Using binominal theorem for expression tree comparing in Intelligent Tutoring Systems

Andrey Chukhray, David Dvinskykh, Vitaliy Narozhnyy, Iryna Trofymova and Tetiana Stoliarenko

National Aerospace University «Kharkiv Aviation Institute», Chkalova 17, 61070 Kharkiv, Ukraine

Abstract

In this paper, a new operation for use in a hybrid expression tree was added. The activation functions for the original and modified tree when applying binomial theorem were described. The possibilities of detecting possible errors by decomposing the expression of the original tree using binomial theorem were extended.

Keywords

Expression tree, binomial theorem, comparing expression tree

1. Introduction

In recent years, there has been a significant growth in interactive learning due to the continuous development of computer capabilities. These include increased processor power, increased memory capacity, increased input-output capabilities, improved networking technologies, and increased efficiency in the development of modern software tools. The surge in machine learning was triggered by the Covid-19 pandemic [4].

Computer-based mathematics learning programmes (CBMPs) as a means of transferring knowledge and skills from teachers to students are in great demand in the modern world. As interest in these programmes grows, so do the expectations placed on them. Over time, CBMPs have evolved from simple testing software and e-textbooks to complex Intelligent Tutoring Systems. These systems are capable of adapting the learning process by adjusting the parameters depending on the individual learner's characteristics [4].

For such systems, it is necessary to have the ability to compare arithmetic expressions in order to determine whether the student has made the correct expression and to determine what errors the student has made. The previous article described a mechanism for comparing arithmetic expressions using non-binary expression trees. It described an algorithm for tree comparison, but there were few possibilities to analyse matched and non-matched nodes. The system allowed to detect missed minus signs and rounding errors, and the available operations were limited to addition, subtraction, multiplication, and division. [1]

In equivariant transformations of expressions, one may encounter a concept called binomial theorem. Particular cases of this theorem can be applied at the subconscious level, which complicates the task of comparing the original expression given by the teacher with the modified expression in the presence of an error, which was introduced by the student.

ProfIT AI 2023: 3rd International Workshop of IT-professionals on Artificial Intelligence (ProfIT AI 2023), November 20–22, 2023, Waterloo, Canada

a.chukhray@khai.edu (A. Chukhray); d.dvinskykh@khai.edu (D. Dvinskykh); v.narozhnyy@khai.edu (V. Narozhnyy); i.trofymova@khai.edu (I. Trofymova); t_stol@ukr.net (T. Stoliarenko);
0000-0002-8075-3664 (A. Chukhray); 0000-0002-5094-4136 (D. Dvinskykh); 0000-0003-1237-8118 (V. Narozhnyy); 0000-0002-1537-5601 (I. Trofymova); 0000-0002-7202-316X (T. Stoliarenko);

The binomial theorem is a result of expanding the non-negative powers of binomials or sums of two terms, where the coefficients of the terms in the expansion are the binomial coefficients. [2]

The most common case of the binomial theorem is (1), where binomial coefficients as (2) [3].

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} = \binom{n}{0} a^n + \binom{n}{1} a^{n-1} b + \cdots + \binom{n}{k} a^{n-k} b^k + \cdots + \binom{n}{n} b^n, \quad (1)$$

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}. \quad (2)$$

These formulas contain an additional mathematical "exponentiation" operation in comparison with the operations described in the previous article, which determines the need to modernise the non-binary expression tree.

The aim of this paper is to develop a method for finding the applied binomial theorem in the original and modified expression tree.

Problems to be solved:

1. Add the possibility of storing the exponentiation operation to a certain degree;
2. Determine the model for searching the presence of transformation by binomial theorem in the modified tree in comparison with the original expression tree.

2. Hybrid Expression Tree

2.1. Definition

A non-binary expression tree contains incoming and outgoing nodes, indicating parent and child nodes respectively. The child nodes had the same priority and could be shuffled at will among themselves.

The exponentiation operation has two operands, which are called base and exponent. Exponentiation does not have the commutativity property, which makes these operands not equivalent and prevents this operation from fitting into the previous expression tree model.

To fit this node denoting exponentiation, we need to define it as a unary operation where the only child node (in the previous non-binary expression tree definition) is the base of the degree. An additional service node is added to denote the exponent of degree. The additional service node is the root of a separate tree, which allows not only natural numbers but also integer expressions to be written in the future. Since the tree contains not only ordinary child nodes, but also service nodes, this tree will be called a hybrid expression tree in the future.

It is necessary not to confuse a hybrid tree with a tree that contains both binary [6] and non-binary nodes. The node of the exponentiation operation, although it resembles a binary node, but unlike the binary node it contains two unequal nodes, so this node is a special case of a hybrid tree node. In the future, when expanding the possible operations of the hybrid expression tree, the nodes will contain not only an unlimited number of child nodes, but also an unlimited number of service nodes.

2.2. Parsing expression into hybrid expression tree

The basic logic of parsing an algebraic expression divided into tokens has not changed. When a token with the value "^" is encountered, the subsequent token is written as an additional service node, and the current node is written as the main single child node.

Exponentiation has level 5 in node levels, and the final hierarchy looks like:

1. addition operation
2. subtraction operation
3. multiplication operation
4. division operation
5. exponentiation operation
6. numerical node

7. special node defining the order of operations (brackets)

Since the current article does not consider complex numbers, obtaining a negative number in the base will cause an error and impossibility in calculating the parent nodes of the expression tree. Therefore, the cases in which a negative number in the base is obtained will not be further considered in this article.

For the example we took expression (3), which in computer form has the form "1+(6+7)^(1+4)+5^4/(5+7)^3+3^(1+2)".

$$1 + (6 + 7)^{1+4} + \frac{5^4}{(6 + 7)^3} + 3^{1+2} \quad (3)$$

When transformed, the hybrid expression tree will look as shown in Figure 1.

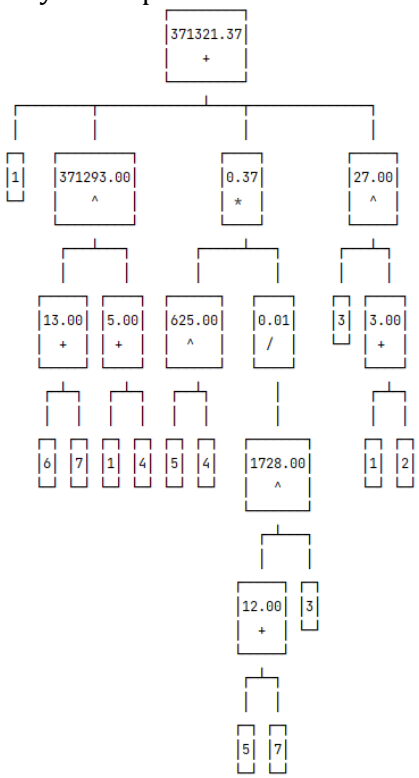

Figure 1: Hybrid Expression Tree for «1+(6+7)^(1+4)+5^4/(5+7)^3+3^(1+2)».

As can be seen in Figure 1, the base of the exponentiation operation is written as the left node, while the service node defining the exponent of the degree is written on the right.

3. Using Multiplication formulas for polynomials to compare hybrid expressions tree

The previous article described the logic for comparing nodes from the original tree to nodes in the modified tree. This logic will not be changed to use binomial theorem, but it will be extended to analyse redundant and missing nodes in the modified tree compared to the original tree.

The lists of missing nodes will be searched using the activation function behind formula (4), where $N_G^-(w)$ indicates the set of child nodes of "w" node; $d_G^-(x)$ is the number of child nodes of "w" node.

$$origin\_activate(w) = type(w) = pow \wedge type\left(N_G^-(w)\right) = sum \wedge d_G^-\left(N_G^-(w)\right) = 2 \quad (4)$$

If the activation function returns true, the original node is expanded using binomial theorem and the child nodes are compared with the redundant nodes from the modified tree using the "matching" function (5) [1]. Then all the found pairs are passed to the "sim" formula (6) [1] and summed up. The obtained sum is divided by the value obtained from the function "Sim_{max}" (7)

[1] in which the original node decomposed by binomial theorem was passed. The generalised formula is shown by (8).

$$\text{matching}\left(W,W^*\right)=\left\{(w,w^*)\in W\times\left\{W^*\left|\underset{w\in W,w^*\in W^*}{\arg\max}\left\{\text{Sim}\left(w,w^*\right)\right\}\right.\right\}\left|\text{Sim}\left(w,w^*\right)<>0\right.\right\} \tag{5}$$

$$\text{Sim}\left(G(x),G^*\left(x^*\right)\right)=\begin{cases}\begin{cases}2,\text{value}\left(G\left(x\right)\right)=\text{value}\left(G^*\left(x^*\right)\right);\\0\end{cases},\text{type}\left(x\right)=\text{value};\\\begin{cases}\text{Sim}_{\max}\left(G\left(x\right)\right),\text{value}\left(G\left(x\right)\right)=\text{value}\left(G^*\left(x^*\right)\right);\\0\end{cases},\text{type}\left(x^*\right)=\text{value};\\1+\sum_{\left(x',x'^*\right)\in\text{matching}\left(N_G^-(x),N_G^-\left(x^*\right)\right)}\text{Sim}\left(G\left(x'\right),G\left(x'^*\right)\right),\begin{matrix}\text{type}\left(x\right)=\text{operation}\wedge\\\wedge\text{type}\left(x^*\right)=\text{operation}\end{matrix};\\0\end{cases}; \tag{6}$$

$$x\in V\left(G\right);d_G^-\left(x\right)=\left|N_G^-\left(x\right)\right|.$$

$$\text{Sim}_{\max}\left(G\left(x\right)\right)=\sum_{x'\in N_G^-(x)}\begin{cases}1+\text{Sim}_{\max}\left(G\left(x'\right)\right),d_G^-\left(x'\right)>0\\2,d_G^-\left(x'\right)=0\end{cases} \tag{7}$$

$$\text{modified\_activate}\left(G\left(w\right),W^*\right)=\frac{\sum_{\left(x',x'^*\right)\in\text{matching}\left(G(w),W^*\right)}\text{Sim}\left(G\left(x'\right),G\left(x'^*\right)\right)}{\text{Sim}_{\max}\left(G\left(w\right)\right)}>0.5 \tag{8}$$

If both functions return true, the comparison assumes that the student has decomposed the expression by binomial theorem, which gives an opportunity to compare the nodes of the just decomposed expression in the original tree and compare them with the extra nodes in the modified tree. As a result, it returns a map [5] of correspondences between nodes to be added, corrected, replaced, or deleted and textual support on how this should be done.

In the previous article [1], the arithmetic expression "1*(-2)+3*4/-5+5*(3+7)" as the original expression and "1*2-3*4/5+6*(1+3)" as the modified expression were considered as an example. For this article, the original and modified expression will be changed by adding "(265+456)^3" and "265^3+3*265^2*456+265*456^2+456^3". As a result, the original expression will look like "1*(-2)+3*4/-5+5*(3+7)+(265+456)^3" (Figure 2).

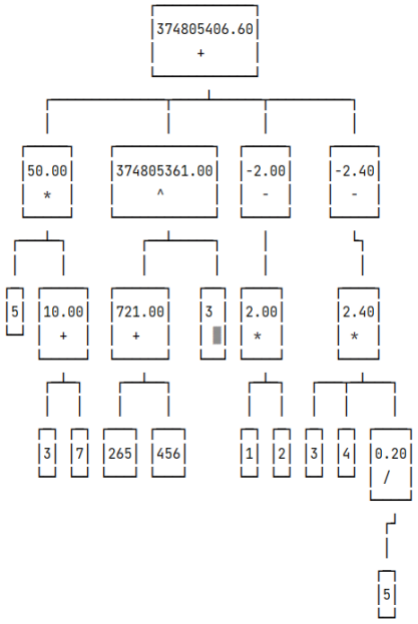

Figure 2: Hybrid Expression Tree for 1*(-2)+3*4/-5+5*(3+7)+(265+456)^3".

The modified expression will look like "1*2-3*4/5+6*(1+3)+265^3+3*265^2*456+ +265*456^2+456^3" (Figure 3).

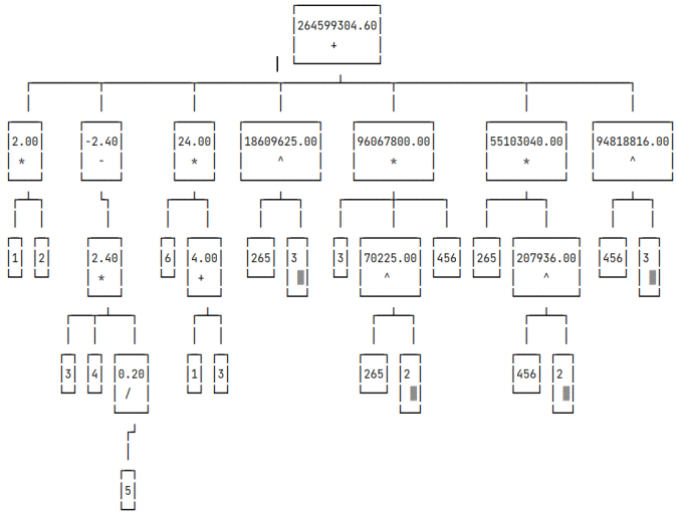

Figure 3: Hybrid Expression Tree for "1*2-3*4/5+6*(1+3)+265^3+3*265^2*456+265*456^2+ +456^3".

During comparing original tree with modified tree, the original activate function and modified activate function returned true value (Figure 4), so binominal theorem was applied to original tree. (Figure 5).

```
original_activated = true
modified_activated = 123/171 >0.5 = true
Matched nodes - '456^3' and '456^3'
Matched nodes - '3*265^2*456' and '3*265^2*456'
Matched nodes - '265^3' and '265^3'
Matched nodes - '-(3*4/5)' and '-(3*4/5)'
Non-matched recorded - '5' and '6'. To fix - 'You should change '6' with '5''
Matched nodes - '3' and '3'
Non-matched recorded - '7' and '1'. To fix - 'You should change '1' with '7''
Matched nodes - '456^2' and '456^2'
Matched nodes - '265' and '265'
Non-matched recorded - '3' and 'null'. To fix - 'You should add additional operand [3] to 265*456^2'
Non-matched recorded - '-(1*2)' and '1*2'. To fix - 'You should add minus for '1*2' part'
```

Figure 4: Activation function results and the suggestions for transforming the modified tree to the original tree.

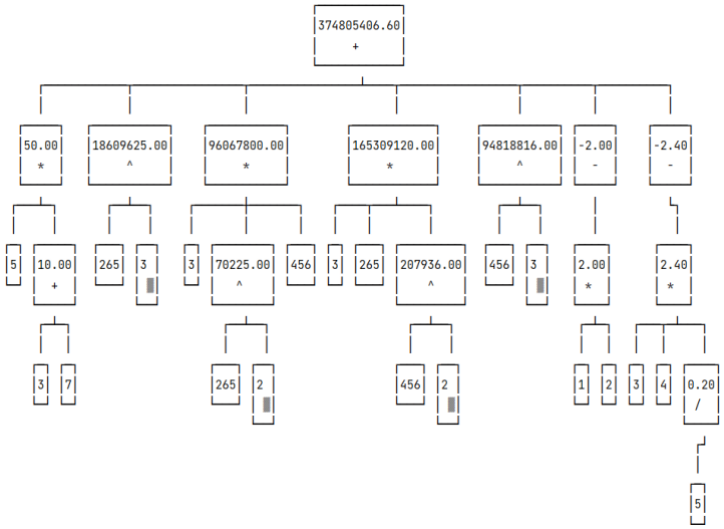

Figure 5: Original expression tree after applying binominal theorem.

The suggestion for transforming original expression tree to modified expression tree is shown in Figure 3. It shows that all nodes are matched except for:

- “6” and this should be replaced with “5”;
- “1” and this should be replaced with “7”;
- “265*456^2” and this non-matched node should be corrected by adding additional “3” operand to multiply operation;
- “1*2” which should be fixed by adding minus.

Conclusions

This system is unique and has no analogues to date, so it is impossible to provide a comparative characteristic with known analogues. In this paper, the existing non-binary tree model was improved by introducing a hybrid tree to be able to use exponentiation node.

When comparing hybrid trees, the comparison model was improved by using binomial theorem. Functions were also implemented to determine usage of binomial theorem in the modified expression tree in order to decompose the expression for further comparison. Without using the binomial theorem, the comparison model would show the error in the decomposition of a sum of two variables of non-negative integer degree, even if the error is elsewhere in the arithmetic expression.

Future articles will apply the expression tree to learn solving systems of linear equations using Cramer's method.

References

[1] A. Chukhray, D. Dvinskykh, V. Narozhnyy, and T. Stoliarenko, “Using an expression tree for adaptive learning,” in 2023 13th International Conference on Dependable Systems, Services and Technologies (DESSERT), 2023.

[2] P. Corn, H. Z. Vee, and K. Jaiswal, “Binomial Theorem,” Brilliant.org. [Online]. Available: https://brilliant.org/wiki/binomial-theorem-n-choose-k/.

[3] E. W. Weisstein, “Binomial Theorem.” [Online]. Available: https://mathworld.wolfram.com/BinomialTheorem.html.

[4] A. Chukhray and E. Yashina, Models and software for intelligent web-based testing system in mathematics, vol. 3003. CEUR-WS, 2021.

[5] Y. D. Liang, J. Bethely, and G. S. Walia, “Visualizing recursion using code and stack animation,” Journal of Computing Sciences in Colleges, vol. 37, no. 4, pp. 41–49, 2021.

[6] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, “Translating a math word problem to a expression tree,” in Learning 2.1 (2007). Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, 2018.