Spreadsheet Error Detection with a Type System

Xiao, Qi, Yan

March 4, 2016

1 Motivation

According to U.S. Bureau of Labor and Statistics, there will be fewer than 3 million professional programmers, but more than 55 million people using spreadsheets and databases at work by 2012 [7]. Despite the popularity of spreadsheet programming, the lack of training of these non-professional end-user programmers leads to many errors in business spreadsheets that on average the cell error rate is 3.9% [6]. These errors cost great loss.

Spreadsheet errors fall in to three categories: logical errors (45%), mechanical errors (23%), and omission errors (31%). Logical errors cover the most part and the main reasons account for logical errors are mathematical and domain knowledge. Mechanical errors refer to the cases in which wrong cells are pointed at or typing errors occurs. Omission errors is similar to mechanical errors which happens due to the cases that leaves out necessary components. Because of the complex dependency in spreadsheet programming, these errors are hard to be detected or fixed clearly through manual check.

In our study, we are interested in how to detect the mechanical errors, omission errors and part of logical errors in spreadsheet using static analysis techniques, like type system?

2 Method

In this section, we will detail the method for our study.

2.1 Definition

2.1.1 Syntax of spreadsheets

Note: e is expression, ε is blank cell, ϵ is error cell, v is value, a is address, ω is operations, n is the number of parameters required by the operation.

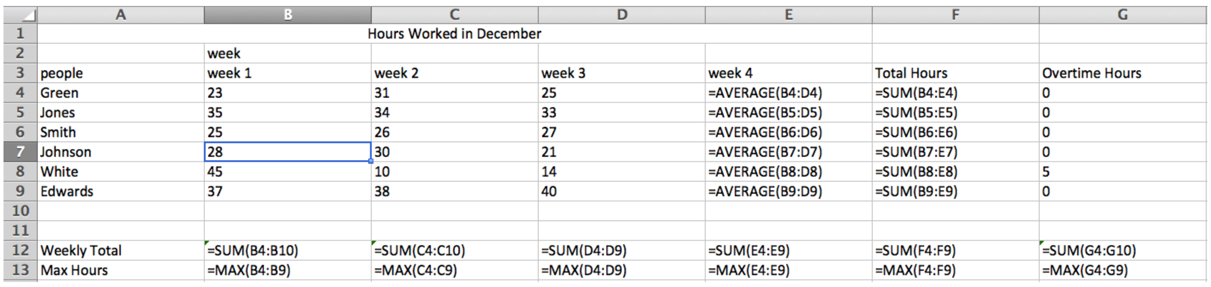
Table 1: Syntax of spreadsheets

Cell address	:	a
Cell expression	:	$e ::= \varepsilon \vert \epsilon \vert v \vert a \vert \omega^n(e_1, ..., e_n)$
Spreadsheet	:	$s ::= (a_1, e_1)\ ;\ .\ .\ .\ ;\ (a_m, e_m) i \neq j \Rightarrow a_i \neq a_j$

2.1.2 Basic Types

Basic Types: Dependent Type, AND Type, and OR Type.

- Dependent Type: Cells with value v and header t are in type $t[v]$, e.g. type for cell B3 is **week[week1]**. It can be recursively defined. e.g. type for cell B7 is **week[week1[28]]**.
- AND Type: Cells with one more types are in type $t1[v]\&t2[v]$, e.g. type for cell B7 is **week[week1[28]]&people[Johnson[28]]**.
- OR Type: Cells with operations referred to other n cells will have a type $t_1[v]|t_2[v]...|t_n[v]$, e.g. type for cell B12 is **week[week1[23]]&people[Green[23]] | ... | week[week1[37]]&people[Edwards[37]]**.



	A	B	C	D	E	F	G
1			Hours Worked in December				
2		week					
3	people	week 1	week 2	week 3	week 4	Total Hours	Overtime Hours
4	Green	23	31	25	=AVERAGE(B4:D4)	=SUM(B4:E4)	0
5	Jones	35	34	33	=AVERAGE(B5:D5)	=SUM(B5:E5)	0
6	Smith	25	26	27	=AVERAGE(B6:D6)	=SUM(B6:E6)	0
7	Johnson	28	30	21	=AVERAGE(B7:D7)	=SUM(B7:E7)	0
8	White	45	10	14	=AVERAGE(B8:D8)	=SUM(B8:E8)	5
9	Edwards	37	38	40	=AVERAGE(B9:D9)	=SUM(B9:E9)	0
10							
11							
12	Weekly Total	=SUM(B4:B10)	=SUM(C4:C10)	=SUM(D4:D9)	=SUM(E4:E9)	=SUM(F4:F9)	=SUM(G4:G10)
13	Max Hours	=MAX(B4:B9)	=MAX(C4:C9)	=MAX(D4:D9)	=MAX(E4:E9)	=MAX(F4:F9)	=MAX(G4:G9)

Figure 1: An example from corpus in formula view

2.1.3 Typing Rules

To check whether every cell is well-typed, we also refer to a few typing rules. By conducting induction on the expressions, we have the rules as shown as follows:

$$\frac{}{v^{\sqcup} : 1}\ (\textsc{Val})$$

$$\frac{(a, s(a)) :: t \qquad \vdash t}{a : t}\ (\textsc{Ref})$$

$$\frac{e : t \qquad t \doteq t'}{e : t'}\ (\textsc{Eq})$$

$$\frac{e_i : t_i \qquad \vdash t_i}{\omega^n(e_1, ...e_n) : \tau(1, t_1, ...t_n)}\ (\textsc{App})$$

2.1.4 Judgments

1. Every value that does not have a header is a well-typed. The type is “1”.
2. If a cell has value v and header t, then it $t[v]$ is well-typed. The type can be recursively definded.
3. For AND Type, if there is no common ancestor, it is well-typed. That is because, e.g. the working hours in the example cannot simultaneously belongs to person A and person B. If so, it is possible to be incorrect.
4. For OR Type, if there is a common header ancestor, it is well-typed. That is because, e.g. the sum of working hours in one week should not contain the numbers from week 1 and week 2. If so, it is possible to be incorrect.

3 Simple Example

Refer to Figure 1.

- By spatial analysis and typing rules, we can know the “type” for each cell.
- For cell B12, by spatial analysis, we know that the type is **WeeklyTotal[193] & week[week1[193]]**. Since there is no common ancestor of **WeeklyTotal[193] and week[week1[193]]**, with Judgment 3, we know it is well-typed.
- For cell B12, by analyzing the operational semantics, we know that the type is **week[week1[23]]&people[Green[23]]** | **...** | **week[week1[37]]&people[Edwards[37]]**. Since there is common ancestor week, thus with Judgment 4, we know it is well-typed.
- If there is a cell with formula “A5 + C4”, via analysis, we know it is typed as **people[Jones]**|**week[week2[31]]**. Since there is no common ancestor, it is ill-typed and it will be marked as a potential wrong cell.

4 Expectation of Project

In this section, we will list the expectation output of our study.

4.1 Implementation

We are going to implement a type system for spreadsheet error detection. The following are the tools and paper we will refer to.

1. Parser: http://ewbi.blogs.com/develops/popular/excelformulaparsing.html
2. Type system:
   (a) basic: [2]
   (b) advanced: [4]
3. Spatial Analysis:
   (a) basic: [1]
   (b) advanced: [3]
4. Evaluation: We will use the benchmark provided by End-user Spreadsheet Corpus [5].

4.2 Report

We will report with a survey with all the papers in the bibtex file. We will propose possible improvement based on current implementation in two possible tracks: (1) Extend typing rules with operational semantics of more spreadsheet functions; (2) Automatic error fixing (creating a fixing language (DSL) and do syntactical heuristic search).

5 Plan

We will finish the implementation of basic type system and basic spatial analysis by the end of March; and extend the system in proposed direction till the mid of April. We will focus on report drafting in the left two weeks.

References

[1] Robin Abraham and Martin Erwig. Header and unit inference for spreadsheets through spatial analyses. In *Visual Languages and Human Centric Computing, 2004 IEEE Symposium on*, pages 165–172. IEEE, 2004.

[2] Robin Abraham and Martin Erwig. Type inference for spreadsheets. In *Proceedings of the 8th ACM SIGPLAN international conference on Principles and practice of declarative programming*, pages 73–84. ACM, 2006.

[3] Chris Chambers and Martin Erwig. Automatic detection of dimension errors in spreadsheets. *Journal of Visual Languages & Computing*, 20(4):269–283, 2009.

[4] Jácome Cunha, Joao Paulo Fernandes, Jorge Mendes, and Joao Saraiva. Embedding, evolution, and validation of model-driven spreadsheets. *Software Engineering, IEEE Transactions on*, 41(3):241–263, 2015.

[5] Marc Fisher and Gregg Rothermel. The euses spreadsheet corpus: a shared resource for supporting experimentation with spreadsheet dependability mechanisms. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–5. ACM, 2005.

[6] Ray Panko. What we don't know about spreadsheet errors today: The facts, why we don't believe them, and what we need to do. *arXiv preprint arXiv:1602.02601*, 2016.

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