EqFix: Fixing LaTeX Equation Errors by Examples

Fengmin Zhu[1,4] Fei He[1,2,3]

[1]School of Software, Tsinghua University

[2]Key Laboratory for Information System Security, MoE, China

[3]Beijing National Research Center for Information Science and Technology

[4]Max Planck Institute for Software Systems

November 1, 2022

Today

Contents

LaTeX: A Universal Syntax for Math Equations

Display math equations in browsers:

MathJax https://www.mathjax.org

WYSIWYG editors like MS Word supports (partial) LaTeX syntax[1]:

[1] This gif is made using the animate package, to view it correctly, open in Acrobat Reader/KDE Okular/PDF-XChange/Foxit Reader.

LaTeX syntax is Cryptic and Mysterious

A novice user expects $\boxed{y^{123}+x}$

types "y^123+x"

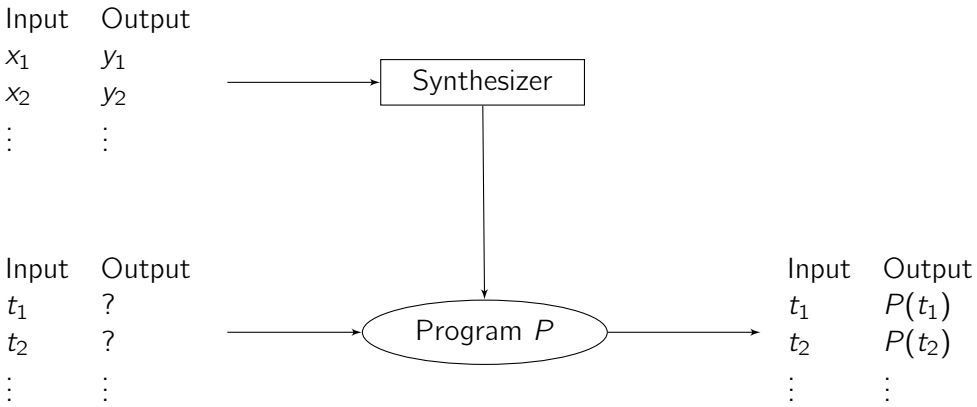
but got unexpected $\boxed{y^{1}23+x}$ (superscript is wrong!)

How to Fix Errors?

Possible methods:

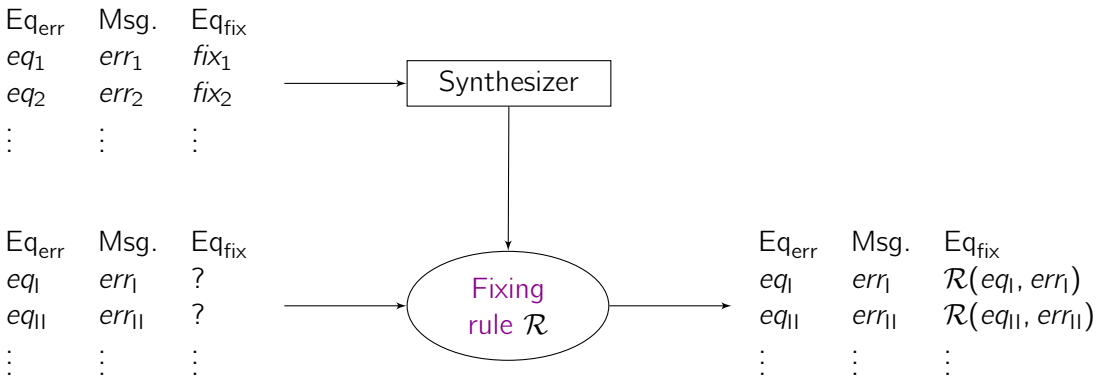
1. Resort to search engine/online help forum
   - Properly describe the issue
   - Adapt the solution (if found)
2. QuickFix: a fixed set of human-specified rules
3. Our approach (**EqFix**): rule-based, but automatically learned from a small set of examples

Programming by Examples (PBE)



Soundness: P is consistent with the input-output examples, i.e., $\forall i, P(x_i) = y_i$

PBE for Fixing LaTeX Equations

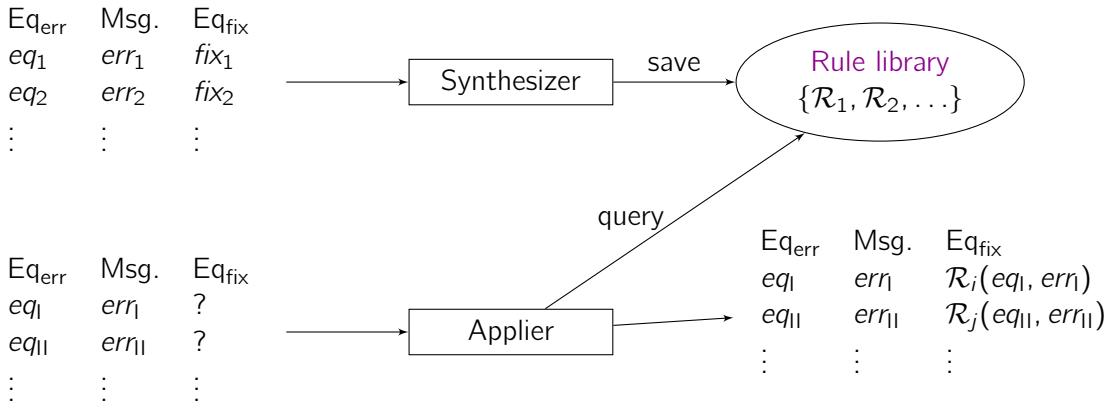


Program: Fixing rule

Input: Erroneous equation (Eq_{err}) + Error message (Msg.)

Output: Fixed equation (Eq_{fix})

Fixing Rules Maintained in Library



Contents

Fixing Rule

A (fixing) rule $\mathcal{R}$ consists of two components:

- An error pattern is a template of the error message
- A transformer specifies the required string transformation on the erroneous parts

Rules are learned from examples!

Input-Output Examples for EqFix

An input-output example consists of an erroneous equation, an error message, and a fixed equation.

Example

#1		#2	
eq_1 :	`$x^10$` x^10	eq_2 :	`$y^123+x$` $y^123 + x$
err_1 :	`superscript 10`	err_2 :	`superscript 123`
fix_1 :	`$x^{10}$` x^{10}	fix_2 :	`$y^{123}+x$` $y^{123} + x$

Fixing Rule (Example)

#1		#2	
eq_1 :	`$x^10$`	eq_2 :	`$y^123+x$`
err_1 :	`superscript 10`	err_2 :	`superscript 123`
fix_1 :	`$x^{10}$`	fix_2 :	`$y^{123}+x$`

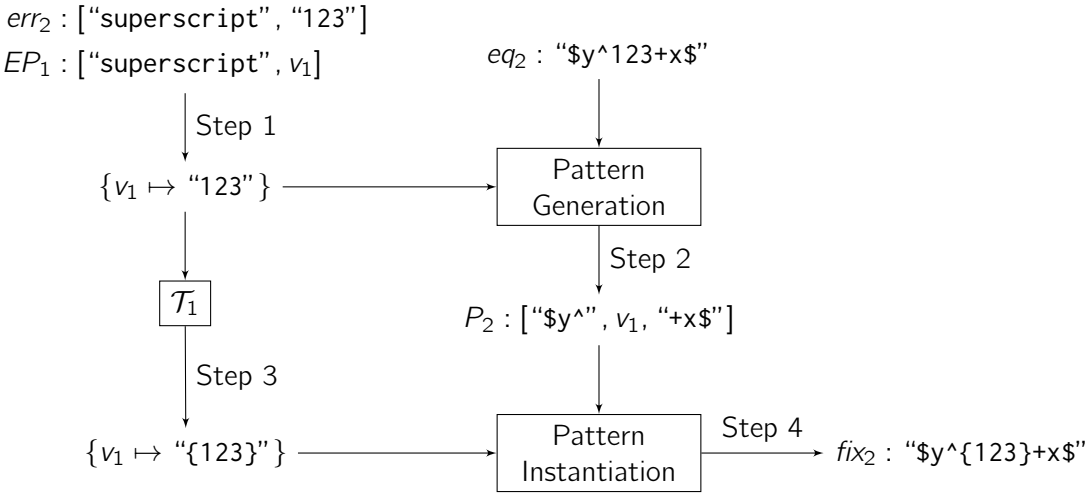
Example

A fixing rule consistent with #1 (and also #2): $\mathcal{R}_1 = \langle EP_1, \mathcal{T}_1 \rangle$ where

- the error pattern $EP_1 = [\text{“}\texttt{superscript}\text{”}, v_1]$ has one variable v_1
- the transformer $\mathcal{T}_1 = \{v_1 \mapsto f_1\}$ has one underlying string transformer (a function of type string $\to$ string) $f_1(x) = \text{“}\{\text{”} + x + \text{“}\}\text{”}$

Rule Application

$\mathcal{R}_1(eq_2, err_2) = fix_2$

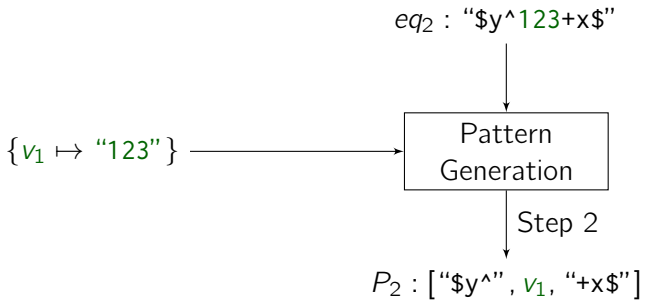


Step 1: Extract Problem-Specific Information

$$err_2 : [\text{“superscript”}, \text{“123”}]$$
$$EP_1 : [\text{“superscript”}, v_1]$$
$$\downarrow \text{ Step 1}$$
$$\{v_1 \mapsto \text{“123”}\}$$

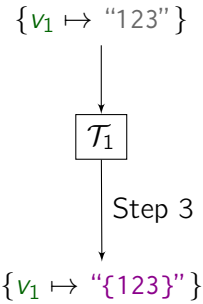
Match the error pattern against the concrete (tokenized) error message: obtain a mapping from variables to values.

Step 2: Equation Pattern Generation



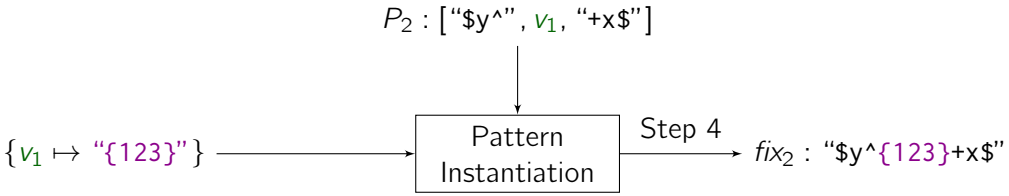
Abstract over the erroneous equation into an equation pattern—this pattern matches both the erroneous and the fixed equation.

Step 3: Transformation on Extracted Values



Apply the underlying string transformer for each variable: obtain a new mapping with transformed values.

Step 4: Equation Pattern Instantiation

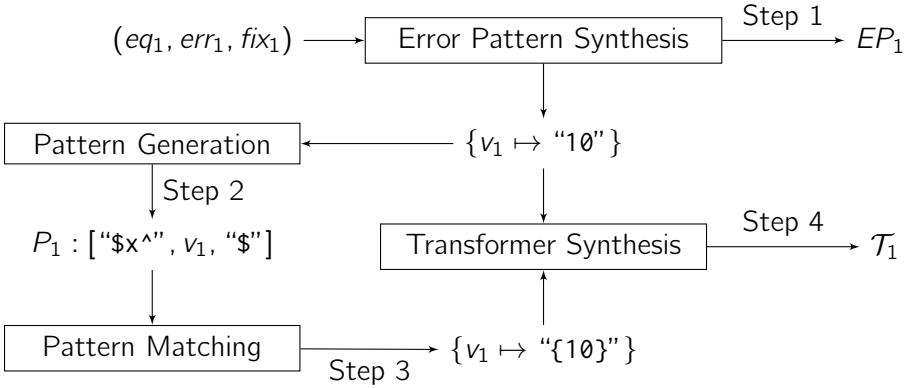


Instantiate the equation pattern with the new mapping: produce the fix.

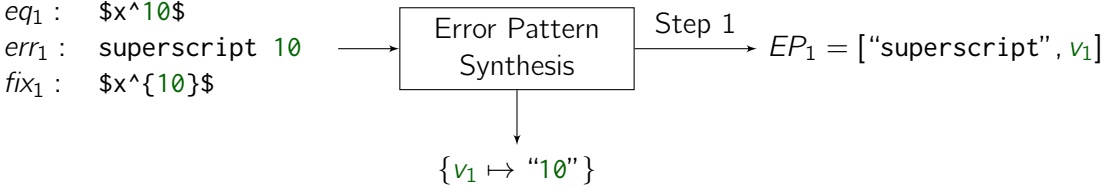
Contents

Rule Synthesis

Learn $\mathcal{R}_1$ by example (eq_1, err_1, fix_1)

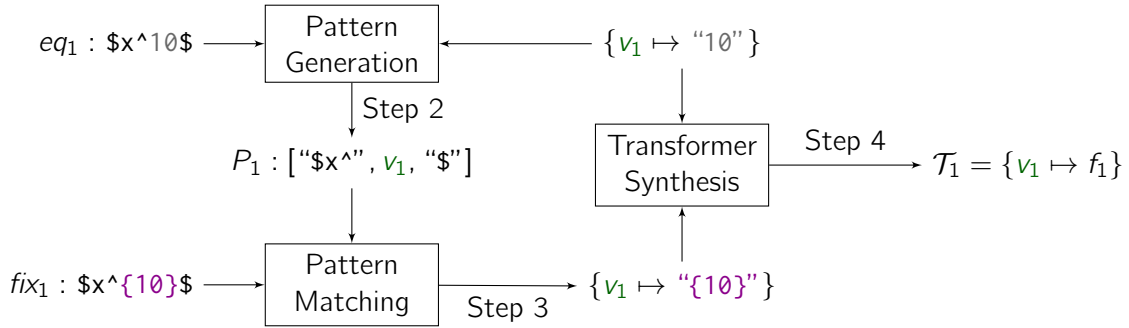


Error Pattern Synthesis



Replace tokens of the error message that occur in the input/output equation as variables in the error pattern.

Transformer Synthesis



The specification for synthesizing f_1 is a set of string input-output examples $\{$"10" $\mapsto$ "{10}"$\}$. We adopt FlashFill's [Gul11] approach for the synthesis.

More on our paper and extended version [ZH22]:

- Formal definition of fixing rules
- Synthesis procedures
- Relaxer extension

Contents

Evaluation Setup

Baseline: FlashFill (Implementation in PROSE[2])

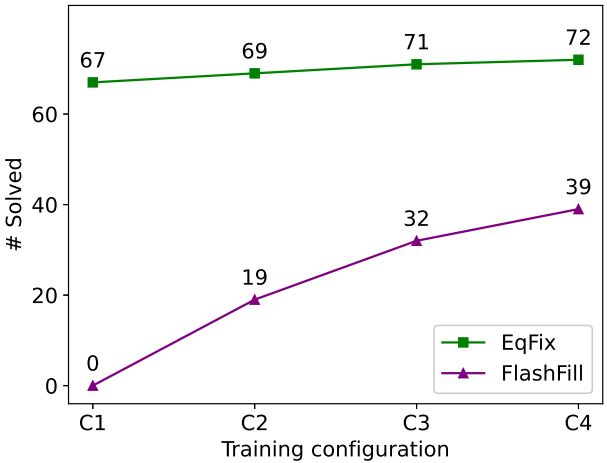
Dataset: 89 examples groups; for each example group:

- take the shortest 1-4 ($C1$-$C4$) examples for learning/training
- take the longest for testing

#	Training				Testing
1	Example 1-1	Example 1-2	Example 1-3	Example 1-4	Example 1-5
2	Example 2-1	Example 2-2	Example 2-3	Example 2-4	Example 2-5
⋮	⋮	⋮	⋮	⋮	⋮

[2] `https://github.com/microsoft/prose`

Solved Testcases



Configuration Ci: learning from the first i examples (for each group).

EqFix Simplifies String Transformer Synthesis

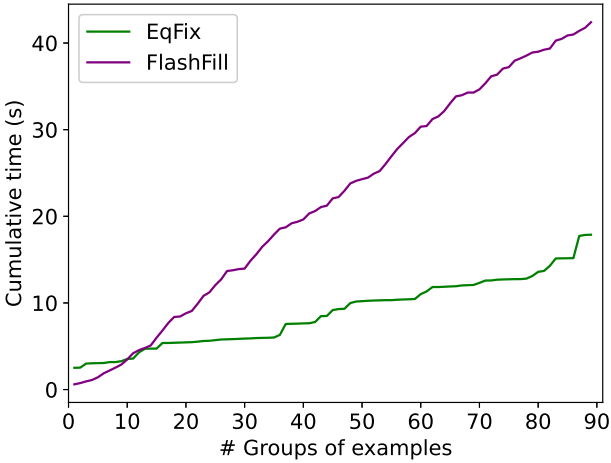
To learn a rule from:

$$
\begin{aligned}
eq_1 &: \quad \texttt{\$x\^{}10\$} \\
err_1 &: \quad \texttt{superscript 10} \\
fix_1 &: \quad \texttt{\$x\^{}\{10\}\$}
\end{aligned}
$$

- FlashFill needs to learn a string transformer for (“\$x^10\$”, err_1) $\mapsto$ “\$x^{10}\$”
- EqFix needs to learn a string transformer for “10” $\mapsto$ “{10}”

Cummulative Synthesis Time

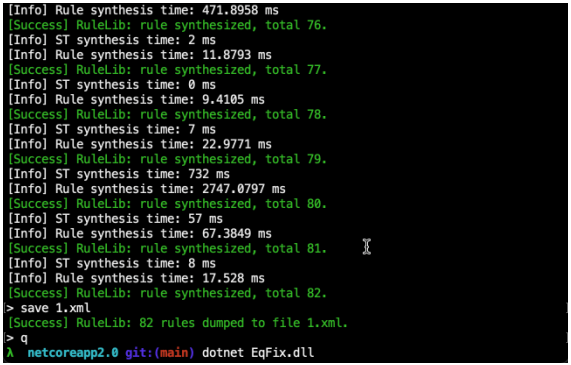
Under configuration C4



Enabling Rule Library

Train and save rules to an XML file:

Load rules from that XML file and test:



References

Sumit Gulwani.

Automating string processing in spreadsheets using input-output examples.

In *Proceedings of the 38th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '11, pages 317–330. ACM, 2011.

doi:10.1145/1926385.1926423.

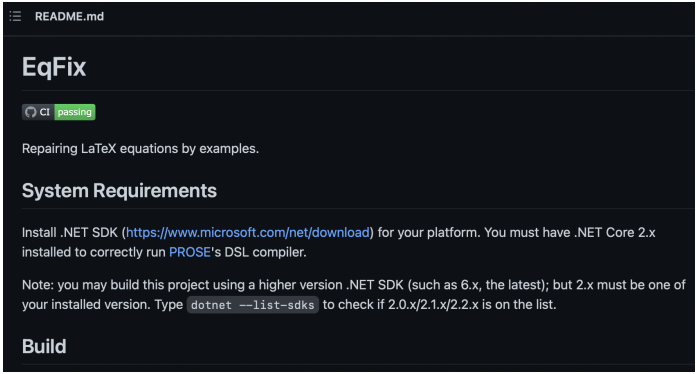
Fengmin Zhu and Fei He.

Eqfix: Fixing latex equation errors by examples, 2022.

URL: https://arxiv.org/abs/2107.00613, doi:10.48550/ARXIV.2107.00613.

Our prototype tool and evaluation data are publicly available:

https://github.com/thufv/EqFix


README.md

EqFix

CI passing

Repairing LaTeX equations by examples.

System Requirements

Install .NET SDK (https://www.microsoft.com/net/download) for your platform. You must have .NET Core 2.x installed to correctly run PROSE's DSL compiler.

Note: you may build this project using a higher version .NET SDK (such as 6.x, the latest); but 2.x must be one of your installed version. Type `dotnet --list-sdks` to check if 2.0.x/2.1.x/2.2.x is on the list.

Build