

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Structured Manipulation of Handwritten Mathematics

João Miguel Gomes Gonçalves



Master's in Informatics and Computing Engineering

Supervisor: Alexandra Sofia Ferreira Mendes

August 24, 2023

Structured Manipulation of Handwritten Mathematics

João Miguel Gomes Gonçalves

Master's in Informatics and Computing Engineering

Approved in oral examination by the committee:

President: José Carlos Medeiros de Campos

External Examiner: Pedro Henrique e Figueiredo Quaresma de Almeida

Supervisor: Alexandra Sofia Ferreira Mendes

August 24, 2023

Abstract

With the advancements in handwriting recognition, the structuring and manipulation of mathematical documents have become more feasible. This technology field has proven effective as it provides a more straightforward presentation and proof handling of the material in question. This project aims to provide a structure editor for handwritten mathematics that uses existing handwriting recognizers. Our focus is on the correct and straightforward structured manipulation of the handwritten content. We focus on the Calculational Method. The Calculational Method is a calculational style developed with the help of the revolutionary work of Dijkstra and Gries, which intensifies the use of mathematical calculations in the design of algorithms. This calculational style is used in this project, supporting both arithmetic and propositional logic as the two main ways of structuring mathematical notations. It allows us to apply correctly multiple algebraic rules and manipulate their mathematical structure, avoiding mistakes in the process. This means that, by the end of this project, there will be a software tool capable of presenting mathematical notations more transparently by allowing the user to write and manipulate these mathematical notations with proper proof handling.

Even though there are some limitations, the results of this work are encouraging. Preliminary results show that a tool such as the one provided has the potential to facilitate the comprehension, interpretation, and teaching of calculational mathematics. It also has the potential to support research projects that aim to create tools to improve the reliability of software systems.

Keywords:

Calculational Method, arithmetic and propositional logic, online handwriting recognition, structured manipulation

ACM Classification:

Human-centered computing → Human computer interaction (HCI) → Interactive systems and tools

Human-centered computing → Human computer interaction (HCI) → Interaction devices

Acknowledgements

First of all, I would like to thank Professor Alexandra Mendes for the supervision provided throughout the development of this thesis. Your help was invaluable to me, and without it, this project would not be possible.

I would also like to thank all my friends who have been there in the best and the worst moments, with whom I have shared many drinks and shared countless memories. I will not name them all as I am sure they all know who they are. May these memories last forever.

Another heartfelt "Thank You" goes to every teacher that helped turn me into the man I am today. Your influence was very important, and I will never forget you.

Last but definitely not least, a special "Thank You" goes to all my family and especially my parents Armanda and Carlos and my sister Inês, who have all always supported me and my dreams and have always been there for me on every occasion. I love you very much!

João Gonçalves

“We all get distracted, the question is, would you bounce back or bounce backwards?”

Kendrick Lamar

Contents

1	Introduction	1
1.1	Calculational Method	1
1.2	Document Structure	2
2	Overview of existing hardware and software	3
2.1	Pen/Touch-based technologies	3
2.2	Handwriting recognition and manipulation software	5
2.2.1	Infty Editor	5
2.2.2	MathPaper	5
2.2.3	AlgoSketch	6
2.2.4	Math speak & write	6
2.2.5	MathBrush	6
2.2.6	MathPad ²	6
2.2.7	MathSpad Tablet	7
2.2.8	GraspableMath	7
2.2.9	Seshat	7
2.2.10	SolveIt	8
2.2.11	OneNote	8
2.2.12	Mathcha	8
2.3	Whiteboards	9
2.3.1	Xournal++	9
2.3.2	cracker0dks/whiteboard	9
3	Functional Design	10
3.1	Supported Notations	11
3.2	Structured Editing	11
3.3	Editing Operations	12
3.3.1	Copy Operation	12
3.3.2	Manipulation Operations	13
3.4	Gesture Support	13
3.5	Tool or User in Control	14
4	Technical Design	16
4.1	Architecture	16
4.2	Input Parsing	16
4.3	Data Structures	18
4.4	Structured Trees	20
4.4.1	Rose Tree Node	21

4.4.2	Construction of Structured Trees	23
4.5	Structured Modification	28
4.5.1	Data Highlighting and Duplication	28
4.5.2	Expression Manipulation Rules	29
4.6	Gestures Implementation	32
5	Validation and Testing	34
5.1	Goals	34
5.2	Experimental Parameters	35
5.3	Case Study Tasks	35
5.4	Results	36
5.4.1	Are the Tool's Functionalities Usable?	36
5.4.2	Is the Tool Straightforward?	37
5.4.3	How Complex is the Tool Versus Ordinary Methods?	37
5.5	Usability	39
5.6	Threats to Validity	41
5.6.1	Selection Bias	41
5.6.2	Confirmation Bias	41
5.7	Conclusions	41
6	Discussion and Conclusions	43
6.1	Future Work	44
	References	46
A	Case Study Guide	48
B	Case Study Form	53

List of Figures

1.1	Example of the Computational Method usage, where ∇ refers to the greatest common divisor.	2
3.1	Example of the distributivity rule being applied without the tool checking its validity	15
4.1	System architecture	17
4.2	Binary tree that represents the expression $a+b+c=d$	21
4.3	Rose tree that represents the expression $a+b+c=d$	22
4.4	First iteration of the construction of the rose tree referring to $a+b \times (c+d) = e$	25
4.5	Second iteration of the construction of the rose tree referring to $a+b \times (c+d) = e$	26
4.6	Third iteration of the construction of the rose tree referring to $a+b \times (c+d) = e$	27
4.7	Fourth iteration of the construction of the rose tree referring to $a+b \times (c+d) = e$	27
4.8	Check gesture associated with the copy of selected sub-expressions	30
4.9	Symmetry	30
4.10	Circle gesture applied on the “+” operator $a+b=c$	31
4.11	Rose tree modification upon applying symmetry on the “=” operator of the expression $a+b=c$	31
4.12	Example of the distributivity rule being applied to the expression $a \times (b+c)$	32
5.1	“I had to clear the whiteboard multiple times in order to complete the task (Task 1).” 1 - (Strongly disagree); 5 - (Strongly agree)	38
5.2	“I had to clear the whiteboard multiple times in order to complete the task (Task 2).” 1 - (Strongly disagree); 5 - (Strongly agree)	38
5.3	“How do you compare the complexity of completing this task (Task 1) by writing all of the steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool?”	39
5.4	“How do you compare the complexity of completing this task (Task 2) by writing all of the steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool?”	40
5.5	“Completing this task (Task 1) using a tablet PC would be...” 1 - (A lot harder); 5 - (A lot easier)	42
5.6	“Completing this task (Task 2) using a tablet PC would be...” 1 - (A lot harder); 5 - (A lot easier)	42

Chapter 1

Introduction

The art of writing, which originated around 3400 B.C. near the Persian Gulf, has a rich history spanning 5400 years. Over time, it has transformed from a sluggish, incoherent form of expression to a more accessible, faster, and expansive approach. Writing has become indispensable for a wide range of everyday tasks, both simple and complex. However, despite its significance, the evolution of writing seems to have reached a standstill. By transitioning to digital platforms, we can unlock further optimization possibilities that positively impact numerous tasks.

Writing is the foundation for various aspects of human life, with education being one of its primary applications. Within the classroom, one of the most crucial activities is the creation of written materials for lectures. The comprehension and understanding of students directly depend on how effectively teachers present and explain the information. Although many schools have adopted digital methods, teaching effectiveness is set back by the reliance on manual writing for all instructional materials. The use of traditional blackboards presents additional challenges, such as the need to optimize limited space, the manual erasing of previous content to make room for new material, and the inability to make real-time structural changes without rewriting the entire content. This issue is particularly prominent in mathematics classes, where the advancements in digital tools can greatly enhance the accessibility and interactivity of the subject matter.

This project aims to develop a structure editor specifically designed for handwritten mathematics. This tool will benefit mathematics teachers during their lectures and a broader range of individuals seeking to optimize their work, including researchers. Users can use this editor to engage in more interactive mathematical writing and manipulation. The tool enables users to apply algebraic rules, manipulate expression structures accurately, and focus on the Computational Method [2], ultimately enhancing the understanding and presentation of mathematical concepts.

1.1 Computational Method

The Computational Method quickly became the primary problem-solving method in the mathematical sciences. It is a powerful tool for solving complex equations and is based on the idea that mathematics can be taught through a systematic approach to problem-solving. This method relies

$$\begin{aligned}
& m \nabla n \\
= & \quad \{ \quad p \nabla n = 1 \text{ and } 1 \text{ is the unit} \\
& \quad \text{of multiplication} \quad \} \\
& m \times (p \nabla n) \nabla n \\
= & \quad \{ \quad \text{distributivity and associativity} \quad \} \\
& (m \times p) \nabla (m \times n) \nabla n \\
= & \quad \{ \quad (m \times n) \nabla n = n \quad \} \\
& (m \times p) \nabla n .
\end{aligned}$$

Figure 1.1: Example of the Calculational Method usage, where ∇ refers to the greatest common divisor.

on symbols, equations, and rules to calculate the solution to a problem, and it simplifies solving problems by breaking them down into smaller parts and then applying the appropriate rules and equations to each part. The Calculational Method is widely used in all areas of mathematics, from introductory algebra to advanced calculus, and it has revolutionized the way mathematical problems are solved, being considered an essential tool for mathematicians and scientists.

The Calculational Method usually has each step accompanied by a hint justifying that step's validity. Figure 1.1 demonstrates the usage of the Calculational Method. In this example, we wish to prove that $m \nabla n$ is equal to $(m \times p) \nabla n$ while knowing that $p \nabla n = 1$. Each step has a hint associated with it to understand it better, and the proof can be achieved through a series of hints and steps.

The advantages of this method are the fact that the user can immediately conclude that $m \nabla n$ is equal to $(m \times p) \nabla n$ just by looking at the first and last step and the fact that it forces the writing of explanations of each step.

1.2 Document Structure

This thesis is divided into five additional chapters. Chapter 2 will cover the state of the art and the current developments in the area. In chapter 3, the functional design is discussed. Chapter 4 will provide this project's technical design and implementation. Chapter 5 refers to the tests performed to validate the tool's functionalities, and chapter 6 provides the discussion and conclusions of the tool, as well as possible future work.

Chapter 2

Overview of existing hardware and software

Over the years, there have been many advancements in handwriting recognition. These developments have focused on addressing the unique challenges of recognizing and processing mathematical expressions and documents.

A notable trend in this domain is the continuous improvement in hardware and software designed to recognise and interpret handwritten mathematics, and these advancements have enabled a wider range of individuals to engage with these technologies as they have become more affordable and available. As a result, more people are gaining exposure to the benefits and capabilities of these innovative tools.

It is worth noting that numerous existing tools have emerged to provide a way to recognize and manipulate handwritten mathematical content, and these tools use various pen and touch-based technologies to accurately interpret mathematical expressions' intricate symbols and structures. Additionally, some of these tools go beyond recognition and enable users to manipulate and edit handwritten mathematics, providing a comprehensive solution for working with mathematical content in its original handwritten form.

In this chapter, we will explore the diverse array of pen and touch-based technologies that have been developed thus far, and we will examine the range of available tools that specifically support handwritten mathematics recognition and manipulation. By examining these existing solutions, we can gain a deeper understanding of the current landscape and the possibilities in handwriting recognition and mathematical expression manipulation.

2.1 Pen/Touch-based technologies

The development of pen and touch-based technologies has been remarkable over the past few decades, and the ability to interact with digital devices and software through these technologies has made them indispensable in various applications. From transforming how we interact with computers to providing a more natural and intuitive interface for digital art and media to enabling

the development of innovative new applications and products, the impact of pen and touch-based technologies has been massive.

One of the earliest examples of pen-based technology was the Apple Newton, a personal digital assistant (PDA) released in 1993. The Newton featured handwriting recognition capabilities, allowing users to enter data by writing on its touchscreen with a stylus. While the Newton was met with low sales and several criticisms, its technical achievements laid the groundwork for future generations of pen-based technology.

In the early 2000s, the popularity of tablet PCs began to take hold. These computers featured larger, more responsive screens, making them ideal for pen-based input. Tablet PCs also incorporated handwriting recognition technology, which allowed users to write directly onto the screen rather than using a keyboard or mouse.

In the years since, pen and touch-based technologies have been integrated into various devices, from smartphones and tablets to laptops and even desktop computers. This has enabled a much more natural and intuitive way to interact with digital devices and software. For instance, Apple's iPad and Microsoft's Surface Pro series feature pressure-sensitive styluses, allowing users to draw and paint more accurately than ever.

The development of pen and touch-based technologies has also had a profound impact on the world of gaming. From the Nintendo DS and Wii U to the PlayStation Vita and Xbox One, many gaming consoles now feature touchscreens that allow users to interact with games in various unique and innovative ways. In particular, the rise of mobile gaming has been enabled by the development of pen and touch-based technologies.

Finally, pen and touch-based technologies have enabled the development of a wide range of innovative new applications and products. For instance, the Microsoft Surface Hub is an 84-inch touchscreen display that can be used for meetings and collaboration. The Surface Hub's interactive whiteboard capabilities allow multiple users to contribute to projects and presentations, while its pen and touch-based interface makes it easy to navigate and use.

The development of pen and touch-based technologies over the past few decades has tremendously impacted how we interact with digital devices and software. The impact of these technologies has been profound, from enabling the development of innovative new applications and products to transforming the way we play and create with digital media.

The versatility of pen and touch-based technologies has also allowed for the development of a wide range of applications in the medical, educational and industrial fields. For example, doctors can use tablet PCs to access medical records, while educators can use interactive whiteboards to create an engaging learning environment. In the industrial sector, pen-based technologies have been used to create innovative tools for design and engineering. Furthermore, pen and touch-based technologies have been instrumental in enabling the development of Virtual Reality (VR) and Augmented Reality (AR) applications, furthering the possibilities of how people interact with digital content.

Pen and touch-based technologies have also been instrumental in developing handwriting and sketch recognition software. This software can interpret written or drawn input and convert it into

machine-readable data, allowing a more natural and intuitive way to interact with digital devices and software. Additionally, these technologies have been used to create digital inking applications, which allow users to draw and make notes directly onto documents and images. This has enabled a much more natural and efficient way to record and share ideas, drawings and sketches.

2.2 Handwriting recognition and manipulation software

In this section, we will introduce a range of innovative tools that have been developed to facilitate the recognition and, in certain cases, manipulation of mathematical expressions and documents based on user preferences. Each of these tools offers unique features and functionalities, which we will explore in detail.

The primary purpose of these tools is to provide users with a seamless and efficient way to input handwritten or digitized mathematical content, enabling the software to recognize and interpret the symbols, equations, and structures accurately. This recognition process is crucial for transforming handwritten or digitized content into a digital format that can be further processed, edited, and shared.

Furthermore, some of these tools go beyond mere recognition and offer additional capabilities that allow users to manipulate the recognized mathematical expressions per their requirements. This means that users can actively interact with the mathematical content, making modifications, reordering equations, or applying mathematical transformations directly within the tool's interface. Such manipulation features empower users to engage more dynamically with their mathematical work and enhance their overall productivity.

To provide a comprehensive overview, a detailed rundown of each of the tools is presented. By examining the specific functionalities, user interfaces, and benefits of these tools, we will gain a deeper understanding of their capabilities and how they can support mathematical workflows.

2.2.1 Infty Editor

The Infty Editor [11] is a software program designed for scientific and technical document editing due to its ability to edit mathematical expressions, scientific formulas and equations. It integrates Optical Character Recognition (OCR) to convert images of printed texts into a digital format. One of this tool's main features is its online recognition of handwritten mathematical formulas. When a letter, number or operator is written, it is rewritten instantly in a proper position and with an appropriate size relative to the rest of the expression. This tool produces output in multiple formats such as \LaTeX , MathML and PDF.

2.2.2 MathPaper

MathPaper [14] is a mathematical sketching tool for pen-based entry and editing of mathematics that supports the recognition of mathematical expressions and sketched algorithmic pseudo-code.

It implements an environment where multiple mathematical expressions and algorithms can be written anywhere. It also supports modification with the use of gestures and widgets.

2.2.3 AlgoSketch

AlgoSketch [8] is a pen-based sketching software with supporting interactive computation. It allows users to write and edit handwritten mathematical expressions with descriptions in pseudo-code, just like MathPaper. This tool provides a graphical interface for users to interact with algorithms, allowing them to create and visualize step-by-step instructions for solving a specific problem. It also provides a debugging feature that allows users to identify and fix errors. AlgoSketch is especially effective as a learning tool, allowing students to understand algorithms better and further helping them improve their programming skills.

2.2.4 Math speak & write

Math Speak & Write [4] is a software application designed to help students with learning disabilities. The software is designed to help students learn basic math concepts in an accessible, interactive way. This is a technology developed at the University of California, and it combines handwriting and voice recognition for inputting mathematical expressions. The system uses a combination of algorithms to recognize characters and symbols written in a standard format and interprets them into an appropriate mathematical expression. It can identify single and multi-line equations and can be used to generate expressions in \LaTeX , MathML and other types of notations. The user can use the voice recognition feature to speak a mathematical expression and have it converted to text.

2.2.5 MathBrush

MathBrush [6] is a handwriting recognition and manipulation system for mathematical expressions developed by researchers at the University of Waterloo. It is able to recognize handwritten mathematical expressions from paper, tablets and other digital devices. MathBrush also allows gestures, such as scratch-out and the use of the back of the pen for erasing, to manipulate these expressions. Another form of manipulation of mathematical expressions supported by this system is the fact that the user is able to select input ink and move it around. This means that the user is able to move expressions around easily, something that cannot be done by using traditional methods such as a blackboard. This software tool also supports problem solving by sending the equation or expression the user wishes to solve to a computer algebra system (Maple and Mathematica) and interpreting the results provided by it.

2.2.6 MathPad²

MathPad² [7] is a pen-based mathematical sketching with gestural interaction for mathematics problem solving. It consists of an interactive graphical interface, which allows users to manipulate

objects on the screen to create equations and diagrams. This software includes a variety of tools for creating and manipulating equations, as well as a library of pre-made equations and diagrams, which can be used as a starting point for exploration.

2.2.7 MathSpad Tablet

MathSpad Tablet's [10] [9] goal is to make the presentation of materials more accessible and interactive by merging the benefits of pen-based technologies with the improvements in the computational power of today's computers. MathSpad Tablet allows the user to interact with the digital content displayed on the tablet's screen in a similar way to how one would interact with a blackboard, using a pen as a pointer. It recognizes handwritten mathematical equations and formulas and structures them so the user can easily manipulate them with simple gestures. These gestures apply manipulation rules and can be changed by the user at any time. For example, one of these gestures is the "semi-circle" gesture, and it applies the distributivity rule. Furthermore, the user can tap on an operator to select it and double-tap it to select the operator and its arguments, which the user can then copy with a gesture. Animations are also supported by MathSpad Tablet in order to demonstrate the use of these rules more clearly. MathSpad Tablet also allows for the redefinition of binary operators as well as the definition of new ones. The primary mathematical method supported by this tool is the Computational Method [2], and, following its principles, this tool has the knowledge to apply rules. However, it does not check if the rules applied are semantically correct. That's the user's responsibility to review.

2.2.8 GraspableMath

GraspableMath [12] is an online math tutoring program that offers interactive learning and problem-solving tools to help students develop their math skills. It includes step-by-step instructions, video tutorials, and practice exercises to help students understand and apply math concepts. GraspableMath is designed to be accessible to students of all ages and ability levels. GraspableMath offers a unique feature that allows students to manipulate mathematical expressions. This feature allows students to interact with and change variables in equations in order to explore the effects of different values on the equation's result. This feature helps students better understand the relationships between different variables, as well as how equations can be manipulated to solve problems. It also provides a built-in calculator that can be used to check the accuracy of students' answers. Even though this tool is capable of manipulating various mathematical expressions according to basic arithmetics and logical rules, it does not offer a handwriting recognizer. All expressions that the user wishes to manipulate must be inputted via a keyboard, and any special symbols must be selected, with the help of the mouse, from a selection of symbols defined by GraspableMath.

2.2.9 Seshat

Seshat [1] is a software program designed to recognize mathematical expressions using probabilistic grammars. It uses various techniques, including tokenization, context-free grammar trees,

and a naïve Bayesian network. It is a web-based application that can recognize handwritten and printed mathematical expressions and symbols. By having a sequence of strokes, Seshat is able to convert it to \LaTeX and other formats like MathML or InkML. This makes this software tool an online and offline handwriting recognizer. Additionally, Seshat can recognize multiple expressions at once, and it can generate multiple versions of the same expression. Seshat also includes an API and a graphical user interface, which allow users to interact with the software, and it can also create mathematical documents and store them in a database.

2.2.10 SolveIt

SolveIt [5] is a mobile handwritten mathematics recognizer which covers useful features like graph plotting, equation solving and arithmetic computation from images. It can process a picture of a handwritten formula or equation with the mobile device's camera. It then forms the corresponding string and parses it using an algebraic computer system, displaying all possible solutions. This is a user-friendly software tool which presents a machine-learning approach to symbol recognition. This application functions as a calculator and is able to handle equations of any degree.

2.2.11 OneNote

OneNote¹ is a digital notebook application from Microsoft that allows users to take notes, store information, create to-do lists, and more. It is available for free across multiple platforms and devices, including Windows, Mac, iOS, and Android. It can be used to organize notes, research, and ideas and shared with others for collaboration. OneNote's handwriting recognition allows the user to write on the page using a stylus or a finger and have it automatically converted to typed text. This makes taking notes and organizing information much easier and more efficient. Handwriting recognition also works with digital ink, so if the user has a tablet or touchscreen device, they can write directly on the page with a digital pen. OneNote can also recognize shapes, so the user can draw shapes and arrows and have them recognized as objects. OneNote has a Math feature that allows users to write or type mathematical expressions and equations and then manipulate them by selecting parts of an equation and changing them. Math Assistant is a feature supported by this software which allows the user to solve equations and use digital inking to draw graphs.

2.2.12 Mathcha

Mathcha² is an online math editor that serves as a powerful tool for creating and editing mathematical content. It offers a user-friendly interface that makes it incredibly easy for users to write and manipulate mathematical expressions according to their needs. This platform provides two convenient methods for inputting equations. The first option is through a \LaTeX -based syntax, allowing users familiar with \LaTeX , to seamlessly express their mathematical ideas. The second

¹<https://www.onenote.com/ink>

²<https://www.mathcha.io/>

option is Mathcha's visual equation editor, which empowers users to effortlessly create complex mathematical expressions with ease.

2.3 Whiteboards

In this section, we will shed light on open-source whiteboards, which offer incredible flexibility and the ability to extend their functionalities. A whiteboard is a digital application or program that simulates the functionality of a physical whiteboard. It allows users to create, draw, write, and collaborate on a virtual canvas. These innovative platforms not only serve as collaborative spaces for sharing ideas and information but also possess the unique capability to accommodate character recognition and structure mathematical content accurately. With their open-source nature, these whiteboards also foster a collaborative and customizable environment, allowing users to enhance their functionalities and tailor them to their specific needs. By harnessing the power of these open-source whiteboards, individuals and teams can explore new ways of interactive and dynamic collaboration, revolutionizing the way mathematical content is created and manipulated.

2.3.1 Xournal++

Xournal++³ is an open-source note-taking, sketching and document annotation application developed in C++. It is the successor of Xournal, a popular Linux-based note-taking application, and is designed to be a more robust and user-friendly alternative. Xournal++ is available for Windows, Mac OS X, and Linux and can be used for various tasks such as taking notes in meetings, sketching ideas, creating diagrams, annotating PDFs, and more.

This whiteboard can be extended in order for it to accommodate the structured manipulation of handwritten mathematics by receiving inputs in the form of symbols associated with its strokes and displaying them on the whiteboard.

2.3.2 cracker0dks/whiteboard

This particular whiteboard⁴, similar to Xournal++, is a remarkable open-source platform built on the NodeJS framework, providing users with extensive customization options. Like its counterpart, Xournal++, this whiteboard possesses the capability to be expanded to cater to the input of strokes corresponding to mathematical symbols. These strokes are then intelligently structured to faithfully represent the intricate composition of the mathematical expression. This meticulous structuring ensures that the mathematical content is accurately depicted, facilitating precise editing and manipulation. Its open-source nature allows for further enhancements and customization, enabling users to tailor the platform to their specific requirements. With its robust features and adaptability, this whiteboard emerges as an exceptional tool for the creation and manipulation of mathematical content.

³<https://xournalpp.github.io/>

⁴<https://github.com/cracker0dks/whiteboard>

Chapter 3

Functional Design

In this chapter, we will delve into the comprehensive functional design of the tool that has been developed, shedding light on the various decisions that were carefully considered and implemented throughout its development process. The primary objective of the tool developed lies in enabling the structured manipulation of handwritten mathematics, and it is this goal that serves as the guiding principle for all the decisions made.

Seeing as this project's main objective is not the recognition of handwritten mathematics, a handwriting recognizer would have to be extended for it to be able to handle these expressions' manipulation. Ideally, this recognizer would output the strokes associated with the original expressions. This expression could then be manipulated, and its internal structure updated accordingly.

However, none of the recognizer tools presented in 2.2 is open-source, so they cannot be extended as described previously. In order to solve this problem, instead of extending an existing handwriting recognizer, an existing whiteboard would have to be extended.

The chosen whiteboard for this purpose is the *cracker0dks/whiteboard* as described in 2.3.2 as it presents a more user-friendly interface and is more easily customizable than Xournal++, which was the second option as is presented in 2.3.1.

Prior to initiating any actual development work, a set of fundamental decisions were established. These decisions, which play an important role in shaping the tool's capabilities and user experience, are as follows:

- **Exclusive Focus on Handwritten Expressions:** In order to simulate the environment of a classroom or a research context, it was determined that the tool should only interact with handwritten expressions. This choice ensures an authentic experience, allowing users to engage with mathematical equations and symbols in their original handwritten form. By accommodating handwritten expressions, the tool captures the essence of traditional mathematical expression manipulation and preserve its pedagogical value.
- **Intuitive and User-Friendly Design:** Recognizing the significance of user experience,

an important consideration was to ensure that the tool remains highly intuitive and user-friendly. Complexity and technical intricacies were minimized to create an accessible interface. The aim is to allow users, regardless of their mathematical expertise, to navigate the tool effortlessly and perform desired operations with ease. By prioritizing simplicity and ease of use, the tool eliminates unnecessary complexity and promotes a smooth interaction between users and the mathematical content.

By adopting these decisions, a framework was established to guide the tool's development process. This chapter will delve into the functional design, exploring the intricate details and providing insights into the features and capabilities that support the structured manipulation of handwritten mathematics.

3.1 Supported Notations

The primary objective of this tool is to provide a consistent and efficient platform for editing mathematical expressions, with a particular focus on supporting simple arithmetic operations. By incorporating these essential functionalities, the tool allows users to manipulate mathematical expressions effortlessly and accurately.

To achieve this goal, the tool ensures support for fundamental arithmetic operations, including addition (“+”), subtraction (“-”), multiplication (“×”), and equality (“=”). These operators are the building blocks for basic mathematical calculations and are crucial for expressing mathematical relationships.

Furthermore, recognizing the significance of parentheses in mathematical notation, the tool should allow and encourage its use. Parentheses play an important role in grouping and prioritizing operations, enabling users to express specific hierarchies and clarify the intended order of mathematical operations. By supporting parentheses, the tool enhances the precision and clarity of mathematical expressions, ensuring accurate computation.

3.2 Structured Editing

Regarding editing approaches, the text model stands out as the most commonly used method, and it involves simply adding or removing text at a specific position within a given document. However, when dealing with mathematical expressions, relying on the text model may not be ideal or efficient. Mathematical notation poses unique challenges that can make using the text model difficult and vulnerable to errors. Therefore, for the purpose of our tool, the text model is deemed inappropriate.

To address these challenges and provide a better editing method, the tool adopts a structured editing approach. In structured editing, the focus shifts from manipulating plain text to altering the underlying structure of the expressions, and this approach offers several advantages when working with mathematical expressions, even though it comes with a few limitations.

One of the key benefits of structured editing is that it allows users to have explicit information about the structure of the expressions they are working with. By understanding and manipulating the expressions' structure, users can make various manipulations while obeying specific mathematical rules. This structured approach facilitates the application of algebraic rules.

Structured editing also empowers users to select sub-expressions within the original expression. This enhances the precision and flexibility of editing, as users can focus on specific portions of the expression and modify them accordingly.

While structured editing can be very advantageous, it is important to acknowledge its limitations. Users using this editing format need to be familiar with the structure accepted by the tool, and this requirement implies a learning curve where users must acquire knowledge about the supported structures and the rules of their manipulation. Additionally, the tool itself may impose certain limitations on the supported structures, restricting the scope of structural editing.

Nevertheless, the benefits of structured editing outweigh its drawbacks, especially when it comes to detecting errors in the expression's structure. With the use of structural information, the tool can perform automated checks and validations, and this functionality adds an extra layer of accuracy and reliability, ensuring that the edited expressions maintain their structural integrity.

While the text model falls short in handling mathematical expressions, the structured editing approach offers a more reliable solution. Despite the learning curve and limitations associated with structured editing, its advantages in applying algebraic rules, selecting sub-expressions, and detecting structural errors make it the preferred method for the tool. With structured editing, users can be provided with a powerful and intuitive editing experience specifically for mathematical expressions.

3.3 Editing Operations

3.3.1 Copy Operation

One of the most frequently performed operations in mathematical expression writing is copying previously written material from one step to another. This repetitive task can lead to multiple errors and inaccuracies in the final output, and recognizing the significance of this operation and the potential errors associated with it, our tool aims to provide users with an easy and intuitive alternative to manually duplicating the same content multiple times.

To facilitate the copying process and minimize the likelihood of errors, our tool should incorporate a dedicated copy operation. By offering this feature, users can effortlessly duplicate expressions, equations, or specific portions of content easily. This copy operation not only saves valuable time but also ensures consistency and accuracy throughout the writing process.

In addition to the copy operation, our tool should also feature a structured selection operation. This functionality would help users to select and copy sub-expressions from the original expression. The structured selection operation allows users to focus on specific components of the expression that require modification, eliminating the need to copy and retype the entire expression.

By facilitating the selection of sub-expressions, users can edit and manipulate their mathematical content more effectively and efficiently.

To ensure a better user experience, both the copy operation and the structured selection operation should be intuitive and user-friendly, and the goal would be to minimize the learning curve associated with these features. By prioritizing simplicity and ease of use, our tool aims to enhance productivity and accuracy when copying and selecting mathematical expressions.

3.3.2 Manipulation Operations

The tool aims to support the application of algebraic rules to the expressions written on the whiteboard, and by incorporating these rules, users can effectively manipulate and transform mathematical expressions. Two key rules stand out among the various algebraic rules: the *symmetry* of operators and the *distributivity* rule. These rules have proven very important in mathematical problem-solving scenarios, making them crucial to include in the tool's capabilities.

The first rule, the *symmetry* of operators, plays a significant role in algebraic transformations. It allows for the swapping of operands of certain operators, such as addition and multiplication, and this way, users can rearrange expressions to facilitate calculations or simplify complex equations by applying this rule. It is then a manipulation rule applicable by the user to swap the order of operands of an operator, even if it isn't mathematically equivalent. The tool is designed to support this rule, allowing users to manipulate expressions by taking advantage of the symmetry of operators.

The second rule, the *distributivity* rule, is another fundamental aspect of algebraic manipulation. This rule governs the distribution of operations across terms within an expression, and it enables users to expand expressions by distributing terms across parentheses. Once again, the user can apply this rule even if it isn't mathematically equivalent. With the *distributivity* rule, users can restructure expressions to facilitate further analysis. The tool should ensure full support for this rule, allowing users to apply it and obtain transformed expressions that obey the distributive property.

To ensure an immersive experience, the tool should support these algebraic rules and present them in a handwritten format. By preserving the handwriting of the mathematical expressions, the tool can replicate the classroom or research context, providing users with a familiar environment. This approach facilitates a more natural and intuitive interaction, enabling users to work with the expressions as they would using traditional methods.

3.4 Gesture Support

As was mentioned before, this project's goal is to offer an efficient and intuitive way to manipulate handwritten mathematics. To achieve this, a straightforward method to trigger actions must be developed. Although widely used, traditional button-based approaches may not always be the most intuitive or efficient means of initiating these actions as they often require users to navigate through menus or locate specific buttons, resulting in potential time-consuming searches.

To address this challenge, an alternative is introduced: the use of gestures as a means to trigger actions. Gestures, which encompass pen or mouse motions, provide a more intuitive and direct way for users to initiate commands without needing explicit button interactions. This way, users can easily and effortlessly execute desired editing actions.

This gesture-based approach offers several advantages over traditional button-based systems. Gestures eliminate the need for users to actively search for the appropriate button or menu option and this way users can rely on basic movements to perform the required gestures, resulting in a more instinctive and efficient interaction.

Gestures should be able to be executed using various input devices, such as pens or mice, accommodating different user preferences. Whether users are working with touchscreens, digital drawing tablets, or conventional computer mice, the tool can recognize and interpret the corresponding gestures accurately.

The primary focus of this tool is to ensure a quick, reliable, and user-friendly editing experience for handwritten mathematics. By employing gestures to trigger actions, the tool introduces a more intuitive and efficient approach to editing, and users can easily and effortlessly perform editing actions.

3.5 Tool or User in Control

Since this tool is primarily intended for classroom and research contexts, it is important to maintain a flexible approach without imposing any specific mathematical definitions. Instead, the tool should allow users to apply arithmetic rules according to their understanding. While the tool possesses a basic understanding of mathematical expressions, it does not perform rigorous checks to ensure the accurate application of these rules.

Therefore, it is the user's responsibility to apply the arithmetic rules correctly while manipulating and editing expressions within the tool. The tool does not impose constraints or limitations on the user's mathematical choices. If a rule appears applicable within the context of an expression, the tool will not intervene to verify its mathematical validity. Figure 3.1 is an example of the distributivity rule applied without the tool checking its validity. This allows users the freedom to explore and experiment with different mathematical operations and transformations without unnecessary restrictions.

By adopting this approach, the tool respects the user's autonomy and encourages independent thinking and problem-solving. Users have the flexibility to interpret and apply mathematical rules based on their own understanding and requirements. The tool serves as a supportive platform, facilitating the expression manipulation process without imposing predefined constraints or predefined notions of correctness.

It is essential to emphasize that the tool does not aim to replace or substitute the user's mathematical proficiency, as it is not an equation solver. Instead, it provides a user-friendly interface for expressing and editing mathematical content, placing the responsibility for accurate manipulation on the user's shoulders.

$$a + (b \times c)$$

$$(a + b) \times (a + c)$$

Figure 3.1: Example of the distributivity rule being applied without the tool checking its validity

Chapter 4

Technical Design

This chapter delves into the technical design of the tool, which can be accessed on GitHub at [SouOCalves/whiteboard](https://github.com/SouOCalves/whiteboard)¹, offering an in-depth exploration of its underlying architecture and implementation. Throughout the following sections, the key features that define the tool’s functionality will be highlighted, and insights into how they were implemented will be provided.

4.1 Architecture

The overall architecture of the system is visually represented in figure 4.1. At the heart of the system is a *user interface* that facilitates the input of handwritten mathematical expressions in the form of InkML files. This user interface serves as the primary way for users to interact with the system and perform various structure manipulations on the expressions.

When a user applies a structure manipulation to an expression, the system sends these manipulations to the structure updaters. The role of the *structure updater* is to process the manipulation instructions received from the *user interface* and accordingly update the structure of the mathematical expression. This process involves analyzing the existing structure, applying the requested manipulation, and generating an updated version of the expression.

Once the *structure updater* has completed its task, the updated mathematical expression is sent back to the *user interface* for displaying, and the user can then observe the modified expression, which reflects the applied structure manipulation. This feedback loop between the *structure updater* and the *user interface* enables an interactive editing experience, allowing users to observe the effects of their manipulations in real-time.

4.2 Input Parsing

Given that this tool assumes that the recognition of the mathematical expressions is done preemptively, its input consists of the parsing of InkML files. InkML stands for Ink Markup Language, an XML-based file format designed to represent digital ink data.

¹<https://github.com/SouOCalves/whiteboard>

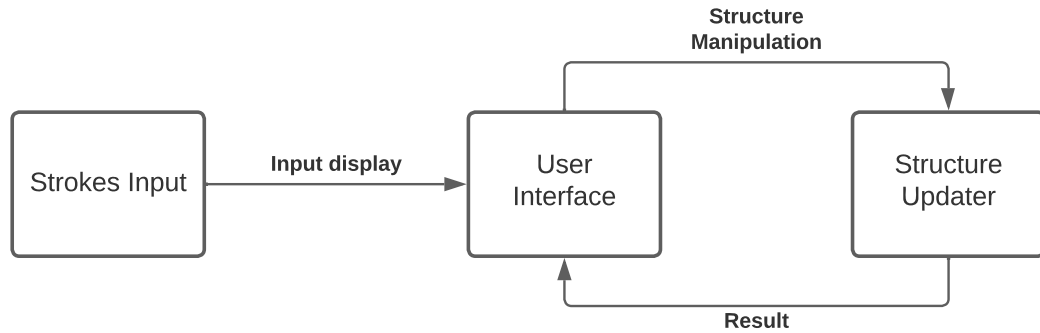


Figure 4.1: System architecture

To establish a framework for the organisation of mathematical expressions, the InkML file associated with it must first be analyzed. To do so, the three main elements of the InkML file will be parsed. These elements are:

- **MathML:** MathML stands for Mathematical Markup Language. It is a standard XML-based language specifically designed for encoding mathematical expressions. Within an InkML document, MathML is used to represent the mathematical content of the ink data. It defines various elements and attributes to express mathematical formulas, equations, symbols, and structures.
- **Traces:** Traces refer to the individual strokes or pen movements captured in the digital ink data. Each stroke is represented as a sequence of coordinates that trace the pen's path. In InkML, traces represent the ink strokes of a mathematical expression.
- **TraceGroups:** Trace groups are used in InkML to group related traces. In mathematical expressions, trace groups group individual traces that form a specific mathematical symbol or component. For example, a trace group may contain the traces of strokes that compose a particular digit, operator, or symbol. Trace groups help organise the ink data and provide a higher-level structure for mathematical expressions, facilitating their recognition and interpretation.

These three elements allow the association of a specific symbol or character to a set of strokes by linking its *id* to the corresponding *traceGroup*, which corresponds to the collection of *trace*'s related to it. With this, the mathematical expression's display and internal structuring are accomplished. A concise depiction of the algorithm's key features and functionalities is shown in Algorithm 1.

Algorithm 1: Display and internal structuring of an expression

```

expression = new Expression();
inkml.normaliseTraces();

// Iterate over each token in the MathML
foreach token  $\in$  inkml.getTokens() do
    // Find the matching trace group for the current token
    matchingTraceGroup = inkml.findMatchingTraceGroup(token);

    if matchingTraceGroup  $\neq$  NULL then
        // Create a new symbol and append it to the expression
        newSymbol = createSymbolFromCharacter(character);
        expression.appendSymbol(newSymbol);

        // Add the traces belonging to the trace group to the
        // draw buffer
        foreach trace  $\in$  matchingTraceGroup.getTraces() do
            drawBuffer.addTrace(trace, matchingTraceGroup.id);
        end
    end
end

```

An *Expression* is created to store all of the symbols belonging to the original expression, and further discussion will be dedicated to this particular data structure, as well as the *Symbol* data structure, which makes up an *Expression*. Furthermore, all of the traces belonging to the InkML are normalised for the coordinates associated with each stroke to fit in the screen, as not every InkML was produced in the same device with identical resolution as one user could have produced an expression on a phone, and another user could have produced an expression using a computer screen. After both these steps have been completed, the algorithm will start to iterate over the InkML. For each token in the MathML, a *Symbol* will be created and added to an *Expression*, and each trace will be added to the draw buffer, which will display each stroke on the whiteboard.

4.3 Data Structures

As previously mentioned, the primary feature of this tool lies in its ability to facilitate the structured manipulation of mathematical expressions. For this functionality to be realized, it is important to establish a data structure that considers the structure within the expressions themselves. By incorporating a data structure that considers the characteristics of mathematical expressions, we can ensure that mathematical rules and operations are applied accurately and precisely, enabling

the manipulation and transformation of the expressions. In this section, the data structures that are used to achieve this are discussed.

A crucial initial step involves storing the expression within an object known as an *Expression*. This *Expression*, as the name suggests, serves as a representation of the mathematical expression under consideration. The expression is represented through an array of *Symbol* entities, as previously mentioned. These *Symbol*'s function as elemental units for both operators and operands. Each symbol contains the following information:

```
class Symbol {
  constructor(symbol, drawId, tagName) {
    this.symbol = symbol;
    this.drawId = drawId;
    this.tagName = tagName;
    this.parent = undefined;
  }
}
```

Through the process of parsing the InkML file, we acquire the first and third variables, the origins of which can be traced back to the MathML element within the file's structure. The definition and characteristics of these variables are as follows:

- *symbol*: refers to the literal symbol that the object refers to and can hold a single character of an operator or multiple characters of an operand.
- *tagName*: refers to the type of symbol that the object refers to, and it can take the values of "mi", "mn" or "mo", which refer to either a variable, a number or an operator, respectively.

The remaining two variables are as follows:

- *drawId*: refers to the *drawId* assigned to the traces stored in the *drawBuffer*, enabling the association of a particular symbol with its representation on the whiteboard.
- *parent*: refers to the symbol's parent node, and initially, every symbol is devoid of a parent until the expression is structured. A comprehensive explanation of this process will be provided in the following sections.

After the successful creation of the symbols, the creation of the *Expression* object is initiated. Each expression contains the following information:

```
class Expression {
  constructor() {
    this.symbols = [];
  }
}
```

```

        this.rootNode = undefined;
    }
}

```

- *symbols*: holds an array that stores all the symbols belonging to the expression. These symbols are progressively added during the parsing of the InkML.
- *rootNode*: refers to the root node of the expression, and its value is established upon the completion of the expression's structuring. A comprehensive explanation of this process will be provided in the upcoming section.

4.4 Structured Trees

The tool at hand serves the critical purpose of facilitating the structured manipulation of mathematical expressions. However, accomplishing this task necessitates the creation of a specialized data structure that can accurately represent the structure of mathematical expressions. Merely dealing with individual symbols or characters in isolation would not suffice.

To enable the manipulation and evaluation of mathematical expressions, a data structure is required that not only captures the symbols but also maintains their hierarchical relationships and dependencies. This ensures that the structure and meaning of the expression are preserved throughout various operations.

The upcoming section will delve into a detailed description of the specific data structure employed for this purpose.

Initially, it was considered to implement a binary search tree, as they are commonly used data structures for solving similar problems. However, it was soon realized that a binary search tree falls short in capturing and preserving the structural information of a mathematical expression.

To illustrate this limitation, let's examine the binary tree depicted in figure 4.2, which represents the expression $a+b+c=d$. In this tree, the hierarchical relationships among the operators and operands are not explicitly evident. Specifically, it is not immediately apparent that the operator "=" holds higher precedence than the operator "+".

In other words, a binary search tree alone fails to capture the inherent structural properties of mathematical expressions. While it can efficiently organize data based on ordering principles, it does not provide a comprehensive representation of the expression's structure, which is crucial for accurately evaluating and manipulating mathematical expressions.

In the context of the MathsPad Tablet [10] [9], the chosen data structure for handling mathematical expressions is the rose tree. This data structure offers a notable departure from the more familiar binary search tree, primarily due to its flexible nature and the ability of each node to have any number of children.

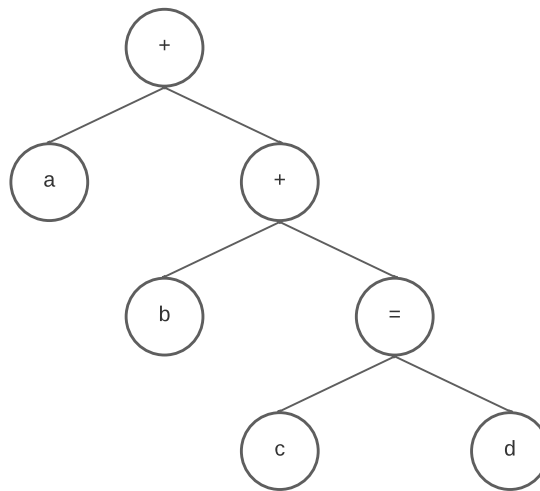


Figure 4.2: Binary tree that represents the expression $a+b+c=d$

While a binary search tree constrains each node to have at most two children, the rose tree breaks free from this limitation. In a rose tree, each node possesses the capability to accommodate an arbitrary number of children, thereby allowing for a more versatile representation of the expression's structure.

In practical terms, this is achieved by equipping each node in the rose tree with a list of subnodes, which acts as a container for its children. This way, the rose tree can capture the complexity of mathematical expressions, accommodating varying numbers of operands, operators, and subexpressions at different levels of the tree.

Figure 4.3 represents the same $a+b+c=d$ expression, but this time it is depicted in a rose tree. By analyzing the tree, it is clear that the expression has two different levels of precedence and all the mathematical rules are captured in this type of tree.

Upon careful examination of the rose tree, it becomes evident that the expression possesses two distinct precedence levels, and the inherent advantage of the rose tree becomes apparent as we explore its branches and nodes. It provides a clear understanding of the relationships and dependencies among different elements of the expression.

This comprehensive representation not only provides a visual overview of the expression but also serves as a valuable tool for accurately performing mathematical operations. It ensures that the correct order of precedence is applied when evaluating the expression, enabling precise calculations and preventing ambiguity.

4.4.1 Rose Tree Node

A notable observation can be made when comparing the nodes of the rose tree, as depicted in figure 4.3, with the nodes of the binary search tree showcased in figure 4.2. These two types of tree nodes exhibit distinct characteristics, highlighting their fundamental differences.

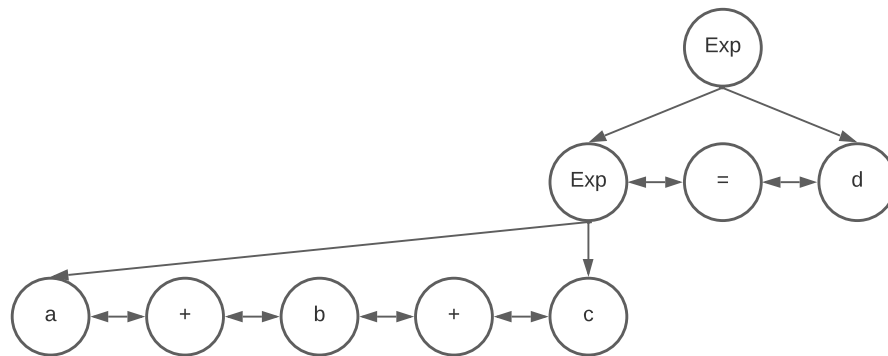


Figure 4.3: Rose tree that represents the expression $a+b+c=d$

A *RoseTreeNode* contains a significantly broader and more extensive range of information than a tree node found in a binary search tree. This additional information is crucial in accurately representing and manipulating mathematical expressions within the rose tree structure.

In contrast to a binary search tree node, a *RoseTreeNode* encompasses various essential attributes that enhance its capabilities. These include:

```

class RoseTreeNode {
    constructor(isExp) {
        this.exp = isExp;
        this.symbol = undefined;
        this.parent = undefined;
        this.backSibling = undefined;
        this.frontSibling = undefined;
        this.firstChild = undefined;
        this.lastChild = undefined;
    }
    ...
}
  
```

- *exp*: holds a binary value indicating whether the node represents a sub-expression. This value serves as a flag or indicator, providing valuable information about the nature of the node within the overall expression hierarchy.
- *symbol*: holds an object of type *Symbol* that refers to the operator or operand associated with the node.

In addition to the previously mentioned attributes, the remaining variables within a *RoseTreeNode*—namely, *parent*, *backSibling*, *frontSibling*, *firstChild*, and *lastChild*—also play significant roles in establishing relationships between nodes within the rose tree structure.

Each of these variables holds an object of type *RoseTreeNode*, representing a specific node associated with its respective name:

- *parent*: points to the *RoseTreeNode* object that serves as the current node's parent. It establishes the hierarchical connection between the current node and its immediate superior within the expression structure.
- *backSibling*: points to the *RoseTreeNode* object that represents the node preceding the current node within the same level of the expression hierarchy. It enables traversal and navigation through siblings that appear before the current node.
- *frontSibling*: points to the *RoseTreeNode* object that represents the node following the current node within the same level of the expression hierarchy. It facilitates traversal and access to siblings that appear after the current node.
- *firstChild*: points to the *RoseTreeNode* object that represents the first child of the current node. It indicates the initial node among the children associated with the current node.
- *lastChild*: points to the *RoseTreeNode* object that represents the last child of the current node. It indicates the final node among the children associated with the current node.

Most of the variables mentioned earlier—except for the *exp* variable—are initialized during the rose tree structure construction. However, it is essential to note that not all variables will have values assigned in every scenario. Their initialization depends on the specific characteristics and relationships of the nodes within the tree.

For instance, when building a sub-expression node, the *symbol* variable remains undefined. This is because sub-expression nodes do not represent individual symbols but rather encompass a group of symbols or sub-expressions. Thus, the *symbol* variable is not applicable in these cases.

Similarly, when a node serves as the *firstChild* of another node, the *backSibling* variable remains undefined. Likewise, when a node is the *lastChild* of another node, the *frontSibling* variable is not defined. This is because, by definition, the *firstChild* does not have a preceding sibling, and the *lastChild* does not have a subsequent sibling within the same level of the expression hierarchy.

An interesting observation derived from this is that if a node is a sub-expression and does not have a parent defined, it signifies that it is the *rootNode* of the entire rose tree structure. This node acts as the starting point and encapsulates the complete mathematical expression.

By understanding the conditions under which these variables remain undefined or take on specific values, we gain insight into the structure and characteristics of the rose tree. These considerations ensure that the rose tree accurately represents the hierarchy and relationships of symbols and sub-expressions within the mathematical expression.

4.4.2 Construction of Structured Trees

In order to delve into the algorithm responsible for constructing the rose tree, let's explore the process of creating the tree that corresponds to the expression $a + b \times (c + d) = e$, where all

the operators are binary. This example will serve as a practical illustration of the construction procedure, highlighting the steps involved and showcasing the resulting rose tree.

The construction algorithm for the rose tree commences by parsing all the operators within the mathematical expression and arranging them in ascending order based on their precedence. This sorting process ensures that the operators are positioned correctly within the resulting array, allowing for the tree's hierarchy to be accurately constructed.

The algorithm identifies the operators present and organizes them in ascending order of precedence. In this case, the operators within the expression are “(”, “×”, “+” and “=”.

To establish the correct order, the algorithm creates an array where each element represents an operator in the expression. Sorting the operators in ascending precedence results in the following array:

$$[(, \times, +, =]$$

It is crucial to note that within this array, only one addition operator (“+”) is present. This is due to the sub-expression enclosed within parentheses being treated as a cohesive unit during the construction of the rose tree. The algorithm recognizes the sub-expression as a single entity and considers its operators a collective unit rather than separate entities within the array.

Following the previous step of sorting the operators in ascending order of precedence, the construction algorithm proceeds to iterate over the resulting array of operators. The first operator encountered during this iteration is an opening parenthesis (“(”). An opening parenthesis introduces a distinctive aspect of the algorithm, as it triggers a recursive process dedicated to handling the expression enclosed within the parentheses.

Upon encountering an opening parenthesis, the algorithm branches off and initiates a separate recursive algorithm to process the expression within the parentheses. This recursive algorithm operates on the sub-expression, treating it as an independent mathematical expression with its own operators and operands. By isolating the sub-expression, the algorithm ensures that its internal structure is captured accurately within the rose tree.

A new array of operators in ascending precedence is generated during this recursive process, specifically for the sub-expression within the parentheses. This new array provides the basis for constructing the rose tree hierarchy within the isolated sub-expression. The resulting sub-expression contains only one operator, the addition operator (“+”). As a result, a new array of operators is formed, consisting of just the addition operator:

$$[+]$$

Following the previous step of generating a new array of operators specific to the sub-expression within the parentheses, the construction algorithm proceeds to iterate over this new array. In this case, the resulting array contains a single element, the addition operator (“+”).

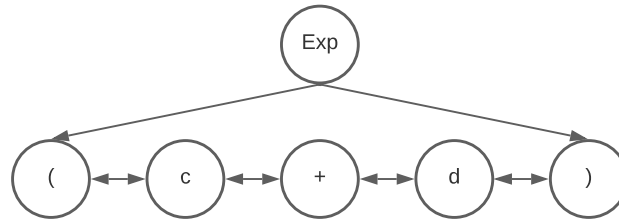


Figure 4.4: First iteration of the construction of the rose tree referring to $a + b \times (c + d) = e$

The algorithm focuses on this operator and examines its associated operands, both on the left and right sides. During this examination, the algorithm considers whether these operands already belong to an existing sub-tree by checking their parent variable.

In this case, the left operand (“c”) and the right operand (“d”) do not have a parent assigned, as indicated by their undefined parent variables.

Since both operands are independent and do not belong to an existing tree, the algorithm creates a new *RoseTreeNode* object to represent this sub-expression. Additionally, it generates three more *RoseTreeNode* objects: one for the addition operator itself and two for the operands, each with their *exp* variable set to *false* to indicate that they are not sub-expression nodes.

Furthermore, the algorithm establishes the necessary relationships between the nodes by defining their parent-child and sibling-sibling connections. The newly created sub-expression node becomes the parent of the addition operator node and both the left and right operand nodes. These connections ensure the sub-expression is properly represented within the rose tree structure.

After this step, the operator’s array is empty, and it returns the root node of the tree constructed so far. It adds two new *RoseTreeNode*’s, one for each parentheses and continues the iteration of the first operator’s array. The rose tree constructed so far is represented in figure 4.4.

After this first iteration, the operator’s array looks like this:

[\times , +, =]

Moving forward with the construction algorithm, the next iteration focuses on evaluating the multiplication (“ \times ”) operator. Similar to the previous addition (“+”) operator, the algorithm examines both the left and right operands associated with the multiplication operator.

Starting with the left operand, which is “b”, the algorithm determines whether it already belongs to an existing tree. In this case, the left operand does not have a parent assigned, indicating that it does not yet belong to any tree. Consequently, the algorithm creates a new *RoseTreeNode* object to represent the left operand. This newly created node becomes a parent node, with the left operand and the multiplication operator as its children. The algorithm establishes the necessary parent-child relationships to reflect the hierarchical structure of the expression.

Moving on to the right operand, which is an opening parenthesis (“(”), the algorithm examines whether it already belongs to a tree. In this particular case, the right operand is part of a sub-expression enclosed within parentheses and already associated with a tree. As a result, the

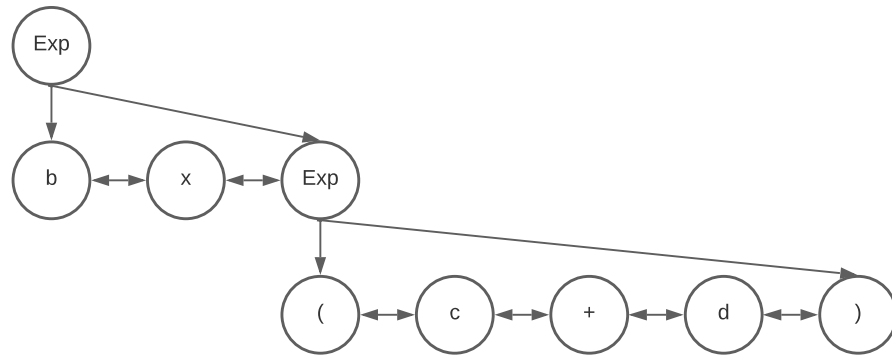


Figure 4.5: Second iteration of the construction of the rose tree referring to $a + b \times (c + d) = e$

algorithm takes a different approach when adding this operand to the newly created node. Instead of adding the right operand itself, the algorithm incorporates the root node of the tree to which the right operand belongs. This decision ensures that the relationships between the nodes accurately represent the original expression's structure and maintain the proper hierarchical arrangement within the rose tree. The rose tree constructed so far is represented in figure 4.5.

After this second iteration, the operator's array looks like this:

[+, =]

Continuing with the construction algorithm, the third iteration focuses on evaluating the addition (“+”) operator. Similar to the previous iterations, the algorithm analyzes both the left (“a”) and right (“b”) operands associated with the addition operator.

In this iteration, the algorithm follows a similar pattern as before. It examines the left operand, “a”, and checks whether it already belongs to an existing tree. As the left operand does not have a parent assigned, indicating that it is not yet part of any tree, the algorithm proceeds to create a new *RoseTreeNode*. This newly created node serves as a parent node, incorporating the left operand and the addition operator. Seeing as the right operand already belongs to a tree, the root node of the said tree will be added as a child of the newly created *RoseTreeNode*. The rose tree constructed so far is represented in figure 4.6.

After this third iteration, the operator's array looks like this:

[=]

The fourth iteration sees the evaluation of the “=” operator, and the process is similar to the previous iterations. The left (“”) and right (“e”) operands are analyzed. The right operand already belongs to an existing tree, and the root node of said tree is added as a child of a new *RoseTreeNode*. After this, the algorithm concludes that the right operand does not belong to any tree and adds the “=” operator and the right operand as children of the newly created *RoseTreeNode*. The rose tree constructed so far is represented in figure 4.7.

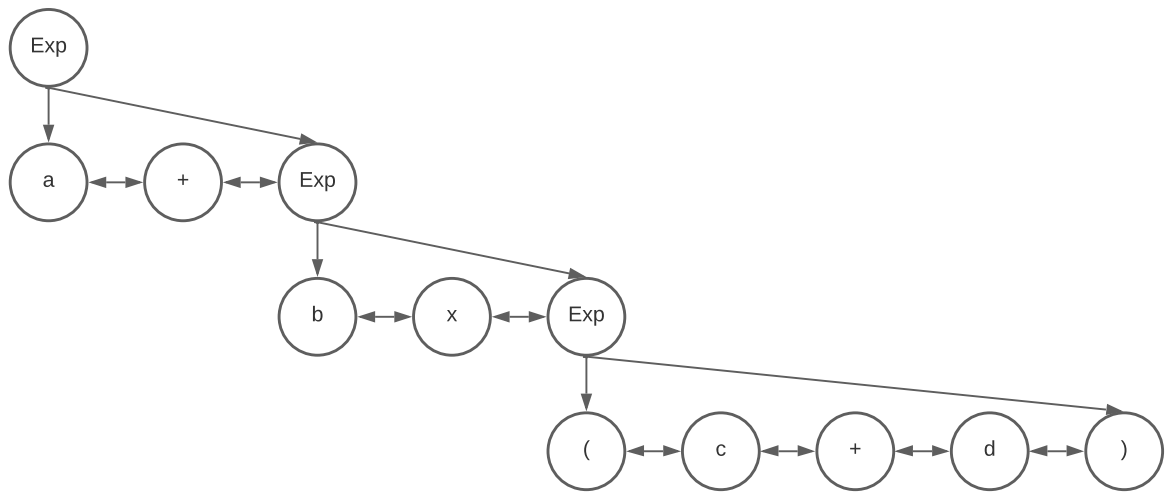


Figure 4.6: Third iteration of the construction of the rose tree referring to $a + b \times (c + d) = e$

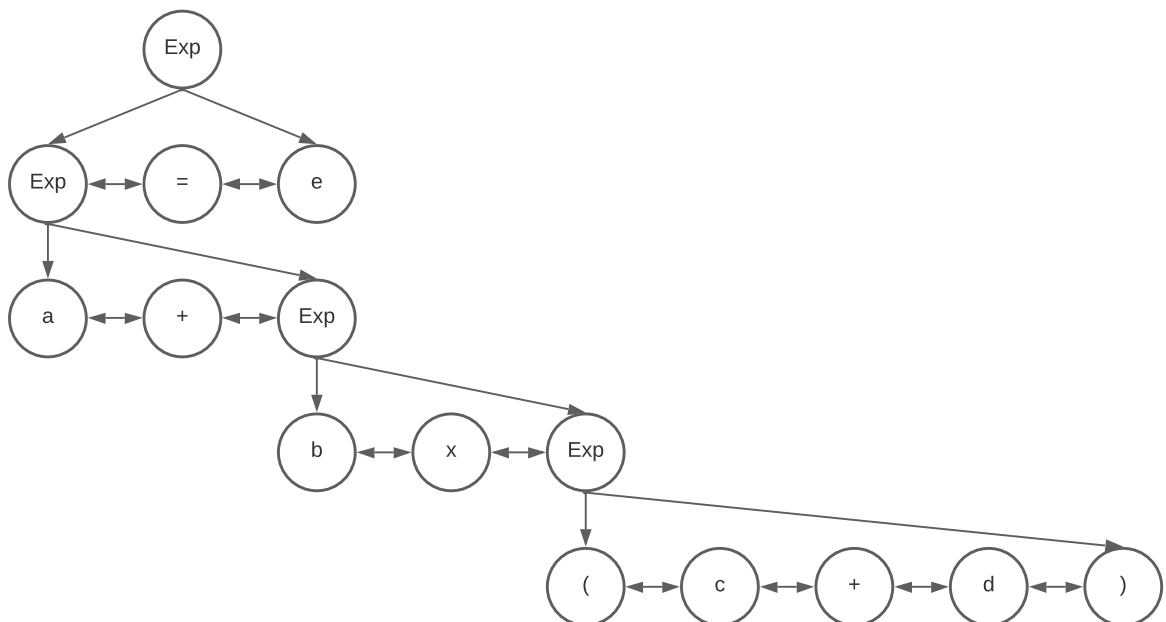


Figure 4.7: Fourth iteration of the construction of the rose tree referring to $a + b \times (c + d) = e$

Upon completing the fourth and final iteration of the algorithm, the operator's array becomes empty, signifying the completion of the tree's construction. At this point, the rose tree fully captures and faithfully represents the entire structure of the mathematical expression.

By constructing the rose tree, we have successfully created a data structure that encapsulates the structure of the mathematical expression.

Provided below is a simplified formulation of the algorithm discussed earlier, referred to as Algorithm 2. This concise representation outlines the key steps involved in constructing the rose tree from a mathematical expression.

Algorithm 2: buildTree: Constructs a tree from the array *expression*

```

clearSymbolsParents(expression.symbols);
operatorArray = orderOperatorsByPrecedenceDescending(expression.symbols);
foreach operator  $\in$  operatorArray do
    if operator.symbol = ( then
        newNode = buildTree(expression[openingParenthesesIndex + 1 :
            closingParenthesesIndex]);
        formBonds(leftSymbol, operator, newNode);
    else
        formBonds(leftSymbol, operator, rightSymbol);
    end
end
return getRootNode(expression[0]);

```

4.5 Structured Modification

4.5.1 Data Highlighting and Duplication

One of the most frequently performed actions when working with mathematical expressions is the need to copy and replicate content repeatedly. Recognizing the significance of this operation, it becomes crucial for the tool to possess a user-friendly copy functionality.

The ability to copy portions of the mathematical expression is essential for various purposes, such as reusing sub-expressions within the same expression or transferring them to other contexts. By enabling users to duplicate content effortlessly, the tool empowers them to efficiently manipulate and rearrange mathematical expressions as needed.

To ensure a better user experience, the copy functionality must be designed with ease and intuitiveness in mind. Users should be able to effortlessly select the desired portion of the expression and trigger the copy operation with minimal effort. The process should be straightforward, enabling users to quickly grasp how to duplicate content without any confusion or unnecessary complexities.

To establish a user-friendly copy mechanism within the tool, it is mandatory to provide an intuitive and natural method for selecting sub-expressions within a larger mathematical expression.

This fundamental aspect ensures that users can effortlessly identify and isolate the specific portions they wish to duplicate.

The selection process should be designed with clarity and simplicity in mind. Users should be able to visually and interactively designate the desired sub-expression by employing intuitive gestures or interactions.

To enhance the user experience, visual cues such as highlighting and colour differentiation are employed to demarcate the selected sub-expression within the larger expression. This immediate visual feedback aids users in confirming their selection and reinforces a sense of confidence and control.

Furthermore, the tool should accommodate various levels of granularity when selecting sub-expressions. Users may desire to copy anything from a single symbol to an entire sub-expression enclosed within parentheses or other mathematical structures. The selection mechanism should allow for this flexibility, enabling users to easily capture the desired level of detail.

To facilitate the easy selection of sub-expressions within the original mathematical expression, a user-friendly approach has been adopted, utilizing single and double-click interactions. By employing these intuitive actions, users can quickly designate the desired portions of the expression for copying or manipulation. Considering the previously mentioned example expression of $a + b \times (c + d) = e$, when a user performs a single click on any symbol within the expression, the selected element will be the symbol itself. This straightforward single-click selection behaviour ensures precision when targeting specific symbols within the expression. For instance, clicking on “a” would result in the selection of “a” as the chosen sub-expression.

However, the selection behaviour differs when a user opts for a double-click interaction. If the user double-clicks on an operand (such as “a”, “b”, “c”, “d”, or “e”), the selection would still focus on the operand itself.

On the other hand, when a user double-clicks on an operator (such as “+”, “×”, or “=”), the selection expands to include the neighbouring symbols associated with that operator. This expansion of the selection enables users to capture entire sub-expressions efficiently. For example, double-clicking on the “×” operator would select the entire sub-expression $b \times (c + d)$. The selection of sub-expressions within the original mathematical expression is made possible through the utilization of the previously constructed structured tree. This tree plays a crucial role in ensuring the accurate and precise selection of sub-expressions.

To enable users to conveniently copy selected content to different sections of the whiteboard, a user-friendly gesture-based approach has been implemented. The designated gesture for copying selected sub-expressions is the “check” gesture presented in figure 4.8. This choice was made from its intuitive nature and its resemblance to an arrow indicating the desired destination for the copied sub-expression. A comprehensive explanation of the gesture process will be provided in further sections.

4.5.2 Expression Manipulation Rules

Two fundamental algebraic rules supported by the tool are *symmetry* and *distributivity*.



Figure 4.8: Check gesture associated with the copy of selected sub-expressions

Users can rearrange an expression's components by using the *symmetry* rule. However, it's the user's responsibility to maintain the equivalence between manipulations. This rule empowers users to swap the positions of symbols or sub-expressions, resulting in an altered expression representation. The ability to apply symmetry within the tool is visually demonstrated by the movement of strokes within the handwritten expression, showcasing the transformation achieved through the rule. The result of applying symmetry to the expression $a+b=c$ is presented in figure 4.9.

In order to facilitate the application of the symmetry rule, the tool incorporates an intuitive gesture-based approach. Users can invoke the symmetry rule by performing a circular gesture around the operator they intend to swap the operands or sub-expressions of. This gesture was thoughtfully selected for its intuitive nature and resemblance to a circular arrow, symbolizing swapping elements around (figure 4.10).

When the user performs the circular gesture, the tool performs the necessary transformations within the underlying rose tree structure representing the expression. By swapping the corresponding nodes in the rose tree, the tool ensures that the structural integrity of the expression is preserved throughout the manipulation process. The modifications on the rose tree of the expression $a+b=c$ when applying the symmetry rule on the "=" operator can be visualized in the figure 4.11.

Algorithm 3 presents a concise high-level description of the operand-swapping algorithm used to interchange the operands of an expression.

$$\begin{array}{c} a + b = c \\ b + a = c \end{array}$$

Figure 4.9: Symmetry

Figure 4.10: Circle gesture applied on the “+” operator $a+b=c$ **Algorithm 3:** Swaps the operands of the operator *node*

```
// Check if the node belongs to an operator
```

```
if node.symbol.tagName = mo then
```

```
    leftNode = node.backSibling;
```

```
    rightNode = node.frontSibling;
```

```
    node.backSibling = rightNode;
```

```
    node.frontSibling = leftNode;
```

```
    // Update connections between nodes
```

```
    swapBonds(leftNode, rightNode);
```

```
    // Update the strokes in the drawBuffer
```

```
    updateDrawBuffer(node, leftSymbol, rightSymbol);
```

```
end
```

The distributivity rule, a powerful tool in mathematical expression manipulation, functions similarly to the symmetry rule. Just as the symmetry rule utilizes existing strokes to rearrange operands or sub-expressions, the distributivity rule uses the strokes to achieve a similar rearrangement. This rule comes into play when a binary expression is enclosed within parentheses, presenting an opportunity to apply the distributivity rule.

To apply the distributivity rule, the operator immediately preceding the parentheses is selectively dragged on top of the operator within the parentheses, as presented in figure 4.12. This action triggers the rearrangement of strokes and ensures the correct implementation of the distributivity rule. Notably, the binary expression enclosed within the parentheses can encompass

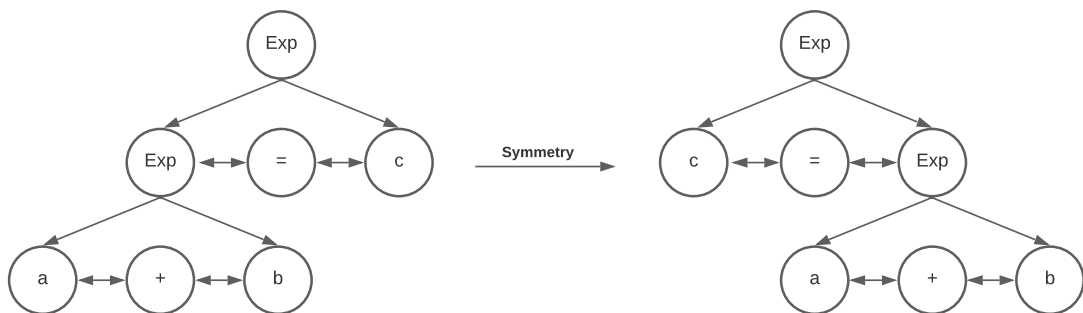
Figure 4.11: Rose tree modification upon applying symmetry on the “=” operator of the expression $a+b=c$



Figure 4.12: Example of the distributivity rule being applied to the expression $a \times (b + c)$

operands or whole sub-expressions.

By employing this intuitive gesture-based mechanism, the tool facilitates the application of the distributivity rule, allowing users to manipulate mathematical expressions while preserving their structural integrity effectively.

4.6 Gestures Implementation

The implementation of gestures within this tool is achieved through a straightforward approach that doesn't rely on external libraries, toolkits, or the training of complex recognition models. Instead, it draws inspiration from the groundbreaking work of Jacob O. Wobbrock, Andrew D. Wilson, and Yang Li, who devised an algorithm known as the *\$1 Recognizer* [13].

The *\$1 Recognizer* algorithm, which serves as the foundation for the gesture recognition system in this tool, enables the accurate identification and interpretation of user gestures. This algorithm combines simplicity with efficiency, making it an ideal choice for the task at hand.

The decision to adopt the *\$1 Recognizer* algorithm was motivated by its robustness and ability to handle various gestures reliably. The algorithm has been extensively studied and validated, providing a solid basis for gesture recognition within the tool. Through the implementation of this algorithm, users can interact with the tool using intuitive and natural gestures, enhancing their overall experience and enabling efficient manipulation of mathematical expressions.

As Wobbrock explains, the algorithm is based on “a simple four-step algorithm”:

- *Step 1 - “Resample the point path”*: Gestures in user interfaces are captured at different speeds, impacting the number of input points recorded. To ensure fair comparisons, gestures are resampled with equidistant points. The number of points 64 is found to be adequate, and this way, the gesture can be directly compared with a set of template gestures, providing a score for each one.
- *Step 2 - “Rotate Once Based on the “Indicative Angle”*”: Determining the optimal rotation angle between two sets of ordered points is a challenging problem. While iterative approaches are typically computationally expensive, the \$1 algorithm is designed to be fast enough for practical use. Instead of relying on brute force, a clever “rotation trick” is employed to speed up the process. The algorithm begins by identifying the indicative angle of a gesture, which is the angle formed between the centroid of the gesture and its first point. By rotating the gesture, this indicative angle is aligned with 0 degrees, resulting in a more efficient alignment process.

- *Step 3 - “Scale and Translate”*: The \$1 algorithm applies scaling and translation operations to the rotated gesture to prepare it for comparison. Scaling is done by fitting the gesture into a reference square, allowing for rotation around the centroid without affecting pairwise point distances. Next, the gesture is translated so that its centroid is positioned at the origin (0,0) for simplicity. These preprocessing steps ensure that the rotated and scaled gestures are standardised for accurate comparison and recognition.
- *Step 4 - “Find the Optimal Angle for the Best Score”*: After preprocessing all candidates and templates by resampling, rotating, scaling, and translating, the recognition process begins. Each candidate is compared to every stored template, calculating the average distance between corresponding points. The template with the smallest path-distance to the candidate is considered the recognized result. This minimum path-distance is converted into a score ranging from 0 to 1.

Chapter 5

Validation and Testing

The following chapter describes the evaluation applied to test the tool’s functionalities. It details the goals of this study in section 5.1, the parameters applied to the experiment in section 5.2, the tasks the participants were asked to conduct in section 5.3, the results that were obtained are discussed in section 5.4 and the tool’s usability is discussed in section 5.5. Section 5.6 details threats to the results’ validity and explains what could have been done to prevent it. Section 5.7 provides the conclusions that were inferred from this study.

5.1 Goals

The primary objective of this case study is to assess the usefulness and efficacy of the tool in manipulating and editing handwritten mathematical expressions while ensuring a structured approach. By evaluating its functionalities, we aim to determine the intuitiveness of the tool and whether it delivers a satisfactory user experience.

The study employed various methodologies and metrics to gather valuable insights to achieve this. Participants volunteered to interact with the tool, allowing us to observe their interactions and collect data on their usage patterns. Through the completion of tasks, we will assess the tool’s effectiveness in handling handwritten mathematical expressions and its ability to structure them for manipulation accurately.

Additionally, the study will evaluate the intuitiveness of the tool’s functionalities through user feedback, surveys, and qualitative assessments to determine the ease of use. We will explore participants’ impressions regarding the tool’s interface, responsiveness, and the overall fluidity of the editing process.

By incorporating both quantitative and qualitative analysis, this case study aims to provide an understanding of the tool’s performance. The data gathered will contribute to further developing the tool’s capabilities, ensuring that it meets the needs and expectations of users.

5.2 Experimental Parameters

The study adopted a set of parameters to ensure a comprehensive and rigorous evaluation of the tool's performance. These parameters were:

- **Tasks:** The study focused on tasks that involved the structured editing and manipulation of imported handwritten mathematical expressions on the whiteboard. These tasks were carefully designed to assess the tool's ability to handle and modify mathematical content effectively.
- **Participants:** The study included a diverse group of 12 participants, who all volunteered to test the tool, ranging in age from 18 to 23. The selection aimed to capture various digital knowledge levels.
- **Environment:** The study was conducted using video calls. Through screen sharing, we interacted and viewed the participants' screens while communicating with them. All participants utilized a mouse as their primary input device to complete the assigned tasks.
- **Recorded Data:** A crucial aspect of the study involved capturing and analyzing the time taken by each participant to complete each task. The recorded data encompassed the duration from when the participant began reading the task instructions to the final step of task execution.
- **Survey:** In addition to task completion data, participants were asked to complete a form that served multiple purposes. Firstly, it aimed to gather demographic information, providing a broader understanding of the participants' backgrounds. Secondly, the survey measured the tool's intuitiveness, usability, and usefulness.

5.3 Case Study Tasks

This case study comprised three essential steps designed to gather valuable insights and assess the tool's functionality and usability. The study followed a structured approach, encompassing a tutorial phase, task execution, and survey assessment.

- **Tutorial:** Before starting tasks, participants were provided with a brief introduction to familiarize themselves with the tool under evaluation. They were instructed to read the introduction and acquaint themselves with the tool's functionalities. During this tutorial, participants were guided through practical exercises that involved performing various actions on an expression.
- **Tasks:** Following the tutorial, participants solved two assigned tasks, as in appendix A. The first task involved importing an InkML file onto the whiteboard and executing a series of transformations on the corresponding mathematical expression. The specific expression in question was " $a+b=c$ ", and participants were required to apply copy and swap operations

on operands to transform it into “ $c=b+a$ ”. The objective of this task was to demonstrate the tool’s ability to manipulate and transform expressions accurately, according to the steps asked in the task. The second task involved importing another InkML file, representing the expression “ $a+b\times(c+d)=e$ ”. Participants were provided with a set of instructions to transform this expression into a non-equivalent form, namely “ $e=(b\times c)+(a+b)\times(a+d)$ ”. This aimed to reinforce the idea that the tool is primarily focused on structured manipulation rather than solving mathematical equations.

- **Survey:** A survey was administered to participants throughout the study, as in appendix B. The survey was divided into multiple parts. Before the tutorial, participants were asked to provide demographic information for a better understanding of the participants’ backgrounds. After each task, participants were prompted to provide feedback on their experience and impressions of the tool’s usability for that specific task. Finally, at the conclusion of all tasks, participants were asked to complete a final section of the survey, providing overall feedback on the tool’s usability.

5.4 Results

5.4.1 Are the Tool’s Functionalities Usable?

To assess the usability of the tool’s functionalities, participants were requested to provide feedback on the difficulty of each functionality for both tasks through the survey.

The results indicated a similar difficulty level reported by participants for Task 1 and Task 2. Initially, it was expected that participants would find Task 2 relatively easier due to their increased familiarity with the functionalities gained from Task 1. However, this was not the case, as many participants found the difficulty level in using the tool’s functionalities the same in both tasks.

One plausible explanation is that the tutorial effectively familiarised participants with the tool’s functionalities. The tutorial ensured that participants understood the functionalities and their usage, leading to a high level of familiarity. This outcome highlights the tool’s intuitiveness and ease of use, as participants quickly grasped the functionalities during the tutorial phase, enabling them to apply them confidently in both tasks.

Table 5.1 provides an overview of the average difficulty scores reported by the participants. These scores, ranging from 1 (indicating a high difficulty level) to 5 (indicating ease of use), offer insights into the participants’ perceptions of the tool’s functionalities across both tasks. The table presents a score for each functionality in each task, offering a comparison of the participant’s experiences.

The average difficulty score for Task 1 was recorded as 4.83, while for Task 2, it was 4.50. These scores shed light on the participants’ perceptions of the difficulty level encountered in each task.

Feature	Task 1	Task 2
Selection	4.58	4.50
Copy	4.50	4.50
Swap	4.50	4.50
Distributivity	n/a	4.33

Table 5.1: Difficulty reported on each functionality (1 - Very hard; 5 - Very easy)

Interestingly, despite Task 2 being considerably more complex than Task 1 and introducing a new functionality, the reported difficulty levels remained relatively similar. This outcome reinforces the notion that the tool possesses inherent intuitiveness and ease of use.

5.4.2 Is the Tool Straightforward?

To assess the tool's user-friendliness, participants were specifically requested to offer insights into the frequency of whiteboard clearances and task restarts required to complete each task successfully.

By considering the number of times participants had to clear the whiteboard and start again, we gained valuable insights into the tool's straightforwardness and ability to facilitate task completion. This measure serves as a significant indicator of the participants' overall experience with the tool and sheds light on the effectiveness of its design.

The connection between the frequency of whiteboard clearances and the participants' perceived task difficulty is connected because if participants found the tasks to be generally easy, it should be expected that they would encounter minimal challenges and have little need to reset the whiteboard and start over from scratch.

This approach adds depth to our assessment, allowing us to delve into the practical aspects of task completion. The data collected regarding whiteboard clearances and task restarts provides valuable quantitative data, and the values recorded by the participants in Task 1 and Task 2 can be examined in figure 5.1 and figure 5.2, respectively.

By analyzing both graphs, it can be concluded that the tool is straightforward as the participants didn't need to start the tasks from fresh multiple times to complete them.

5.4.3 How Complex is the Tool Versus Ordinary Methods?

To evaluate how much complex or simpler using the tool to complete the tasks is, the participants were asked to give their opinion on this exact matter. After completing each task, they were asked to compare the complexity of completing the same task using a pen and paper or a blackboard versus using the tool's functionalities.

The results obtained after Task 1 are very mixed, with around half the participants considering the tool's functionalities less complex and the other half considering them more complex. This can be observed in figure 5.3.

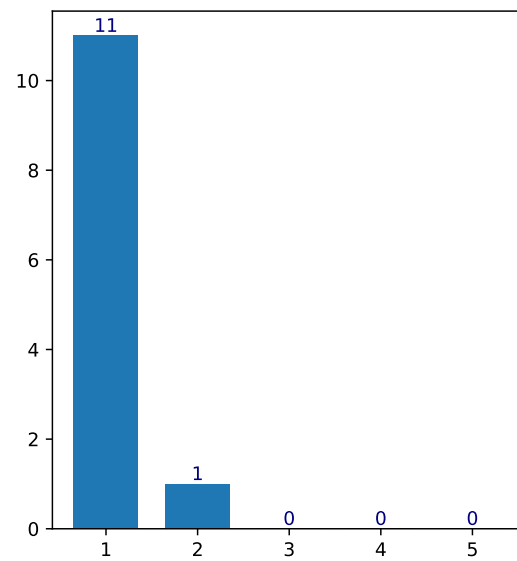


Figure 5.1: “I had to clear the whiteboard multiple times in order to complete the task (Task 1).”
1 - (Strongly disagree); 5 - (Strongly agree)

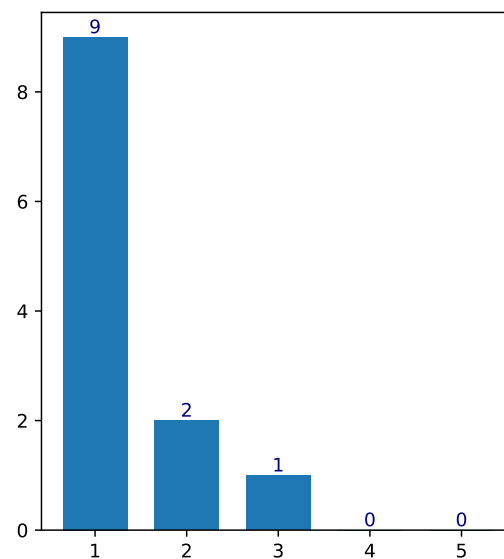


Figure 5.2: “I had to clear the whiteboard multiple times in order to complete the task (Task 2).”
1 - (Strongly disagree); 5 - (Strongly agree)

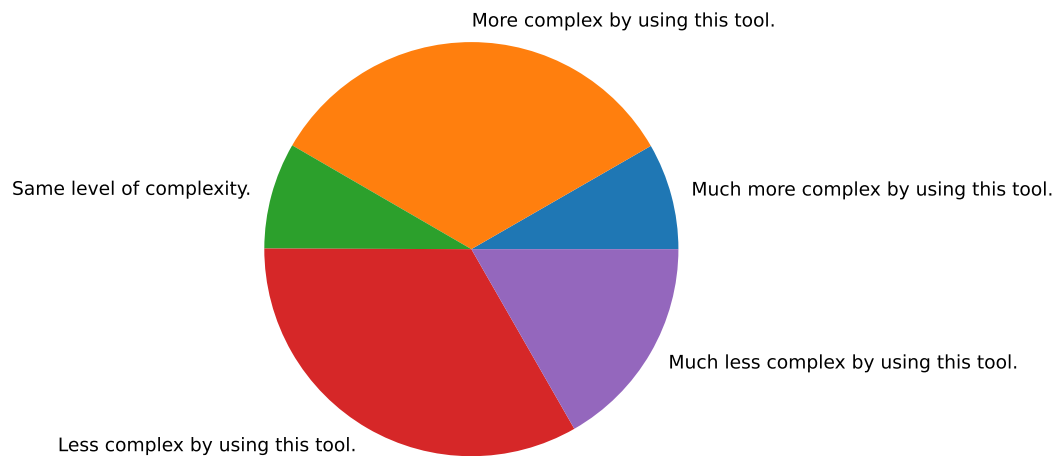


Figure 5.3: “How do you compare the complexity of completing this task (Task 1) by writing all of the steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool?”

On the other hand, the results obtained after Task 2 indicate that the participants think that completing the task using the tool’s functionalities is overall less complex than completing it using a piece of paper and pen or a blackboard. Figure 5.4 shows the results obtained.

The different results obtained in both tasks can be explained due to Task 2 being more complex than Task 1. This way, the participants involved think that completing difficult tasks that involve multiple steps and expression modifications is less complex using the tool’s functionalities. Another small factor in the different results can be the further familiarization of the participant with the tool from Task 1 to Task 2.

With these results, it can be concluded that the use of the tool’s functionalities for less complex tasks is not as optimal as simply using ordinary methods. The same cannot be said for more complex tasks, which the participants think are less complex to complete using the tool.

5.5 Usability

To evaluate the usability of the tool’s functionalities, the *System Usability Scale* (SUS) [3] was used. It consists of 10 questions that the participant must answer with a score from 1 to 5, from “Strongly disagree” to “Strongly agree”. The usability score of the tool’s functionalities was then calculated by adding the average scores of each question. The score for questions 1, 3, 5, 7 and 9 is calculated by subtracting 1 from the scale position, and the score for questions 2, 4, 6, 8 and 10 is 5 minus the scale position. These scores are then multiplied by 2.5. The scores for each question and the final score can be analyzed in table 5.2.

According to *System Usability Scale*, a score above 68 is considered above average in terms of usability. The score obtained by the tool is 87.167, which attributes high usability to it.

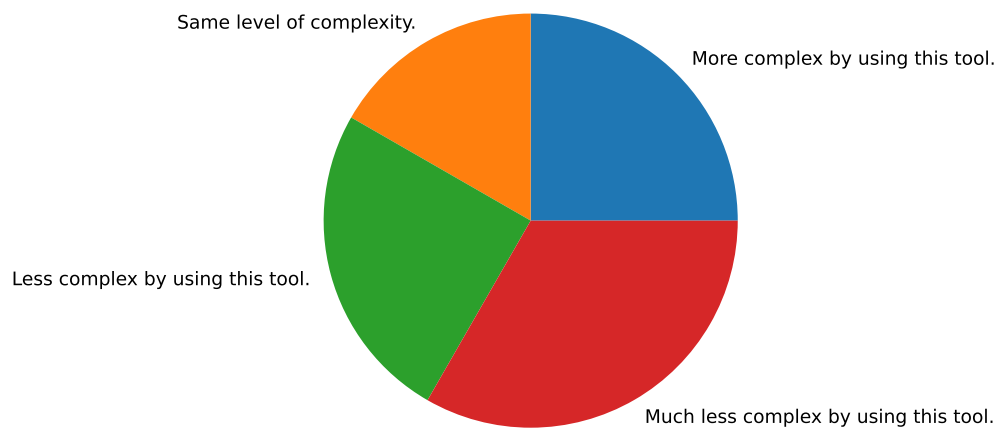


Figure 5.4: “How do you compare the complexity of completing this task (Task 2) by writing all of the steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool?”

System Usability Scale (SUS) Questions	Average Scores
I think that I would like to use these functionalities frequently.	7.708
I found the functionalities unnecessarily complex.	9.667
I thought the functionalities were easy to use.	8.333
I think I would need the support of a technical person to be able to use the functionalities.	9.375
I found the various functions of this system functionalities were well integrated.	8.542
I thought there was too much inconsistency in this system’s functionalities.	8.125
I would imagine that most people would learn to use these functionalities very quickly.	8.542
I found the functionalities very cumbersome to use.	8.958
I felt very confident using these functionalities.	8.750
I needed to learn a lot of things before I could get going with these functionalities.	9.167
Total SUS Score	87.167

Table 5.2: Table showing the average System Usability Scale (SUS) score for each individual question.

5.6 Threats to Validity

This section discusses some biases that could have affected the study. These biases derive from the “Types of Bias in Research”¹, which refer to various types of biases that may be introduced in studies.

5.6.1 Selection Bias

Selection bias refers to bias introduced in the study through the study population. All the participants who volunteered for this study are between the ages of 18 and 23. This selection does not entirely represent the population that should be analyzed among the potential users of this tool’s functionalities. The small gap in the participant’s age limits the study, as people outside this age group might have slightly different perspectives. This can lead to an age-related bias. The same can be said because all participants used a regular PC with a mouse for task completion. Once again, this does not capture all potential users of the tool’s functionalities, as someone using a tablet PC instead could have different user experiences while using the tool. Most of the participants would consider that the tasks would be made simpler if they were performed using a tablet PC, as can be seen in figure 5.5 and figure 5.6.

5.6.2 Confirmation Bias

Confirmation bias refers to bias introduced when pre-existing beliefs heavily support the decisions made. This could have been introduced in the study, as many participants had difficulties with mathematical concepts in the past, and their opinions on the usefulness of the tool’s functionalities might be a little skewed as they might perceive the tool’s functionalities as more beneficial than they actually are.

5.7 Conclusions

Even though many of the results gathered from the participants regarding the tool’s usefulness and complexity may be skewed due to the factors discussed in section 5.6, there is no denying that the results are very positive. The use of gestures was very well received by the participants, and the functionalities were deemed as useful, even though they were not easy to use at first. Many participants found the use of the tool’s functionalities less complex than the use of ordinary methods like a sheet of paper and pen or a blackboard, and these opinions regarding the complexity of the tool were more positive when dealing with more complex expressions, as is the case of Task 2. The tool was also considered above average in terms of usability, according to the *System Usability Scale* [3], obtaining a score of 87.167.

¹<https://www.scribbr.com/research-bias>

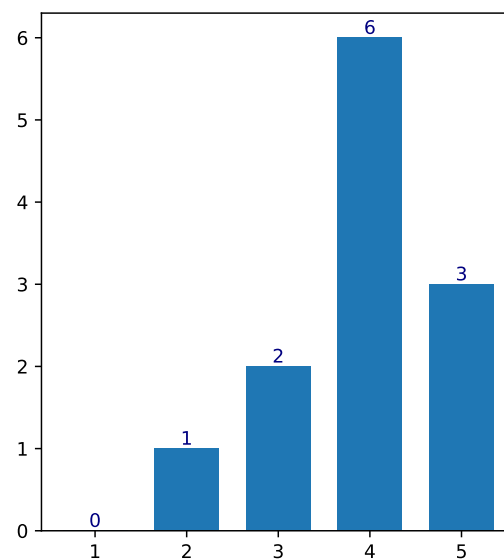


Figure 5.5: “Completing this task (Task 1) using a tablet PC would be...” 1 - (A lot harder); 5 - (A lot easier)

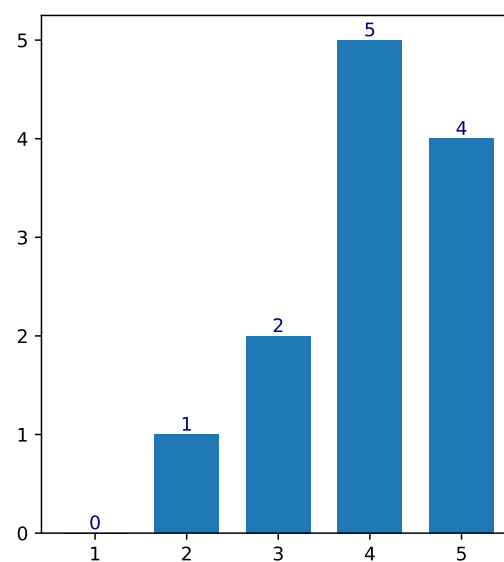


Figure 5.6: “Completing this task (Task 2) using a tablet PC would be...” 1 - (A lot harder); 5 - (A lot easier)

Chapter 6

Discussion and Conclusions

This project's main objective was to develop a structured editor for handwritten mathematics that facilitates teaching mathematics in the classroom and eases research tasks. This is done by providing a manipulator with features that has the potential to reduce the complexity of doing mathematics.

The use of traditional methods, such as pen and paper or a blackboard, for teaching mathematics or research purposes has slowed down its evolution and is being complemented by more advanced methods. The goal of using digital tools designed for this purpose is to overcome the difficulties presented by traditional methods, such as rewriting content, space management and erasing important content. The tool presented in this thesis has the potential to overcome all those difficulties while maintaining a low complexity compared to the traditional methods, as was proven in chapter 5, where the tool's functionalities were highly praised by the participants of the study conducted.

The sub-expression selection functionality is a very valuable feature when dealing with mathematical expressions as it deals with the internal structure of the expression itself. It allows users to correctly assess the precedence of the operators and the order of operations they must follow. It also acts as the foundation for all the other functionalities.

The copy functionality is probably the most performed action when dealing with mathematics using traditional methods, and it can introduce many errors. This functionality prevents just that by allowing the copy of sub-expressions, without introducing errors, easily with gestures.

The swap of operands introduces a functionality that allows users to perform actions that may not be mathematically correct, putting the responsibility of correctly applying this rule on the user's shoulders.

The distributivity rule is also a rule that is applied a lot in mathematical contexts and, just like the swap of operands functionality, it can be applied in contexts that are not mathematically correct, enhancing the manipulation power of the user.

Data gathered from the user studies conducted seem to indicate that tool's functionalities are promising and, through further development, could lead to its application in the classroom or

research contexts. This is proven by the high usability rating given by the participants and the low complexity recorded by them.

Even though there are some limitations, the use of this tool's functionalities can help improve the application of mathematical concepts in the contexts of teaching and research as it aims to replace the pen and paper and facilitate the manipulations of mathematical expressions. Based on the tests performed, the tool's functionalities seem useful in these environments and can lay the groundwork for future developments.

6.1 Future Work

In this section, various suggestions for the tool's improvement are provided. These suggestions originate from user feedback and ideas for the tool's progression that were not implemented. These are:

- **Mathematical recognition:** Even though handwriting recognition is not the goal of this project, there is no denying that this tool would be a lot more useful as a whole if the inputs didn't have to be made via InkML files, but rather through the raw input of mathematical expressions on the whiteboard. This would facilitate the usability of the tool, making it more intuitive and interactive for the user. This exact idea is being developed on a separate project parallel to this and the next step would be to merge both projects.
- **Implementation of extra mathematical functionalities:** The tool's usefulness could be further improved with the introduction of more functionalities such as the *substitution of equals*, where a user can substitute a variable with a sub-expression. The extra functionalities could provide a higher usability rating for the tool and make it more essential in certain mathematical contexts.
- **User customization:** Through the definition of new operators or the editing of the existing operators' precedence, the tool could become more useful by shaping itself to the user's need. This could take expression manipulation to the next level by giving the user the power to alter the operators' qualities. This way, the user could alter the operators in such a way that the addition (“+”) operator could have priority over the multiplication (“×”) operator.
- **Equation solver:** Another idea for the further development of this tool is through the implementation of an equation solver within the tool's functionalities. This way the user would be able to solve mathematical expressions that had been previously manipulated by him, and when pairing this with the previous idea mentioned, it would output some very interesting results.
- **User interface:** The most common suggestion provided by the participants of the user tests is the further improvement of the interface, making it more user-friendly. This could be achieved by displaying the gestures performed by the user on the screen or by the display

of information regarding each expression. The information could range from its internal structure to the rules that are applicable to it.

References

- [1] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. An integrated grammar-based approach for mathematical expression recognition. *Pattern Recognition*, 51:135–147, 2016.
- [2] Roland Backhouse. *Program Construction: Calculating Implementations from Specifications*. John Wiley & Sons, Inc., USA, 2003.
- [3] John Brooke. Sus: A quick and dirty usability scale. *Usability Eval. Ind.*, 189, 11 1995.
- [4] Cassandra Guy, Michael Jurka, Steven Stanek, and Richard Fateman. Math speak & write, a computer program to read and hear mathematical input. *University of California Berkeley*, 2004.
- [5] Sagar Bharadwaj KS, Vilas Bhat, and Arvind Sai Krishnan. Solveit: an application for automated recognition and processing of handwritten mathematical equations. In *2018 4th International Conference for Convergence in Technology (I2CT)*, pages 1–8. IEEE, 2018.
- [6] George Labahn, Edward Lank, Scott MacLean, Mirette Marzouk, and David Tausky. Mathbrush: A system for doing math on pen-based devices. In *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, pages 599–606. IEEE, 2008.
- [7] Joseph J LaViola Jr and Robert C Zeleznik. Mathpad2: a system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2006 Courses*, pages 33–es. 2006.
- [8] Chuanjun Li, Timothy S Miller, Robert C Zeleznik, and Joseph J LaViola Jr. Algosketch: Algorithm sketching and interactive computation. *SBIM*, 8:175–182, 2008.
- [9] Alexandra Mendes. *Structured editing of handwritten mathematics*. PhD thesis, University of Nottingham, 2012.
- [10] Alexandra Mendes, Roland Backhouse, and Joao F Ferreira. Structure editing of handwritten mathematics: Improving the computer support for the calculational method. In *Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces*, pages 139–148, 2014.
- [11] Masakazu Suzuki, Fumikazu Tamari, Ryoji Fukuda, Seiichi Uchida, and Toshihiro Kanahori. Infty: an integrated ocr system for mathematical documents. In *Proceedings of the 2003 ACM symposium on Document engineering*, pages 95–104, 2003.
- [12] Erik Weitnauer, David Landy, and Erin Ottmar. Graspable math: Towards dynamic algebra notations that support learners better than paper. In *2016 Future Technologies Conference (FTC)*, pages 406–414. IEEE, 2016.

- [13] Jacob O Wobbrock, Andrew D Wilson, and Yang Li. Gestures without libraries, toolkits or training: a \$1 recognizer for user interface prototypes. In *Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 159–168, 2007.
- [14] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J LaViola. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *International Symposium on Smart Graphics*, pages 20–32. Springer, 2008.

Appendix A

Case Study Guide

This appendix refers to the case study guide that all participants followed in order to conduct the evaluation of the tool's functionalities.

User Study

Structured Manipulation of Handwritten Mathematics

Introduction

Thank you for taking this interview which aims to obtain your opinion about a tablet PC tool designed for the structured manipulation of handwritten mathematics. You will then have to manipulate and edit mathematical expressions.

This document provides a tutorial to get you acquainted with the tool to solve the tasks you're asked to.

You will be asked to:

- Go through a tutorial
- Manipulate some expressions
- Answer some questions

Requirements:

- Basic understanding of mathematical concepts
- Internet connection
- A mouse is recommended

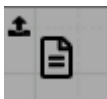
Before starting, please answer the first three sections of this [form](#).

Tutorial

Start by following [these](#) instructions in order to download and install the tool.

First Steps

Download the file “secondExpression.inkml” and load it to the whiteboard




using the button.

After this step, the screen should look like this:



$$a + b \times (c + d) = e$$



Now select  tool and follow the next steps.


Now select the whole expression by double-clicking the “=” sign, and copy it to another place on the screen by pressing down and making a “V” gesture with your mouse/pen in the place you want the expression to be copied to.

Now swap the order of the operands in the addition sub-expression by making a “circle” gesture around the “+” sign with your mouse/pen.

Now apply the distributivity rule to the “ $b \times (c + d)$ ” sub-expression by dragging the “ \times ” sign on top of the “+” sign. Make sure that the mouse starts and ends on the strokes of each operator.

Now select the sub-expression “ $(b \times c) + (b \times d)$ ” by double-clicking the “+” sign and copy it to another place on the screen.



Once you’re finished, press the  button twice to clear the whiteboard and start the tasks.

Task 1

Download the “firstExpression.inkml” file and import it to the whiteboard. The screen should look like this:



$$a + b = c$$



Now select tool and follow the next steps.

Manipulate the expression, step by step, in order to obtain the expression: “c=b+a”. Make sure all steps are displayed on the whiteboard by using the “copy” functionality.

- Start by selecting the whole expression by double-clicking the “=” sign and copying it to another area of the screen by using the “V” gesture.
- Next, swap the “a” and “b” operands by circling the “+” operator on the expression you just copied.
- Next, select the whole expression by double-clicking the “=” sign and copy it to another area of the screen by using the “V” gesture.
- Next, swap the sub-expression “b+a” and “c” by circling the “=” operator on the expression you just copied.



If you are having difficulties you can start over by pressing the button twice.

Now please answer the section about task 1 in the same form as above.



Once you’re finished, press the button twice to clear the whiteboard and start task 2.

Task 2

Download the “secondExpression.inkml” file and import it to the whiteboard. The screen should look like this:



$$a + b \times (c + d) = e$$




Now select tool and follow the next steps.

Apply the required transformations to the expression in order to obtain the expression: $e = (b * c) + (a + b) * (a + d)$.

Remember that this is not equivalent to the original expression.

- Start by selecting the whole expression by double-clicking the “=” operator and copy it to another place on the screen. Now perform the next steps on the copied expression.
- Next, apply the distributivity rule to $b * (c + d)$ by dragging the “*” operator on top of the “+” operator.
- Next, swap the sub-expression $(b * c)$ with “a” by circling the “+” operator.
- Next, apply the distributivity rule again to $a + (b * d)$ by dragging the “+” operator on top of the “*” operator.
- Finally, swap the sub-expression $(b * c) + (a + b) * (a + d)$ with “e” by circling the “=” operator.

If you are having difficulties you can start over by pressing the  button twice.

Now, please finish answering the form.

Appendix B

Case Study Form

This appendix refers to the user study form that all participants were asked to answer in order to obtain information about their experience using the tool's functionalities.

Structured Manipulation of Handwritten Mathematics - User Experience

The purpose of this form is to capture the user's experience with the tool and gather their perspective on its usability.

* Indica uma pergunta obrigatória

1. The primary aim of this form is to gather your opinion on a tablet PC tool *
designed for manipulating handwritten mathematics in a structured manner.

In the initial sections of the form, we collect demographic information along with details about the environment in which you are utilizing the tool.

In this study, **all the data collected will remain anonymous**. This data may be utilized for presentations at conferences, academic events, similar gatherings, and scientific publications.

Please note that your participation in this form is **voluntary**, and you can withdraw at any time without facing any penalties.

By selecting "**Yes**" below, you indicate your consent for the processing, storage, and utilization of the data as described above. You confirm that you have read and understood this consent form and that **you are above the age of 18**. If you select "**No**," you must immediately discontinue the study.

Marcar apenas uma oval.

- ☐ Yes *Avançar para a pergunta 2*
- ☐ No

Demographic Data

In this section, demographic data is collected, and all data collected will be handled with confidentiality.

2. Age *

Marcar apenas uma oval.

☐ 18 - 23

☐ 24 - 30

☐ 31 - 40

☐ 41 - 50

☐ > 51

3. Gender *

Marcar apenas uma oval.

☐ Male

☐ Female

☐ Prefer not to say

☐ Other

4. How would you rate your digital literacy? *

Marcar apenas uma oval.

Very weak

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very strong

5. What are you currently using? *

Marcar apenas uma oval.

- ☐ Regular PC *Avançar para a pergunta 6*
- ☐ Tablet PC *Avançar para a pergunta 9*

Regular PC

Regarding the regular PC, please provide the remaining information.

6. What are you currently using? *

Marcar apenas uma oval.

- ☐ Mouse
- ☐ Trackpad

7. What operating system are you using? *

Marcar apenas uma oval.

- ☐ Windows
- ☐ Mac
- ☐ Linux
- ☐ Other

8. What browser do you use? *

Marcar apenas uma oval.

- ☐ Chrome
- ☐ Safari
- ☐ Firefox
- ☐ Edge
- ☐ Opera
- ☐ DuckDuckGo
- ☐ Brave
- ☐ Other

Avançar para a pergunta 12

Tablet PC

Regarding the tablet PC, please provide the remaining information.

9. Are you using a stylus/digital pen? *

Marcar apenas uma oval.

- ☐ Yes
- ☐ No

10. What operating system are you using? *

Marcar apenas uma oval.

- ☐ Android
- ☐ iOS
- ☐ Windows
- ☐ Other

11. What browser do you use? *

Marcar apenas uma oval.

- ☐ Chrome
- ☐ Safari
- ☐ Firefox
- ☐ Edge
- ☐ Opera
- ☐ DuckDuckGo
- ☐ Brave
- ☐ Other

Avançar para a pergunta 19

Task 1 (Regular PC)

The questions below regard task 1 using a regular PC.

12. Completing task 1 was... *

Marcar apenas uma oval.

Very difficult

1

☐

2

☐

3

☐

4

☐

5

☐

Very easy

13. The sub-expression selection functionality is... *

Marcar apenas uma oval.

Very difficult to use

1

☐

2

☐

3

☐

4

☐

5

☐

Very easy to use

14. The copy functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

15. The swap of operands functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

16. I had to clear the whiteboard multiple times in order to complete the task. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

17. How do you compare the complexity of completing this task by writing all of steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool? *

Marcar apenas uma oval.

☐ Much more complex by using this tool.

☐ More complex by using this tool.

☐ Same level of complexity.

☐ Less complex by using this tool.

☐ Much less complex by using this tool

18. Completing this task using a tablet PC would be... *

Marcar apenas uma oval.

A lot harder

1

2

3

4

5

A lot easier

Avançar para a pergunta 26

Task 1 (Tablet PC)

The questions below regard task 1 using a tablet PC.

19. Completing task 1 was... *

Marcar apenas uma oval.

Very difficult

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy

20. The sub-expression selection functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

21. The copy functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

22. The swap of operands functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

23. I had to clear the whiteboard multiple times in order to complete the task. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

24. How do you compare the complexity of completing this task by writing all of steps asked in the task on a piece of paper or on a blackboard, versus writing it using this tool? *

Marcar apenas uma oval.

☐ Much more complex by using this tool.

☐ More complex by using this tool.

☐ Same level of complexity.

☐ Less complex by using this tool.

☐ Much less complex by using this tool

25. Completing this task using a regular PC would be... *

Marcar apenas uma oval.

A lot harder

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

A lot easier

Avançar para a pergunta 34

Task 2 (Regular PC)

The questions below regard task 2 using a regular PC.

26. Completing task 2 was... *

Marcar apenas uma oval.

Very difficult

1

☐

2

☐

3

☐

4

☐

5

☐

Very easy

27. The sub-expression selection functionality is... *

Marcar apenas uma oval.

Very difficult to use

1

☐

2

☐

3

☐

4

☐

5

☐

Very easy to use

28. The copy functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

29. The swap of operands functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

30. The distributivity rule functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

31. I had to clear the whiteboard multiple times in order to complete the task. *

Marcar apenas uma oval.

Strongly disagree

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Strongly agree

32. How do you compare the complexity of completing this task by writing it on a piece of paper or on a blackboard, versus writing it using this tool? *

Marcar apenas uma oval.

- ☐ Much more complex by using this tool.
- ☐ More complex by using this tool.
- ☐ Same level of complexity.
- ☐ Less complex by using this tool.
- ☐ Much less complex by using this tool

33. Completing this task using a tablet PC would be... *

Marcar apenas uma oval.

A lot harder

1

☐

2

☐

3

☐

4

☐

5

☐

A lot easier

Avançar para a pergunta 42

Task 2 (Tablet PC)

The questions below regard task 2 using a tablet PC.

34. Completing task 2 was... *

Marcar apenas uma oval.

Very difficult

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy

35. The sub-expression selection functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

36. The copy functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

37. The swap of operands functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

38. The distributivity rule functionality is... *

Marcar apenas uma oval.

Very difficult to use

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Very easy to use

39. I had to clear the whiteboard multiple times in order to complete the task. *

Marcar apenas uma oval.

Strongly disagree

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Strongly agree

40. How do you compare the complexity of completing this task by writing it on a piece of paper or on a blackboard, versus writing it using this tool? *

Marcar apenas uma oval.

- ☐ Much more complex by using this tool.
- ☐ More complex by using this tool.
- ☐ Same level of complexity.
- ☐ Less complex by using this tool.
- ☐ Much less complex by using this tool

41. Completing this task using a regular PC would be... *

Marcar apenas uma oval.

A lot harder

1

☐

2

☐

3

☐

4

☐

5

☐

A lot easier

Avançar para a pergunta 42

Usability

This section poses questions about the system functionalities' quality in terms of usability.

42. I think that I would like to use these functionalities frequently. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

43. I found the functionalities unnecessarily complex. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

44. I thought the functionalities were easy to use. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

45. I think I would need the support of a technical *
person to be able to use the functionalities.

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

46. I found the various functions of this system functionalities were well integrated. *

Marcar apenas uma oval.

Strongly disagree

1

2

3

4

5

Strongly agree

47. I thought there was too much inconsistency in this system functionalities. *

Marcar apenas uma oval.

Strongly disagree

1

2

3

4

5

Strongly agree

48. I would imagine that most people would learn to use these functionalities very quickly. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

49. I found the functionalities very cumbersome to use. *

Marcar apenas uma oval.

Strongly disagree

1

☐

2

☐

3

☐

4

☐

5

☐

Strongly agree

50. I felt very confident using these functionalities. *

Marcar apenas uma oval.

Strongly disagree

1 ☐

2 ☐

3 ☐

4 ☐

5 ☐

Strongly agree

51. I needed to learn a lot of things before I could get going with these functionalities. *

Marcar apenas uma oval.

Strongly disagree

1

2

3

4

5

Strongly agree

52. Observations and improvement suggestions.

Este conteúdo não foi criado nem aprovado pela Google.

Google Formulários