AN ABSTRACT OF THE THESIS OF

Chris Chambers for the degree of Master of Science in Computer Science presented on June 3, 2010.

Title: Dimension Checking Tools for Spreadsheets

Abstract approved: ______________________________

Martin Erwig

We present the evolution of a reasoning system for inferring dimension information in spreadsheets. The three papers included in this thesis show how the initial system can be used to check the consistency of spreadsheet formulas and thus is able to detect errors in spreadsheets, and the evolution to a system that can check both label and dimension errors.

The approach for these systems is based on three static analysis components. First, the spatial structure of the spreadsheet is analyzed to infer the labels for specific cells. Second, those cells that are identified as labels are analyzed to determine dimension information. Once this is completed the system, will look at formulas and, using specific rules, will determine if the dimensions and labels are correct. 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.

These systems have been implemented as a add-in for Excel, and this prototype has allowed us to perform several evaluations on the systems. These evaluations show that the systems can be effective in detecting dimension errors, with the initial system

detecting errors in 50% of the investigated spreadsheets, and the subsequent systems having similar success. In addition these evaluations show that by adding label checking, the effectiveness and efficiency of the system is improved with many previously undetected errors being found.

Dimension Checking Tools for Spreadsheets

by

Chris Chambers

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 3, 2010
Commencement June 2011

ACKNOWLEDGEMENTS

I would like to thank Martin Erwig for his help regarding research and for working with me on several papers. I would also like to thank Gil and Bobbie Beck who sponsered me during the first 5 years of college.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

DEDICATION

To my friends and family that have made this possible and without whom the journey would not have been worth taking.

Chapter 1 – Introduction

Each year hundreds of millions of spreadsheets are created for a variety of tasks, including grades and finances. Spreadsheets allow users to easily store and manipulate data. However, there is nothing in place to ensure that these manipulations are valid. An invalid manipulation can diminish the correctness of a spreadsheet and cause errors.

Several studies have shown that existing spreadsheets contain an alarming number of errors [33, 30, 12]. Some studies even report that 90% or more of real-world spreadsheets contain errors [40]. These errors can often cause severe problems for corporations, and there have been many examples where companies have lost thousands or even millions of dollars just because of a simple error in a spreadsheet. One recent example of this is Kodak. An error in a spreadsheet led to $11 million in losses, caused them to have to restate their earnings, and damaged their reputation and their employees trust [28].

There are several approaches that can be used to improve the quality of spreadsheets. These approaches are generally focused in three main areas: prevention, detection, and removal of errors from spreadsheets.

Prevention approaches are varied and involve everything from guidelines to spreadsheet design, spreadsheet template creation, and the automatic generation of spreadsheets. Often these preventive approaches require a new application and can be disruptive to the common spreadsheet creation process. For example, while the spreadsheet generator Gencel can be useful as a way to apply templates to spreadsheets and thereby reducing errors, it requires an additional program in a new language that must be learned by the user. This can be very time consuming.

Because of this, much research has focused on the detection and removal of errors. Due to the varied nature of spreadsheets, there are many different types of errors that can occur, including formula errors, format errors, and data entry errors. Errors such as those that occur in formulas can sometimes be a basic check to determine if the correct cells are being used. However, there are also ways that the structure and design of the spreadsheet can be taken advantage of.

A primary example is label based reasoning, and there are several systems that are designed to directly take advantage of the labels and the structure of spreadsheets. These systems, such as UCheck [4] or the system described in [9], are designed to find formula errors caused by inconsistent label usage and typically operate in two distinct phases. The first phase will identify label information for the entire spreadsheet. Some systems require users to annotate the labels for every cells, while some are able to infer the correct labels. In the second phase this information is analyzed to find errors, such as inconsistent labels in addition formulas.

Another type of error that can be detected in spreadsheets is dimension errors, which occur when units of measurement are used incorrectly in formulas. Units of measurements are well known among end users [10] and are used to characterize different kinds of values, much like the type systems used in general-purpose programming languages. However, there is an important difference between this and traditional type systems, namely that objects, such as integers, which have just one type, are able to represent different kinds of quantities, such as length or time values.

Several systems [10, 14, 17] have been developed in order to deal with unit of measurement errors. Dimension inference [14], a method presented later in this thesis, can be used to automatically find dimension errors in spreadsheets. This approach has been shown to work reliably and effectively in many cases, and can detect many cases of dimension errors in spreadsheets.

While systems for label-based reasoning and unit-of-measurement errors have been shown to work effectively on spreadsheets, they are rather specific and focus on simply one type of error. However, in most cases users would like to check for multiple types of errors. To more effectively bring type systems to spreadsheets, these methods could be combined to take full advantage of the information provided in the spreadsheet and allow it to check for both types of mistakes.

Both label analysis and dimension analysis systems rely on header and label information. However, dimension analysis does not take advantage of the structure, it simply checks to make sure that formulas are dimension correct. By combining dimension analysis with the purely label-based approaches the structure of the spreadsheet could be used to help strengthen the reasoning of the system.

This thesis is divided into three sections, each containing a published paper that describes a method to check spreadsheets for errors. These papers are all related, and, starting with the description of the original Dimension Inference, show the natural progression of the system to its current state.

The first paper [14] presents the Dimension Inference system that will check a spreadsheet for unit-of-measurement errors. This system can be used to check the consistency of formulas and detect dimension errors in spreadsheets as well as infer dimensions. This system was developed as a prototype add-in to Excel which allowed us to test the system to determine how well it detected these errors. To analyze Dimension Inference, this paper presents a study which involved running the system on 40 spreadsheets to determine how many errors were detected and how many were missed.

The second paper contained in this thesis [15] is an expansion upon and an update of the previous paper. It contains an updated set of rules as well as an expanded results section. In addition it contains an in-depth example of a rule instantiation which shows in more detail how the system works and how errors are detected.

The final paper [16] describes a way to combine label-based reasoning with dimension inference. In many cases a spreadsheet contains dimensions on only one axis, either row or column, while the other axis typically contains labels that do not map to any dimension. These labels, which go unused in dimension inference, can be used to provide structural information that can be exploited by the reasoning system behind UCheck. This paper presents a system that combines these two reasoning systems and a comparison of this method and Dimension Inference when run on the EUSES spreadsheet repository.

These papers show the evolution of a type system for spreadsheets. This system integrates the benefits of label checking with the increased effectiveness of detecting unit errors.

Dimension Inference in Spreadsheets

Chris Chambers and Martin Erwig

VLHCC ’08: Proceedings of the 2008 IEEE Symposium on Visual

Languages and Human-Centric Computing

Best Paper Award

Chapter 2 – Dimension Inference in Spreadsheets

2.1 Introduction

Spreadsheets are widely used [44] end-user programs that contain many errors [38] with a substantial negative impact on society [23]. To improve the quality of spreadsheets a variety of approaches to prevent, detect, and remove errors from spreadsheet have been investigated. 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 two major approaches to detect errors are testing/auditing and static (type) checking. For example, the "What You See Is What You Test" approach [42] that uses data-flow adequacy and coverage criteria to give the user feedback on how well tested the spreadsheet is. Test-case generation systems [25, 2] can support users in their testing efforts. However, a problem with testing is that is suffers from oracle mistakes (that is, incorrect decisions made by users during testing) [34]. Even though some of these problems can be alleviated by automating parts of the testing/debugging process [3], testing is also problematic because it requires substantial effort on part of the user, which poses a serious challenge since many of the spreadsheet users are end users who mainly want to get their job done and are much less motivated than professional software developers to spend additional time on their spreadsheets for testing purposes.

This latter aspect makes type checking approaches attractive since they promise mostly automatic error detection. Two immediate problems with type checking are that they are limited in the kinds of errors they find and 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 [29]. 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 [22, 9]. 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 [7]. Second, we can employ units of measurements, or dimensions, as a concrete notion of types that is well known among end users [10]. 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.

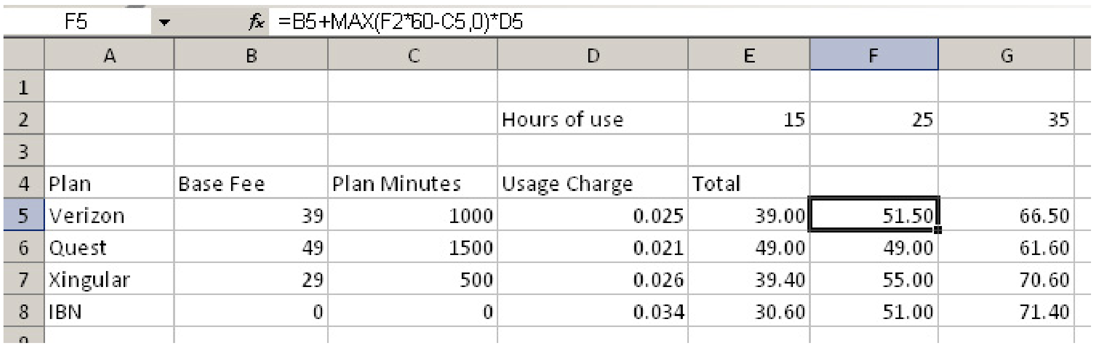
In this paper we describe dimension inference, a method to automatically find dimension errors in spreadsheets. Our work builds on previous approaches and extends them in several important ways. First, through incorporating header inference [1], the presented system does not have to rely on additional 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 all 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.

The rest of this paper is structured as follows. In Section 2.2 we illustrate the issues involved in dimension checking and inference with a small example. In Section 2.3 we formalize spreadsheets and a model of dimensions. We introduce in Section 2.4 a characterization of valid dimensions that is expected to be useful in practice to detect more errors. The process of dimension inference is then described in Section 2.5. In Section 2.6 we report on an evaluation of a prototypical implementation of a tool for dimension analysis. We discuss related work in Section 2.7 and give conclusions and ideas for future work in Section 2.8.

2.2 An Example

Figure 2.1 shows a spreadsheet for computing costs of different phone plans for different companies and 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 hours, the formula in cell F5 is as follows.

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



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

Figure 2.1: Example spreadsheet

By inspecting the labels in the spreadsheet we can see that the value in cell B5, 39, represents a money amount, which could be without further information given in any currency. It makes sense for a system to assume whatever 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 at all what dimension the value in D5 has.

Given (partial) information about the dimensions of values we can reason about 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/header and is so far unknown. In the example, we see that C5 is subtracted from F2*60. Since all additive operations require that the arguments have the same dimension, we can conclude that F2*60 must be minutes, which is possible if the constant 60 has the dimension minutes/hour. In fact, only the value 60 has this dimension.[1] In other words, any other factor (or no factor at all) would have caused a mistake in this formula.

Since the dimension behavior of MAX is the same as that of other addition operators, we can infer that 0 and the whole expression MAX(F2*60-C5,0) also have the dimension minutes. Here we can observe that the ability to infer dimensions 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 final two steps are to figure out the correct dimension for D5 so that the sum with B5 is dimension correct. Since B5 is in \$, the product MAX(F2*60-C5,0)*D5 must have the same dimension. Since the MAX expression is in minutes, we can therefore conclude that D5 must have the dimension \$/minute.

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

2.3 Spreadsheets and Dimensions

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) or references to other cells ($\uparrow a$), or are constructed using arithmetic ($+$ or $*$), aggregating (**count**), or conditional operators.

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

Here $+$ and $*$ represent, respectively, a whole class of additive operators (including $-$ and MAX) and multiplicative operators (including $/$).

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). (We can view a dimension as 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 (δ). If a dimension contains only one component, it is called a *singleton dimension*, whereas a dimension that contains more than one component is called a *composite dimension*. The *identity dimension* $\{\}$ is used for dimensionless values.

$$\begin{aligned} d \quad &::= \quad \{c, \ldots, c\} \\ c \quad &::= \quad b_f^n \mid \delta \end{aligned}$$

For each base dimension we identify a default unit with factor 1. For example, the default for length is meter (m), that is, m $=$ length_1^1, which also means that cm $=$ $\text{length}_{0.01}^1$ and ft $=$ $\text{length}_{0.3}^1$. In general, the following relationship holds (where x is a dimensionless number and b is an arbitrary base).

$$x b_f^n = x f b_1^n$$

We may also omit conversion factors and exponents of 1 for brevity, that is, we write more shortly b^n for b^n_1, b_f for b^1_f, 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 [24]. The quantities and their default units are shown in Table 2.1.

Table 2.1: 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)

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

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

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

We have seen examples of conversion factors in Figure 2.1, namely min = time_{60} and hr = 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 money_ϕ, but it is not clear in which currency. If B4 were added in some formula to a value that is known to be of dimension \$ = money_1 or 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 dimension \$ or cent, respectively.

Representing conversion by just a 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 out of 487 spreadsheets contained dimensions that could not be converted using the presented model.

The use of dimensions in computations effectively restricts the allowed computations in the sense of typing annotations. Consider the following definition of the semantics for addition.

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

When the semantics is based on values that are annotated with dimensions, the rule becomes the following.

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

This definition 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, $b^n_f \in d \wedge b^m_{f'} \in d' \wedge f \neq f'$. Otherwise, we have

$$d \bowtie d' = \{b^{n+m}_f \mid b^n_f \in d \wedge b^m_f \in d'\} \cup d \triangle d'$$

where the symmetric difference, $d \triangle d'$, 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 seconds when its speed is 950 km/hr, one has to use a conversion factor with dimension hr/s in the multiplication, otherwise $\bowtie$ is undefined, and the rule cannot be applied.

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

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.

2.4 Dimension Validity

Dimensions span a 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 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.

It is an interesting scientific (or even philosophical) question 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 [37]. 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 three exceptions. The allowed exponent ranges are defined by the table below.

Table 2.2: Dimension Validy

b	$\mathscr{R}(b)$
length	-3 .. 3
electric current	-2 .. 1
time	-3 .. 2
all others	-1 .. 1

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

$$\mathscr{E}(d) = d = \mathrm{kg}^{-1}\mathrm{m}^{-2}\mathrm{s}^{4}\mathrm{A}^{2} \vee \tag{2.1}$$

$$d = \mathrm{kg}^{-1}\mathrm{m}^{-2}\mathrm{s}^{3}\mathrm{A}^{2} \tag{2.2}$$

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.

2.5 Dimension Analysis

The dimension analysis of a spreadsheet goes through the following four distinct steps. The last step applies only in those cases when the third step produces underspecified dimensions, that is, when it results in inferred dimensions that contain dimension variables.

1. Header inference
2. Label analysis

3. Dimension inference
4. Dimension instantiation

In the following we will describe these four steps in some detail.

2.5.1 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 Figure 2.1 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. It can also take into account layout information. Techniques for header inference have been described in detail elsewhere [1, 4]. In the context of this paper we simply reuse those techniques.

2.5.2 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 in Figure 2.1 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. If no part of a header label can be mapped to a dimension other than $\{\}$, the label is mapped to a dimension variable δ, which indicates that the dimension is at this time unknown.

2.5.3 Dimension Inference

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

$$\frac{H(a)=\{a\} \qquad S(a)\Rightarrow d}{S,H\vdash a:d} \qquad \frac{H(a)=\varnothing}{S,H\vdash a:\delta} \qquad \textsc{Val}\ \frac{S,H\vdash a:d}{S,H\vdash(a,v):d}$$

$$\textsc{Ref}\ \frac{S,H\vdash(a,S(a')):d \qquad S,H\vdash a:d}{S,H\vdash(a,\uparrow a'):d}$$

$$\textsc{Add}\ \frac{\begin{array}{c}S,H\vdash(a,e_1):\{b^n_{f_1}\}\cup d\\ S,H\vdash(a,e_2):\{b^n_{f_2}\}\cup d \qquad c_1=f_1/f \qquad c_2=f_2/f \qquad S,H\vdash a:\{b^n_f\}\cup d\end{array}}{S,H\vdash(a,c_1*e_1+c_2*e_2):\{b^n_f\}\cup d}$$

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

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

Figure 2.2: Dimension inference rules

The third step of dimension analysis is dimension inference, which inspects each cell containing a formula and derives for it a dimension using a system of rules given in Figure 2.2. 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 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 it does not apply.

Dimension inference is defined through the following three judgments that tie together dimensions inferred from headers/labels, known 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 and prior knowledge of conversion factors, such as $60 \Rightarrow \text{min}/\text{hr}$, $60 \Rightarrow \text{s}/\text{min}$, or $100 \Rightarrow \text{cm}/\text{m}$. 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 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 Figure 2.1 we have $S,H \vdash \text{C5} : \text{min}$.

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 Figure 2.1 and H is the corresponding header relationship, then we obtain $S,H \vdash (\text{F5}, \text{B5+MAX(F2*60-C5,0)*D5}) : \$$. How this result is obtained was explained informally in Section 2.2. The rules given in Figure 2.2 formalize this process.

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. Figure 2.2 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, SUM, etc.). 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.

2.5.4 Dimension Instantiation

An inferred dimension might contain dimension variables and/or conversion-factor variables. The occurrence of variables happens whenever the spreadsheet doesn't 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, length$_\phi$ can be m, cm, or any other length dimension that can be obtained by substituting values for ϕ. 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 2.4). Of those valid dimensions we can then select the one that is most common (as indicated by the numbers to be reported in Section 2.6).

2.6 Evaluation

We have implemented a tool for performing automatic dimension analysis as an add-in to Microsoft Excel. This tool reuses the header analysis implementation [1] of the UCheck tool [4]. 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.

2.6.1 Experiments

To answer RQ1 we have employed the EUSES spreadsheet corpus [24], 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.

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.

2.6.2 Results

The occurrence of dimensions in the spreadsheets from the EUSES corpus containing formulas is detailed in Table 2.3, 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 2.3 since several spreadsheets contain more than one type of dimension.

Table 2.3: Occurrences of dimensions

Quantity/ Dimension	**Occurrences**	
	Total	**In Spreadsheets**
Money	390	279
Time	351	237
Length	35	26
Mass	27	20
$/hr	20	15
Area	12	8
Velocity	10	5
Temperature	10	5
kW	3	3
Mole	3	3
Luminous Intensity	3	2
Total	864	603

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

Header inference was able to infer the correct header information in 30 of the 40 cases. 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.

Dimension inference was then successfully run on all spreadsheets and was able to map 222 headers into 188 singleton and 34 composite dimensions. The tool detected 21 dimension errors in 17 of the 40 spreadsheets. To verify these errors, the spreadsheets were manually inspected, leading to the discovery of three false positives (caused by the label analysis) and zero false negatives. Of the 18 correctly identified dimension errors, 2 were invalid dimensions ($\$^2$ in both cases), and 1 error was detected in an **if** formula, and the remaining 15 errors were all due to violations of the rule ADD. The found 18 different errors had, due to copies in several rows/columns, altogether 105 error instances.

2.6.3 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.

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.

2.7 Related Work

Most closely related to our work is the Xelda system [10] 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 does not only include data cells, but also all formula cells. While analyzing a spreadsheet Xelda checks the annotated units against the results of formulas to insure correctness. 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 [4] 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.

While UCheck works completely automatically, some other related approaches require the user to annotate the spreadsheet with label information [22, 9]. The same advantages and disadvantages that we have mentioned for Xelda apply here as well.

SLATE [17] 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 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.

2.8 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. This 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 dimension is.
- *Dimension inference.* The system even works well in situations when only partial label/dimension information is given since constraints can also be propagated upstream of computations. This aspect contributes to flexibility and robustness.
- *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 handles such conversions smoothly.
- *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 evaluation has demonstrated that the system works well in practice and can detect errors in many cases. The evaluation has also revealed two promising directions for future research. First, we could improve the system with a more accurate header inference. Second, a combination of dimension inference with the purely label-based approaches as pursued by UCheck [4] or the system described in [9] could strengthen the reasoning of the system. To some degree this was already tried in the SLATE approach [17]. 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.

Automatic Detection of Dimension Errors in Spreadsheets

Chris Chambers and Martin Erwig

Journal of Visual Languages & Computing

Volume 20, Issue 4

Chapter 3 – Automatic Detection of Dimension Errors in Spreadsheets

3.1 Introduction

End users engage in a variety of programming activities, including the creation and maintenance of spreadsheets [44]. However, it has been shown that spreadsheets contain many errors [38, 18], and that these errors are often the cause of substantial negative impacts on society [23].

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 [41, 47, 26, 36, 38] and techniques for the automatic generation of spreadsheets [20, 21] from visual [8] or object-oriented specification [19, 11]. 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 [32, 43, 31], 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 [46]. The only systematic testing approach for spreadsheets is the "What You See Is What You Test" approach [42, 13] that uses data-flow adequacy and coverage criteria to give the user feedback on how well tested the spreadsheet is.

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 [25, 2, 6] 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 [34] might introduce more errors into spreadsheets. Some of these problems can be alleviated by automating parts of the testing/debugging process [3, 5].

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.

This 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 [29]. 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 [22, 9]. 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 [7].

Second, we can employ units of measurements as a concrete notion of types that is well known among end users [10]. 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 [27]. Since one pound is equal to 4.48 Newtons, the craft slowly drifted off course. Over time the orbiter dropped 60 miles closer to the surface of Mars and was destroyed.

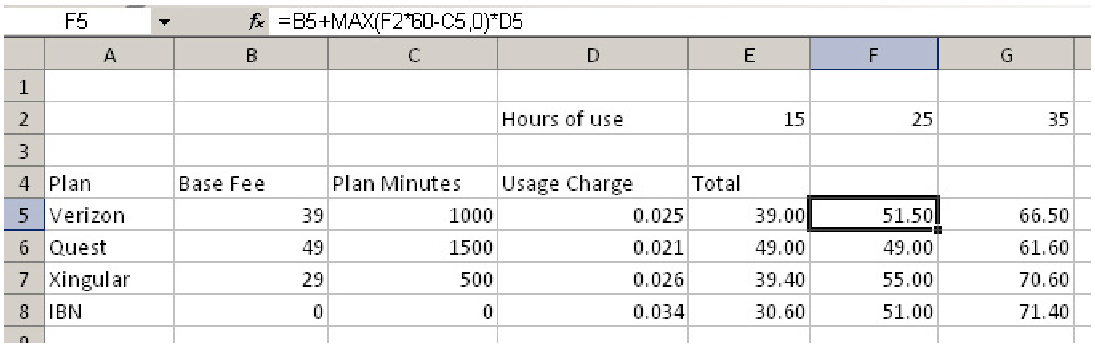
As we will demonstrate in this paper, dimension errors occur frequently in spreadsheets. Therefore, an approach to detect such errors can be an important part of a tool suite to improve the quality of spreadsheets. In this paper we describe *dimension inference*, a method to automatically find dimension errors in spreadsheets. Our work builds on previous approaches and extends them in several important ways. First, through incorporating header inference [1], 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 [14] 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 3.2 we illustrate the issues involved in dimension checking and inference with a small example. In Section 3.3 we formalize spreadsheets and a model of dimensions. The process of dimension inference is then described in Section 3.4. In Section 3.5 we report on an evaluation of a prototypical implementation of a tool for dimension analysis. We discuss related work in Section 3.6 and give conclusions and ideas for future work in Section 3.7.

3.2 Determining Dimensions of Spreadsheet Formulas

Consider the spreadsheet shown in Figure 3.1 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 hours, the formula in cell F5 is as follows.



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

Figure 3.1: A spreadsheet for computing the costs of phone plans under different usage scenarios.

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

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

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.

3.3 A Formal Model of Spreadsheets and Dimensions

3.3.1 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.

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

3.3.2 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 (δ). 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.

$$\begin{aligned} d \ &::= \ \{c,\ldots,c\} \\ c \ &::= \ b_f^n \mid \delta \end{aligned}$$

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, m $=$ length_1^1, which also means that cm $=$ $\text{length}_{0.01}^1$ and ft $=$ $\text{length}_{0.3048}^1$. In general, the following relationship holds (where x is a dimensionless number and b is an arbitrary base).

$$x b_f^n = x f b_1^n$$

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 [24]. The quantities and their default units are shown in Table 3.1.

Table 3.1: 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)

Examples of composite dimensions are speed, measured in $\mathrm{m/s}$, that is $\{\text{length}, \text{time}^{-1}\}$, or force, measured in $\mathrm{kg\,m/s^2}$, 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 3.2. Default units are boxed.

A conversion factor can be either a real number (r) or a conversion variable (ϕ), which

Table 3.2: Basic and derived dimensions and corresponding units

	Dimension	**Units**
basic	length	$\boxed{\text{m}}$, cm, km, ft, ...
	time	$\boxed{\text{s}}$, min, hr, ...
	mass	$\boxed{\text{kg}}$, pounds, ...
	$\dots$	$\dots$
derived	speed	$\frac{\text{m}}{\text{s}}$, $\frac{\text{km}}{\text{hr}}$, ...
	force	$\frac{\text{kg}\,\text{m}}{\text{s}^2}$ = Newton, dyne, ...
	pressure	$\frac{\text{kg}}{\text{m}\,\text{s}^2}$ = Pascal, psi, atm, ...
	$\dots$	$\dots$

serves as a placeholder to be used during dimension inference.

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

Examples of conversion factors can be found in the spreadsheet shown in Figure 3.1, namely min $=$ time$_{60}$ and hr $=$ 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 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 \$ $=$ money$_1$ or cent $=$ 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.

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 out of 487 spreadsheets contained dimensions that could not be converted using the presented model.

3.3.3 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].

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

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

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

The effect of the dimension annotation is that values that are 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 $b^n_f \in d \wedge b^m_{f'} \in d' \wedge f \neq f'$. Otherwise, we have the following definition.

$$d \bowtie d' = \{b^{n+m}_f \mid b^n_f \in d \wedge b^m_f \in d'\} \cup d \triangle d'$$

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 seconds when its speed is 950 km/hr, one has to use a conversion factor with dimension hr/s in the multiplication, otherwise $\bowtie$ is undefined, and the rule cannot be applied.

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

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 doesn't make sense to multiply one side length, denoted by meters, with the other side length denoted by centimeters. What is the meaning of 5 m * 7 cm = 35 cm $*$ 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 seconds multiplied with 950 km/hr. With no conversion factors the result would be: 4750 s $*$ km/hr, which doesn't 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 hr/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 hr/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.

3.3.4 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 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 [37]. 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 3.3.

Table 3.3: Valid dimension exponent ranges

b	$\mathscr{R}(b)$
length	-3 .. 3
electric current	-2 .. 1
time	-3 .. 2
all others	-1 .. 1

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

$$\mathscr{E}(d) = d = \mathrm{kg}^{-1}\mathrm{m}^{-2}\mathrm{s}^{4}\mathrm{A}^{2} \vee \tag{3.1}$$

$$d = \mathrm{kg}^{-1}\mathrm{m}^{-2}\mathrm{s}^{3}\mathrm{A}^{2} \tag{3.2}$$

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.

3.4 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.

3.4.1 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 Figure 3.1, B4 is a header for B5, B6, B7, and B8, that is, $H^{-1}(\text{B4}) = \{\text{B5}, \text{B6}, \text{B7}, \text{B8}\}$, and A5 and B4 are headers of B5, that is, $H(\text{B5}) = \{\text{A5}, \text{B4}\}$. 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 [1, 4]. In the context of this paper we simply reuse those techniques.

3.4.2 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 in Figure 3.1 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 \$/hr. 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 δ, which indicates that the dimension is at this time unknown.

3.4.3 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 Figure 3.2. 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 3.3.4 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 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.

$$60 \Rightarrow \mathrm{min}/\mathrm{hr}$$
$$60 \Rightarrow \mathrm{s}/\mathrm{min}$$
$$100 \Rightarrow \mathrm{cm}/\mathrm{m}$$
$$\vdots$$

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.

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

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

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

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

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

$$\textsc{If}\quad \frac{S,H \vdash e_2 : d \qquad S,H \vdash e_3 : d}{S,H \vdash \mathbf{if}(e_1,e_2,e_3) : d}$$

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

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

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

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

Figure 3.2: Dimension inference rules

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 Figure 3.1 we have $S,H \vdash \text{C5} : \text{min}$.

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 Figure 3.1 and H is the corresponding header relationship, then we obtain $S,H \vdash (\text{F5}, \text{B5+MAX(F2*60-C5,0)*D5}) : \$$. How this result is obtained was explained informally in Section 3.2. The rules given in Figure 3.2 formalize this process, and we will illustrate the formal derivation of this result in Section 3.4.5.

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.

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

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

[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.

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. Figure 3.2 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, SUM, etc.). 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.

3.4.4 Dimension Instantiation

An inferred dimension might contain dimension variables and/or conversion-factor variables. The occurrence of variables happens whenever the spreadsheet doesn't 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, length$_\phi$ can be m, cm, or any other length dimension that can be obtained by substituting values for ϕ. 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 3.3.4). Of those valid dimensions we can then select the one that is most common (as indicated by the numbers to be reported in Section 3.5).

3.4.5 Example Derivation

We now will illustrate how the rule system shown in Figure 3.2 works by showing an example derivation for cell F5 taken from Figure 3.1, which contains the following formula.

B5+MAX(F2*60-C5,0)*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.

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

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.

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

So we know what dimension the cell will have to have, but we don't 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.

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

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.

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

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 $\uparrow B5$. Therefore, the instantiated REF rule looks as follows.

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

The premise of the rule results from the lookup $S(\text{B5}) = 39$. 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 $S,H \vdash \text{B5} : \$$ using the SINGLEHDR rule.

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

To derive the second premise of the ADD rule we have to employ the MULT rule, which is instantiated as follows.

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

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.

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

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).

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

Since the dimension of C5 will be minutes, which is derived in an analogous way to the dimension of B5 (see above), 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 $d_3 = \text{min}$ for F2*60, as can be seen by invoking the MULT rule again. The resulting rule instance is driven by the fact that F2*60 should have the dimension min and that F2 can be shown to be in hours as follows.

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

These two constraints force the invocation of the value judgment $60 \Rightarrow \text{min/hr}$, which then leads to the following instance of the MULT rule.

$$\textsc{Mult}\ \frac{S,H \vdash \text{F2} : \text{hr} \qquad S,H \vdash 60 : \text{min/hr} \qquad \text{min} = \text{hr} \bowtie \text{min/hr} \qquad \mathscr{V}(\text{min})}{S,H \vdash \text{F2*60} : \text{min}}$$

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

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

$$\text{ADD}\ \frac{\begin{array}{c} S,H \vdash \text{A1} : \{\text{length}_1^1\} \cup \{\} \\ S,H \vdash \text{A2} : \{\text{length}_1^{0.01}\} \cup \{\} \qquad c_1 = 1/1 \qquad c_2 = 0.01/1 \end{array}}{S,H \vdash c_1 * \text{A1} + c_2 * \text{A2} : \{\text{length}_1^1\} \cup \{\}}$$

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 can not only point out the omission of a conversion factor, but can also immediately suggest a corrected formula.

3.5 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 [1] of the UCheck tool [4].

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.

3.5.1 Experiments

To answer RQ1 we have employed the EUSES spreadsheet corpus [24], 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.

3.5.2 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 3.4, 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 3.4 since several spreadsheets contain more than one type of dimension.

We found that certain headers were more prevalent than others and had a greater im-

Table 3.4: Occurrences of dimensions

Quantity/ Dimension	**Occurrences**	
	Total	**In Spreadsheets**
Money	390	279
Time	351	237
Length	35	26
Mass	27	20
$/hr	20	15
Area	12	8
Velocity	10	5
Temperature	10	5
kW	3	3
Mole	3	3
Luminous Intensity	3	2
Total	864	603

pact. 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.

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

The dimension inference component able to detected 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, 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.

RQ3: *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.

RQ4: *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 hr, 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

3.5.3 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.

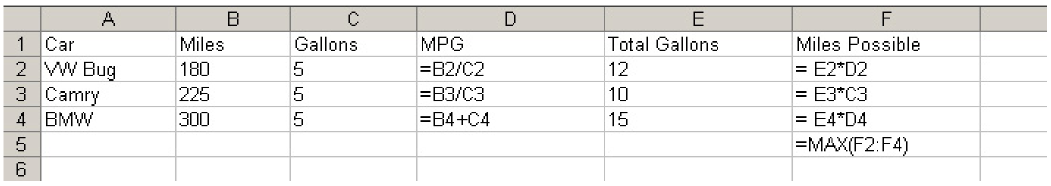
Label analysis and dimension inference worked very reliably. The weakest link in the chain of steps for dimension analysis was clearly the header inference, which is not too surprising since labeling practices vary widely across spreadsheets.

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 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.

3.6 Related Work

We have already discussed several approaches to reduce the occurrence of errors in spreadsheets in Section 3.1. 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 Figure 3.3 that computes for different cars miles-per-gallon numbers and from that their potential range.



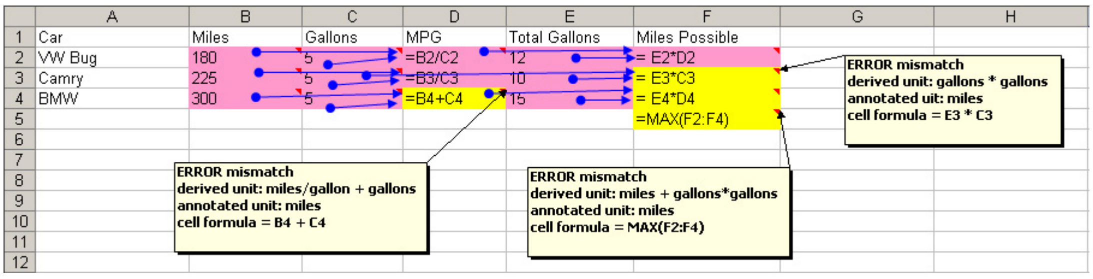
	A	B	C	D	E	F
1	Car	Miles	Gallons	MPG	Total Gallons	Miles Possible
2	VW Bug	180	5	=B2/C2	12	= E2*D2
3	Camry	225	5	=B3/C3	10	= E3*C3
4	BMW	300	5	=B4+C4	15	= E4*D4
5						=MAX(F2:F4)
6						

Figure 3.3: System Comparison Example

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 [10] 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 Figure 3.3 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. Figure 3.4 shows the the feedback produced by XeLda when run on the sheet in Figure 3.3 after it has been annotated.



	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

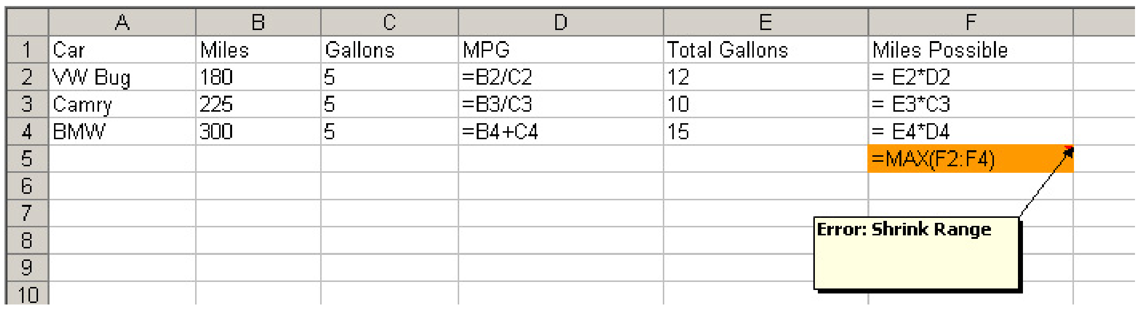
ERROR mismatch
derived unit: miles/gallon + gallons
annotated unit: miles
cell formula = B4 + C4

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

Figure 3.4: XeLda Results

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 [4] 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 Figure 3.5.



	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						
7						
8						
9						
10						

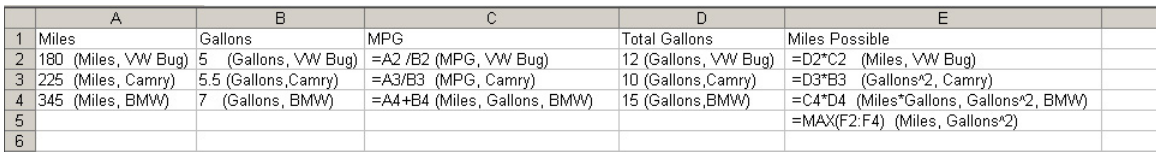
Error: Shrink Range

Figure 3.5: UCheck Results

While UCheck works completely automatically, some other related approaches require the user to annotate the spreadsheet with label information [22, 9]. The same advantages and disadvantages that we have mentioned for XeLda apply here as well.

SLATE [17] 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 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 Figure 3.6.

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

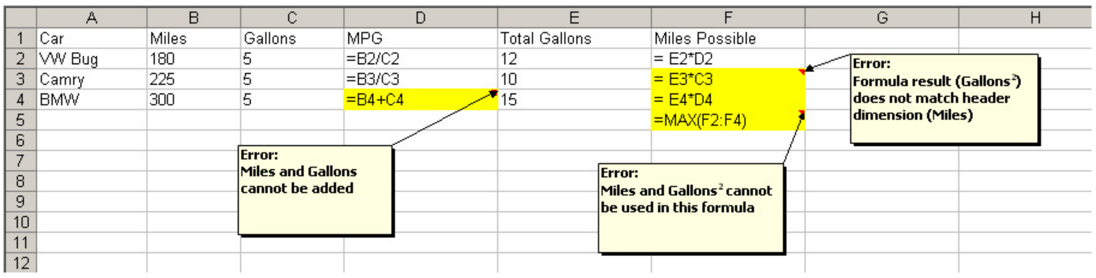


	A	B	C	D	E
1	Miles	Gallons	MPG	Total Gallons	Miles Possible
2	180 (Miles, VW Bug)	5 (Gallons, VW Bug)	=A2 /B2 (MPG, VW Bug)	12 (Gallons, VW Bug)	=D2*C2 (Miles, VW Bug)
3	225 (Miles, Camry)	5.5 (Gallons,Camry)	=A3/B3 (MPG, Camry)	10 (Gallons,Camry)	=D3*B3 (Gallons^2, Camry)
4	345 (Miles, BMW)	7 (Gallons,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

Figure 3.6: SLATE Results

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, Figure 3.7 shows the results of dimension inference on the spreadsheet in Figure 3.3. Dimension Inference is able to highlight the errors in the spreadsheet and does so without requiring any annotation of the spreadsheet.



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

Error:
Miles and Gallons cannot be added

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

Figure 3.7: Dimension Inference Results

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 3.5.3.

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.

3.7 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 also can be used to better resolve ambiguous cases.

Second, a combination of dimension inference with the purely label-based approaches as pursued by UCheck [4] or the system described in [9] could strengthen the reasoning of the system. To some degree this was already tried in the SLATE approach [17]. 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 Figure 3.3. 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.

Thirdly, 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 doesn’t 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.

Combining Spatial and Semantic Label Analysis

Chris Chambers and Martin Erwig

VLHCC ’09: Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing

Chapter 4 – Combining Spatial and Semantic Label Analysis

4.1 Introduction

Spreadsheets are widely used [45] end-user programs that contain many errors [39]. To improve the quality of spreadsheets a variety of approaches to prevent, detect, and remove errors from spreadsheets have been investigated. 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.

One type of error that can be detected in spreadsheets is dimension errors, which occur when units of measurement are used incorrectly in formulas. Units of measurements can be employed as a concrete notion of types that is well known among end users [10], and are used to characterize different kinds of values, much like traditional, more abstract, type systems used in general-purpose programming languages. For example, a floating point number, which has just one type, can nevertheless represent different kinds of quantities, such as length or time values.

Several systems [14, 17, 10] have been developed in order to deal with unit of measurement errors. Among these dimension inference [14] is a method that can be used to automatically find dimension errors in spreadsheets. This approach has been shown to work reliably and effectively in many cases, however it does not take full advantage of the information provided in the spreadsheet as it does not utilize the structure of the spreadsheet and focuses on ensuring that formulas are dimension correct.

In contrast, there are several systems that are designed to directly take advantage of the labels and the structure of spreadsheets. These purely label-based approaches, such as UCheck [4] or the system described in [9], are designed to find formula errors caused by inconsistent label usage. This technique operates in two distinct analysis phases. The first phase defines header or label information for the entire spreadsheet. UCheck is able to infer this while most others require users to annotate the labels for every cell. Once the headers are determined for a sheet, labels are assigned to cells based on headers and formulas. In the second phase this information is analyzed to find errors.

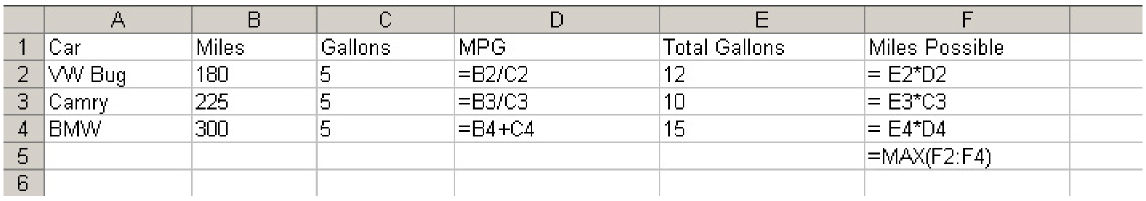
One thing to note when looking at label analysis and dimension analysis is that both systems rely on header and label information, however, how this information is used is quite different. By combining dimension analysis with the purely label-based approaches the structure of the spreadsheet could be used to help strengthen the reasoning of the system. To some degree this was already tried in the SLATE approach [17]. However, SLATE only transforms labels and dimensions and does not identify errors. Moreover, the fact that SLATE is a stand-alone spreadsheet system that cannot be integrated into Excel and the time it takes for users to annotate a spreadsheet renders the approach currently impractical.

In this paper we describe a way to combine label based reasoning with dimension inference. This approach is achieved by gathering both label and dimension information about a cell. In many cases a spreadsheet contains dimensions on only one axis, while the other axis typically contains labels that do not map to any dimension. These labels, which go unused in dimension inference can help to provide structural information that can be exploited by the reasoning system behind UCheck.

The rest of this paper is structured as follows. In Section 2 we illustrate the issues involved in adding label reasoning to dimension inference with a small example. In Section 3 we formalize spreadsheets and present models of dimensions and labels. The combined analysis method is 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 in Section 7.

4.2 An Example

To explain how the integration of spatial and semantic label analysis works, we will show how both dimension inference and the integrated system work on the spreadsheet in Figure 4.1. This spreadsheet is calculating how far specific cars can travel on a full tank of gas. This data is correlated based on the result of a drive using only five gallons of gas.



	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						

Figure 4.1: Example spreadsheet

When the spreadsheet is checked with dimension inference, it would first identify the headers for all cells. When the headers are analyzed for dimension information, B1, C1, D1, E1, and F1 all map to a valid dimension. This would allow the system to check all formulas in this spreadsheet for dimension correctness. In this case, the system would detect that there is an error in cell D4 where the formula is trying to add miles and gallons.

Upon further inspection it could be noted that the spreadsheet contains another error. In this particular example, the cell F2 has the formula E2*D3. The dimension for E2 is Gallons, and the dimension for D3 is Miles per Gallon, which, when multiplied together result in the dimension Miles. This result contains no dimension errors, but it does not seem right. E3 is actually total Gallons for the Camry, while D2 is the MPG for the VW Bug. Logically, the result does not make sense, however, plain dimension inference would have no way to catch this.

By integrating label reasoning and checking that formulas are both dimension *and* label correct, the system presented in this paper is able to identify a previously unnoticed error. The first step is to determine which header axis (row or column) will be used as the dimension axis. In this case there are several dimensions on the horizontal axis (row 1), but dimensionless labels on the vertical axis (column A). The system then identifies labels and dimensions for each cell. 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 dimension and label correct. When the system checks the formula in F2 it can identify that it is multiplying a cell, E2, with the unit Gallons and the label “VW Bug” with the cell D2, which has the unit Miles / Gallon and the label “Camry”. While the dimensions work out in this formula, the system will identify an inconsistency with the labels and be able to report this to the user.

4.3 Syntactic Representation

In this section, we will formalize the notions of spreadsheets, dimensions, and labels as a preparation for the formal rule system given in Section 4.3.4.

4.3.1 Spreadsheets

Spreadsheets (S) are functions that is a map addresses ($a \in A$) to expressions (e), in particular, $S(a)$ yields the expression stored at address a in the spreadsheet S. Expressions can be values (v) or references to other cells ($\uparrow a$), or are constructed using arithmetic ($+$ or $*$), aggregating (**count**), or conditional operators.

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

The operations $+$ and $*$ represent, respectively, a whole class of additive operators (including $-$ and MAX) and multiplicative operators (including $/$).

4.3.2 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). A dimension component can also be a dimension variable (δ). The *identity dimension* $\{\}$ is used for dimensionless values.

$$\begin{aligned} d &::= \{c, \ldots, c\} \\ c &::= b^n_f \mid \delta \end{aligned}$$

For each base dimension we identify a default unit with factor 1. For example, the default for length is meter (m), that is, m $=$ length^1_1, which also means that cm $=$ $\text{length}^1_{0.01}$ and ft $=$ $\text{length}^1_{0.3}$. In general, the following relationship holds (where x is a dimensionless number and b is an arbitrary base): $x b^n_f = x f b^n_1$.

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 [24].

A more detailed discussion of dimensions can be found in [14].

4.3.3 Labels

The labeling structure in this integrated system is a simplified version of the formal model presented in [22]. In particular, since labels will be used only for one axis, we can omit the concept of AND labels, which leads to much simplified rules for combining labels.

The syntax that we use for labels is shown below. (Note the difference between an OR-label $\ell_1 \,|\, \ell_2$ and the vertical bar $\mid$ to separate grammar alternatives.)

$$\ell \;::=\; v \mid \ell_1[\ell_2] \mid \ell_1 \,|\, \ell_2 \mid 1$$

To show the different possible types of labels we will look at several cells in Figure 4.1. A chain of labels, $\ell_1[\ell_2]$, represents cells that may have a second level label. In this example, the cell B3 has the header A3, which contains the label Camry. Since A3 has as its header the cell A1 with label Car, the label associated with B3 is Car[Camry].

An OR label is used when cells with labels are added together. In general, when two cells are added together, their labels are ORed to produce a resulting label. The cell B5 is a SUM, which adds the three value cells in column B. The three labels used in this formula are Car[VW Bug], Car[Camry], and Car[BMW]. The resulting label is Car[VW Bug] | Car[Camry] | Car[BMW]. Since OR distributes over label chains [22], we can factor this expression to Car[VW Bug | Camry | BMW]. This label expression can be generalized to Car [4].

The ADD rule requires a compatibility of labels expressed by the following label simplification rule, $\ell \wr \ell \rightarrow \ell$, defined as follows.

$$\begin{aligned} \ell \wr \ell &\rightarrow \ell \\ \ell \wr \ell[\ell_1] &\rightarrow \ell \\ \ell[\ell_1] \wr \ell &\rightarrow \ell \end{aligned}$$

This operation works only for specific arguments, and if it fails in the premise of an inference rule, this corresponds to the identification of a label error.

4.3.4 On Semantic vs. Syntactic Label Analysis

Before we describe our integrated reasoning tool, we want to point out a principal difference of the two underlying approaches, because even though both UCheck and dimension inference uses labels to determine errors, they use this information in quite different ways.

UCheck essentially exploits the relative position of labels, but it does not actually interpret labels. This means that we can rename labels without changing the functionality of the system (at least if the renaming is done systematically). For example, the labels Camry or BMW have no meaning to the system and could be replaced by any other strings. It is how these labels are combined with other labels in formulas that forms the basis of error detection.

Dimension inference, on the other hand, derives some meaning from labels. Instead of treating labels as simple strings, labels in dimension inference actually are interpreted to have some extrinsic semantics. This means that renaming a label could cause errors in a spreadsheet. Using the sheet in Figure 4.1, if the label in cell B1 is renamed to “KM” the formulas in column D will not be dimension correct as the label for that column is “Miles / Gallon”, but the resulting dimension is “KM / Gallons”. To keep the formula dimension correct the labels in D1 and F1 would have to be renamed as well.

4.4 Integrated Label and Dimension Analysis

The combined label and dimension analysis of a spreadsheet goes through five distinct steps. The last step applies only in those cases when the fourth step produces underspecified dimensions, that is, when it results in inferred dimensions that contain dimension variables.

1. Header inference
2. Label analysis
3. Identify dimension and label axes
4. Dimension inference
5. Dimension instantiation

Header inference, label analysis, and dimension instantiation are components that we have adopted unchanged from our previous work [14], and they are therefore only briefly described here. Steps 3 and 4 will be described in greater detail in the following.

4.4.1 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, cell B1 in Figure 4.1 is a header for B2, B3, and B4, that is, $H^{-1}(\mathrm{B1}) = \{\mathrm{B2}, \mathrm{B3}, \mathrm{B4}\}$, and A2 and B1 are headers of B2, that is, $H(\mathrm{B2}) = \{\mathrm{A2}, \mathrm{B1}\}$. Header inference essentially works by analyzing the spatial relationships between different kinds of formulas. It can also take into account layout information. Techniques for header inference have been described in detail elsewhere [1, 4]. In the context of this paper we simply reuse those techniques.

4.4.2 Label Analysis

In the second phase of the integrated system 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 E1 in Figure 4.1 is a header cell and is therefore subject to label analysis. Its value can be split into the two words “Total” and “Gallons”, and the plural of “Gallons” can be removed. The resulting “Gallon” can then be mapped to the dimension gal. In contrast, “Total” cannot be mapped into any dimension and will thus be mapped to $\{\}$. Finally, the combination of both dimensions yields “Gallons”. If no part of a header label can be mapped to a dimension other than $\{\}$, the label is mapped to a dimension variable δ, which indicates that the dimension is at this time unknown.

4.4.3 Identify Dimension and Label Axes

The goal of our system is to exploit one axis for dimension checking and the other for label checking. In the formal rule system, this separation of analysis is reflected by two new judgments, $S, L \vdash a : \ell$ and $S, D \vdash a : d$. These judgments specify how the integrated system gets labels and dimensions for each cell. The header relationship H identified in the header inference phase has to be partitioned into two parts $H = L \cup D$ where:

- L is the set of headers that define labels
- D is the set of headers that define dimensions

If a cell has two headers, one of which defines dimensions and the other which defines a plain label, both of these pieces of information are exploited to make the inference stronger, as can be seen in the COMBOHDR rule in Figure 4.2.

To facilitate the partitioning of the header relationship, we have to identify table regions in a spreadsheet and for each table its horizontal and vertical label axes. The information provided by label analysis allows us to the identify the following three cases for axes:

1. No Dimension Axis
2. One Dimension Axis
3. Two Dimension Axes

In the following we will describe each of the possibilities in some detail.

No Dimension Axis In the simplest case, that is a spreadsheet that has no units of measurement, there will be no dimension axis. If this is the case, then there is no need to run dimension inference at all. The previous system would not have been able to detect any errors. However, with the integration of label-based reasoning we can run UCheck to detect label errors. While this is not the goal of the system, it does give the tool a function for spreadsheets without dimensions.

One Dimension Axis The situation where this system is most useful is when there is one axis that contains dimensions in a spreadsheet. If this is the case, the system will be able to combine label-based reasoning with dimension inference. The system will use the identified dimension and label axes to assign labels and dimensions for each cell in a spreadsheet. This information can then be used to detect errors.

As an example, we again look at Figure 1. The two identified axes will be row 1 and column A. In this case these share a cell, A1, which has been identified as a header for A2, A3, and A4 (for details, see [1]). As it is a header for all of the cells in the vertical axis, it is included with them and ignored in the horizontal axis. To correctly identify the dimension axis, the headers are mapped to a dimension. In this example, every header in row 1 and none of the headers in column A maps to a dimension. This makes the decision easy, and the horizontal axis, row 1, is chosen as the dimension axis, with column A being used for labels.

The labels for all headers in the dimension axis are defined as the one unit, 1, and the dimensions for all headers in the label axis are defined as the unit dimension, $\{\}$. This will give the system the flexibility to use both label-based reasoning and dimension inference.

Two Dimension Axes The final case occurs when dimensions exist in both axes. Should this arise the system will not attempt to use label-based reasoning to assist dimension inference. It will instead do a pure dimension analysis using the method described previously.

4.4.4 Combined Label and Dimension Inference

The fourth step of the integrated system is a "label-aware" dimension inference, which inspects each cell containing a formula and derives for it a dimension and a label using the system of rules given in Figure 4.2. In the previous incarnation of this system a rule application could result only when a dimension could not be inferred. Now an additional failure point has been added. If the labels cannot be combined as described in Section 4.3.3, then the formula is also declared erroneous.

The relationship between formulas and dimensions is formalized through the following judgments that tie together the idea of dimension and label axes and the previous judgments from dimension inference.

$$\boxed{S, L \vdash a : \ell} \qquad\qquad \boxed{S, D \vdash a : d}$$

$$\textsc{LabHdr}\ \frac{L(a) = \{a_1\} \qquad S(a_1) = \ell}{S, L \vdash a : \ell} \qquad\qquad \textsc{DimHdr}\ \frac{D(a) = \{a_1\} \qquad S(a_1) \Rightarrow d}{S, D \vdash a : d}$$

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

$$\textsc{NoHdr}\ \frac{H(a) = \varnothing}{S, H \vdash a : 1, \delta} \qquad\qquad \textsc{ComboHdr}\ \frac{S, L \vdash a : \ell \qquad S, D \vdash a : d}{S, H \vdash a : \ell, d}$$

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

$$\textsc{Val}\ \frac{}{S, H \vdash v : 1, \delta} \qquad \textsc{Ref}\ \frac{S, H \vdash (a, S(a)) : \ell, d}{S, H \vdash \uparrow a : \ell, d} \qquad \textsc{Count}\ \frac{S, H \vdash e_i : \ell, d}{S, H \vdash \mathbf{count}(e_1, \ldots, e_n) : \ell, \{\}}$$

$$\textsc{If}\ \frac{S, H \vdash e_2 : \ell, d \qquad S, H \vdash e_3 : \ell, d}{S, H \vdash \mathbf{if}(e_1, e_2, e_3) : \ell, d}$$

$$\textsc{Add}\ \frac{\begin{array}{c} S, H \vdash e_1 : \ell_1, \{b^n_{f_1}\} \cup d \\ S, H \vdash e_2 : \ell_2, \{b^n_{f_2}\} \cup d \qquad \ell_1 \wr \ell_2 \rightarrow \ell \qquad c_1 = f_1/f \qquad c_2 = f_2/f \end{array}}{S, H \vdash c_1 * e_1 + c_2 * e_2 : \ell, \{b^n_f\} \cup d}$$

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

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

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

Figure 4.2: Combined System rules

1. Value Judgment

   $v \Rightarrow d$ says the value v, if used as a label or factor, describes the dimension d.

2. Header Judgments

   $S,L \vdash a : \ell$ and $S,D \vdash a : d$ transform the header information into dimension and label assignments for addresses. These judgments rely on the separation of H into L and D performed by the dimension/label axis identification step. Specifically, $S,L \vdash a : \ell$ ($S,D \vdash a : d$) says that in the spreadsheet S and given label (dimension) header L (D), the location given by address a has dimension d (label ℓ).

3. Expression Judgment

   $S,H \vdash e : \ell, d$ says that in the spreadsheet S and given the header structure H, the expression e has label ℓ and dimension d.

4. Cell Judgment

   $S,H \vdash (a,e) : \ell, d$ says the cell (a,e) in the spreadsheet S has the label ℓ and dimension d under the given header relationship H.

To show how these rules are applied to a spreadsheet, we will investigate how they are used on the cells containing errors in Figure 4.1.

The first such cell is D4. This cell contains an addition and thus will be checked by the ADD rule, which instantiates as follows for D4.

$$\frac{\begin{array}{c} S,H \vdash \text{B4} : \text{Car[BMW]}, \{\text{Miles}\} \cup \{\} \qquad S,H \vdash \text{C4} : \text{Car[BMW]}, \{\text{Gallons}\} \cup \{\} \\ \text{Car[BMW]} \wr \text{Car[BMW]} \rightarrow \text{Car[BMW]} \qquad c_1 = f_1/f \qquad c_2 = f_2/f \end{array}}{S,H \vdash \text{B4} + \text{C4} : \text{BMW}, ?}$$

Since the dimensions in this formula are not compatible, a result dimension cannot be derived, and an error is produced.

The behavior of multiplication errors can be seen by looking at the cell F2, for which the rule MULT is employed.

$$\frac{S,H \vdash \text{E2} : \text{Car[VW Bug]}, \text{Gallons} \qquad S,H \vdash \text{D3} : \text{Car[Camry]}, \text{MPG} \qquad d = \text{Miles} \qquad \mathscr{V}(\text{Miles})}{S,H \vdash \text{E2} * \text{D3} : ?, \text{Miles}}$$

While the dimensions in this multiplication are fine, the labels Car[VW Bug] and Car[Camry] are not compatible. Thus an error has been identified. For this formula to be correct D3 would have to be changed to D2, which also has the label Car[VW Bug].

4.4.5 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.

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. Of those valid dimensions we can then select the one that is most common (as indicated by the numbers to be reported in Section 4.5).

4.5 Evaluation

We have extended the dimension inference tool with the ability to perform automatic dimension analysis with label-based reasoning. This tool reuses the header analysis implementation [1] of the UCheck tool [4] and is based on the dimension inference tool in [14]. In this section we describe an evaluation of this system to answer the following research questions.

RQ1: *How does the combined system compare to pure dimension inference?*
Dimension inference can be an effective tool to check formulas and spot errors in spreadsheet computations, but it ignores much of the structure of spreadsheets. By integrating label analysis we expect to see a considerable improvement in the detection of errors.

RQ2: *Do we lose anything by adding label-based reasoning?*
By integrating label-based reasoning with dimensions it is possible that the system could lose the ability to detect dimension errors. Due to how dimension axes are treated this seems unlikely, but is an important question to answer.

RQ3: *By turning off labels on one axis do we lose errors?*
Most label-based systems involve using labels from both axes in a spreadsheet. With one axis being used to define dimensions, how many label errors are not caught by the system?

RQ4: *How many tables map cleanly into two axes?*
If the tables in a spreadsheet do not map cleanly into two axes, it will be difficult for the integrated system to run successfully. Knowing how many tables do not map cleanly will provide a negative estimate of how useful the system can be.

RQ5: *How many tables contain dimensions on only one axis?*
The primary case for this tool is when dimensions occur on only one axis of a table. We will try to determine exactly how many spreadsheets in the EUSES corpus fall into the category of having one dimension axis and one label axis. This will allow us to further gauge the usefulness of this tool.

4.5.1 Experiments

To answer these research questions we have employed the EUSES spreadsheet corpus [9], which currently contains 4498 spreadsheets collected from various sources. Of these 4498 spreadsheets only 487 contain both dimensions and formulas. Dimension inference, the integrated version, and UCheck were run on all of these spreadsheets to determine how well each system fared and how many errors each found.

For RQ1 and RQ2, we compared the results of dimension inference and the integrated system. Specifically, how many additional errors did the new system find that were not able to be caught by the old system, and how many errors were not able to be caught when label checking is introduced.

To investigate RQ3 we compared the results of the new system with those of UCheck. By doing this comparison we were able to identify possible label errors that the integrated system was not able to detect, but that were caught when both axes are used for a pure label-based system.

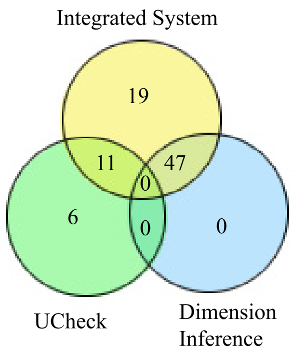
Finally, for RQ4 and RQ5 the mapping of axes for all the 487 spreadsheets with dimensions was investigated to see how many mapped cleanly and to how many dimension axes. Since spreadsheets can have multiple tables, this mapping is performed on all 567 tables in these spreadsheets.

4.5.2 Results

The main experiment was running the three systems on the subset of dimension sheets in the EUSES corpus. This process gave us the necessary information to be able to determine how well the new system operates.

Dimension inference was able to detect 47 spreadsheets with a total of 241 dimension errors, the integrated system was able to detect 77 spreadsheets with errors and a total of 674 errors, and UCheck identified errors in 17 sheets with 151 total errors. The overlap of errors is shown in Figure 4.3.

To determine false positives, the spreadsheets with errors were looked at more closely. We found that the integrated system produced 11 false positive instances that resulted in 78 total errors, whereas dimension inference found 7 false positive instances that resulted in 49 total errors. The 7 false positives of dimension inference were all inherited by the integrated system. Of the 7 false positives, 6 were caused by label analysis, and one was caused by incorrect header inference. Of the additional 4 false positives in the integrated



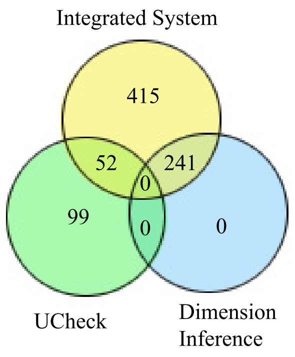


Figure 4.3: System comparison: Identified erroneous spreadsheets (left) and total errors (right)

system one was caused by label analysis and 3 by incorrect header inference.

To determine how many tables contain dimensions in both axes, the axes mapping was logged for every relevant table. When this mapping was performed 128 out of a total of 567 tables contained dimensions in both axes.

There were also 22 tables that did not map well into axes. In general, these were caused by a failure in header inference, in which the headers were not able to be determined.

4.5.3 Discussion

Due to how the new system handles the axes in a spreadsheet the new system did not "lose" any dimension errors. All of the dimension errors are caught by both systems. This is unsurprising as label integration in no way changes how dimensions are handled. If both axes contain dimensions then the system will ignore label-based reasoning and simply perform dimension inference. If dimensions only occur on one axis, then labels will not reduce the number of errors that can be detected.

The final comparison brings up one shortcoming of the system. When we look at the results from UCheck and the results from the new system, there are 6 spreadsheets and

99 additional label errors that the new system was not able to catch. These errors all are caused by a label error based on the dimension axis. Since that axis was being used to define the dimensions in the spreadsheet, the labels were not used to catch label errors.

The numbers for the axes mappings are very encouraging. In only 22 tables or 3% of the total spreadsheets, the headers were not able to be mapped to axes. This defect was due one of two facts. Either labels were only present on one axis, or the header inference was not able to determine proper headers for that sheet.

This number is encouraging as it means that most of the sheets will be able to be used by the system. In general 128 tables mapped to two dimension axes, which means that 22% of the tables with dimensions will not have different results when the integrated system is run on them. This is another encouraging number since it means that a vast majority of sheets will be able to gain more error checking capabilities with the integrated system.

4.6 Related Work

The system that is most closely related to our work is SLATE [17], which separates the unit from the object of measurement and defines semantics for spreadsheets so that the unit *and* the object of measurement are considered.

SLATE is the only system that attempts to measure both labels and dimensions, and it does this by assigning three attributes to every expression: a value, a 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)”.

For this system to work correctly it requires a user to annotate a spreadsheet, which involves adding the units and labels to every non-formula cell. The system then analyzes the formula cells and determines the unit and label for these cells. This information is then displayed in the cell. One of the primary problems with this approach is that is does not actually detect errors, it simply displays labels and units for each cell.

Another related system is XeLda [10] which is designed to check a spreadsheet for units of measurement, such as meters, grams, and seconds. Much like SLATE, XeLda requires the user to annotate the units for all of the cells in a spreadsheet. Note that this does not only include data cells, but also all formula cells. While analyzing a spreadsheet XeLda checks the annotated units against the results of formulas to insure correctness.

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.

With respect to the error discussed in the Section 4.2 that requires a combined label and dimension analysis for its discovery, SLATE would infer a proper dimension together with a label (VW Bug, Camry), which is meant to draw the attention of the user and hopefully point out something that might be wrong with the formula, but it is not marked as an error. XeLda would not be able to catch this error as it depends entirely on units of measurement annotated by the user.

UCheck [4] was designed to check for labels 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 labels for the cells and checks for label errors.

While UCheck works completely automatically, some other related approaches require the user to annotate the spreadsheet with label information [22, 9]. The same advantages and disadvantages that we have mentioned for XeLda apply here as well.

4.7 Conclusions and Future Work

We have introduced a system that integrates label-based reasoning with dimension analysis. This integration has strengthened the system considerably with the evaluation showing almost a twofold increase in the number of errors that the system is able to detect.

In future work, we will try to strengthen the integration of UCheck and dimension inference further. We have seen that some label errors were missed because some of the labels were exclusively used for dimension analysis. We could potentially catch more errors by running the combined system *and* UCheck and taking the union of errors reported by both systems. However, this might also increase the chance of hitting more false positives. To study different options for an integrated system, we can employ the system architecture that we have developed in previous work [29].

Additionally, the manual and automatic approaches could be combined so that users could easily view and correct errors, as well as provide additional annotation as needed.

Chapter 5 – Conclusion

Throughout this thesis we have introduced a system for inferring and checking dimensions in spreadsheets. Starting from pure dimension analysis and then adding the ability to check for labels, the system can identify errors in formulas, because dimensions place constraints on how operations can act on values. While the systems described in this thesis have slightly different functionalities, the main goal of each is the same, to check a spreadsheet for dimension errors. Despite the differences, they all share several traits in common, which separates them from previous approaches in several ways.

- *No user annotation required.* While many systems require user input to define dimensions or units for cells, our systems are able infer labels for cells automatically and use this information to determine the expected dimension. This minimizes the amount of work a user has to do in order to use the system as well as making it easier to use.

- *Dimension inference.* There can be cases when a spreadsheet contains only partial dimension information. For example, only a few columns have labels that can be inferred as a dimension, while the other have ambiguous labels. These systems can determine these undefined dimensions through the use of 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 fact that these systems differ from other, similar, systems, does no inherently mean they are better or even that they are effective in finding dimension errors in spreadsheets. To show these several evaluations were performed. These evaluations showed these systems to be efficient in determining dimension errors. In addition the combined system showed great improvement over the original and is a significant upgrade in terms of functionality and the number of errors detected and could prove to be very valuable to end users.

In addition to showing that the systems work efficiently the evaluations have also revealed several promising directions for future research.

First, we could improve these systems with a more accurate header inference. One of the biggest problems that these systems have is with incorrect or missing header information. If this is the case, the systems can infer the wrong dimensions for specific cells. This arises from the fact that the header inference can at times be rather heuristic. One way around this would be to employ 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 also can be used to better resolve ambiguous cases.

Some preliminary work has been done to determine the different types of header patterns. With this information in hand a system that looks at the spatial aspects of a spreadsheet and matches it with one of the predetermined header patterns may be able to more accurately map the headers of a spreadsheet leading to fewer header mismatches.

Secondly, ways to combine manual and automatic dimension checking techniques should be explored. This combination would allow users to change or supplement dimensions that are determined by the system with dimensions that are either more correct or add information. There can be cases where there is not enough information to determine if a formula is actually correct. For example, if the header of the resulting formula does not contain any dimension information, nothing can be done to validate the result. Once the system has determined this it can be pointed out to the user, giving them an opportunity to manual add a dimension for that header.

Thirdly, one could imagine ways to improve this in the eyes of users. These systems could be tested with user systems to see if the information that they present makes sense and if there are better ways to display these error messages that make them more manageable or easier to understand. These user studies could also provide ideas that would possibly lead to future research opportunities.

As the evaluations show the systems presented in this thesis can be effective in finding dimension errors. It has been demonstrated that the majority of errors and can be found with minimal false positives. The fact that these systems can operate automatically is beneficial and one of the properties that make these systems unique.

The work of combining label and dimension analysis created a much stronger system and demonstrates the power of integrated analysis. This system was able to find all label and dimension errors caught by the independent systems as well as many errors that were not caught by either label or dimension analysis. The strength of this system shows how powerful and useful these tools can be to end users and that they provide a simple way to check a spreadsheet for multiple types of errors.

Bibliography

[1] R. Abraham and M. Erwig. Header and Unit Inference for Spreadsheets Through Spatial Analyses. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 165–172, 2004.

[2] R. Abraham and M. Erwig. AutoTest: A Tool for Automatic Test Case Generation in Spreadsheets. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 43–50, 2006.

[3] R. Abraham and M. Erwig. GoalDebug: A Spreadsheet Debugger for End Users. In *29th IEEE Int. Conf. on Software Engineering*, pages 251–260, 2007.

[4] R. Abraham and M. Erwig. UCheck: A Spreadsheet Unit Checker for End Users. *Journal of Visual Languages and Computing*, 18(1):71–95, 2007.

[5] R. Abraham and M. Erwig. Test-Driven Goal-Directed Debugging in Spreadsheets. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 131–138, 2008.

[6] R. Abraham and M. Erwig. Mutation Operators for Spreadsheets. *IEEE Transactions on Software Engineering*, 35(1):94–108, 2009.

[7] R. Abraham, M. Erwig, and S. Andrew. A Type System Based on End-User Vocabulary. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 215–222, 2007.

[8] R. Abraham, M. Erwig, S. Kollmansberger, and E. Seifert. Visual Specifications of Correct Spreadsheets. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 189–196, 2005.

[9] Yanif Ahmad, Tudor Antoniu, Sharon Goldwater, and Shriram Krishnamurthi. A type system for statically detecting spreadsheet errors. *18th IEE Int. Conf. on Automated Software Engineering*, 0:174–183, 2003.

[10] Tudor Antoniu, Paul A. Steckler, Shriram Krishnamurthi, Erich Neuwirth, and Matthias Felleisen. Validating the unit correctness of spreadsheet programs. In *26th IEEE Int. Conf. on Software Engineering*, pages 439–448. IEEE Computer Society, 2004.

[11] J.-C. Bals, F. Christ, G. Engels, and M. Erwig. ClassSheets – Model-Based, Object-Oriented Design of Spreadsheet Applications. *Journal of Object Technologies*, 6, 2007.

[12] Polly S. Brown and John D. Gould. An experimental study of people creating spreadsheets. *ACM Trans. Inf. Syst.*, 5(3):258–272, 1987.

[13] M. M. Burnett, A. Sheretov, B. Ren, and G. Rothermel. Testing Homogeneous Spreadsheet Grids with the "What You See Is What You Test" Methodology. *IEEE Transactions on Software Engineering*, 29(6):576–594, 2002.

[14] C. Chambers and M. Erwig. Dimension inference in spreadsheets. *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pages 123–130, Sept. 2008.

[15] C. Chambers and M. Erwig. Automatic Detection of Dimension Errors in Spreadsheets. *Journal of Visual Languages and Computing*, 20(3), 2009.

[16] Chris Chambers and Martin Erwig. Combining spatial and semantic label analysis. In *VLHCC '09: Proceedings of the 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, pages 225–232, Washington, DC, USA, 2009. IEEE Computer Society.

[17] Michael J. Coblenz, Andrew J. Ko, and Brad A. Myers. Using objects of measurement to detect spreadsheet errors. In *VLHCC '05: Proceedings of the 2005 IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 314–316, Washington, DC, USA, 2005. IEEE Computer Society.

[18] S. Ditlea. Spreadsheets Can be Hazardous to Your Health. *Personal Computing*, 11(1):60–69, 1987.

[19] G. Engels and M. Erwig. ClassSheets: Automatic Generation of Spreadsheet Applications from Object-Oriented Specifications. In *20th IEEE/ACM Int. Conf. on Automated Software Engineering*, pages 124–133, 2005.

[20] M. Erwig, R. Abraham, I. Cooperstein, and S. Kollmansberger. Automatic Generation and Maintenance of Correct Spreadsheets. In *27th IEEE Int. Conf. on Software Engineering*, pages 136–145, 2005.

[21] M. Erwig, R. Abraham, S. Kollmansberger, and I. Cooperstein. Gencel – A Program Generator for Correct Spreadsheets. *Journal of Functional Programming*, 16(3):293–325, 2006.

[22] M. Erwig and M. M. Burnett. Adding Apples and Oranges. In *4th Int. Symp. on Practical Aspects of Declarative Languages*, LNCS 2257, pages 173–191, 2002.

[23] EuSpRIG. European Spreadsheet Risks Interest Group. `http://www.eusprig.org/`.

[24] Marc Fisher and Gregg Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *WEUSE I: Proceedings of the first workshop on End-user software engineering*, pages 1–5, New York, NY, USA, 2005. ACM.

[25] M. Fisher II, M. Cao, G. Rothermel, C. Cook, and M. M. Burnett. Automated Test Case Generation for Spreadsheets. In *24th IEEE Int. Conf. on Software Engineering*, pages 141–151, 2002.

[26] T. Isakowitz, S. Schocken, and H. C. Lucas, Jr. Toward a Logical/Physical Theory of Spreadsheet Modelling. *ACM Transactions on Information Systems*, 13(1):1–37, 1995.

[27] Douglas Isbell. Mars Climate Orbiter Team Finds Likely Cause of Loss, September 1999. `http://mars.jpl.nasa.gov/msp98/news/mco990930.html`.

[28] Jelter. J. Kodak restates, adds $9 million to loss., 2005. http://www.marketwatch.com/News/Story/Story.aspx?guid=276FD56F-00CA-42AE-AD70-C66DF571FC77.

[29] J. Lawrence, R. Abraham, M. M. Burnett, and M. Erwig. Sharing Reasoning about Faults in Spreadsheets: An Empirical Study. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 35–42, 2006.

[30] F. J. Lerch, M. M. Mantei, and J. R. Olson. Skilled financial planning: the cost of translating ideas into action. In *CHI '89: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 121–126. ACM, 1989.

[31] R. Mittermeir and M. Clermont. Finding High-Level Structures in Spreadsheet Programs. In *9th Working Conference on Reverse Engineering*, pages 221–232, 2002.

[32] R. R. Panko. Applying Code Inspection to Spreadsheet Testing. *Journal of Management Information Systems*, 16(2):159–176, 1999.

[33] Raymond R. Panko. Spreadsheet errors: What we know. what we think we can do. *CoRR*, abs/0802.3457, 2008.

[34] A. Phalgune, C. Kissinger, M. Burnett, C. Cook, L. Beckwith, and J. Ruthruff. Garbage In, Garbage Out? An Empirical Look at Oracle Mistakes by End-User Programmers. In *IEEE Int. Symp. on Visual Languages and Human-Centric Computing*, pages 45–52, 2005.

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

[36] S. G. Powell and K. R. Baker. *The Art of Modeling with Spreadsheets: Management Science, Spreadsheet Engineering, and Modeling Craft*. Wiley, 2004.

[37] R. Rowlett. A Dictionary of Units of Measurement, July 2005. `http://www.unc.edu/~rowlett/units/index.html`.

[38] K. Rajalingham, D. Chadwick, B. Knight, and D. Edwards. Quality Control in Spreadsheets: A Software Engineering-Based Approach to Spreadsheet Development. In *33rd Hawaii Int. Conf. on System Sciences*, pages 1–9, 2000.

[39] Kamalasen Rajalingham, David Chadwick, Brian Knight, and Dilwyn Edwards. Quality control in spreadsheets: A software engineering-based approach to spreadsheet development. In *33rd Hawaii Int. Conf. on System Sciences-Volume 4*, pages 1–9, 2000.

[40] Kamalasen Rajalingham, David R. Chadwick, and Brian Knight. Classification of spreadsheet errors. *CoRR*, abs/0805.4224, 2008.

[41] B. Ronen, M. A. Palley, and H. C. Lucas, Jr. Spreadsheet Analysis and Design. *Communications of the ACM*, 32(1):84–93, 1989.

[42] G. Rothermel, M. M. Burnett, L. Li, C. DuPuis, and A. Sheretov. A Methodology for Testing Spreadsheets. *ACM Transactions on Software Engineering and Methodology*, pages 110–147, 2001.

[43] J. Sajaniemi. Modeling Spreadsheet Audit: A Rigorous Approach to Automatic Visualization. *Journal of Visual Languages and Computing*, 11:49–82, 2000.

[44] C. Scaffidi, M. Shaw, and B. Myers. Estimating the Numbers of End Users and End User Programmers. In *IEEE Symp. on Visual Languages and Human-Centric Computing*, pages 207–214, 2005.

[45] Christopher Scaffidi, Mary Shaw, and Brad Myers. Estimating the numbers of end users and end user programmers. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 207–214. IEEE Computer Society, 2005.

[46] E. J. Weyuker, S. N. Weiss, and D. Hamlet. Comparison of Program Testing Strategies. In *Fourth Symposium on Software Testing, Analysis and Verification*, pages 154–164, 1991.

[47] A. G. Yoder and D. L. Cohn. Real Spreadsheets for Real Programmers. In *Int. Conf. on Computer Languages*, pages 20–30, 1994.