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 [27]. 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% [26]. These errors cost a great loss.

Spreadsheet errors fall into 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 are similar to mechanical errors that happen due to the cases leaving 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.

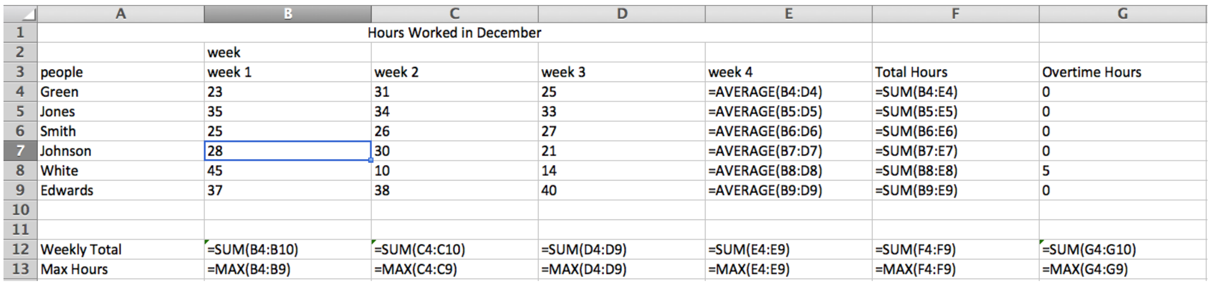
Table 1: Syntax of spreadsheets

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 shown as follows:

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

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

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

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

2.1.4 Judgments

1. Every value that does not have a header is well-typed. The type is “1”.
2. If a cell has value v and header t, it is well-typed, and the type is $t[v]$. 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 simultaneously 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 potentially wrong cell.

4 Expectation of Project

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

4.1 Implementation

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

1. Parser: http://ewbi.blogs.com/develops/popular/excelformulaparsing.html
2. Type system:
   (a) basic: [3]
   (b) advanced: [18]
3. Spatial Analysis:
   (a) basic: [1]
   (b) advanced: [14]
4. Evaluation: We will use the benchmark provided by End-user Spreadsheet Corpus [21].

4.2 Report

We will report with a survey with all the papers (not limited to) in the reference. 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 are going to do research on related work during the first two weeks of March. Based on what we have learned, we will finish the implementation of basic type system and basic spatial analysis by the end of March. After that, to get a step further, we will extend the system in proposed direction till the mid of April. Finally, we will focus on report drafting and finish the report 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. How to communicate unit error messages in spreadsheets. In *ACM SIGSOFT Software Engineering Notes*, volume 30, pages 1–5. ACM, 2005.

[3] 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.

[4] Robin Abraham and Martin Erwig. Ucheck: A spreadsheet type checker for end users. *Journal of Visual Languages & Computing*, 18(1):71–95, 2007.

[5] Robin Abraham and Martin Erwig. Mutation operators for spreadsheets. *Software Engineering, IEEE Transactions on*, 35(1):94–108, 2009.

[6] Robin Abraham, Martin Erwig, and Scott Andrew. A type system based on end-user vocabulary. In *Visual Languages and Human-Centric Computing, 2007. VL/HCC 2007. IEEE Symposium on*, pages 215–222. IEEE, 2007.

[7] Rui Abreu, Birgit Hofer, Alexandre Perez, and Franz Wotawa. Using constraints to diagnose faulty spreadsheets. *Software Quality Journal*, 23(2):297–322, 2015.

[8] Rui Abreu, André Riboira, and Franz Wotawa. Constraint-based debugging of spreadsheets. In *CIbSE*, pages 1–14, 2012.

[9] Sorin Adam and Ulrik Pagh Schultz. Towards tool support for spreadsheet-based domain-specific languages. In *Proceedings of the 2015 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, pages 95–98. ACM, 2015.

[10] Yanif Ahmad, Tudor Antoniu, Sharon Goldwater, and Shriram Krishnamurthi. A type system for statically detecting spreadsheet errors. In *Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on*, pages 174–183. IEEE, 2003.

[11] Tudor Antoniu, Paul A Steckler, Shriram Krishnamurthi, Erich Neuwirth, and Matthias Felleisen. Validating the unit correctness of spreadsheet programs. In *Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on*, pages 439–448. IEEE, 2004.

[12] Margaret Burnett and Martin Erwig. Visually customizing inference rules about apples and oranges. In *Human Centric Computing Languages and Environments, 2002. Proceedings. IEEE 2002 Symposia on*, pages 140–148. IEEE, 2002.

[13] Chris Chambers and Martin Erwig. Dimension inference in spreadsheets. In *Visual Languages and Human-Centric Computing, 2008. VL/HCC 2008. IEEE Symposium on*, pages 123–130. IEEE, 2008.

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

[15] Chris Chambers and Martin Erwig. Combining spatial and semantic label analysis. 2009.

[16] Chris Chambers and Martin Erwig. Reasoning about spreadsheets with labels and dimensions. *Journal of Visual Languages & Computing*, 21(5):249–262, 2010.

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

[18] 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.

[19] Wensheng Dou, Shing-Chi Cheung, and Jun Wei. Is spreadsheet ambiguity harmful? detecting and repairing spreadsheet smells due to ambiguous computation. In *Proceedings of the 36th International Conference on Software Engineering*, pages 848–858. ACM, 2014.

[20] Martin Erwig and Margaret Burnett. Adding apples and oranges. In *Practical Aspects of Declarative Languages*, pages 173–191. Springer, 2002.

[21] 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.

[22] Felienne Hermans and Danny Dig. Bumblebee: A refactoring environment for spreadsheet formulas. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 747–750. ACM, 2014.

[23] Felienne Hermans, Martin Pinzger, and Arie van Deursen. Detecting and refactoring code smells in spreadsheet formulas. *Empirical Software Engineering*, 20(2):549–575, 2015.

[24] Birgit Hofer and Franz Wotawa. Why does my spreadsheet compute wrong values? In *Software Reliability Engineering (ISSRE), 2014 IEEE 25th International Symposium on*, pages 112–121. IEEE, 2014.

[25] Joseph Lawrance, Robin Abraham, Margaret Burnett, and Martin Erwig. Sharing reasoning about faults in spreadsheets: An empirical study. In *Visual Languages and Human-Centric Computing, 2006. VL/HCC 2006. IEEE Symposium on*, pages 35–42. IEEE, 2006.

[26] 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.

[27] 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.