Graz University of Technology
And University of Porto

Generation of Relevant Spreadsheet Repair Candidates

Birgit Hofer, Rui Abreu, Alexandre Perez and Franz Wotawa

Abstract

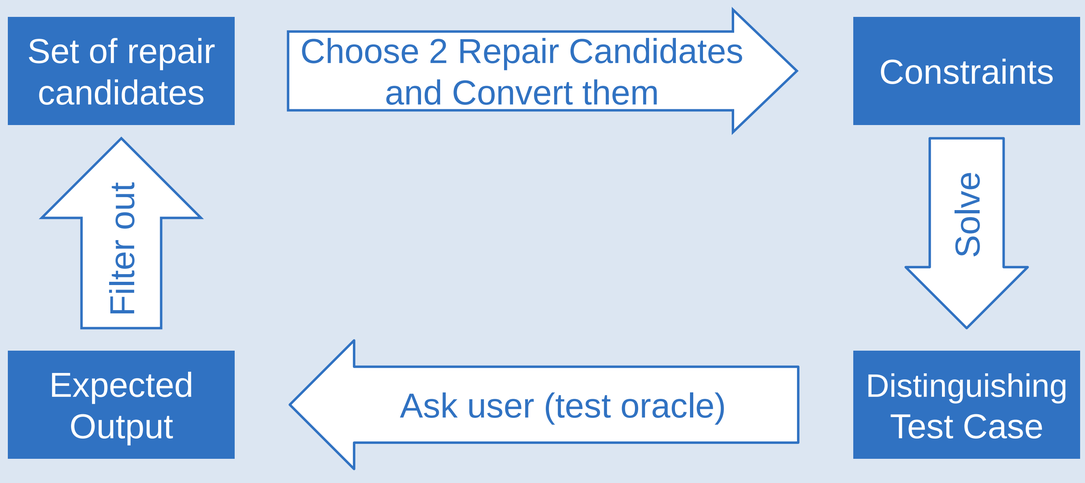
Locating and fixing faults in **spreadsheets** is important. A state-of-the-art technique uses **genetic programming** for generating repair candidates, but this technique computes **too many repair candidates** which hinders real-world application. Therefore, we propose an approach that uses **distinguishing test cases to narrow down the number of repair candidates**.

1 Spreadsheet Repair

Spreadsheets are used **in nearly every company** and **important decisions** are often based on spreadsheets. Unfortunately, they often **contain errors**. Locating and **correcting faults** in spreadsheets can be time consuming and **frustrating**. Therefore several approaches have been proposed which automatically create repair candidates, e.g. Repair by Genetic Programming [1] and GoalDebug [2].

Unfortunately, these approaches often compute a **large set of repair candidates**. A large set often **overwhelms a user**. Therefore, we propose an approach which automatically **narrows down the number of repair candidates**. This is done with the help of **distinguishing test cases** [3]. The user only has to indicate the expected output for the automatically generated distinguishing input.

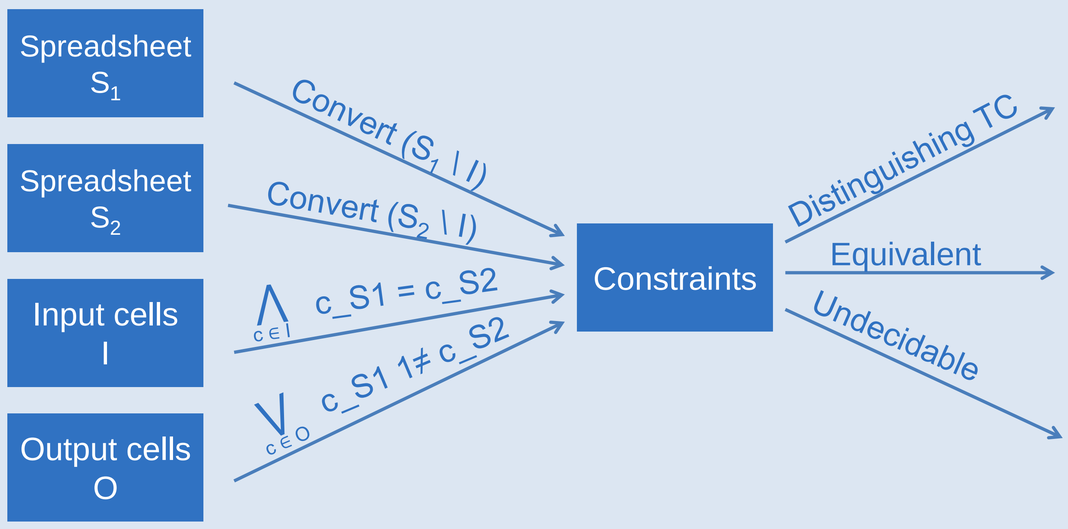
2 MuSSCo (Mutation Supported Spreadsheet Correction)



Pick two mutants which have not (yet) been identified as equivalent or undecidable and convert them into constraints. Stop when you cannot find such mutants. Present remaining repair candidates to the user.

3 Computing Distinguishing Test Cases

A **distinguishing test case** [3] for two spreadsheets leads to at least one different output for the same input.



The selected **repair candidates** are **converted into constraints**. Thereby, all variables of Spreadsheet S_1 get the postfix "_S1" and all variables of Spreadsheet S_2 "_S2" in order to distinguish them. The input cells of the spreadsheets are not encoded into constraints since the solver should find values for these cells. To ensure that **S_1 and S_2** have the **same input values**, we add the corresponding **constraint** to the constraint system. In addition, we add a constraint ensuring that at **least one output cell has a different value for S_1 and S_2**. For such a constraint system, a solver could either return a solution (a distinguishing test case), no solution (in case of equivalence) or unknown (when the solver cannot decide if there exists a solution for the given constraint system).

4 Example

Initial situation: A spreadsheet where a wrong output is observed

	A	B
1	**Cardiogenic Shock Estimator**	
2	**End Diastolic Volume**	120 mL
3	**End Systolic Volume**	60 mL
4	**Heart Rate**	72 bpm
5	**Body Surface Area**	2 m2
6	Stroke Volume	2 mL
7	Cardiac Output	144 mL/min
8	**Cardiac Index**	72 mL/min/m2

Input cells

Formula cells

=B2/B3
=B6*B4
=B7/B5

Output cell

Should be 2160

Repair suggestions: Repair Tools (e.g. [1,2]), often create several solutions that lead to the desired output.

2	**End Diastolic Volume**		120 mL
3	**End Systolic Volume**		60 mL
4	**Heart Rate**		72 bpm
5	**Body Surface Area**		2 m2
6	Stroke Volume	=B2-B3	60 mL
7	Cardiac Output	=B6*B4	4320 mL/min
8	**Cardiac Index**	=B7/B5	2160 mL/min/m2

Repair Candidate S_1

2	**End Diastolic Volume**		120 mL
3	**End Systolic Volume**		60 mL
4	**Heart Rate**		72 bpm
5	**Body Surface Area**		2 m2
6	Stroke Vol	=B2/B3	2 mL
7	Cardiac Ou	=B6*B4*30	4320 mL/min
8	**Cardiac In**	=B7/B5	2160 mL/min/m2

Repair Candidate S_2

MuSSCO: Generation of a distinguishing test case

2	**End Diastolic Volume**		30 mL
3	**End Systolic Volume**		30 mL
4	**Heart Rate**		30 bpm
5	**Body Surface Area**		1 m2
6	Stroke Volume	=B2-B3	0 mL
7	Cardiac Output	=B6*B4	0 mL/min
8	**Cardiac Index**	=B7/B5	0 mL/min/m2

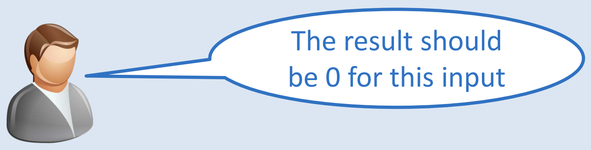
Repair Candidate S_1

Automatically created input

2	**End Diastolic Volume**		30 mL
3	**End Systolic Volume**		30 mL
4	**Heart Rate**		30 bpm
5	**Body Surface Area**		1 m2
6	Stroke Volu	=B2/B3	1 mL
7	Cardiac Out	=B6*B4*30	900 mL/min
8	**Cardiac Ind**	=B7/B5	900 mL/min/m2

Repair Candidate S_2

User: Tells what output he/she expects



References

[1] B. Hofer, and F. Wotawa: „Mutation-based spreadsheet debugging.“ International Workshop on Program Debugging (IWPD) – ISSRE (Supplemental Proceedings) , pp. 132–137, 2013.
[2] R. Abraham, and M. Erwig: “GoalDebug: A spreadsheet debugger for end users”, International Conference on Software Engineering (ICSE ’07 - Proceedings), pp. 251–260, 2007.
[3] M. Nica, S. Nica, and F. Wotawa: “On the use of mutations and testing for debugging.” Software : Practice & Experience 43(9), pp. 1121–1142 , 2013.

ECAI – 21st European Conference on Artificial Intelligence
August, 18 - 22, 2014 – Prague, Czech Republic