Adopting Tree Overlapping Algorithm for MathML Equation Structural Similarity Evaluation

Mikhail Ponomarev
Peter the Great St. Petersburg Polytechnic University
29 Polytechnicheskaya st.
195251 St. Petersburg Russia
ponmike92@gmail.com

Evgeny Pyshkin
University of Aizu
Software Engineering Lab
Tsuruga, Ikki-machi, Aizu-Wakamatsu
Fukushima, Japan 965-8580
pyshe@u-aizu.ac.jp

ABSTRACT

The paper is focused on an approach to computing structural syntactic similarity of mathematical equations presented in MathML. We examine a modification of a tree overlapping algorithm adopted to a purpose of describing mathematical equation similarity patterns.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.5.1 [**Pattern Recognition**]: Models—*structural*

General Terms

Algorithms, Human Factors

Keywords

Syntactic similarity, Mathematical equations, MathML, Tree overlapping

1. INTRODUCTION

Nowadays there is a few models of adopting natural language processing (NLP) algorithms related to syntax similarity to the specific notations used in mathematics. Mathematical equations (with their unique structural syntax with a big variety of semantically equivalent constructions) provide a non-trivial case for information retrieval [6]. Many reported implementations are focused on exact matching of mathematical constructions rather than on their similarity [5, 4]. Indeed, for a case of mathematical equations, syntactical similarity is defined rather fuzzy by using several structural syntactical similarity patterns. However, such a model would be very useful while developing searching and classification tools, especially used in education by math learners and tutors that would allow selecting suitable tasks to nail down a topic presented during a classroom session.

For example, there is an obvious use case which is accessing a set of relevant mathematical equations to be used for learner's training while doing preparation works for an examination. One more option is using such tools for searching an equation by its syntactical structure, the latter being often easier to recall comparing to exact mathematical formulas.

For the reason that most structural notations used for mathematical expression representation are in fact based on directed graphs, syntactic similarity can be defined by using tree structural similarity. Specifically, in this work we use the expressions uniformly presented in MathML[1] which is one of widely used structural XML based mathematical notations. In turn, if we use better structural forms to represent a math equation, we can expect more efficient and accurate retrieval in contrast to the not rare case of using image based equation representation on many web sites. At the same time we accept a possible criticism that not an every mathematical expression retrieval difficulty could be addressed under limitations of MathML representability.

2. STRUCTURAL SIMILARITY OF MATHEMATICAL EQUATIONS

In [1] similarity of two trees is defined on the base of recursive examination of their subtrees. In [3] the following mathematical expressions similarity patterns are defined:

Mathematical equivalence: Equations E_1 and E_2 are mathematically equivalent if they are semantically the same (but not obligatorily syntactically the same, for example $\frac{d(sin(x))}{dx}$ and $(sin(x))'$, $sin^2(x) + cos^2(x)$ and 1.

Identity: E_1 and E_2 are identical if they are exactly the same.

Syntactical identity: E_1 and E_2 are syntactically identical if they are identical after normalization (dealing with variable names and numeric values). For example $sin(a)$ and $sin(b)$, $\frac{1}{sin(x)}$ and $\frac{5}{sin(x)}$.

Expression n–similarity: Normalized equations E_1 and E_2 are n-similar if similarity (in a certain sense) $sim(E_1, E_2) \geqslant n$, n being a parametric value determining the threshold. There are two specific cases of n–similarity which are particularly important for our work:

1. **Subexpression n–similarity**: There is a subexpression n–similarity for E_1 and E_2, if they are n–similar and the corresponding trees both contain the common

[1]MathML – Mathematical Markup Language

subtree which in turn contains all the terminal nodes of both trees. Figure 1 shows an example for the case of $sin(x)^2$ and $\frac{sin(x)}{2}$.

2. **Structural n–similarity**: E_1 and E_2 are structurally n–similar if they are n–similar (in a common sense) and there is a common part in both trees rooted at root nodes of the compared trees with the production rules being the same for all the nodes in this part. Figure 2 illustrates this case for the equations $x + \sqrt{sin(a)}$ and $x + \sqrt{2b}$.

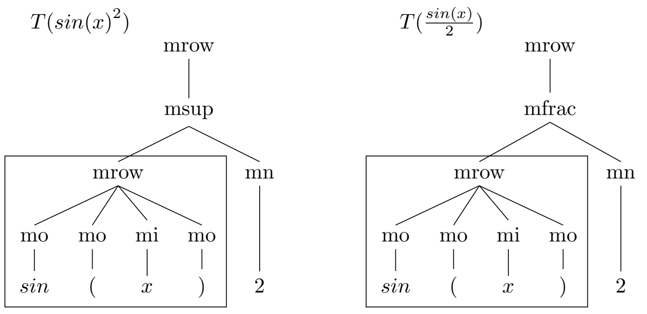


Figure 1: Equations $sin(x)^2$ and $\frac{sin(x)}{2}$ are structurally n–similar for any value of $n \geqslant \frac{18}{26}$

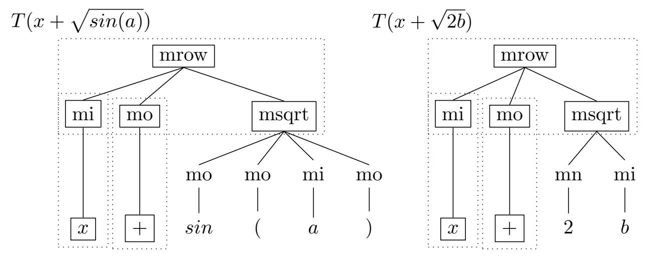


Figure 2: $x + \sqrt{sin(a)}$ and $x + \sqrt{2b}$ are structurally n–similar for any value of $n \geqslant \frac{12}{24}$

Note that in Figures 1 and 2 for n-values we use nodes ratio where a fraction's numerator corresponds to the number of the common nodes in both trees while its denominator corresponds to the number of all nodes in both trees.

2.1 Tree Overlapping Algorithm

A basic tree overlapping algorithm is described in [2] for a case of sentence similarity which is defined as follows. When putting an arbitrary node n_1 of a tree T_1 on a node n_2 of a tree T_2, there might be the same production rule overlapping in T_1 and T_2. Tree similarity is defined as a number of such overlapping production rules.

2.2 Modifying Tree Overlapping Algorithm for Math Equation Structural Similarity

In contrast to the base algorithm from [2] where tree terminals are naturally excluded, for a case of mathematical equations we propose to include terminal nodes as if they had the same production rules (*Relaxation 1*). Also we relax the strictness of the base algorithm and include the pairs of corresponding nodes which are in the same order among their siblings but do not have the same production rules for their child nodes (*Relaxation 2*). Below there is a formal definition of our modification.

Suppose $L(n_1, n_2)$ represents a set of overlapping node pairs when putting n_1 on n_2. It means that if $ch(n, i)$ is i-th child of node n then $L(n_1, n_2)$ is being generated by the following rules:

1. $(n_1, n_2) \in L(n_1, n_2)$
2. If $(m_1, m_2) \in L(n_1, n_2)$, then $(ch(m_1, i), ch(m_2, i)) \in L(n_1, n_2)$
3. $L(n_1, n_2)$ includes all the pairs generated recursively by the rule No. 2.

A number $N_{TO}(n_1, n_2)$ of production rules in question (according to the *Relaxation 1*) is defined as follows:

$$N_{TO}(n_1,n_2) = \left\{ (m_1,m_2) \middle| \begin{array}{l} m_1 \in nodes(T_1) \\ \wedge\ m_2 \in nodes(T_2) \\ \wedge\ (m_1,m_2) \in L(n_1,n_2) \\ \wedge\ PR(m_1) = PR(m_2) \end{array} \right\} \quad (1)$$

In equation 1 $nodes(T)$ is a set of nodes (including terminals) in a tree T, while $PR(n)$ is a production rule rooted at the node n.

Figure 3 shows an example of overlapping tree modification algorithm for $N_{TO}(d_1, d_2) = \{(d^1, d^2), (f^1, f^2), (g^1, g^2)\}$.

According to the *Relaxation 2*, suppose $P_{WPR}(n_1, n_2)$ is a set of nodes which is represented as a path from (n_1, n_2) to the top last pair of nodes being in the same order among their siblings. Suppose n_i and m_i are nodes of a tree T_i, $ch(n, i)$ is i-th child of node n. P_{WPR} is defined as follows:

1. $(n_1, n_2) \notin P_{WPR}$
2. If $PR(parent(n_1)) \neq PR(parent(n_2))$
$\wedge\ ch(parent(n_1), i) = ch(parent(n_2), i)$
$\wedge\ ch(parent(n_1), i) = n_1$
$\wedge\ h(parent(n_2), i) = n_2$,
$(parent(n_1), parent(n_2)) \in P_{WPR}$
3. $P_{WPR}(n_1, n_2)$ includes only pairs generated by applying rule No. 2.

Then the second component for an integral similarity measure can be defined bu using the above introduced P_{WPR} as follows:

$$P_{TO}(n_1,n_2) = \left\{ (m_1,m_2) \middle| \begin{array}{l} (p_1,p_2) \in N_{TO}(n_1,n_2) \\ (m_1,m_2) \in P_{WPR}(p_1,p_2), \\ \qquad \text{if} \quad top(m_1,m_2) = (n_1,n_2) \end{array} \right\} \quad (2)$$

In equation 2 $top(n_1, n_2)$ is the last pair in set $P_{WPR}(n_1, n_2)$:

$$top(n_1,n_2) = p_{last}(n_1,n_2), \qquad p_{last} \in P_{WPR} \quad (3)$$

Thus, for two nodes the resulting combined similarity measure is defined as follows:

$C_{TO}(n_1, n_2) = |N_{TO}(n_1, n_2)| + |P_{TO}(n_1, n_2)|$

For the whole trees, we get:

$$S_{TO}(T_1,T_2) = \max_{n_1 \in nodes(T_1), n_2 \in nodes(T_2)} C_{TO}(n_1,n_2) \quad (4)$$

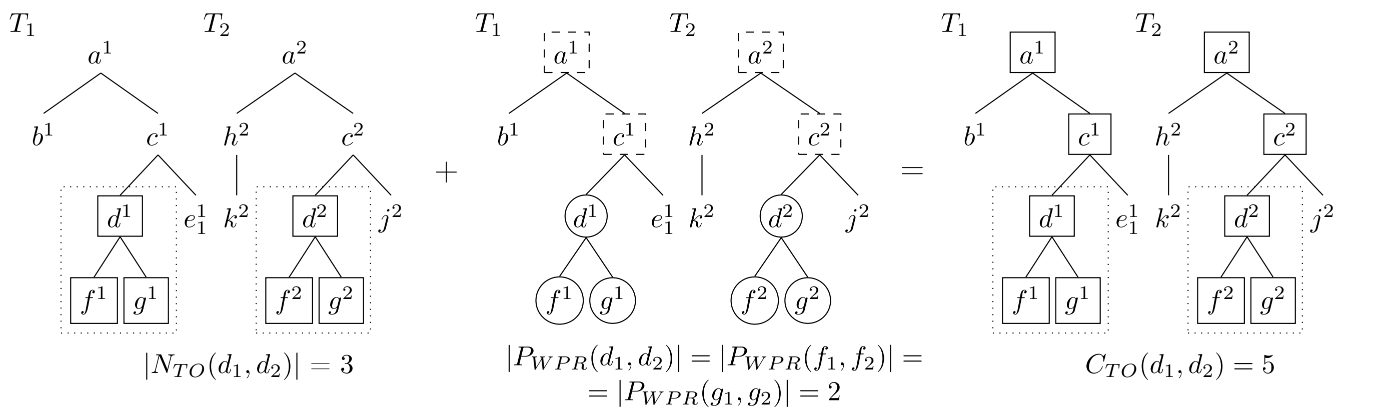
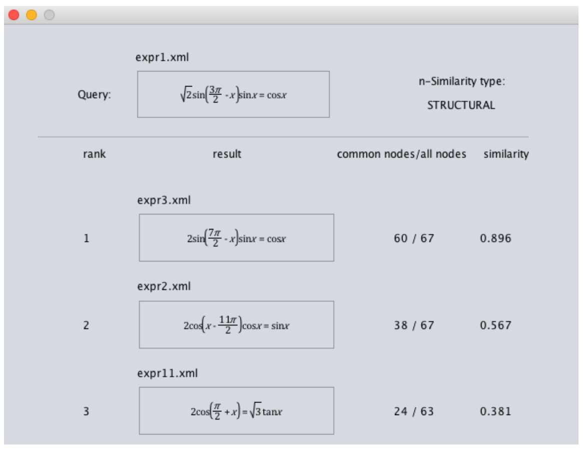


Figure 3: Modified Tree-Overlapping Algorithm: Example

2.3 Software Implementation

We developed a software prototype in order to arrange a series of experiments for our modification of the tree overlapping algorithm for a case of mathematical equations. Figure 4 gives a hint of how the application user interface is organized.



expr1.xml

Query: $\sqrt{2}\sin\left(\frac{3\pi}{2} - x\right)\sin x = \cos x$

n–Similarity type:

STRUCTURAL

rank	result	common nodes/all nodes	similarity
1	expr3.xml $2\sin\left(\frac{7\pi}{2} - x\right)\sin x = \cos x$	60 / 67	0.896
2	expr2.xml $2\cos\left(x - \frac{11\pi}{2}\right)\cos x = \sin x$	38 / 67	0.567
3	expr11.xml $2\cos\left(\frac{\pi}{2} + x\right) = \sqrt{3}\tan x$	24 / 63	0.381

Figure 4: Structural similarity component: GUI

For displaying mathematical equations defined in MathML the library *net.sourceforge.jeuclid* is used.

3. EXPERIMENTS

There is a significant problem we faced while attempting to evaluate our modification algorithm. We discovered that, unlike to the NLP domain, there is no substantial corpus of mathematical equation syntactical similarity classes. So, for our rather preliminary analysis we selected a number of typical trigonometry problems from the set of tasks used in Russian National Exam on Mathematics. Then we involved several experts experienced in teaching mathematics. With their help we classified a selection of expressions in order to proceed with preliminary analysis of our approach.

3.1 Test Corpora

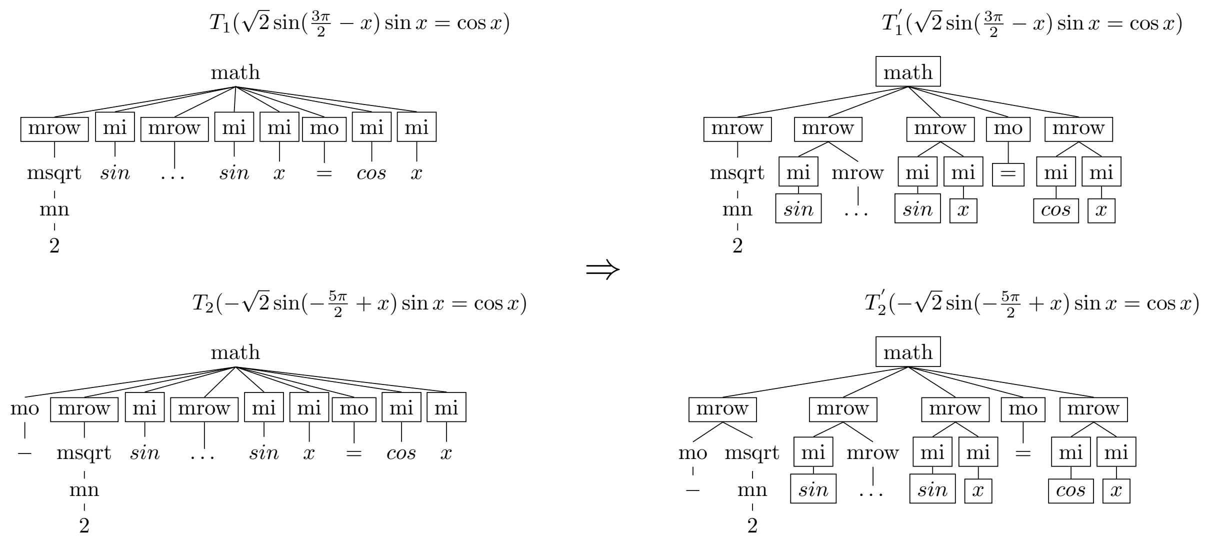
For our initial experiments we created a set of equations classified according their structural similarity (being limited by the paper size we skip here our tests for subexpression similarity). Table 1 lists the equations we used in our experiments.

Table 1: Expression Classification on Structural Similarity

No.	Expression	Class
1	$\sqrt{2}\sin(\frac{3\pi}{2} - x)\sin x = \cos x$	①
2	$2\cos(x - \frac{11\pi}{2})\cos x = \sin x$	
3	$2\sin(\frac{7\pi}{2} - x)\sin x = \cos x$	
4	$-\sqrt{2}\sin(-\frac{5\pi}{2} + x)\sin x = \cos x$	
5	$\cos 2x - 3\cos x + 2 = 0$	②
6	$\cos 2x + 3\sin x - 2 = 0$	
7	$3\cos 2x - 5\sin x + 1 = 0$	
8	$\cos 2x - 5\sqrt{2}\cos x - 5 = 0$	
9	$\cos(\frac{\pi}{2} + 2x) = \sqrt{2}\sin x$	③
10	$\cos 2x = \sin(x + \frac{\pi}{2})$	
11	$2\cos(\frac{\pi}{2} + x) = \sqrt{3}\tan x$	
12	$2\sin^4 x + 3\cos 2x + 1 = 0$	④
13	$4\sin^4 2x + 3\cos 4x - 1 = 0$	
14	$4\cos^4 x - 4\cos^2 x + 1 = 0$	
15	$(2\cos x + 1)(\sqrt{-\sin x} - 1) = 0$	⑤
16	$(2\sin x - 1)(\sqrt{-\cos x} + 1) = 0$	
17	$\sqrt{\cos^2 x - \sin^2 x}(\tan 2x - 1) = 0$	
18	$\cos^2 x - \frac{1}{2}\sin 2x + \cos x = \sin x$	⑥
19	$\frac{1}{2}\sin 2x + sin^2 x - \sin x = \cos x$	
20	$\tan x + \cos(\frac{3\pi}{2} - 2x) = 0$	⑦
21	$\cos x + \cos(\frac{\pi}{2} + 2x) = 0$	
22	$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$	⑧
23	$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$	

3.2 Tests

Though a corpus presented in Table 1 isn't representative enough, it allows us to have some preliminary similarity precision estimation. Let us note that the preliminary experiments described in this work serves us as a prove-of-concept example for investigating further necessary improvements of the developed algorithm. In the future tests a standard cross-fold validation procedure will be required in order to get trustworthy precision evaluation results.


$T_1(\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x=\cos x)$
$T_1'(\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x=\cos x)$
$T_2(-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x=\cos x)$
$T_2'(-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x=\cos x)$
math
mrow
mi
mo
msqrt
mn
2
sin
cos
x
=
−
...
$\Rightarrow$

$$S_{TO}(T_1,T_2)=\frac{0+0}{34+38}=\frac{0}{72}=0 \qquad S_{TO}(T_1',T_2')=\frac{17+17}{37+41}=\frac{34}{78}=0.44$$

Figure 5: Tree structure normalization to avoid a false negative case

3.3 Analysis

In Table 2 we listed 5 expressions with the best scores for the query expression $\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x=\cos x$ (belonging to the class 1).

Table 2: Query: $\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x=\cos x$

Compared expressions	Nodes ratio	Similarity
$2\sin(\frac{7\pi}{2}-x)\sin x=\cos x$	60/67	0.896
$2\cos(x-\frac{11\pi}{2})\cos x=\sin x$	40/67	0.597
$2\cos(\frac{\pi}{2}+x)=\sqrt{3}\tan x$	24/63	0.381
$3\cos 2x-5\sin x+1=0$	12/60	0.200
$\tan x+\cos(\frac{3\pi}{2}-2x)=0$	10/67	0.149

Two best scores are for the equations which also belong to the class 1, except the equation $-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x=\cos x$ (No. 4 in Table 1), not recognized as a similar expression. To explain this phenomenon we have to go back to MathML equation structure. As you can see from Figure 5 (left side), two compared equations (both belonging to the class 1 of our corpus) have rather similar structure (at least, from the human point of view). However, their tree roots have different number of child nodes, hence their production rules are (formally) different. It means that we have to enhance equation normalization factor (currently limited by only variable names and numerical values): in the above mentioned case the issue can be resolved by restructuring a tree based equation representation as Figure 5 (right side) shows: both trees in the right side are semantically equivalent to those from the left side. However, similarity score increases from 0 (in the "left" case) to 0.44 (in the "right" case).

4. CONCLUSION

In our study of mathematical equation similarity patterns we adopted a tree overlapping algorithm (used originally in NLP) for mathematical equation syntactical similarity. After arranging a set of experiments, we discovered that our modification fits well a selection of equations from college-level teaching practice. We examined some drawbacks and argued that in order to improve precision the further steps towards equation normalization are required.

5. REFERENCES

[1] R. Bod. Beyond grammar. *An Experienced-Based Theory of Language. CSLI Lecture Notes*, 88, 1998.

[2] I. Hiroshi, H. Keita, H. Taiichi, and T. Takenobu. Efficient sentence retrieval based on syntactic structure. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 399–406. Association for Computational Linguistics, 2006.

[3] S. Kamali and F. W. Tompa. Improving mathematics retrieval. *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*, pages 37–48, 2009.

[4] K. Sain, A. Dasgupta, and U. Garain. Emers: a tree matching–based performance evaluation of mathematical expression recognition systems. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(1):75–85, 2011.

[5] K. Yokoi and A. Aizawa. An approach to similarity search for mathematical expressions using mathml. *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*, pages 27–35, 2009.

[6] Q. Zhang and A. Youssef. *An Approach to Math-Similarity Search*, pages 404–418. Springer International Publishing, Cham, 2014.