empirical analysis of how dimension inference works in practice. This paper is an extended version of [33] and contains a revised and refactored rule system, a comparison of different dimension checking systems, and an expanded discussion of results.

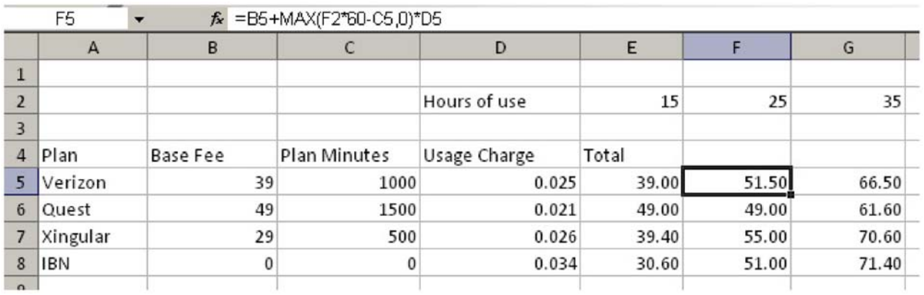
The rest of this paper is structured as follows. In Section 2 we illustrate the issues involved in dimension checking and inference with a small example. In Section 3 we formalize spreadsheets and a model of dimensions. The process of dimension inference is then described in Section 4. In Section 5 we report on an evaluation of a prototypical implementation of a tool for dimension analysis. We discuss related work in Section 6 and give conclusions and ideas for future work in Section 7.

## 2. Determining dimensions of spreadsheet formulas

Consider the spreadsheet shown in Fig. 1 that shows the details of different phone plans and that computes the costs for different usage profiles. The monthly totals for each plan and a particular hours-of-use value is computed by adding the base fee and the cost for the minutes exceeding the free minutes. For example, for the plan in row 5 and for the use of 25 h, the formula in cell F5 is as follows:

$$\mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5}$$

What unit of measurement, or unit for short, does the value computed by this formula have? First, by inspecting the labels in the spreadsheet we can try to infer what the dimensions of the stored data values are. For example, the value 39 in cell B5 represents a money amount, which could be without further information given in any currency. It makes sense for a system to assume whatever



F5 =B5+MAX(F2*60-C5,0)*D5

|   | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 |  |  |  |  |  |  |  |
| 2 |  |  |  | Hours of use | 15 | 25 | 35 |
| 3 |  |  |  |  |  |  |  |
| 4 | Plan | Base Fee | Plan Minutes | Usage Charge | Total |  |  |
| 5 | Verizon | 39 | 1000 | 0.025 | 39.00 | 51.50 | 66.50 |
| 6 | Quest | 49 | 1500 | 0.021 | 49.00 | 49.00 | 61.60 |
| 7 | Xingular | 29 | 500 | 0.026 | 39.40 | 55.00 | 70.60 |
| 8 | IBN | 0 | 0 | 0.034 | 30.60 | 51.00 | 71.40 |

**Fig. 1.** A spreadsheet for computing the costs of phone plans under different usage scenarios.

currency is set to be the default, which we assume here to be $. Similarly, we can conclude that C5 is a time value. In this case there is no doubt about the unit, which is minutes. The same applies to F2, which contains an hour value. However, it is not clear what dimension the value in D5 has, because "charge" could indicate a money amount or an electrical charge.

Second, given the potentially incomplete information about the dimensions of values, we can reason about the structure of formulas to find out the dimension of the computed value, or identify an error in case the formula combines dimensions incorrectly. In the course of determining the dimension of a formula we can also infer dimensions for values whose dimension could not be determined from a label and is so far unknown, as for the value in cell D5 for instance.

In the example, we see that C5 is subtracted from $F2 * 60$. Since all additive operations require that the arguments have the same unit of measurement, we can conclude that $F2 * 60$ must be minutes, which is possible if the constant 60 has the unit minutes/hour. In fact, only the value 60 has this unit.[1] In other words, the use of any other factor or the omission of a factor would have meant a fault in this formula.

The dimension behavior of MAX is the same as that of other addition operators. Therefore, we can infer that 0 and the whole expression $\mathrm{MAX}(F2 * 60 - C5, 0)$ also have the unit minutes. Here we can observe that the ability to infer dimensions/units in arbitrary directions, that is, for arguments from results (instead of only being able to reason from arguments to results) is crucial for obtaining a flexible and user-friendly reasoning system, because requiring the user to annotate 0 with minutes and 60 with minutes/hour would mean a big impact on usability.

The next step is to determine the unit for D5 so that the sum with B5 is dimension correct. Since B5 is in $, the product $\mathrm{MAX}(F2 * 60 - C5, 0) * D5$ must have the same unit. Since the MAX expression is in minutes, we can therefore conclude that D5 must have the unit $/minute. Finally, since B5 and the product expression have the same unit, we can conclude that the formula is dimension correct and has the unit of minutes.

[1] Constants can have multiple dimensions or units, for example, 60 also has the unit seconds/minute.

## 3. A formal model of spreadsheets and dimensions

### 3.1. Abstract syntax of spreadsheets

We work with the following simple model of spreadsheets. A spreadsheet ($S$) is a mapping from addresses ($a \in A$) to expressions ($e$). We write $S(a)$ to refer to the expression stored at address $a$ in the spreadsheet $S$. Expressions can be values ($v$), references to other cells ($\uparrow a$), or are constructed using operators. Each arithmetic operators, such as, $+$, represents a whole class of binary operations (here additive operations, such as $-$ or MAX) as well as corresponding aggregation operations (here: SUM). In addition, we have a dimension-consuming aggregation (**count**) and a conditional operator.

$$e ::= v \mid \uparrow a \mid e + e \mid e * e \mid \mathbf{count}(e, \ldots, e) \mid \mathbf{if}(e, e, e)$$

### 3.2. Representation of dimensions

A dimension ($d$) is given by a set of dimension components ($c$). Each component is given by a base ($b$), a conversion factor ($f$), and an integer exponent ($n$). Essentially, a dimension is a partial mapping from base dimensions to pairs ($n$,$f$). For the purpose of dimension inference, a dimension component can also be a dimension variable ($\delta$). If a dimension contains only one component, it is called a *singleton dimension*, whereas a dimension that contains two or more components is called a *composite dimension*. The *identity dimension* $\{\}$ is used for dimensionless values

$$d ::= \{c, \ldots, c\}$$
$$c ::= b_f^n \mid \delta$$

Dimensions that only differ in conversion factors describe a similar quantity, and for different factors there often exist different names, which are called *units of measurement*.

For each base dimension we identify a default unit with factor 1. For example, the default for length is meter (m), that is, $\mathrm{m} = \mathrm{length}_1^1$, which also means that $\mathrm{cm} = \mathrm{length}_{0.01}^1$ and $\mathrm{ft} = \mathrm{length}_{0.3048}^1$. In general, the following relationship holds (where $x$ is a dimensionless number and $b$ is an arbitrary base):

$$xb_f^n = xfb_1^n$$

We may also omit conversion factors and exponents of 1 for brevity, that is, we write more shortly $b^n$ for $b_1^n$, $b_f$ for $b_f^1$, and simply $b$ for $b_1^1$.

In general, the choice of dimensions is arbitrary and depends on the application. For the task of analyzing dimensions in arbitrary spreadsheets, we have chosen the seven SI units and some further units that we have found in the EUSES spreadsheet corpus [34]. The quantities and their default units are shown in Table 1.

Examples of composite dimensions are speed, measured in m/s, that is $\{\text{length}, \text{time}^{-1}\}$, or force, measured in kg m/s$^2$, which is $\{\text{mass}, \text{length}, \text{time}^{-2}\}$.

The relationship between basic and derived dimensions and units is illustrated with several examples in Table 2. Default units are set in boldface.

A conversion factor can be either a real number ($r$) or a conversion variable ($\phi$), which serves as a placeholder to be used during dimension inference

$$f ::= r \mid \phi$$

Examples of conversion factors can be found in the spreadsheet shown in Fig. 1, namely $\text{min} = \text{time}_{60}$ and $\text{h} = \text{time}_{3600}$. We can also illustrate the effect of conversion variables using that example. The label "Base Fee" in cell B4 can be mapped to a dimension $\text{money}_\phi$, but it is not clear in which currency. If B4 were added in some formula to a value that is known to be of unit $\$ = \text{money}_1$ or $\text{cent} = \text{money}_{0.01}$, the requirement of both arguments of addition to be of the same dimension would cause the unification of both dimensions and create the substitution $\{\phi \mapsto 1\}$ or $\{\phi \mapsto 0.01\}$, respectively, and thus B4 would also receive the unit \$ or cent, respectively.

**Table 1**
Base dimensions with default units.

| Quantity | Default unit |
|---|---|
| Length | Meter (m) |
| Mass | Kilogram (kg) |
| Time | Second (s) |
| Electric current | Ampere (A) |
| Temperature | Kelvin (K) |
| Amount of substance | Mole (mol) |
| Luminous intensity | Candela (cd) |
| Money | Dollar ($) |
| Angle | Degree (deg) |

**Table 2**
Basic and derived dimensions and corresponding units.

| | Dimension | Units |
|---|---|---|
| Basic | Length | **m**, cm, km, ft, ... |
| | Time | **s**, min, h, ... |
| | Mass | **kg**, pounds, ... |
| | ... | ... |
| Derived | Speed | $\frac{\text{m}}{\text{s}}, \frac{\text{km}}{\text{h}}, \ldots$ |
| | Force | $\frac{\text{kg m}}{\text{s}^2}$ = Newton, dyne, ... |
| | Pressure | $\frac{\text{kg}}{\text{ms}^2}$ = Pascal, psi, atm, ... |
| | ... | ... |

Our approach to represent conversions between different units within a dimension by a simple factor is not general enough to cover some conversions, such as degrees Fahrenheit to degrees Celsius. Nevertheless, we have chosen this simple model because it keeps the unification of dimensions feasible and works in most cases. This restriction is not too severe since in the spreadsheet repository that we have tested our prototype implementation on only 2 of 487 spreadsheets contained dimensions that could not be converted using the presented model.

### 3.3. Dimension-aware semantics

The rationale for introducing dimensions into computations is that they effectively restrict the meaningful computations in the sense of typing annotations. Consider, for example, the following operational semantics definition for the addition operation [35]:

$$\frac{e_1 \longrightarrow v_1 \quad e_2 \longrightarrow v_2}{e_1 + e_2 \longrightarrow v_1 + v_2}$$

We can refine the semantics definition by considering values that are annotated with dimensions. In that case, the rule becomes the following:

$$\frac{e_1 \longrightarrow v_1 : d \quad e_2 \longrightarrow v_2 : d}{e_1 + e_2 \longrightarrow v_1 + v_2 : d}$$

The effect of the dimension annotation is that values that are annotated with different dimensions are considered to be incompatible. By requiring that both arguments of the addition operation evaluate to values that are annotated by the same dimensions $d$, this definition effectively leaves the addition of expressions that evaluate to values with different dimensions undefined.

Multiplication transforms the dimensions of values according to the function $\bowtie$, which is defined as follows. First, $d \bowtie d'$ is undefined if $d$ and $d'$ contain two dimension components with the same base $b$ but different conversion factors, that is, if $b_f^n \in d \wedge b_{f'}^m \in d' \wedge f \neq f'$. Otherwise, we have the following definition:

$$d \bowtie d' = \{b_f^{n+m} \mid b_f^n \in d \wedge b_f^m \in d'\} \cup d \triangle d'$$

We use the symmetric difference of sets, $d \triangle d'$, which is defined as all the dimension components that are a variable or have a base that is in either $d$ or $d'$, but not in both.

The dimension-aware semantics for multiplication is then given by the following rule, which enforces the use of proper conversion factors in multiplications. For example, to calculate the distance a plane travels in 5 s when its speed is 950 km/h, one has to use a conversion factor with dimension h/s in the multiplication, otherwise $\bowtie$ is undefined, and the rule cannot be applied

$$\frac{e_1 \longrightarrow v_1 : d_1 \quad e_2 \longrightarrow v_2 : d_1 \quad d_1 \bowtie d_2 = d}{e_1 * e_2 \longrightarrow v_1 * v_2 : d}$$

This definition prevents the multiplication of two values that have dimensions with the same base dimension but different factors. For example, when determining the area of a square it does not make sense to multiply one side length, denoted by meters, with the other side length denoted by centimeters. What is the meaning of $5\,\mathrm{m} \times 7\,\mathrm{cm} = 35\,\mathrm{cm} \times \mathrm{m}$? While this technically is not an illegal operation, it seems more reasonable and practical to try and catch these situations. Therefore, a conversion factor has to be applied to one of the two dimensions being multiplied.

In the example above we have 5 s multiplied with 950 km/h. With no conversion factors the result would be $4750\,\mathrm{s} \times \mathrm{km/h}$, which does not provide the desired information. Since dimension inference requires the resulting dimension to have only one dimension of each base type, this would be an error. However, if the conversion factor 1/3600 h/s is included in the multiplication, the resulting value and dimension is 1.3194 km, which is certainly a valid dimension and provides a useful value. The one problem with requiring a conversion factor is that it can reduce flexibility in certain instances. For example, the conversion factor h/s could be applied in a later formula. In general, though, it makes more sense to try and catch this where it occurs.

The shown rules are a bit over-simplified because they ignore the notion of dimension validity discussed in the next section. The purpose of the rules was to show that incorporating a dimension concept into the semantics yields a more precise notion of what correct computations are, which forms the basis for an approach to identify errors based on dimension analysis.

### 3.4. Dimension validity

The dimension system defines an $n$-dimensional space, and values having a certain dimension can be regarded as points in this space. The traditional handling of dimensional values requires arguments of addition to have the same dimension, but places no constraints on the argument (or the result) dimension for multiplication. However, in practice dimensions cannot be multiplied arbitrarily. For example, no reasonable value can have the dimension $\mathrm{kg}^3$. Ruling out such unreasonable dimensions can strengthen dimensional analysis by effectively placing a validity constraint on the multiplication of dimensional values, that is, the result dimension of a multiplication must be a valid dimension.

An interesting scientific (or even philosophical) question is this. What, in principle, is a valid dimension? Since we are not aware of any general rules that could be used to determine the validity of dimensions, we have taken a pragmatic approach and have gathered dimensions that have been reported and documented [36]. The set of the thus obtained dimensions is taken as a definition of the predicate $\mathscr{V}(d)$ that yields true if and only if $d$ is a valid dimension. This predicate can be defined as a test of the exponents of all base dimensions occurring in $d$ with two exceptions. The allowed exponent ranges are defined by function $\mathscr{R}$ shown in Table 3.

**Table 3**
Valid dimension exponent ranges.

| $b$ | $\mathscr{R}(b)$ |
|---|---|
| Length | $-3,\ldots,3$ |
| Electric current | $-2,\ldots,1$ |
| Time | $-3,\ldots,2$ |
| All others | $-1,\ldots,1$ |

The exceptions to this table are the valid dimensions (1) farads and (2) Siemens, captured by the following predicate:

$$\mathscr{E}(d) = (d = \mathrm{kg}^{-1}\,\mathrm{m}^{-2}\,\mathrm{s}^{4}\,\mathrm{A}^{2}) \;\vee \qquad (1)$$
$$(d = \mathrm{kg}^{-1}\,\mathrm{m}^{-2}\,\mathrm{s}^{3}\,\mathrm{A}^{2}) \qquad (2)$$

With the definitions for $\mathscr{R}$ and $\mathscr{E}$ we can define the dimension validity predicate as follows:

$$\mathscr{V}(d) = (\forall b_f^n \in d : n \in \mathscr{R}(b)) \vee \mathscr{E}(d)$$

This predicate is still only a crude approximation since it considers quite a few non-existing dimensions as valid, for example, kg m. Ultimately, the best approach to realize $\mathscr{V}$ might be to simply store a table of all valid dimensions.

## 4. Dimension analysis

Dimension analysis of a spreadsheet happens in four phases that exploit different aspects of the information presented in the spreadsheet:

1. Header inference.
2. Label analysis.
3. Dimension inference.
4. Dimension instantiation.

Header inference identifies spatial relationships between labels and values/formulas in the spreadsheet. Label analysis derives basic information about units and dimensions from the textual content of labels employed in the spreadsheet. Dimension inference derives the dimensions for formulas using a formal rule system that encodes the laws of proper dimension handling by operations. Dimension inference has two main purposes: (1) it propagates dimension inference across the spreadsheet and (2) it identifies cases of computations that are dimension incorrect. Finally, dimension instantiation substitutes concrete units for dimension variables. This step applies only in those cases when the third step produces underspecified units that contain dimension variables. In the following we will describe these four steps in some detail.

### 4.1. Header inference

Header inference analyzes the structure of a spreadsheet and returns a set of headers for each cell. A header is simply the address of another cell. Therefore, header inference produces a binary relation $H \subseteq A \times A$ such that $(a, a') \in H$ says that $a'$ is a header of $a$. In general, one cell can be a header for many cells, and any particular cell can have zero, one, or more headers. For example, in Fig. 1, B4 is a header for B5, B6, B7, and B8, that is, $H^{-1}(\mathrm{B4}) = \{\mathrm{B5}, \mathrm{B6}, \mathrm{B7}, \mathrm{B8}\}$, and A5 and B4 are headers of B5, that is, $H(\mathrm{B5}) = \{\mathrm{A5}, \mathrm{B4}\}$. Header inference essentially works by analyzing the spatial relationships between different kinds of formulas, and it can also take into account layout information. Techniques for header inference have been described in detail elsewhere [32,37]. In the context of this paper we simply reuse those techniques.

### 4.2. Label analysis

In the second phase of dimension analysis we try to derive a dimension for each label contained in a cell that has been identified as a header by header inference. This process works by (a) splitting labels into separate words, (b) removing word inflections, (c) mapping word stems to dimensions, and (d) combining dimensions into one dimension. For example, cell C4 shown in Fig. 1 is a header cell and therefore subject to label analysis. Its value can be split into the two words "Free" and "Minutes", and the plural of "Minutes" can be removed. The resulting "Minute" can then be mapped to the dimension min. In contrast, "Free" cannot be mapped into any dimension and will thus be mapped to $\{\}$. Finally, the combination of both dimensions yields min.

An example of a label that produces a complex dimension is "Miles per Gallon" or "MPG" or "Miles/Gallon". Label analysis uses the divide symbol to infer that gallons will have an exponent of $-1$. Another example is "Hourly Pay Rate" or "Dollars per Hour", which will be deconstructed into parts and reassembled into the unit \$/h. Names for derived units are dealt with in principally the same way. For example, the label "Newton" would be identified as a unit and mapped to the dimension $\mathrm{kg\,m\,s^{-2}}$.

If no part of a header label can be mapped to a dimension other than $\{\}$, the label is mapped to a dimension variable $\delta$, which indicates that the dimension is at this time unknown.

### 4.3. Dimension inference

The third step of dimension analysis is dimension inference, which inspects each cell containing a formula and derives for it a dimension using the system of rules given in Fig. 2. Whenever the rule application fails, the formula for which no dimension could be inferred has been identified as erroneous. Moreover, derived dimensions that are not valid according to the predicate $\mathscr{V}$ defined in Section 3.4 also indicate formula errors. Since the derived dimension can be the identity dimension $\{\}$, the system simply ignores (areas of) spreadsheets that do not involve any headers or identified dimensions, that is, dimension analysis works smoothly on any kind of spreadsheet and is not disruptive in cases where it does not apply.

The relationship between formulas and dimensions is formalized through the following four judgments that tie together dimensions derived from headers/labels, known

$$\boxed{S,H \vdash a : d}$$

$$\textsc{NoHdr}\ \frac{H(a) = \varnothing}{S,H \vdash a : \delta} \qquad \textsc{SingleHdr}\ \frac{H(a) = \{a'\} \quad S(a') \Rightarrow d}{S,H \vdash a : d}$$

$$\textsc{DoubleHdr}\ \frac{H(a) = \{a_1,a_2\} \quad S(a_1) \Rightarrow d_1 \quad S(a_2) \Rightarrow d_2 \quad d \in d_1 \otimes d_2 \quad \mathscr{V}(d)}{S,H \vdash a : d}$$

$$\boxed{S,H \vdash e : d}$$

$$\textsc{Val}\ \frac{}{S,H \vdash v : \delta} \qquad \textsc{Ref}\ \frac{S,H \vdash (a,S(a)) : d}{S,H \vdash \uparrow a : d} \qquad \textsc{Count}\ \frac{S,H \vdash e_i : d}{S,H \vdash \mathbf{count}(e_1,\ldots,e_n) : \{\}} \qquad \textsc{If}\ \frac{S,H \vdash e_2 : d \quad S,H \vdash e_3 : d}{S,H \vdash \mathbf{if}(e_1,e_2,e_3) : d}$$

$$\textsc{Add}\ \frac{S,H \vdash e_1 : \{b^n_{f_1}\} \cup d \quad S,H \vdash e_2 : \{b^n_{f_2}\} \cup d \quad c_1 = f_1/f \quad c_2 = f_2/f}{S,H \vdash c_1 * e_1 + c_2 * e_2 : \{b^n_f\} \cup d}$$

$$\textsc{Mult}\ \frac{S,H \vdash e_1 : d_1 \quad S,H \vdash e_2 : d_2 \quad d = d_1 \bowtie d_2 \quad \mathscr{V}(d)}{S,H \vdash e_1 * e_2 : d}$$

$$\boxed{S,H \vdash (a,e) : d}$$

$$\textsc{Cell}\ \frac{S,H \vdash e : d \quad S,H \vdash a : d}{S,H \vdash (a,e) : d}$$

**Fig. 2.** Dimension inference rules.

dimensions for conversion factors, and dimension transformations in expressions.

*Value dimensions*. The judgment $v \Rightarrow d$ says the value $v$, if used as a label or factor, describes the dimension $d$. This judgment combines the result of the label analysis process, which provides judgments, such as Money $\Rightarrow$ \$, and prior knowledge of conversion factors, such as the following:

$$\begin{aligned} 60 &\Rightarrow \mathrm{min/h} \\ 60 &\Rightarrow \mathrm{s/min} \\ 100 &\Rightarrow \mathrm{cm/m} \\ &\vdots \end{aligned}$$

Note that the judgment $v \Rightarrow d$ is *not* a function, that is, one value can generally indicate different dimensions. This flexibility allows dimension inference to select the correct interpretation based on the context, that is, based on usage in formulas.

*Location dimensions*. The judgment $S, H \vdash a : d$ says that in the spreadsheet $S$ and given the header structure $H$, the location given by address $a$ has dimension $d$. This judgment combines the result of label analysis and header analysis into a judgment about the expected dimensions for cell locations. For example, in Fig. 1 we have $S, H \vdash \mathrm{C5} : \mathrm{min}$.

*Expression dimensions*. The judgment $S, H \vdash e : d$ says that in the spreadsheet $S$ and given the header structure $H$, the expression $e$ has dimension $d$. This judgment uses specific rules for expressions to determine the expected dimension, and is based on the standard rules for dimensions. For example, addition requires both expressions to have the same base dimension.

*Cell dimensions*. The judgment $S, H \vdash (a, e) : d$ says the cell $(a, e)$ in the spreadsheet $S$ has the dimension $d$ under the given header relationship $H$. For example, if $S$ represents the spreadsheet shown in Fig. 1 and $H$ is the corresponding header relationship, then we obtain $S, H \vdash (\mathrm{F5}, \mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5}) : \$$. How this result is obtained was explained informally in Section 2. The rules given in Fig. 2 formalize this process, and we will illustrate the formal derivation of this result in Section 4.5.

Since a cell can have more than one header[2] we have to define how to deal with the cases when both headers are identified as dimensions. Do we just take one dimension? If so, which one do we choose? Or shall we combine the dimensions somehow? As with the mapping of values to dimensions, the correct interpretation depends in many cases on the context, so that for the purpose of dimension inference it is best to principally allow all possibilities. We can realize this approach through the definition of a function that generates all possible dimensions that can be obtained from the combination of two[3] dimensions.

$$d \otimes d' = \{d, d', d \bowtie d', d \bowtie \bar{d}', \bar{d} \bowtie d'\}$$

[2] In practice, a cell has almost always at most two headers (row and column). This fact depends, however, on the method that is used for header inference.

[3] Since we are working with a header inference that produces at most two headers for any cell, the restriction to considering only two dimensions is appropriate. It would not be difficult to extend the definition to an arbitrary number of headers.

Here the operation $\bar{d}$ computes the inverse of a dimension, which is obtained by negating all exponents in all components

$$\bar{d} = \{b_f^{-n} \mid b_f^n \in d\}$$

Now we can provide the rules that define the dimension inference. Fig. 2 shows the three rules for the location judgment covering the cases when a cell has two, one, or zero headers, and a rule for each possible expression to define the cell judgment. Note that a rule like Add actually represents a whole class of rules covering the dimension inference for all "additive" operations (including MAX and SUM). Moreover, combinations of rules like Add and Count yield rules for correspondingly derived operations like AVG.

We can observe the following four principal kinds of dimension rules:

1. Dimension generators (Val and Add).
2. Dimension preservers (Add, Ref, and If).
3. Dimension composers (Mult).
4. Dimension consumers (Count).

Consistency checks are contained in some form or another in all rules but Ref. The most restrictive rules are If and Count since they require arguments to have the same dimensions. A little less restrictive is the rule Add that requires its arguments to have the same base, but allows for differences in the conversion factors as long as the arguments are scaled accordingly. Effectively, all conversions between dimensions happen within the rule Add. Rule Mult is least restrictive since it allows the multiplication of any quantities as long as the result is a valid dimension.

## 4.4. Dimension instantiation

An inferred dimension might contain dimension variables and/or conversion-factor variables. The occurrence of variables happens whenever the spreadsheet does not provide enough information to precisely narrow down the dimensions. In these cases we have to find substitutions for the variables to obtain proper dimensions. In fact, a dimension involving variables describes a whole class of possible dimensions. For example, $\mathrm{length}_\phi$ can be m, cm, or any other length dimension that can be obtained by substituting values for $\phi$. Similarly, the dimension $\{\mathrm{m}, \delta\}$ can be instantiated to velocity or acceleration using the substitution $\{\delta \mapsto \mathrm{s}^{-1}\}$ or $\{\delta \mapsto \mathrm{s}^{-2}\}$, respectively.

The instantiation of dimensions can be realized by generating substitutions for conversion-factor variables so that default dimensions are obtained and by generating substitutions for dimension variables that produce valid dimensions (as defined in Section 3.4). Of those valid dimensions we can then select the one that is most common (as indicated by the numbers to be reported in Section 5).

## 4.5. Example derivation

We now will illustrate how the rule system shown in Fig. 2 works by showing an example derivation for cell F5 taken from Fig. 1, which contains the following formula:

$\mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5}$

To determine the dimension of the cell, the rule Cell is employed, which requires the inference of the dimension for both the address, F5, and the stored formula.

Regarding the dimension of F5, we observe that header inference yields $H(\mathrm{F5}) = \mathrm{E4}$, and the value judgment maps the text "Total" to \$. In general, "Total" is an ambiguous label with respect to what dimension it denotes. However, our example is taken from the financial field, where it makes sense to map Total to \$. Therefore the application of the header rule yields the following:

$$\textsc{SingleHdr}\,\frac{H(\mathrm{F5}) = \mathrm{E4} \quad \mathrm{Total} \Rightarrow \$}{S, H \vdash \mathrm{F5} : \$}$$

The concluding judgment of the above rule instantiates the variable $d$ to \$ in the application of the Cell rule, which therefore takes the following form:

$$\textsc{Cell}\,\frac{S, H \vdash \mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5} : \$ \quad S, H \vdash \mathrm{F5} : \$}{S, H \vdash (\mathrm{F5}, \mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5})) : \$}$$

So we know what dimension the cell will have to have, but we do not know yet whether the expression in F5 is dimension correct. Therefore, we have to apply expression rules to establish that the formula produces indeed a \$ value, and since the outermost operation applied is $+$, we have to employ the Add rule, which is instantiated as follows:

$$\textsc{Add}\,\frac{S, H \vdash \mathrm{B5} : \{\$\} \cup \{\} \quad S, H \vdash \mathrm{MAX}(\mathrm{F2} * 60\text{-}\mathrm{C5}, 0) * \mathrm{D5} : \{\$\} \cup \{\} \quad c_1 = f_1/f \quad c_2 = f_2/f}{S, H \vdash c_1 * \mathrm{B5} + c_2 * \mathrm{MAX}(\mathrm{F2} * 60\text{-}\mathrm{C5}, 0) * \mathrm{D5} : \{\$\} \cup \{\}}$$

We can observe that all involved factors, $f$, $f_1$, and $f$ are simply 1, and therefore $c_1$ and $c_2$ are also 1, which means that we can simplify the rule instance for a more convenient future handling as follows:

$$\textsc{Add}\,\frac{S, H \vdash \mathrm{B5} : \{\$\} \cup \{\} \quad S, H \vdash \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5} : \{\$\} \cup \{\}}{S, H \vdash \mathrm{B5} + \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5} : \{\$\} \cup \{\}}$$

The first premise can be derived using the Ref rule. We have to notice that references in formulas, such as B5, are represented in the abstract syntax of the formal spreadsheet model as $\uparrow \mathrm{B5}$. Therefore, the instantiated Ref rule looks as follows:

$$\textsc{Ref}\,\frac{S, H \vdash (\mathrm{B5}, 39) : \$}{S, H \vdash \uparrow \mathrm{B5} : \$}$$

The premise of the rule results from the lookup $S(\mathrm{B5}) = 39$. Why is the premise of this rule true? Because due to the Val rule we can have $S, H \vdash 39 : \$$, and we can derive $S, H \vdash \mathrm{B5} : \$$ using the SingleHdr rule

$$\textsc{Cell}\,\frac{S, H \vdash 39 : \$ \qquad \dfrac{H(\mathrm{B5}) = \{\mathrm{B4}\} \quad S(\mathrm{B4}) \Rightarrow \$}{S, H \vdash \mathrm{B5} : \$}\,\textsc{SingleHdr}}{S, H \vdash (\mathrm{B5}, 39) : \$}$$

To derive the second premise of the Add rule we have to employ the Mult rule, which is instantiated as follows:

$$\textsc{Mult}\,\frac{S, H \vdash \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) : d_1 \quad S, H \vdash \mathrm{D5} : d_2 \quad \{\$\} = d_1 \bowtie d_2 \quad \mathscr{V}(\$)}{S, H \vdash \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5} : \$}$$

In this example, D5 was not assigned a dimension through label analysis. Therefore, the dimension is left as a variable $d_2$, to be possibly instantiated later. The first premise is derived using the rule for the MAX operation, which is the same as the Add rule

$$\textsc{Max}\,\frac{S, H \vdash \mathrm{F2} * 60 - \mathrm{C5} : d_1 \quad S, H \vdash 0 : d_1}{S, H \vdash \mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) : d_1}$$

The dimension of the constant 0 is left open for now, and the derivation of the dimension for the subtraction expression requires another instance of the Add (here called Sub to match the $-$ operation)

$$\textsc{Sub}\,\frac{S, H \vdash \mathrm{F2} * 60 : d_3 \quad S, H \vdash \mathrm{C5} : \mathrm{min} \quad c_3 = f_1/f \quad c_4 = f_2/f}{S, H \vdash c_1 * \mathrm{F2} * 60 - c_2 * \mathrm{C5} : \mathrm{min}}$$

Since the dimension of C5 will be minutes, which is derived in an analogous way to the dimension of B5 (see above), $\mathrm{F2} * 60$ must have a dimension that can at least be converted to minutes, using the conversion factors made available by the rule.

In fact, we can derive $d_3 = \mathrm{min}$ for $\mathrm{F2} * 60$, as can be seen by invoking the Mult rule again. The resulting rule instance is driven by the fact that $\mathrm{F2} * 60$ should have the dimension min and that F2 can be shown to be in hours as follows:

$$\textsc{Cell}\,\frac{S, H \vdash 25 : \mathrm{h} \qquad \dfrac{H(\mathrm{F2}) = \{\mathrm{D2}\} \quad S(\mathrm{D2}) \Rightarrow \mathrm{h}}{S, H \vdash \mathrm{F2} : \mathrm{h}}\,\textsc{SingleHdr}}{\textsc{Ref}\,\dfrac{S, H \vdash (\mathrm{F2}, 25) : \mathrm{h}}{S, H \vdash \uparrow \mathrm{F2} : \mathrm{h}}}$$

These two constraints force the invocation of the value judgment $60 \Rightarrow \mathrm{min/h}$, which then leads to the following instance of the Mult rule:

$$\textsc{Mult}\,\frac{S, H \vdash \mathrm{F2} : \mathrm{h} \quad S, H \vdash 60 : \mathrm{min/h} \quad \mathrm{min} = \mathrm{h} \bowtie \mathrm{min/h} \quad \mathscr{V}(\mathrm{min})}{S, H \vdash \mathrm{F2} * 60 : \mathrm{min}}$$

Having thus established the unit min for the expression $\mathrm{F2} * 60 - \mathrm{C5}$ (i.e., $d_3 = \mathrm{min}$), we can instantiate the dimension variable $d_1$ also to min.

We are now back to the dimension of the formula $\mathrm{MAX}(\mathrm{F2} * 60 - \mathrm{C5}, 0) * \mathrm{D5}$ for which we had our first instance of the Mult rule. With $d_1 = \mathrm{min}$, we obtain the constraint $\{\$\} = \mathrm{min} \bowtie d_2$ as the third premise. This constraint can be resolved by letting $d_2 = \$/\mathrm{min}$, which finally resolves the dimension for the cell D5. This essentially completes the derivation process for the formula.

A feature of the inference system that was not exhibited by the above example is the suggestion of conversion factors through the ADD rule. We will describe this aspect briefly in the following.

Consider the formula, $A1+A2$, with A1 having unit meters and A2 with the unit centimeters. What happens if we apply the ADD rule to derive the unit of the formula? The first thing we can note is that while the base dimension of both arguments is the same, the dimension component of each cell contains a different factor. This prevents the derivation of a unit for the formula—unless a conversion factor is added. In this case the conversion factor $c_2$ is instantiated to $0.01/1 = 0.01$, which allows the unit of the formula to be meters

$$\textsc{Add}\,\frac{S,H \vdash \mathrm{A1} : \{\mathrm{length}_1^1\} \cup \{\} \quad S,H \vdash \mathrm{A2} : \{\mathrm{length}_1^{0.01}\} \cup \{\} \quad c_1 = 1/1 \quad c_2 = 0.01/1}{S,H \vdash c_1 * \mathrm{A1} + c_2 * \mathrm{A2} : \{\mathrm{length}_1^1\} \cup \{\}}$$

This rule instance says that the formula $A1 + 0.01 * A2$ has the unit meters. The fact that one of the conversion factors had to be instantiated to a value different from 1 indicates a conversion error. Moreover it suggests a remedy for the error, that is, an error message presented to the user cannot only point out the omission of a conversion factor, but also immediately suggest a corrected formula.

## 5. Evaluation

We have implemented a prototype system for performing automatic dimension analysis as an add-in to Microsoft Excel. This tool reuses the header analysis implementation [32] of the UCheck tool [37].

In this section we describe an evaluation of this dimension analysis system to answer the following research questions.

RQ1: *How wide-spread is the use/occurrence of dimensions in spreadsheets*?

Dimension inference can be an effective tool to check formulas and spot errors in spreadsheet computations, but only if those computations involve dimensions, or, to be more precise, if the tool can identify the dimensions involved in the computations. We expect a considerable number of spreadsheets to contain dimensions.

RQ2: *Does dimension inference run effectively on spreadsheets involving dimensions*?

For those spreadsheets that contain dimensions, we would like to know whether or not dimension inference runs correctly, that is, whether it can infer the proper dimensions for values and formulas and whether it can find errors based on inconsistent dimension use in formulas.

RQ3: *To what degree is dimension analysis dependent on the underlying header inference and label analysis*?

Header inference is the first step in dimension analysis. If this step fails to work properly, dimension analysis cannot take off. Following header inference, label inference is the crucial link that ties header information to dimension information. In general, label analysis is complicated by the fact that the process in inherently ambiguous.

Anything that can improve header or label analysis has potentially a great impact on the applicability and accuracy of dimension analysis.

RQ4: *Does dimension validity matter*?

The concept of dimension validity was introduced to make the inference rule MULT stronger so that more dimension errors can be detected. If this additional test helps in practice to detect dimension errors, we can refine the definition of $\mathscr{V}$ to make it even stronger.

### 5.1. Experiments

To answer RQ1 we have employed the EUSES spreadsheet corpus [34], which currently contains 4498 spreadsheets collected from various sources. Dimension analysis is relevant only for those 1977 spreadsheets containing formulas. We ran label analysis on those spreadsheets to find which dimensions occur how often and in how many sheets.

To investigate RQ2 and RQ3 we ran our tool on a subset of 40 spreadsheets randomly selected from the 1977 spreadsheets that contain formulas. We inspected all results, and in cases the header inference or label analysis was not working, we adjusted that information "by hand" and ran only the dimension inference part of the tool.

To investigate RQ4 we have categorized the dimension errors that were reported according to which decision in the inference process led to their discovery. To perform the experiments we had to write some additional scripts and had to perform a few minor instrumentations for the prototype.

### 5.2. Results

RQ1: *How wide-spread is the use/occurrence of dimensions in spreadsheets*?

The distribution of dimensions in the spreadsheets from the EUSES corpus containing formulas is detailed in Table 4, which shows the number of occurrences of dimensions in total and in different spreadsheets.

Altogether, dimensions were found in 487 spreadsheets, that is, in only 1/4th of the spreadsheets with formulas. This number is smaller than the total of 603 from Table 4 since several spreadsheets contain more than one type of dimension.

We found that certain headers were more prevalent than others and had a greater impact. For example, for the dimension money, the two most common results were "Dollars" and "Money", with 201 and 159 occurrences, respectively. For the dimension time, the results were distributed more evenly, with the most common, "Year", occurring 91 times and the least common, "Month", occurring 28 times.

**Table 4**
Occurrences of dimensions.

| Quantity/ | Occurrences | |
|---|---|---|
| dimension | Total | In spreadsheets |
| Money | 390 | 279 |
| Time | 351 | 237 |
| Length | 35 | 26 |
| Mass | 27 | 20 |
| $/h | 20 | 15 |
| Area | 12 | 8 |
| Velocity | 10 | 5 |
| Temperature | 10 | 5 |
| kW | 3 | 3 |
| Mole | 3 | 3 |
| Luminous Intensity | 3 | 2 |
| Total | 864 | 603 |

*RQ*2: *Does dimension inference run effectively on spreadsheets involving dimensions*?

The dimension inference component was able to detect 21 dimension errors in 17 of the 40 randomly selected spreadsheets, that is, we were able to find dimension errors in 42.5% of the spreadsheets that were selected for this study. Not only does this show the effectiveness of the inference mechanism, but it also demonstrates that dimension inference is an effective approach to find errors in spreadsheets.

Due to the number of spreadsheets used in this study, we were able to inspect them manually to verify errors. By looking at the dimensions involved in formulas we were able to discover three false positives, caused by faulty label analysis, and zero false negatives, which would have been any error that was in the spreadsheet but was not caught. Due to the specificity of dimension errors it is possible, if time consuming, to verify that a formula does not contain errors. The first step in this verification was to look at the headers and the dimensions assigned to cells. If the dimension did not match the label this would be a case of faulty label analysis. Once the dimensions were verified, the formulas using these cells were investigated. In the 40 spreadsheets selected for this study there were no cases of dimension errors in formulas that were not caught by this approach.

The found 18 different errors had, due to copies in several rows/columns, altogether 105 error instances.

*RQ*3: *To what degree is dimension analysis dependent on the underlying header inference and label analysis*?

Header inference was able to infer the correct header information in 30 of the 40 spreadsheets that were randomly selected from the 487 candidates. In the 10 other cases headers were placed too distant of the data they were labeling, in some cases having other unrelated data in between the headers and the data. Label analysis worked correctly in all cases. To take the limitations of header inference out of the analysis of dimension inference, we have altered those 10 spreadsheets so that dimension inference could start with proper header information.

In general header inference works very well on spreadsheets that contain a mostly tabular format. The primary problem in most cases in header location as headers are often placed in offsetting columns or rows to the relevant data. One common example occurred in budget spreadsheets where certain labels took up several rows. The system only used the last row as the relevant header, which in some cases caused the loss of dimension information.

We then ran label analysis on all spreadsheets, which was able to map 222 headers into 188 singleton and 34 composite dimensions.

*RQ*4: *Does dimension validity matter*?

Of the 18 correctly identified dimension errors, 2 were invalid dimensions ($\$^2$ in both cases), 1 was detected in an **if** formula, and the remaining 15 errors were all due to violations of the rule Add. Some examples of these include direct formula errors, such as adding the dimensions $ and h, and others involve conflicts with headers, such as the header expecting the result to be $ with the result containing the dimension m.

In general it is hard to determine if the values caused by these errors are incorrect for two primary reasons. First, while the system treats headers as an absolute, one could easily envision a user inputting a header containing a dimension, when in fact one should not be applied. If the headers are assumed to be correct.

### *5.3. Discussion*

We were initially surprised by the overall low occurrence rate of dimensions in spreadsheets, but we later found by inspecting spreadsheets containing formulas that had no dimension that this result is largely due to the kind of spreadsheets that are collected in the repository. For example, among the 1977 spreadsheets with formulas are over 700 grading spreadsheets, which have no dimensions at all.

Label analysis and dimension inference worked very reliably. The weakest link in the chain of steps for dimension analysis was clearly the header inference, which is not too surprising since labeling practices vary widely across spreadsheets.

On the other hand, label analysis can also fail. In fact, the false positives were all caused by label analysis, which in some cases inferred dimension too aggressively. This can happen whenever labels are used to distinguish between different kinds of numbers and are not intended to define the dimension of the numbers. For example, in a table that sums up the number of hourly and day passes, the addition formula would fail because the numbers of different units are added without the use of a conversion factor.

The fact that dimension errors were found in almost half of the selected spreadsheets shows that dimension analysis is an effective tool for uncovering errors in spreadsheets. Even though we found two instances of an error that was due to the concept of invalid dimensions, the number of spreadsheets studied was too small to draw conclusions about the importance of this concept/feature.

## 6. Related work

We have already discussed several approaches to reduce the occurrence of errors in spreadsheets in Section 1. In this section we compare our approach in some more detail with related work that is concerned with label- or annotation-based error checking in spreadsheets. To compare how the different approaches work we use the spreadsheet shown in Fig. 3 that computes for different cars miles-per-gallon numbers and from that their potential range.

There are four errors in this spreadsheet, two caused by incorrect formulas and two caused through the propagation of erroneous dimensions. The first formula error can be found in cell D4, where instead of dividing B4 by C4, the two cells are being added together. Since B4 has the unit Miles and C4 has the unit Gallons, this addition is not dimension correct. The second formula error is located in F3 where the cells E3, with the unit Gallons, and the cell C3, also with the unit Gallons, are being multiplied. Normally, this multiplication would not result in an error; however, the header for this cell, F2, has the unit Miles, which requires all the cells in this column to result in the dimension Miles.

The two propagation errors are caused through the use of cells that have errors. The first is located in cell F4 where D4, which contains an error, is multiplied with E4. The second propagation error is located in F5, which is using the MAX function of column F. MAX requires that all the cells have the same dimension; however, due to the incorrect multiplication in cell F3, this is not the case.

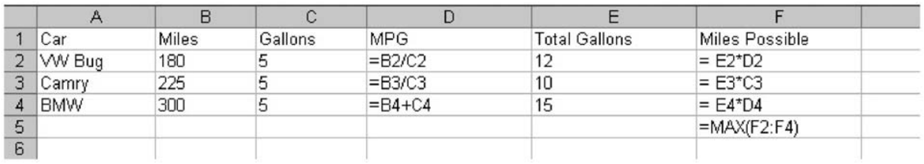
Most closely related to our work is the XeLda system [30] that is designed to check a spreadsheet for units of measurement, such as meters, grams, and seconds. XeLda requires the user to annotate the units for all of the cells in a spreadsheet. Note that this not only includes data cells, but also all formula cells. For example, a user could annotate Fig. 3 by specifying that the cells B2, B3, and B4 are (Miles, 1), which is the XeLda notation for a dimension that has one component, Miles, whose exponent is 1. The cells C2, C3, and C4 would have to be annotated by the unit (Gallons, 1), and the cells D2, D3, and D4 would be annotated with the unit (Miles, 1) (Gallons, −1), representing Miles per Gallon, and so on. While analyzing a spreadsheet XeLda checks the annotated units against the results of formulas to insure correctness. For example, if the unit (Meters, 1) is multiplied with another unit of (Seconds, 1), the result will be (Meters, 1) (Seconds, 1). The unit determined with the formula is then compared to the annotated unit to determine any inconsistencies, which are shaded yellow and contain error messages. Fig. 4 shows the feedback produced by XeLda when run on the sheet in Fig. 3 after it has been annotated.

The advantage of the XeLda approach is that it works well independently of the spreadsheet layout, whereas our approach depends on header and label analysis. On the other hand, XeLda's disadvantage is the huge amount of extra work required by the user whereas our approach is fully automatic. Moreover, XeLda cannot infer conversion factors.

UCheck [37] was designed to check for units in a spreadsheet, and as such it does not handle dimensions. UCheck works by inferring headers for all the cells in a spreadsheet, based on the structure and content of the spreadsheet. Once these headers are inferred, the system derives units for the cells and checks for unit errors. In the given example, UCheck can find only the error in the aggregation formula in F5 that is caused by inconsistent units derived for the argument cells. (The error message is also misleading in this case.) The results for UCheck are shown in Fig. 5.

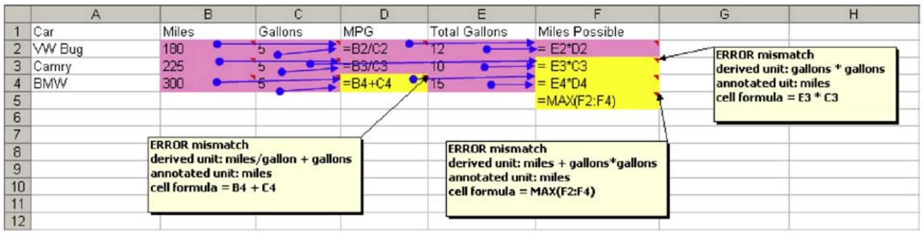
While UCheck works completely automatically, some other related approaches require the user to annotate the spreadsheet with label information [27,28]. The same advantages and disadvantages that we have mentioned for XeLda apply here as well.

SLATE [38] separates the unit from the object of measurement and defines semantics for spreadsheets so that the unit *and* the object of measurement are considered. In SLATE, every expression has three attributes: a value, a



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Car | Miles | Gallons | MPG | Total Gallons | Miles Possible |
| 2 | VW Bug | 180 | 5 | =B2/C2 | 12 | = E2*D2 |
| 3 | Camry | 225 | 5 | =B3/C3 | 10 | = E3*C3 |
| 4 | BMW | 300 | 5 | =B4+C4 | 15 | = E4*D4 |
| 5 | | | | | | =MAX(F2:F4) |
| 6 | | | | | | |

**Fig. 3.** System comparison example.



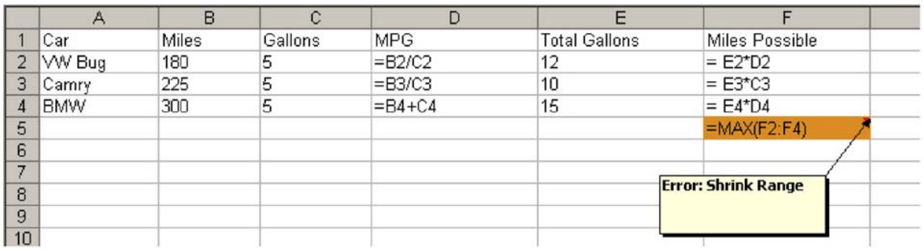
| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Car | Miles | Gallons | MPG | Total Gallons | Miles Possible | | |
| 2 | VW Bug | 180 | 5 | =B2/C2 | 12 | = E2*D2 | | |
| 3 | Camry | 225 | 5 | =B3/C3 | 10 | = E3*C3 | | |
| 4 | BMW | 300 | 5 | =B4+C4 | 15 | = E4*D4 | | |
| 5 | | | | | | =MAX(F2:F4) | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |

ERROR mismatch
derived unit: gallons * gallons
annotated uit: miles
cell formula = E3 * C3

ERROR mismatch
derived unit: miles/gallon + gallons
annotated unit: miles
cell formula = B4 + C4

ERROR mismatch
derived unit: miles + gallons*gallons
annotated unit: miles
cell formula = MAX(F2:F4)

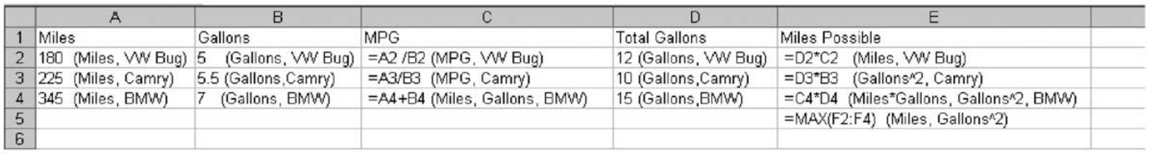
**Fig. 4.** XeLda results.



| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Car | Miles | Gallons | MPG | Total Gallons | Miles Possible |
| 2 | VW Bug | 180 | 5 | =B2/C2 | 12 | = E2*D2 |
| 3 | Camry | 225 | 5 | =B3/C3 | 10 | = E3*C3 |
| 4 | BMW | 300 | 5 | =B4+C4 | 15 | = E4*D4 |
| 5 | | | | | | =MAX(F2:F4) |

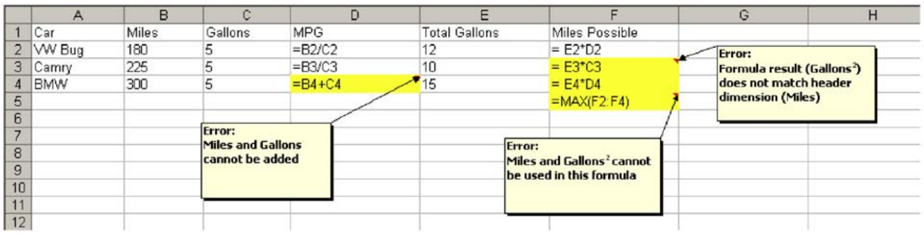
Error: Shrink Range

**Fig. 5.** UCheck results.



| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | Miles | Gallons | MPG | Total Gallons | Miles Possible |
| 2 | 180 (Miles, VW Bug) | 5 (Gallons, VW Bug) | =A2 /B2 (MPG, VW Bug) | 12 (Gallons, VW Bug) | =D2*C2 (Miles, VW Bug) |
| 3 | 225 (Miles, Camry) | 5.5 (Gallons,Camry) | =A3/B3 (MPG, Camry) | 10 (Gallons,Camry) | =D3*B3 (Gallons^2, Camry) |
| 4 | 345 (Miles, BMW) | 7 (Gallons, BMW) | =A4+B4 (Miles, Gallons, BMW) | 15 (Gallons,BMW) | =C4*D4 (Miles*Gallons, Gallons^2, BMW) |
| 5 | | | | | =MAX(F2:F4) (Miles, Gallons^2) |
| 6 | | | | | |

**Fig. 6.** SLATE results.



| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Car | Miles | Gallons | MPG | Total Gallons | Miles Possible | | |
| 2 | VW Bug | 180 | 5 | =B2/C2 | 12 | = E2*D2 | | |
| 3 | Camry | 225 | 5 | =B3/C3 | 10 | = E3*C3 | | |
| 4 | BMW | 300 | 5 | =B4+C4 | 15 | = E4*D4 | | |
| 5 | | | | | | =MAX(F2:F4) | | |

Error:
Miles and Gallons cannot be added

Error:
Miles and Gallons$^2$ cannot be used in this formula

Error:
Formula result (Gallons$^2$) does not match header dimension (Miles)

**Fig. 7.** Dimension inference results.

unit, and a label. The value is what is contained in a cell. Units, such as meters, kilograms, and seconds, capture information about the scale at which the measurement was taken and the dimensions of the measurement. The final attribute, labels, defines characteristics of the objects of measurement. For example, a cell referring to 25 pounds of apples might read “25 lbs. (apples)”. Like XeLda, this system requires the user to annotate the spreadsheet before the analysis can begin. This annotation involves adding the units and labels to all cells containing no references to other cells. The system then analyzes the cells with formulas containing references and determines the unit and label for these cells. The annotations and results from this system are shown in Fig. 6.

This system does not explicitly show errors, but by investigating the system-generated labels the user can see which cells may have problems. For example, the system-generated annotation for cell C4 is (Miles, Gallons, BMW). Compared to the rest of the cells in this column, this annotation appears out of place. This annotation should cause the user to inspect the formula and see that instead of doing a division, the cell is performing an addition. The other errors in this spreadsheet have similar annotation problems.

As a comparison, Fig. 7 shows the results of dimension inference on the spreadsheet in Fig. 3. Dimension inference is able to highlight the errors in the spreadsheet and does so without requiring any annotation of the spreadsheet.

The two most closely related systems with regard to dimension checking are probably XeLda and the presented dimension inference. They both seem to do the best job of recognizing dimension errors. However, both of these systems handle dimensions differently and because of this have different strengths and weaknesses.

XeLda requires that users annotate all the cells in a sheet before the analysis will work properly. This can be time consuming for the user and represents the major weakness of the system. It is hard to make sure that the user will correctly annotate the cells. If this is not done, it is not possible for the system to check for errors.

On the other hand, if the annotation is done correctly, the system will know the units for all of the cells. This enables it to be confident of the results. The strength of XeLda also shows one weakness of dimension inference. Automatically determining the dimensions in a spreadsheet depends on proper header information and also on a mapping of the header labels to dimensions, which can be ambiguous, as we have mentioned in Section 5.3.

One of the strengths of dimension inference is the fact that users do not have to specifically enter the dimensions for any of the cells. As long as the headers are fairly clear and contain dimensions, checking a spreadsheet is as easy as clicking a button. As our experiments have shown dimension inference can find errors and does a good job of checking spreadsheets when the headers are in order.

## 7. Conclusions and future work

We have introduced a system for inferring and checking dimensions in spreadsheets. By interpreting headers as dimensions and relating those to formulas, the system can identify errors in formulas, because dimensions place constraints on how operations can act on values. Our system differs from previous approaches in the following ways:

- *No user annotation required.* The system can infer labels for cells automatically and use them to determine what the expected dimension of a cell is. This aspect greatly enhances the usability of the system since it minimizes the amount of work a user has to do in order to use the system.
- *Dimension inference.* The system even works well in situations when only partial dimension information is available since dimension constraints can also be propagated upstream of computations through the defined dimension inference rules. This aspect contributes to the flexibility and robustness of the presented system.
- *Conversion factors allow different units of measurement.* Some formulas in a spreadsheet require conversion of quantities whenever non-compatible dimensions, such as meters and feet, are involved in additive operations. The described rule system can identify required conversion factors.
- *Dimension instantiation.* As a consequence of dimension inference there are situations in which dimension variables remain unsolved at the end of the analysis. These “dimension templates” can be instantiated to the most likely dimensions expected.

The presented evaluation has demonstrated that our system works well in practice and can detect errors in many cases. The evaluation has also revealed three promising directions for future research.

First, we could improve the system with a more accurate header inference. One way to do this would be to use *header patterns*, which categorize the different ways that headers are used in spreadsheets. This information can be used to increase the ability to recognize headers. For example, if a spreadsheet can be identified as having a certain header pattern, then the headers will be set up in a certain way. This may reduce the work it takes to correctly identify headers and it can also be used to better resolve ambiguous cases.

Second, a combination of dimension inference with the purely label-based approaches as pursued by UCheck [37] or the system described in Ahmad et al. [28] could strengthen the reasoning of the system. To some degree this was already tried in the SLATE approach [38]. However, SLATE only transforms labels and dimensions and does not identify errors. Moreover, the fact that SLATE is a stand-alone spreadsheet system and cannot be integrated into Excel renders the approach currently impractical.

One way to combine these two methods is to try and gather both label and dimension information about a cell. In many spreadsheets containing dimensions only one axis contains any relevant dimension information. The other axis typically contains labels, which provides structural information that can be exploited by the reasoning system behind UCheck.

To explain this idea in more detail, we will again use the example from Fig. 3. This spreadsheet contains several dimensions on the horizontal axis (Row 1), but dimensionless labels on the vertical axis (Column A). In this particular example, we will focus on the formula E2*D2, which is located in cell F2. Assume now that the formula is changed to E3*D2, that is, mistakenly referencing E3 instead of E2. Dimension inference would have no problem with this formula, because the result is still valid and matches the rest of the dimensions in the column.

To try and catch this error we can assign the labels from Column A to specific rows. For example, the cells in Row 2 would have the label “VW Bug”. With this information assigned, the system can then check to ensure that formulas are also label correct. The formula E3*D2 is multiplying a cell, E3, with the unit Gallons and the label “Camry” with the cell D2, which has the unit Miles/Gallon and the label “VW Bug”. The system would be able to catch this label inconsistency and report it to the user.

Third, the system may be able to be combined with manual dimension checking techniques, to get the best possible results. In many cases there is not enough dimension information to determine if a function is actually correct. For example, any multiplication resulting in a valid dimension is accepted. However, this does not insure that the formula is actually correct. If the header of the resulting formula does not contain any dimension information, nothing can be done to validate the result. By pointing out these problem areas the system could help the user determine where to add dimension information to labels.

## Acknowledgments

This work is partially supported by the National Science Foundation under the Grant ITR-0325273 and by the EUSES Consortium (http://EUSESconsortium.org).

## Appendix

Below is a list of the 40 spreadsheets that were randomly selected from those 487 spreadsheets of the

EUSES spreadsheet corpus [34] that had formulas and dimensions in them.

```
cs101/posey_Q1.xls
cs101/ACT3_sec23_smith.xls
database/cost_spreadsheet.xls
financial/2001financialstatements.xls
financial/financialanalysis.xls
financial/ratioanal.xls
financial/English%20financial%252#A7F43.xls
financial/FIN%20YEAR%201%20APRI#A7B18.xls
financial/financial_table_overview.xls
finanical/gaap_q4_03.xls
financial/INCOMESTMT.xls
financial/Q3_Final.xls
financial/summ0602.xls
financial/XI-Sample-Budget-Fina#A7BA6.xls
financial/am_skandia_fin_supple#A80EE.xls
financial/5_year_summary3.xls
financial/90001.xls
financial/FinStmtEx.xls
financial/NOTES2-00xls1.xls
financial/Stats%202001_results.xls
financial/quaterlyreport.xls
financial/3yrsegment_table.xls
finanial/Financial%20Compariso#A7ED8.xls
filby/ACTENER.xls
filby/PLANK.xls
filby/TOUGHEX.xls
forms3/treasury.reichwja.xl97
inventory/a2-33.xls
inventroy/InvCtrlForm.xls
inventory/dairywateruse1.xls
inventory/pigskin.xls
homework/chart.xls
homework/compost homework.xls
homework/expenses_ans.xls
homework/DfGH35Clnf.xls
homework/HMWK92903.xls
homework/HW01.D.xls
homework/hw8.xls
homework.hw08a.xls
homework/Evns4-5.xls
```

## References

[1] C. Scaffidi, M. Shaw, B. Myers, Estimating the numbers of end users and end user programmers, in: IEEE Symposium on Visual Languages and Human-Centric Computing, 2005, pp. 207–214.
[2] K. Rajalingham, D. Chadwick, B. Knight, D. Edwards, Quality control in spreadsheets: a software engineering-based approach to spreadsheet development, in: 33rd Hawaii International Conference on System Sciences, 2000, pp. 1–9.
[3] S. Ditlea, Spreadsheets can be hazardous to your health, Personal Computing 11 (1) (1987) 60–69.
[4] EuSpRIG, European spreadsheet risks interest group. ⟨http://www.eusprig.org/⟩.
[5] B. Ronen, M.A. Palley, H.C. Lucas Jr., Spreadsheet analysis and design, Communications of the ACM 32 (1) (1989) 84–93.
[6] A.G. Yoder, D.L. Cohn, Real spreadsheets for real programmers, in: International Conference on Computer Languages, 1994, pp. 20–30.
[7] T. Isakowitz, S. Schocken, H.C. Lucas Jr., Toward a logical/physical theory of spreadsheet modelling, ACM Transactions on Information Systems 13 (1) (1995) 1–37.
[8] S.G. Powell, K.R. Baker, The Art of Modeling with Spreadsheets: Management Science, Spreadsheet Engineering, and Modeling Craft, Wiley, New York, 2004.
[9] M. Erwig, R. Abraham, I. Cooperstein, S. Kollmansberger, Automatic generation and maintenance of correct spreadsheets, in: 27th IEEE International Conference on Software Engineering, 2005, pp. 136–145.
[10] M. Erwig, R. Abraham, S. Kollmansberger, I. Cooperstein, Gencel—a program generator for correct spreadsheets, Journal of Functional Programming 16 (3) (2006) 293–325.
[11] R. Abraham, M. Erwig, S. Kollmansberger, E. Seifert, Visual specifications of correct spreadsheets, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2005, pp. 189–196.
[12] G. Engels, M. Erwig, ClassSheets: automatic generation of spreadsheet applications from object-oriented specifications, in: 20th IEEE/ACM International Conference on Automated Software Engineering, 2005, pp. 124–133.
[13] J.-C. Bals, F. Christ, G. Engels, M. Erwig, ClassSheets—model-based, object-oriented design of spreadsheet applications, Journal of Object Technologies 6 (2007).
[14] R.R. Panko, Applying code inspection to spreadsheet testing, Journal of Management Information Systems 16 (2) (1999) 159–176.
[15] J. Sajaniemi, Modeling spreadsheet audit: a rigorous approach to automatic visualization, Journal of Visual Languages and Computing 11 (2000) 49–82.
[16] R. Mittermeir, M. Clermont, Finding high-level structures in spreadsheet programs, in: 9th Working Conference on Reverse Engineering, 2002, pp. 221–232.
[17] E.J. Weyuker, S.N. Weiss, D. Hamlet, Comparison of program testing strategies, in: Fourth Symposium on Software Testing, Analysis and Verification, 1991, pp. 154–164.
[18] G. Rothermel, M.M. Burnett, L. Li, C. DuPuis, A. Sheretov. A methodology for testing spreadsheets, ACM Transactions on Software Engineering and Methodology (2001) 110–147.
[19] M.M. Burnett, A. Sheretov, B. Ren, G. Rothermel, Testing homogeneous spreadsheet grids with the "What You See Is What You Test" methodology, IEEE Transactions on Software Engineering 29 (6) (2002) 576–594.
[20] M. Fisher II, M. Cao, G. Rothermel, C. Cook, M.M. Burnett, Automated test case generation for spreadsheets, in: IEEE International Conference on Software Engineering, 2002, pp. 141–151.
[21] R. Abraham, M. Erwig, AutoTest: a tool for automatic test case generation in spreadsheets, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2006, pp. 43–50.
[22] R. Abraham, M. Erwig, Mutation operators for spreadsheets, IEEE Transactions on Software Engineering 35 (1) (2009) 94–108.
[23] A. Phalgune, C. Kissinger, M. Burnett, C. Cook, L. Beckwith, J. Ruthruff, Garbage in, garbage out? An empirical look at oracle mistakes by end-user programmers, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2005, pp. 45–52.
[24] R. Abraham, M. Erwig, GoalDebug: a spreadsheet debugger for end users, in: 29th IEEE International Conference on Software Engineering, 2007, pp. 251–260.
[25] R. Abraham, M. Erwig, Test-driven goal-directed debugging in spreadsheets, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2008, pp. 131–138.
[26] J. Lawrence, R. Abraham, M.M. Burnett, M. Erwig, Sharing reasoning about faults in spreadsheets: an empirical study, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2006, pp. 35–42.
[27] M. Erwig, M.M. Burnett, Adding apples and oranges, in: 4th International Symposium on Practical Aspects of Declarative Languages, LNCS, vol. 2257, 2002, pp. 173–191.
[28] Y. Ahmad, T. Antoniu, S. Goldwater, S. Krishnamurthi, A type system for statically detecting spreadsheet errors, in: 18th IEEE International Conference on Automated Software Engineering, 2003, pp. 174–183.
[29] R. Abraham, M. Erwig, S. Andrew, A type system based on end-user vocabulary, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2007, pp. 215–222.
[30] T. Antoniu, P.A. Steckler, S. Krishnamurthi, E. Neuwirth, M. Felleisen, Validating the unit correctness of spreadsheet programs, in: 26th IEEE International Conference on Software Engineering, 2004, pp. 439–448.
[31] D. Isbell, Mars climate orbiter team finds likely cause of loss, September 1999. ⟨http://mars.jpl.nasa.gov/msp98/news/mco990930.html⟩.

[32] R. Abraham, M. Erwig, Header and unit inference for spreadsheets through spatial analyses, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2004, pp. 165–172.

[33] C. Chambers, M. Erwig, Dimension inference in spreadsheets, in: IEEE International Symposium on Visual Languages and Human-Centric Computing, 2008, pp. 123–130.

[34] M. Fisher, G. Rothermel, The EUSES spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanism, in: 1st Workshop on End-User Software Engineering, 2005, pp. 47–51.

[35] B.C. Pierce, Types and Programming Languages, MIT Press, Cambridge, MA, 2002.

[36] R. Rowlett, A dictionary of units of measurement, July 2005. ⟨http://www.unc.edu/~rowlett/units/index.html⟩.

[37] R. Abraham, M. Erwig, UCheck: a spreadsheet unit checker for end users, Journal of Visual Languages and Computing 18 (1) (2007) 71–95.

[38] Michael J. Coblenz, Andrew J. Ko, Brad A. Myers, Using objects of measurement to detect spreadsheet errors, in: IEEE Symposium on Visual Languages and Human-Centric Computing, 2005, pp. 314–316.