

SETPAD: A SKETCH-BASED TOOL FOR EXPLORING DISCRETE MATH
SET PROBLEMS

by

TRAVIS COSSAIRT
B.S.E., University of Michigan, 2001

A thesis submitted in partial fulfillment of the requirements
for the degree of Master of Science
in the Department of Electrical Engineering and Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2012

Major Professor: Joseph J. LaViola Jr.

© Copyright 2012 Travis Cossairt

ABSTRACT

We present SetPad, a new application prototype that lets computer science students explore discrete math problems by sketching set expressions using pen-based input. Students can manipulate the expressions interactively with the tool via pen or multi-touch interface. Likewise, discrete mathematics instructors can use SetPad to display and work through set problems via a projector to better demonstrate the solutions to the students. We discuss the implementation and feature set of the application, as well as results from both an informal perceived usefulness evaluation for students taking a computer science foundation exam in addition to a formal user study measuring the effectiveness of the tool when solving set proof problems. The results indicate that SetPad was well received, allows for efficient solutions to proof problems, and has the potential to have a positive impact when used as an individual student application or an instructional tool.

ACKNOWLEDGMENTS

This thesis has been supported by the patience and encouragement of my wife Amy. I would also like to thank my father Mikel and my mother Margery for their support during both my undergraduate and graduate studies. Special thanks to my brother Jacob for always challenging me academically and also to my friends and coworkers.

My faculty advisor, Dr. Joseph LaViola, deserves praise for his constant guidance and support throughout my graduate experience. Likewise, I would like to thank the members of my committee, Dr. Charles Hughes, and Dr. Hassan Foroosh for their contributions. Finally, I would like to thank the past and present members of the Interactive Systems and User Experience lab for their help and feedback.

TABLE OF CONTENTS

LIST OF FIGURES	viii
LIST OF TABLES	xi
CHAPTER 1 INTRODUCTION	1
Section 1.1 Motivation	1
Section 1.2 Statement of Research Question	3
Section 1.3 Thesis Overview	3
CHAPTER 2 RELATED WORK	4
CHAPTER 3 USER INTERFACE	7
Section 3.1 Expression Entry and Visualization	7
Section 3.2 Interactive Algebraic Law Manipulation	9
Section 3.3 Expanding Simple Expressions	22
Section 3.4 Defining Subsets	26
Section 3.5 X Element Of Notation	29
Section 3.6 Elements In Set Problems	30
Section 3.7 Previous Step Branching	31
Section 3.8 Workspace Maneuvering	32
Section 3.9 Real-time Observation	32
CHAPTER 4 ARCHITECTURE	34

Section 4.1	Frameworks	35
Section 4.1.1	Modifications Made to starPad SDK Framework	36
Section 4.2	Components	37
Section 4.2.1	Sketching and Solving GUI	37
Section 4.2.2	Algebraic Rule System	39
Section 4.2.3	Venn Diagram Renderer	42
Section 4.2.4	Solution Viewer GUI	43
CHAPTER 5	USER STUDY	44
Section 5.1	Informal Perceived Usefulness Study	44
Section 5.2	Pilot Test	48
Section 5.3	Experimental Study	49
Section 5.3.1	Subjects and Apparatus	50
Section 5.3.2	Experimental Task	50
Section 5.3.3	Experimental Design and Procedure	51
Section 5.3.4	Results	52
CHAPTER 6	DISCUSSION, FUTURE WORK, AND CONCLUSION	58
Section 6.1	Discussion	58
Section 6.2	Uses For Instructors	59
Section 6.2.1	Whiteboard Replacement	59
Section 6.2.2	Set Problem Generation	60
Section 6.2.3	Grading Set Problems	60
Section 6.3	Future Work	60
Section 6.4	Conclusions	61

APPENDIX A: QUESTIONNAIRE GIVEN TO STUDY SESSION PARTICIPANTS	63
APPENDIX B: PRE-QUESTIONNAIRE GIVEN TO ALL USER STUDY PARTICIPANTS	66
APPENDIX C: POST-QUESTIONNAIRE GIVEN TO EXPERIMENT PARTICIPANTS	68
APPENDIX D: CHEAT SHEET GIVEN TO CONTROL PARTICIPANTS	72
LIST OF REFERENCES	75

LIST OF FIGURES

Figure 3.1	Using SetPad to compare two equivalent set expressions.	7
Figure 3.2	Venn diagram visualization for a selected set expression.	8
Figure 3.3	Sketching an expression and selecting a sub-expression that SetPad visualizes in red.	9
Figure 3.4	Interactively manipulating an expression using multi-touch and pen input. (1. A set expression is sketched. 2. A user touches set A, and drags it without releasing to set B, causing a potential Commutative transformation to appear. 3. A user releases the touch, causing the Commutative transformation to be applied and the expression to be changed and redrawn below.)	11
Figure 3.5	Commutativity applied by dragging a set or expression to another to switch positions.	12
Figure 3.6	Distribution applied by dragging a set or expression into another expression to distribute over.	13
Figure 3.7	Associative parentheses removed by dragging a parenthesis to any operand or operator in the parent expression.	14
Figure 3.8	Expressions with their complement are simplified by dragging the set or expression into its complement.	15
Figure 3.9	Identity expressions simplified by dragging a set or expression into the identity.	16

Figure 3.10	Idempotent expressions are simplified by dragging a set or expression into itself.	17
Figure 3.11	Domination expressions are applied by dragging a set or expression into the universal or empty set.	18
Figure 3.12	Absorption Law applied by dragging a set or expression into the expression that contains itself.	19
Figure 3.13	De Morgan’s Laws are applied by dragging a complement over and into the inner expression.	20
Figure 3.14	Double negations are simplified by dragging the complement into another complement.	21
Figure 3.15	Relative Complement transformations are applied by dragging the relative complementing operand in the other.	22
Figure 3.16	Radial menu options to expand set C	23
Figure 3.17	Expanding expression by double-tapping set C to bring up radial menu and selecting $C \cup \emptyset$	24
Figure 3.18	Expanding an expression via the radial menu and identity laws. . .	25
Figure 3.19	Expanding an expression via the radial menu and idempotent laws.	25
Figure 3.20	Expanding an expression via the radial menu and involution (double negation) law.	26
Figure 3.21	Defining A as a subset of B with resulting changes to the rendering of those sets.	27
Figure 3.22	Transforming $A \cap B$ into set A after defining A as a subset of B . . .	28
Figure 3.23	Antisymmetry is applied by dragging an equality expression into itself.	29
Figure 3.24	Converting to “x element of” notation.	30

Figure 3.25	Entering elements in a set. 1. A user first selects set A, opens the special element entry window, and sketches elements “1 2 4”. 2. User selects set B and repeats the steps to add “2 4 6”. 3. User selects complex expression and opens to see the intersecting elements between A and B.	31
Figure 3.26	The “Solution Viewer” showing a visualization of solution paths and their frequency. (Line thickness indicates the number of students that have taken that step.)	33
Figure 4.1	SetPad Software Architecture.	34
Figure 5.1	SetPad used as an instructional tool for a study session.	45
Figure 5.2	Mean results for study session post-questionnaire survey questions.	47
Figure 5.3	Results for mean time spent working on the problems.	52
Figure 5.4	Number of participants able to correctly solve the problems.	54
Figure 5.5	Mean results for post-questionnaire survey questions of SetPad’s effectiveness in the experiment.	56
Figure 5.6	Mean results for post-questionnaire survey questions of SetPad’s potential effectiveness in a classroom.	57

LIST OF TABLES

Table 5.1	Questions presented to students in the study session post-questionnaire.	46
Table 5.2	The five problems that study participants were asked to solve. . . .	50
Table 5.3	One way ANOVA where tool was the independent variable (pen and paper, or SetPad), and time on task was the dependent variable. . .	53
Table 5.4	Fisher's exact test for experiment vs. control correctness results. . .	54
Table 5.5	Questions presented to participants in experiment post-questionnaire.	55

CHAPTER 1: INTRODUCTION

SetPad is a new pen and touch-based application prototype that aims to foster understanding of discrete math and set relationships. SetPad lets users sketch out set expressions and allows selection via pen and touch input to provide graphical visualization and confirmation via Venn diagrams of the set relationships. SetPad also provides direct interactive manipulation of the set expressions by simply dragging a term into another to apply various laws of set algebra. By using the tool in this manner, it allows a full formal step-by-step proof to be performed for a given expression of arbitrarily chosen sets.

Section 1.1 Motivation

As part of a computer science curriculum, it is important that students gain an understanding of core concepts such as discrete mathematics and set theory. Traditionally, an instructor uses a whiteboard to work out set theory problems while students solve similar problems on their own using traditional pen and paper. However, these methods can be limited because they cannot provide instant feedback while the student or instructor explores a problem and can lead to errors in logic and misunderstanding of the laws of set algebra. Furthermore, traditional methods are limited in workspace and suffer from handwriting legibility issues.

To overcome these limitations, we have developed SetPad. By limiting manipulations to

follow the laws of set algebra, we posit that students will be more likely to explore set proof problems and thus learn why and how expressions can be manipulated to solve the proofs. Additionally, we reinforce the names of the algebraic laws by displaying the applicable laws as the user moves the terms around in an expression. As such, SetPad could replace traditional pen and paper when used by students for exploring and solving set problems.

Additionally, SetPad has strong potential as an instructional tool when combined with a projector. A discrete mathematics instructor can use SetPad instead of the typical dry-erase whiteboard or overhead to show students how to solve set problems. By doing so, instructors gain nearly unlimited workspace, have guaranteed legible expressions, and have each step displaying the applied set algebra law automatically. SetPad also provides a simple interface for instructors to easily create new set proof problems. By starting out with any basic expression, they can use the tool to transform it to find equivalent mathematical relationships with derived expressions for the purposes of generating a step-by-step proof problem.

Lastly, SetPad also assists instructors when their students use the tool in the classroom. Student solutions to a set problem can be stored, aggregated, and distributed to instructors in real time. This aggregation is presented to instructors with a special tree visualization for use in observation and teaching analysis. We believe this ability holds great potential for giving instructors information about where students may be struggling with a particular set problem or in applying a particular set algebra law.

Section 1.2 Statement of Research Question

In creating SetPad, we wanted to explore how we could best develop a system to allow users to explore and solve discrete math set problems when used as both an instructional tool and a student exploration tool. To do so, we have built into the software a large amount of flexibility for users to solve and manipulate set expressions in many different ways. By observing students using this tool and evaluating their experiences, we believe we have obtained some insight into this question.

Section 1.3 Thesis Overview

This thesis is separated into several chapters. Chapter 2 is a discussion of related work on touch and pen-based mathematical systems, especially anything related to algebraic manipulation. Chapter 3 will cover the user interface elements of SetPad, and Chapter 4 will cover its system architecture. Chapter 5 will present both an informal user evaluation as well as a formal user study that we performed to evaluate how SetPad's effectiveness in exploring and solving set proof problems. Finally, Chapter 6 will present a discussion of the potential current applications of SetPad with its current feature set, some possible future directions for the tool, and concludes with a summary of our findings.

CHAPTER 2: RELATED WORK

There are several areas of work using pen and touch interfaces to create a digital workspace for exploring mathematics concepts. Hands-On Math [15] is a recent effort that uses a digital workspace (in this case a Microsoft Surface) to allow freeform sketching and visualizing of simple algebraic expressions. Thimbleby and Thimbleby developed a simple calculator that combines stylus and touch input for solving simple algebra problems [14]. Like SetPad, these systems allow direct touch manipulation to transform expressions in an effort to explore the solution. SetPad differs in its gestures to manipulate the expressions, as well as its focus on set visualization through Venn diagrams and set algebra.

MathPad² [8] also presents a pen and gesture-based UI for sketching mathematical expressions. It features the ability to visualize and manipulate the graphical representation of the mathematics. VectorPad [3] lets users write down vector mathematics and presents animations illustrating different vector operations. Although these systems are similar to SetPad, they do not support set expression manipulation or visualization or try to provide an understanding of the application of mathematical laws (i.e., set algebra laws in SetPad). Other systems have been developed for entering mathematics using a pen-based interface but they also are not focused on set expression entry and manipulation or designed to assist in discrete mathematics instruction [7, 9, 16]. Commercially, AlgebraTouch [13] is a touch-based math application available on the App

Store for the iPhone. Instead of starting with an open workspace, AlgebraTouch attempts to teach basic algebra by working problem by problem with instructions on how to manipulate the expression using touch gestures. Although it does not use sets in any of its problems and only works with basic algebraic laws like commutativity, the animation and gestures it presents are visually appealing and intuitive.

SetPad also differs from these works in that it combines the use of a pen-based sketch system together with multi-touch capabilities. Combining these two approaches into one system was explored by Hinckley, Pahud, and Buxton [6] where they advocated a "division of labor between pen and touch where the pen writes, touch manipulates, and the combination of pen+touch yields new tools." SetPad follows these principles by using the pen mostly for writing new expressions, and focusing on using the multi-touch interface to manipulate the expressions or for workspace maneuvering. SetPad did not however combine both systems together for any gestures for the sake of simplicity, but it certainly could be pursued in future work.

With regard to other high-level logic software, both Tarski's World [1] and Fitch [2] were developed to allow students to work with first order logic and proofs. Tarski's World provides an environment where students create a 3D graphical representation of a chess-like world, and then must describe that world using first order logic sentences. Fitch is a tool used to enter and build formal proofs that, like SetPad, gives instant feedback to the user when working through each step of the proof and marks steps that do not check out properly. While similar in that regard, neither of these applications is specialized or capable in solving and manipulating the set proof problems like SetPad.

In terms of software tools specifically for assisting students and instructors with discrete

mathematics, there has been little reported work in the literature [12]. One of the few examples was a software system coupled with Hall and O'Donnell's discrete math textbook [5]. This software was text-based and similar to a computer algebra system, which is in direct contrast to SetPad's pen-and-paper style interface. To the best of our knowledge, SetPad is the first application that supports discrete mathematics instruction, in particular set theory, using pen and multi-touch input.

CHAPTER 3: USER INTERFACE

Section 3.1 Expression Entry and Visualization

In SetPad, users start with a blank open workspace where they can begin to sketch an expression in either set notation (e.g., “ $A \cap B$ ”) or element of notation (e.g., “ $x \in A$ ”). As they sketch out expressions, the tool renders a Venn diagram graphic in real-time displaying the relationship between the sets described by the expression (See Figure 3.1).

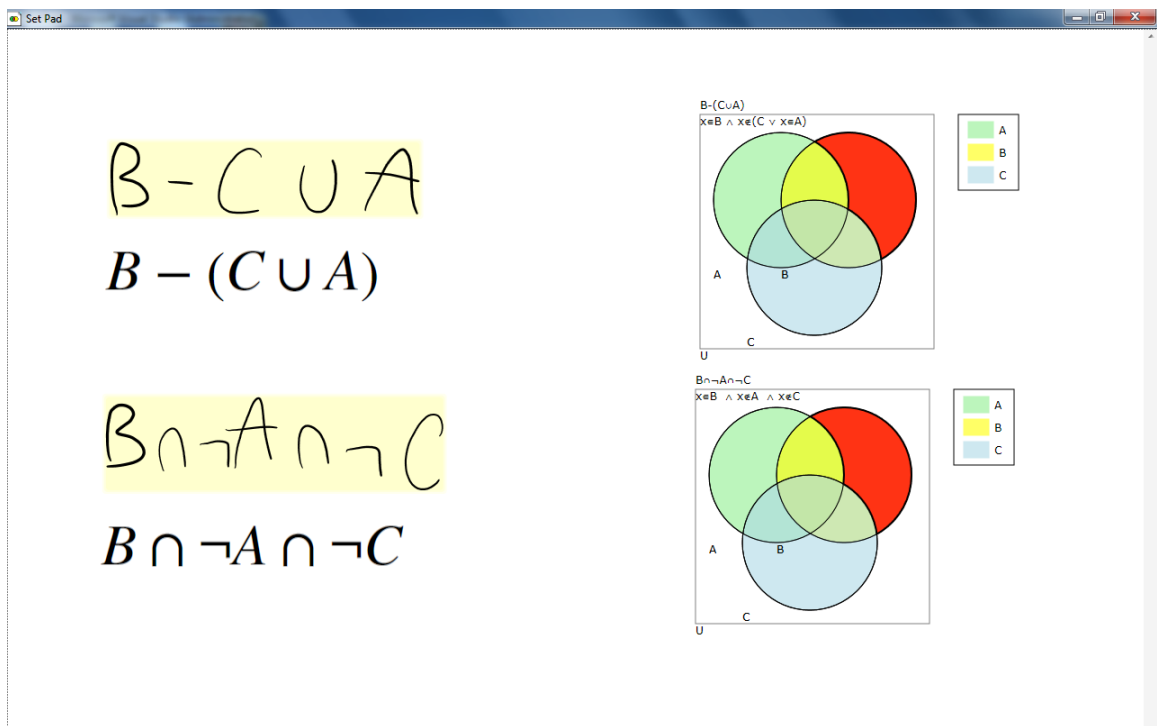


Figure 3.1: Using SetPad to compare two equivalent set expressions.

SetPad also displays information in the rendered graphic area using the original expression notation they have sketched as well as alternate notation (i.e., set notation as well as “x element of” notation) (See Figure 3.2). This alternate notation provides additional information to aid in understanding what the expression is describing.

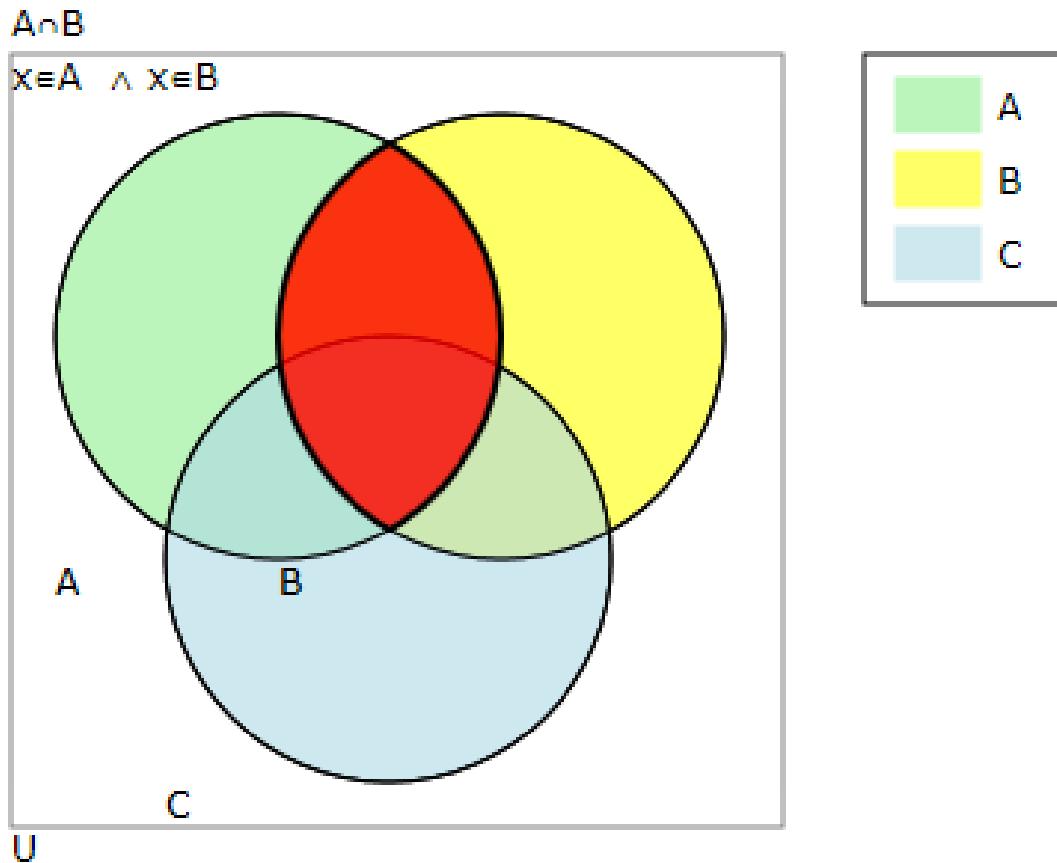


Figure 3.2: Venn diagram visualization for a selected set expression.

Additionally, after the expression is recognized by the starPad framework, users can choose to tap on any part of the expression to see just that sub-expression displayed in the rendering area (See Figure 3.3).

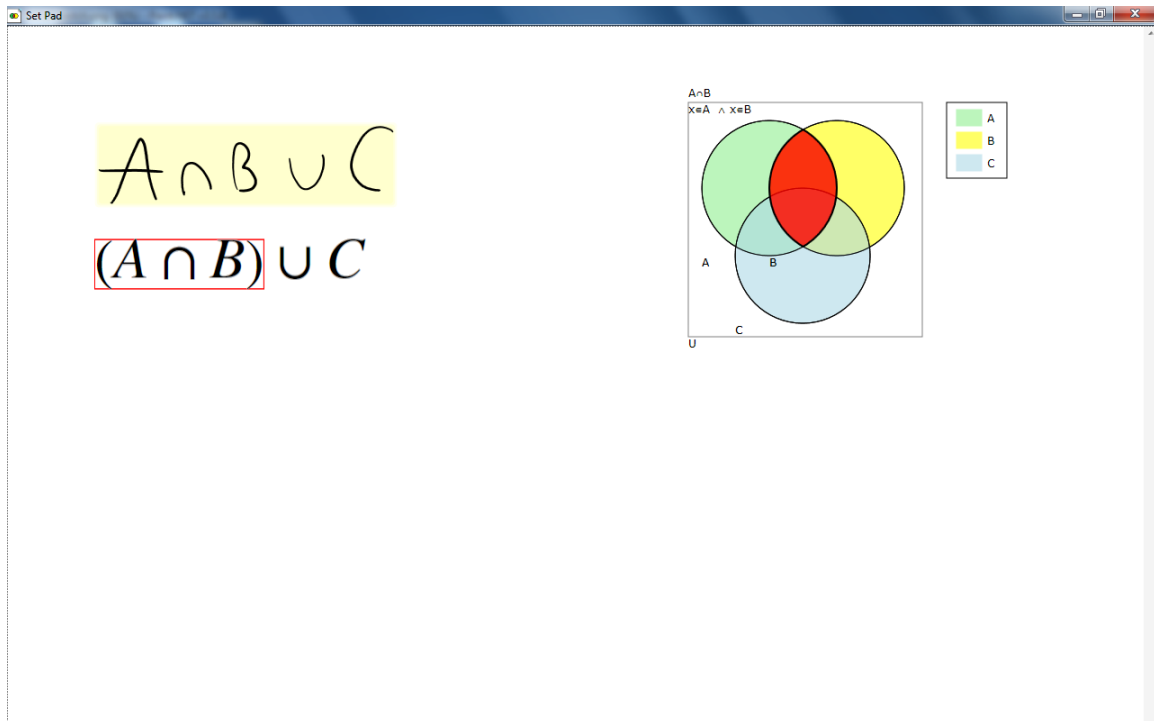


Figure 3.3: Sketching an expression and selecting a sub-expression that SetPad visualizes in red.

Once a student or instructor has completed sketching the initial expression, they can sketch below that recognized expression what they think the next step would be in a proof. By using the Venn diagram visualization, they can verify equivalence between the two expressions (as shown in Figure 3.1). Alternatively, they can use the interactive algebraic law manipulation feature described in Section 3.2.

Section 3.2 Interactive Algebraic Law Manipulation

One of the key features of SetPad is that students and instructors can explore set expressions and set algebra, without fearing that they have violated any set algebra laws. By dragging expressions around with the pen or finger, the tool displays what valid

operations are allowed, alongside the law that applies to make that operation valid. By displaying the law that would apply, along with the input expression and resulting expression, we believe SetPad reinforces to the student when certain transformations can take place and why, even if the operation is not the correct next step or the one they eventually decide to use.

If the user lifts the pen or touch for a valid move, SetPad applies that law to the expression and moves on to the next step below in the workspace while also showing what law was applied to get the expression to the new form (See Figure 3.4). The user can then proceed to manipulate the new active expression, or choose to sketch the next step using the pen.

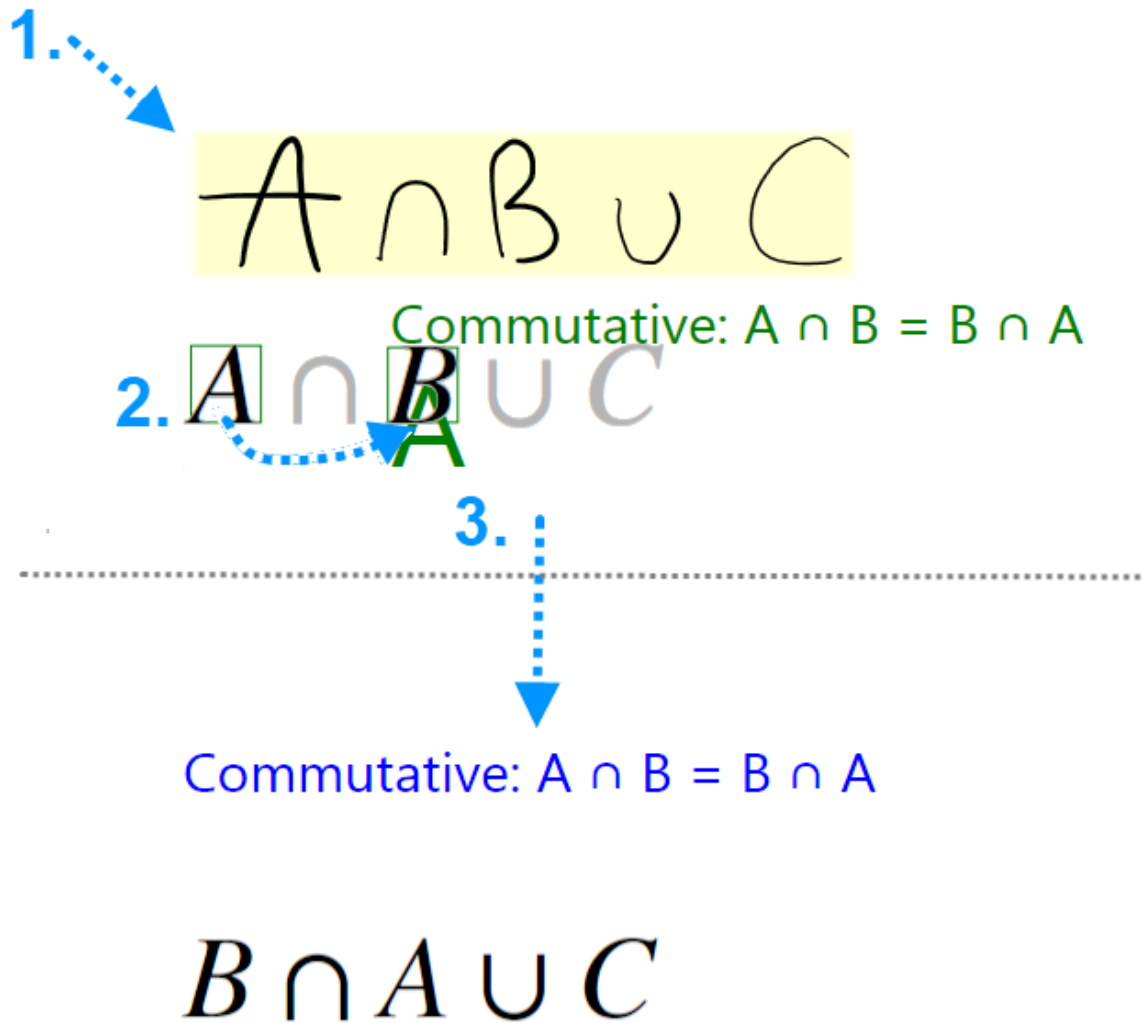


Figure 3.4: Interactively manipulating an expression using multi-touch and pen input. (1. A set expression is sketched. 2. A user touches set A, and drags it without releasing to set B, causing a potential Commutative transformation to appear. 3. A user releases the touch, causing the Commutative transformation to be applied and the expression to be changed and redrawn below.)

Based on feedback given during initial pilot testing, the interactive gestures were designed to be very simple using just one touch or pen down, and then typically dragging one operand into another. Additionally, pilot test participants requested that gestures work

bi-directionally, that is, that they could either drag operand 1 into operand 2 or operand 2 into operand 1 with the same results.

Currently SetPad recognizes and implements the following laws of set algebra:

Commutative Laws

$A \cap B \cup C$
 Commutative: $A \cap B = B \cap A$
 $(A \cap B) \cup C$

Commutative: $A \cap B = B \cap A$

$(B \cap A) \cup C$

$(B \cap A) \cup C$
 $(x \in B \wedge x \in A) \vee x \in C$

A
 B
 C
 U

Figure 3.5: Commutativity applied by dragging a set or expression to another to switch positions.

A user can initiate the law of commutativity by dragging one expression or set into another within the same parent expression. If the law is applied, the operands switch positions with each other.

Distributive Laws

Set Pad

$A \cup B \cap C$

Distributive: $A \cup (B \cap C) = A \cup B \cap A \cup C$

$A \cup (B \cap C)$

Distributive: $A \cup (B \cap C) = A \cup B \cap A \cup C$

Distributive: $A \cup B \cap A \cup C = A \cup (B \cap C)$

$(A \cup B) \cap (A \cup C)$

$(x \in A \vee x \in B) \wedge (x \in A \vee x \in C)$

Legend: A (green), B (yellow), C (blue)

Figure 3.6: Distribution applied by dragging a set or expression into another expression to distribute over.

A user can initiate the distributive law by dragging an expression over the operator of the expression it is to be distributed over. If the law is applied, the operand is distributed in the resulting two sub expressions.

A reverse distribution can be performed by the user dragging a common factor out from either sub expression back into the distributing operator. When applied, the expression is transformed back to the original simplified expression. Originally, we enforced that the common factors had to be the first terms in both sub expressions (as explicitly written in the law) but discovered that pilot test user's found the extra step of moving via commutative unnecessary and tedious.

Associative Laws

Set Pad

$(A \cap B) \cap C$

Associative: $(A \cap B) \cap C = A \cap B \cap C$

$(A \cap B) \cap C$

Associative: $(A \cap B) \cap C = A \cap B \cap C$

$A \cap B \cap C$

$A \cap B \cap C$
 $x \in A \wedge x \in B \wedge x \in C$

A
B
C
U

Figure 3.7: Associative parentheses removed by dragging a parenthesis to any operand or operator in the parent expression.

A user can initiate the law of associativity to remove parentheses in an expression by dragging any parenthesis into any operand operator within that associative parent expression. Because a single parenthesis can sometimes be difficult for user's to touch and drag due to its small size, we also support dragging the entire grouped expression into the parent's operator.

If the law is applied, all associative parentheses are removed from the expression. Initially we experimented with dragging a specific parenthesis around to expand or contract a grouping, but we observed from pilot testing that user's were generally confused by this method and preferred to just quickly remove the entire associative grouping altogether to

solve the proofs.

Complement Laws

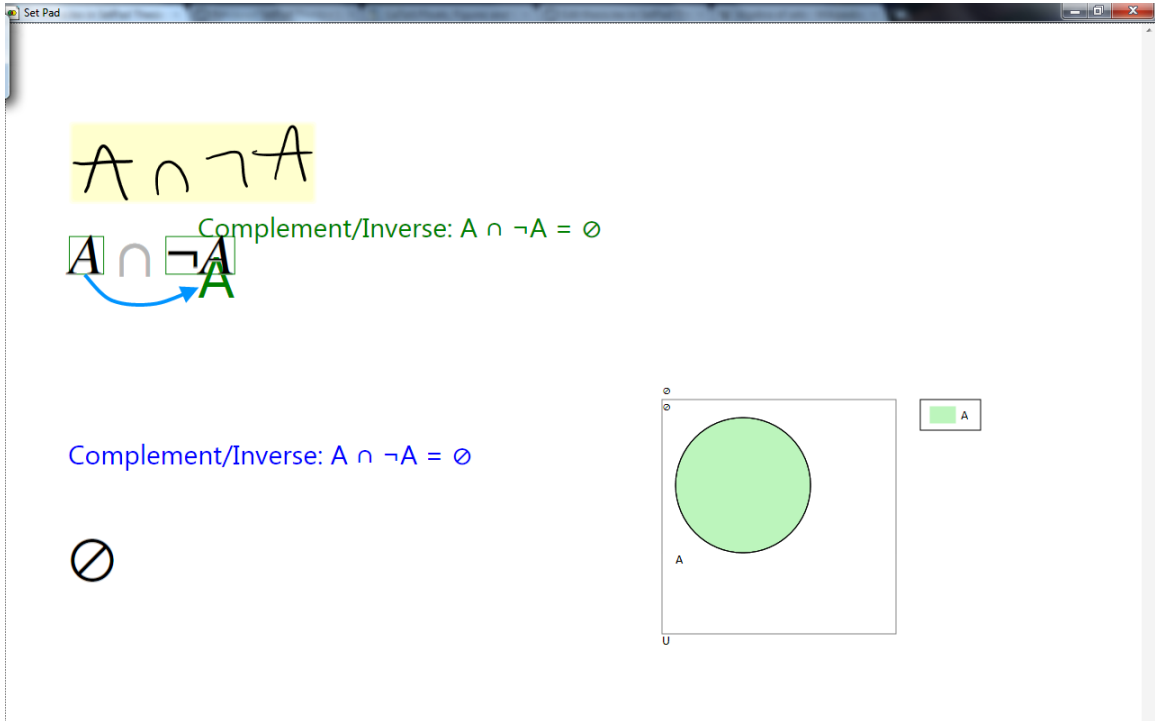


Figure 3.8: Expressions with their complement are simplified by dragging the set or expression into its complement.

A user can simplify a set that is unioned or intersected with its complement by dragging one into the other. When applied, it will transform the expression into either the empty or universal set depending on which operator joined the two.

Identity Laws

The screenshot shows a window titled "Set Pad" with the following content:

- At the top, the expression $A \cap U$ is written in yellow.
- Below it, the text "Identity: $A \cap U = A$ " is displayed in green.
- Underneath, the expression $A \cap U$ is shown with each term in a separate box. A blue arrow points from the A box to the U box, and another blue arrow points from the U box to the A box.
- To the right of this is a Venn diagram consisting of a square labeled U at the bottom and A at the top. Inside the square is a red circle labeled A . To the right of the square is a legend with a green square and the letter A .
- Below the Venn diagram, the text "Identity: $A \cup \emptyset = A$ " is written in blue.
- At the bottom left, the letter A is enclosed in a pink box.

Figure 3.9: Identity expressions simplified by dragging a set or expression into the identity.

Identities can be simplified by the user dragging the operand into the identity, or the identity into the operand. Identities recognized are intersections into the universal set and unions with the empty set.

Idempotent Laws

The screenshot shows a window titled "Set Pad" with a white background. At the top left, the text "A ∩ A" is written in black, with a yellow highlight behind it. Below this, the text "Idempotent: A ∩ A = A" is written in green. Further down, the text "A ∩ A" is written in black, with a blue arrow pointing from the first "A" to the second "A". Below this, the text "Idempotent: A ∩ A = A" is written in blue. At the bottom left, the letter "A" is written in black. On the right side, there is a Venn diagram consisting of a square labeled "U" at the bottom left corner, containing a red circle. The top left corner of the square is labeled "A" and "x ∈ A". To the right of the square is a small green square labeled "A".

Figure 3.10: Idempotent expressions are simplified by dragging a set or expression into itself.

Idempotents can be simplified by the user dragging the operand into the matching second operand within the same sub expression. Both union and intersection idempotents are recognized.

Domination Laws

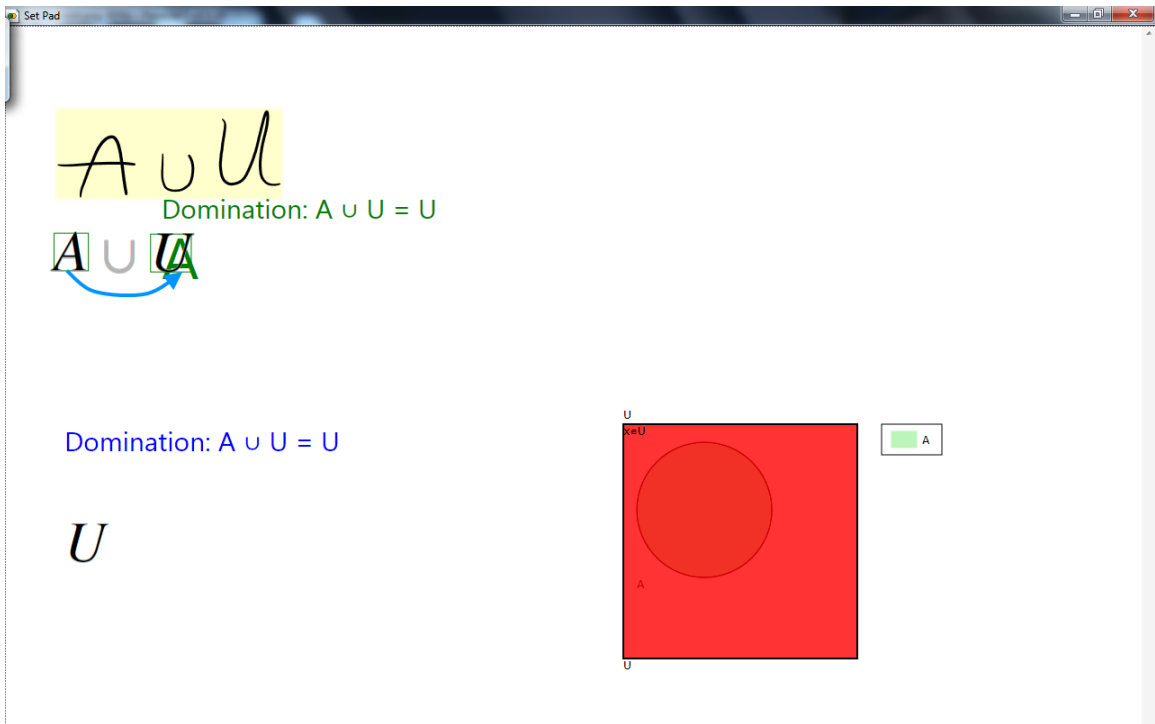


Figure 3.11: Domination expressions are applied by dragging a set or expression into the universal or empty set.

Dominations can be simplified by the user dragging the operand into the dominating value within the same sub expression, or the dominating value into the operand. SetPad recognizes both intersections dominated by the empty set and unions dominated by the universal set.

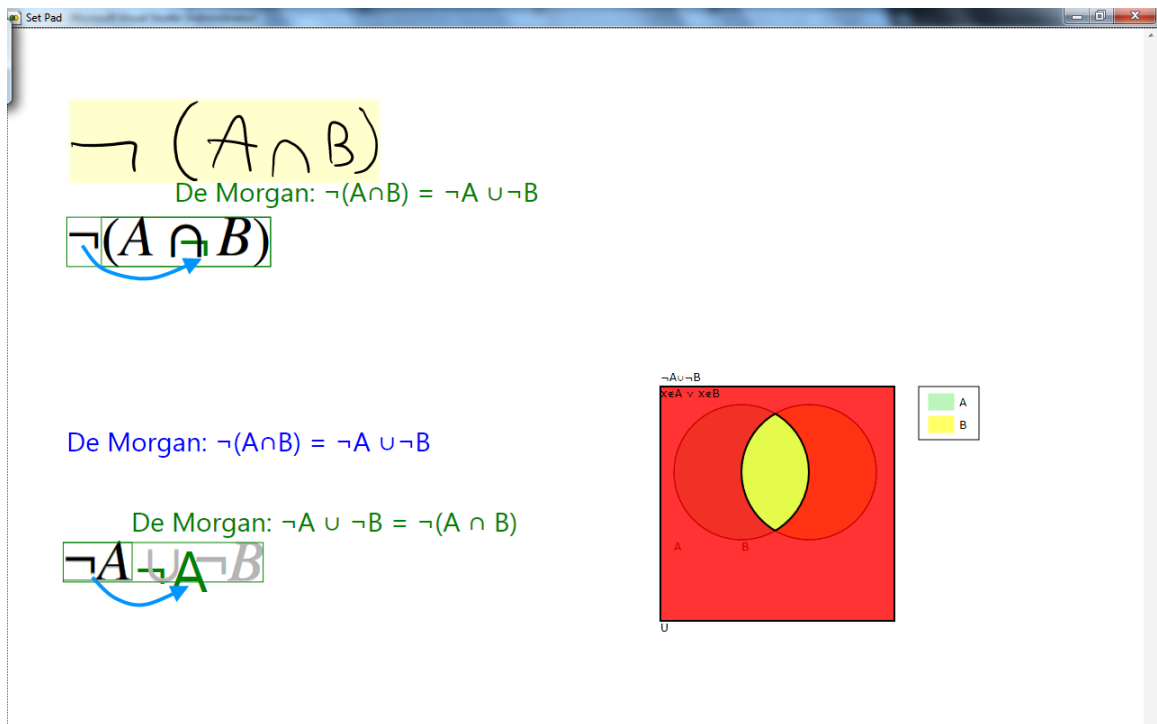
Absorption Laws

The screenshot shows a window titled "Set Pad" with a text area containing the expression $A \cup (A \cap B)$. A green arrow points from the A in the second term to the A in the first term. Below the expression, the text "Absorption: $A \cup (A \cap B) = A$ " is displayed. To the right, there are two Venn diagrams. The top diagram shows two overlapping circles, A (green) and B (yellow), with their intersection shaded red. The bottom diagram shows the same two circles, but the entire area of circle A is shaded red. A legend to the right of each diagram identifies A as green and B as yellow.

Figure 3.12: Absorption Law applied by dragging a set or expression into the expression that contains itself.

Absorption can be performed by dragging the set operand into the adjacent expression containing that same set. When applied, the expression is simplified to become just the single set. Based on user request, we also added support to drag the set operand into the second matching set within the adjacent expression to apply the law.

De Morgan's Laws



The screenshot shows a window titled "Set Pad" containing a text editor with the following content:

- The expression $\neg(A \cap B)$ is highlighted in yellow.
- Below it, the text "De Morgan: $\neg(A \cap B) = \neg A \cup \neg B$ " is displayed in green.
- A second expression $\neg(A \cap B)$ is shown with a green box around the \cap operator. A blue arrow points from the \neg symbol to the \cap operator.
- Further down, the text "De Morgan: $\neg(A \cap B) = \neg A \cup \neg B$ " is shown in blue.
- Below that, another expression $\neg A \cup \neg B$ is shown with a green box around the \cup operator. A blue arrow points from the \neg symbol to the \cup operator.
- Next to the text is a Venn diagram with two overlapping circles, A and B, inside a universal set U. The area outside both circles is red. The intersection of A and B is highlighted in yellow. A legend to the right shows a green square for A and a yellow square for B.

Figure 3.13: De Morgan's Laws are applied by dragging a complement over and into the inner expression.

De Morgan's Laws can be initiated by dragging a complement over the operator of the expression it is to be distributed over. If the law is applied, the complement is distributed in the resulting two sub expressions.

Inversely, a reverse De Morgan transformation can be performed by the user dragging the common complement out from either sub expression back into the distributing operator.

When applied, the expression is transformed back to the original simplified complementing expression.

Involution Law (Double Negation)

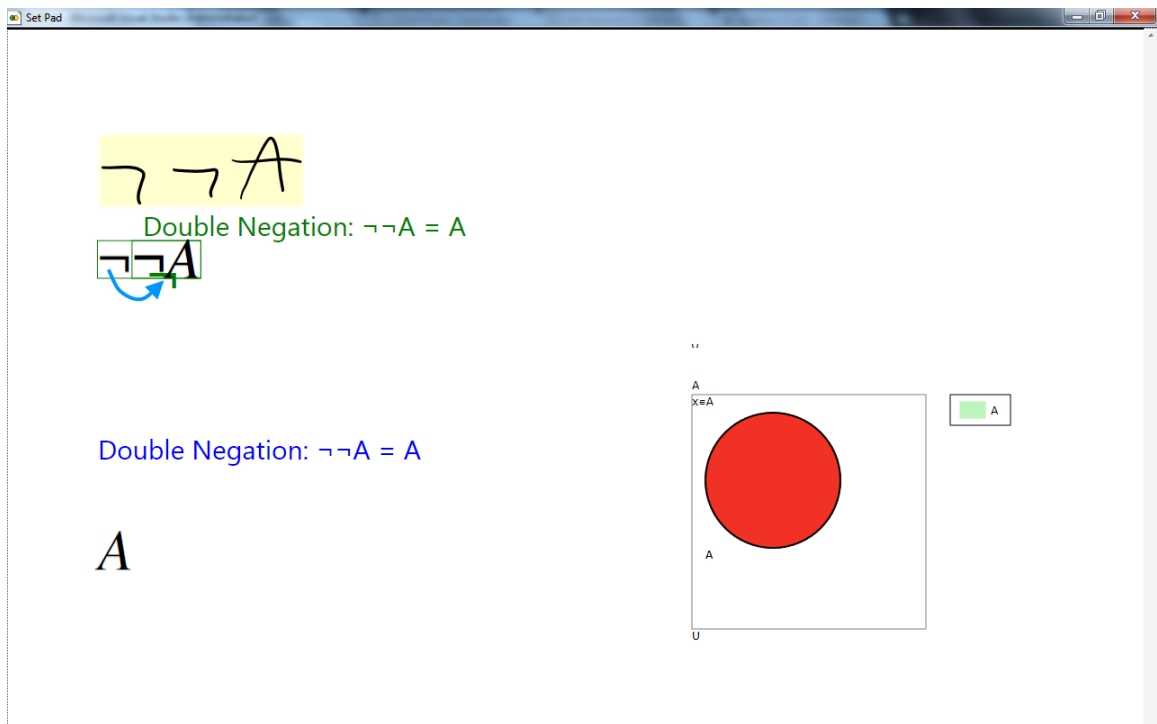


Figure 3.14: Double negations are simplified by dragging the complement into another complement.

The Involution Law (Double Negation) is enacted when user drags the one complement into the neighboring complement. When applied, the complements cancel each other out leaving the resulting expression.

Relative Complements

Set Pad

$A - B$

Relative Complement: $A - B = A \cap \neg B$

$A - B$

Relative Complement: $A - B = A \cap \neg B$

$A \cap \neg B$

Relative Complement: $A \cap \neg B = A - B$

$A \cap \neg B$

$x \in A \wedge x \notin B$

A

B

U

Figure 3.15: Relative Complement transformations are applied by dragging the relative complementing operand in the other.

Users can swap between the two definitions of relative complement (i.e., $A - B$ vs. $A \cap \overline{B}$) by dragging the first operand into the second operand in the relative complement expression. Only after the expression is in the intersection form can the other laws of algebra be applied.

Section 3.3 Expanding Simple Expressions

Because a proof problem can be solved by starting out with either the left or right hand side of the equation, we believe it is important to be able to expand out a simple

expression via identity and other rules. For example, a simple set A is equivalent to $A \cup \emptyset$ via the law of set identities. To support user expansion of the set, we have created a radial menu invoked by double-tapping any set or sub-expression. This action brings up a context sensitive radial menu that when an item is selected allows expanding the current expression based on laws of identity, idempotent, and involution (double negation) as shown in Figure 3.16.

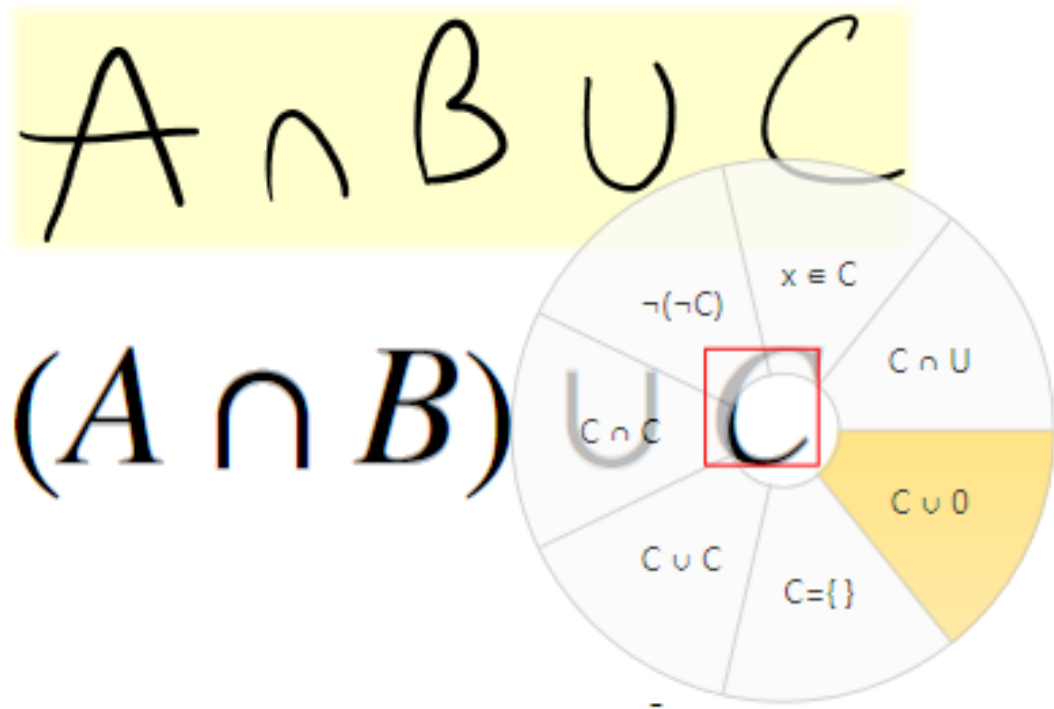
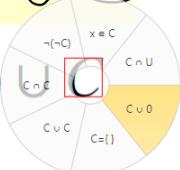


Figure 3.16: Radial menu options to expand set C .

Set Pad

$A \cap B \cup C$

$(A \cap B) \cup C$



$x \in C$

$\neg(\neg C)$

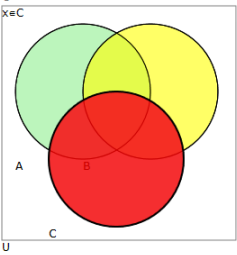
$C \cap U$

$C \cap \emptyset$

$C \cup C$

$C \cup \emptyset$

$C = \{\}$



C

$x \in C$

A

B

C

U

A

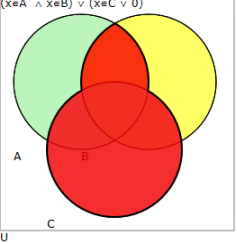
B

C

Identity: $C = C \cup \emptyset$

$(A \cap B) \cup (C \cup \emptyset)$

$(x \in A \wedge x \in B) \vee (x \in C \vee \emptyset)$



$(A \cap B) \cup (C \cup \emptyset)$

$(x \in A \wedge x \in B) \vee (x \in C \vee \emptyset)$

A

B

C

U

A

B

C

Figure 3.17: Expanding expression by double-tapping set C to bring up radial menu and selecting $C \cup \emptyset$.

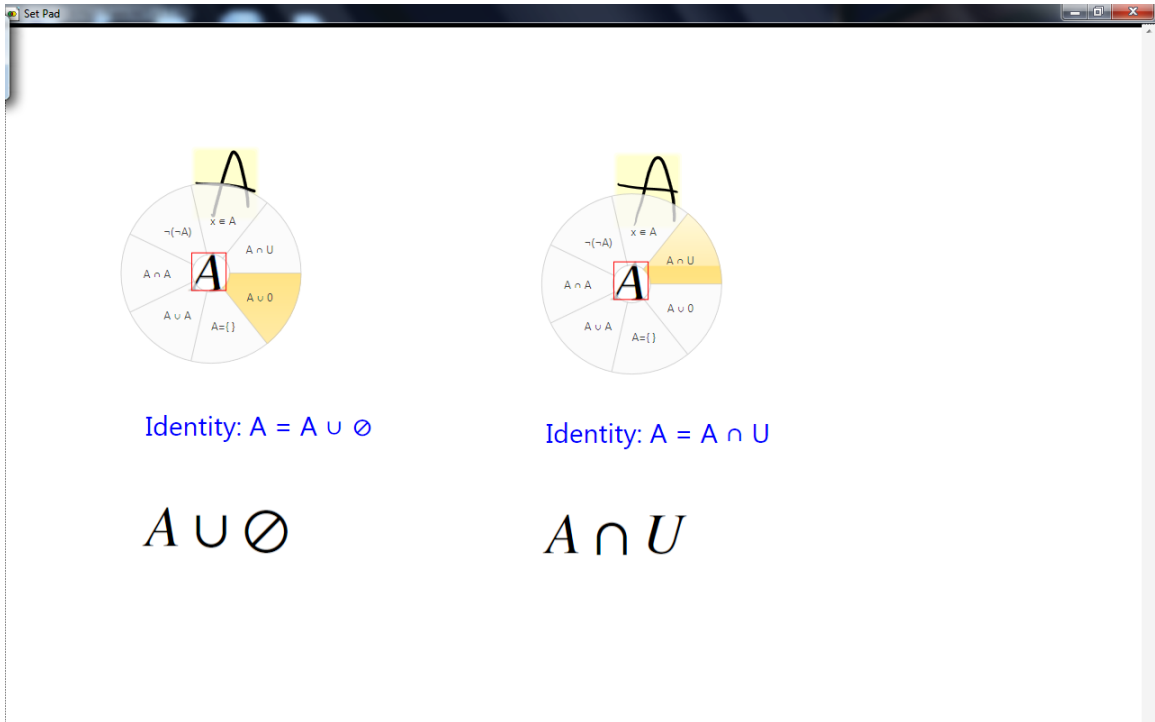


Figure 3.18: Expanding an expression via the radial menu and identity laws.

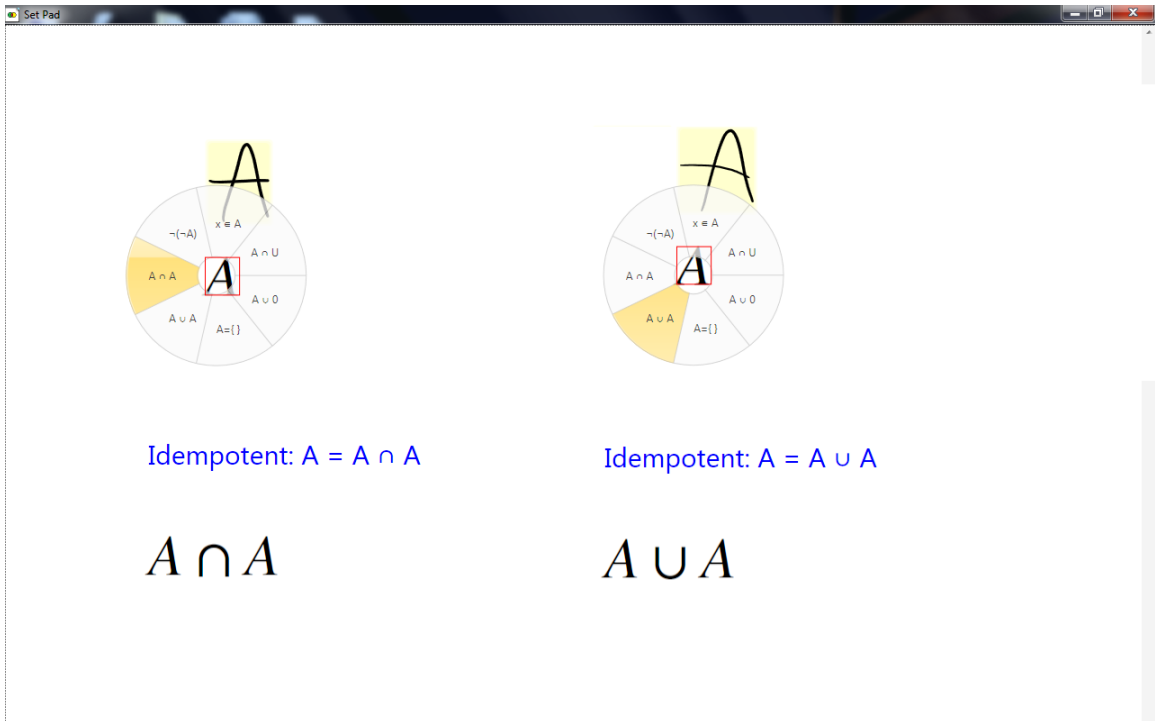


Figure 3.19: Expanding an expression via the radial menu and idempotent laws.

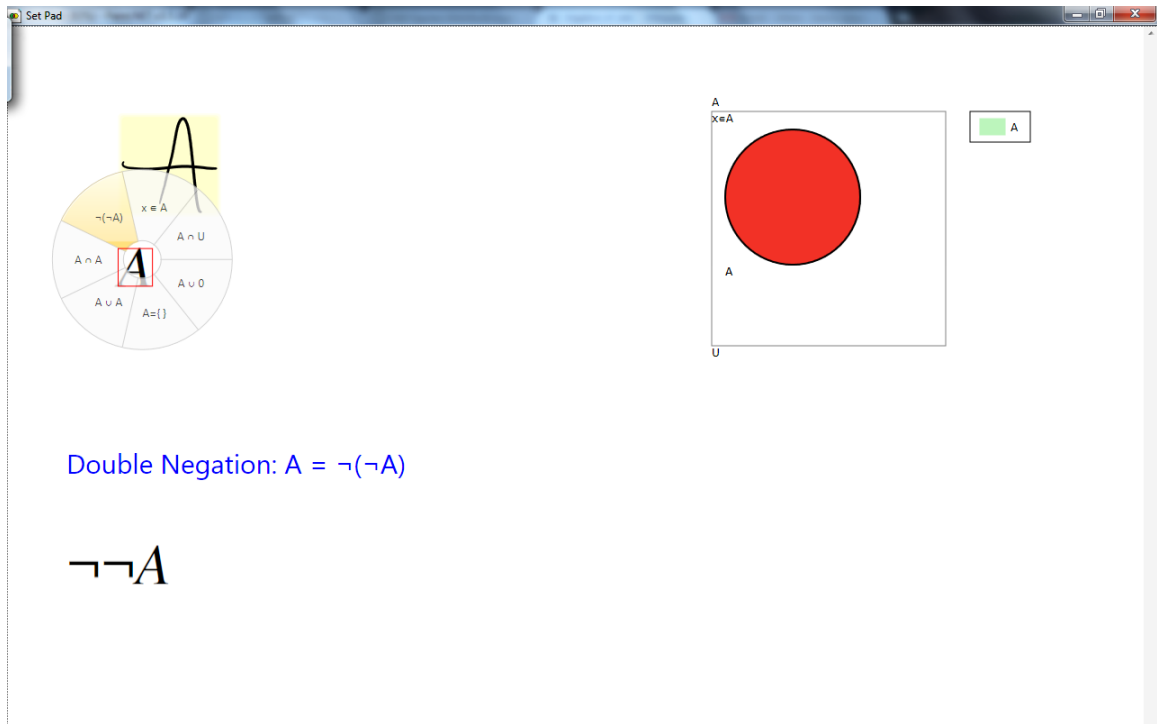


Figure 3.20: Expanding an expression via the radial menu and involution (double negation) law.

Section 3.4 Defining Subsets

Users of SetPad are also capable of defining that one set is a subset of another. To do so, a user simply needs to sketch the expression showing that the two sets are related via the subset of symbol. After doing this, SetPad changes how that set is rendered in the Venn Diagram when in relation to the set that it is a subset of as shown in Figure 3.21.

Moreover, once a set is defined as a subset of another, SetPad allows the user to perform several transformations to be made allowing simplification of expressions that are necessary for certain proof problems.

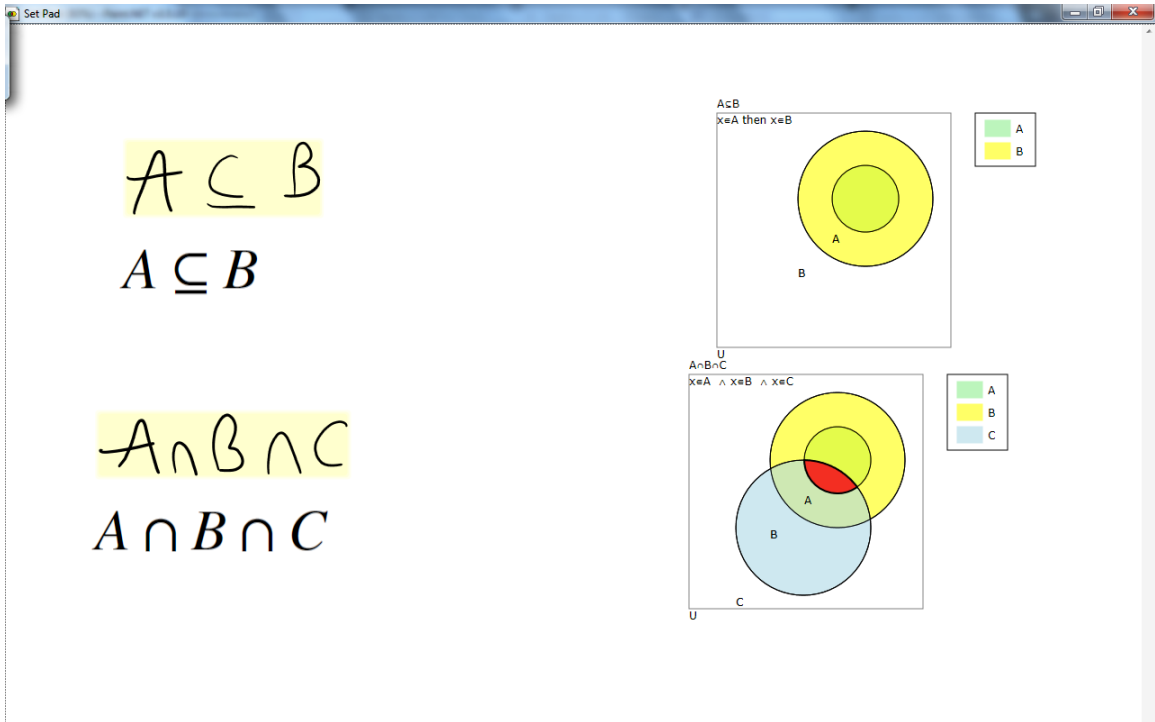


Figure 3.21: Defining A as a subset of B with resulting changes to the rendering of those sets.

Set Pad

$A \subseteq B$

$A \subseteq B$

$A \cap B \cap C$

Implied by Subset: $A \cap B = A$

$A \cap B \cap C$

Implied by Subset: $A \cap B = A$

$A \cap C$

The image shows a screenshot of a software window titled "Set Pad". On the left side, there is a list of mathematical expressions: $A \subseteq B$ (highlighted in yellow), $A \subseteq B$, $A \cap B \cap C$ (highlighted in yellow), "Implied by Subset: $A \cap B = A$ ", $A \cap B \cap C$ (with a blue arrow pointing from the A in the first term to the A in the second term), "Implied by Subset: $A \cap B = A$ ", and $A \cap C$. On the right side, there are two Venn diagrams. The top diagram is titled "A ⊆ B" and "x ∈ A then x ∈ B". It shows a large yellow circle labeled "B" containing a smaller green circle labeled "A". A legend to the right shows a green square for "A" and a yellow square for "B". The bottom diagram is titled "A ∩ C" and "x ∈ A ∧ x ∈ C". It shows three overlapping circles: a green circle "A", a yellow circle "B", and a blue circle "C". The intersection of "A" and "B" is shaded red, and the intersection of "A" and "C" is shaded green. A legend to the right shows a green square for "A", a yellow square for "B", and a blue square for "C".

Figure 3.22: Transforming $A \cap B$ into set A after defining A as a subset of B .

Antisymmetry

SetPad also allows users to define subsets via the property of antisymmetry which allows a set equation to transform into equivalent subset definitions as shown in Figure 3.23.

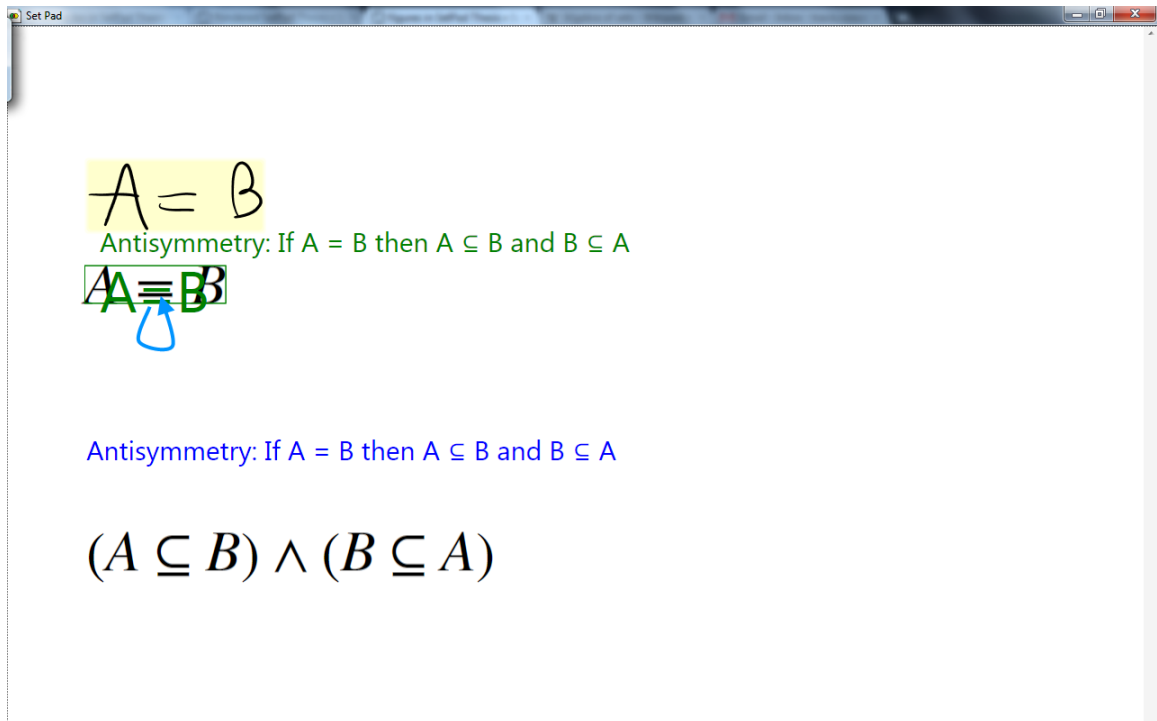


Figure 3.23: Antisymmetry is applied by dragging an equality expression into itself.

Section 3.5 X Element Of Notation

During early evaluations, several students suggested that SetPad should support the current standard set notation as well as Set Builder notation (i.e., “x is an element of”). We built a special radial menu option (e.g., “ $x \in A$ ”) that will convert the selected expression into its logically equivalent Set Builder notation (See Figure 3.24). Once in this notation, SetPad still supports most of the manipulations including commutative, associative, distributive, and the identities. Users can revert back to the standard notation using the same radial menu option.

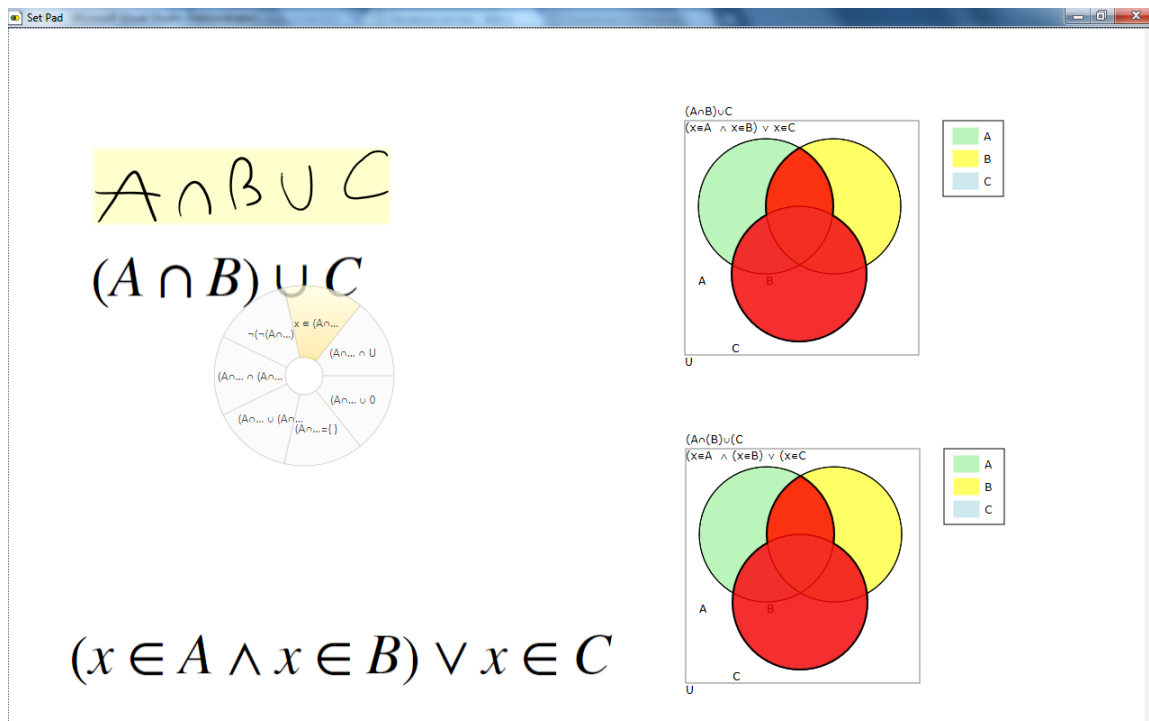


Figure 3.24: Converting to “x element of” notation.

Section 3.6 Elements In Set Problems

Another feature available from the radial menu is the ability to specify the actual elements that exist in a certain set. A user can double-tap on any set to bring up the radial menu, and select the special element entry option (e.g., “ $A = \{ \}$ ”), which causes a new special pen entry window to appear. In the window, a user can sketch numerical values that exist in the set (See Figure 3.25). When they are finished, they simply close the entry window and can continue to other sets. Once all set elements have been entered for each individual set, users can sketch a new complex expression containing those sets, (e.g., “ $A \cap B$ ”). A new window will then properly display the elements that exist in that complex expression based on the set operation that was sketched and the elements that

were previously entered.

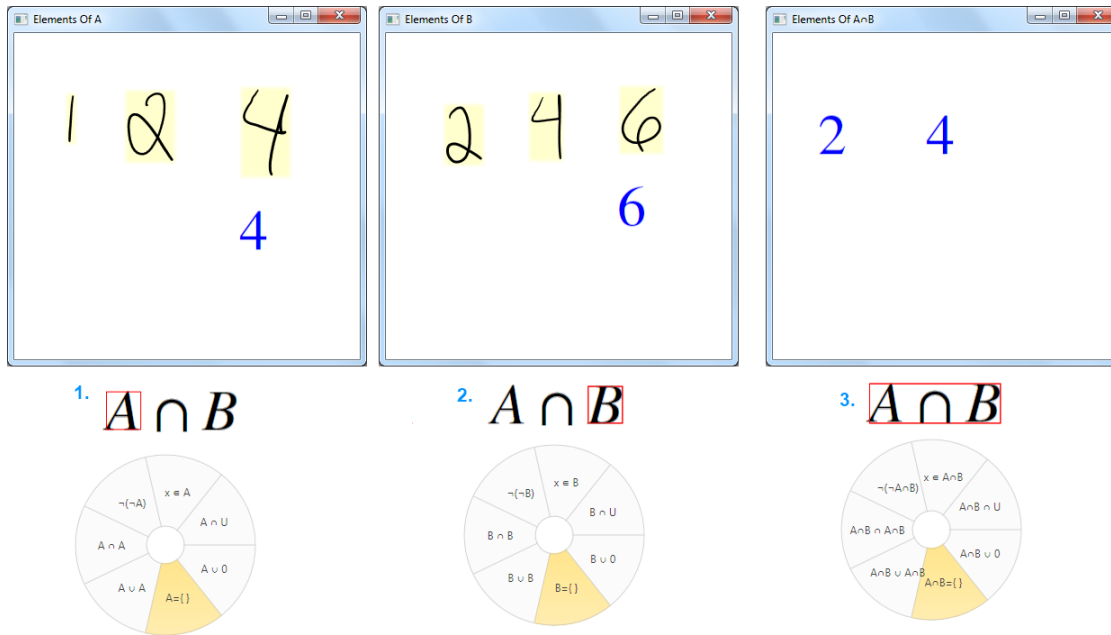


Figure 3.25: Entering elements in a set. 1. A user first selects set A, opens the special element entry window, and sketches elements “1 2 4”. 2. User selects set B and repeats the steps to add “2 4 6”. 3. User selects complex expression and opens to see the intersecting elements between A and B.

Section 3.7 Previous Step Branching

As students are expected to make mistakes and encouraged to explore the set algebra rules, SetPad also provides a means to branch back quickly to a previous step. By scrolling back to a previous step, users can then double-tap on any previous step to branch out and work from that point on. When this occurs, the application pans to the right of the branched step and replaces the current active expression with the selected branched step. Because SetPad allows a near unlimited workspace, this branching provides a considerable benefit over traditional pen and paper or whiteboard.

Section 3.8 Workspace Maneuvering

SetPad provides several basic features to easily and quickly control the workspace as the user moves from step to step of the problem. By using a multi-touch enabled device, users can zoom in and out using familiar pinch-to-zoom controls. The workspace can also be quickly translated in all directions to review previous steps by dragging a single finger on any part of the workspace not displaying an expression.

Section 3.9 Real-time Observation

Another aspect of SetPad is the “Solution Viewer” feature for instructors. This separately running process renders a tree showing all steps attempted thus far to solve a specific problem by any number of students running separate SetPad instances. This lets instructors observe in real-time exactly what steps students are taking to solve the current problem, as well as which solution paths are being taken most frequently (via the thickness of the line connecting the step nodes, as well as a tooltip). We believe that instructors would be able to then infer the key algebraic laws that students may be struggling with, by examining the point in the solution tree where students stray from the accepted solution or fail to proceed.

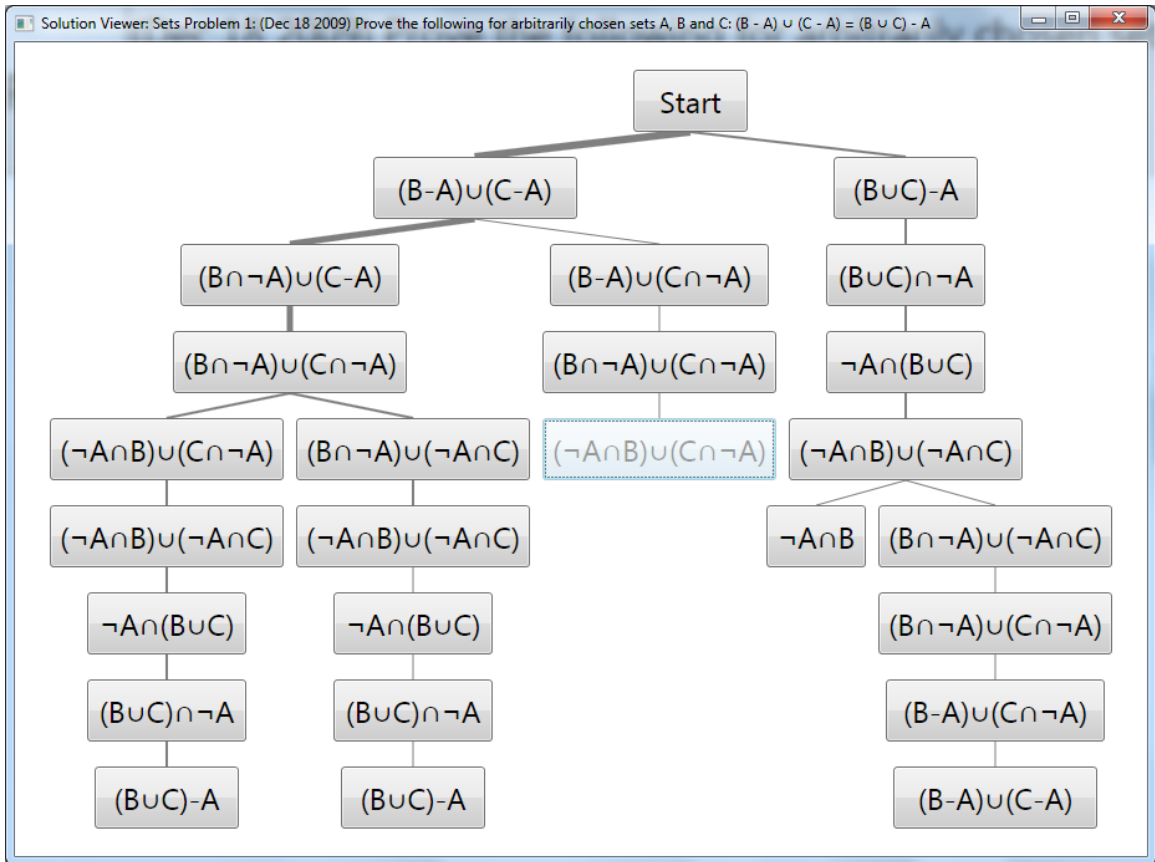


Figure 3.26: The “Solution Viewer” showing a visualization of solution paths and their frequency. (Line thickness indicates the number of students that have taken that step.)

CHAPTER 4: ARCHITECTURE

In this chapter, we present the software architecture used in SetPad including the frameworks that were used and the modifications made to them, followed by the new components we created to implement the system.

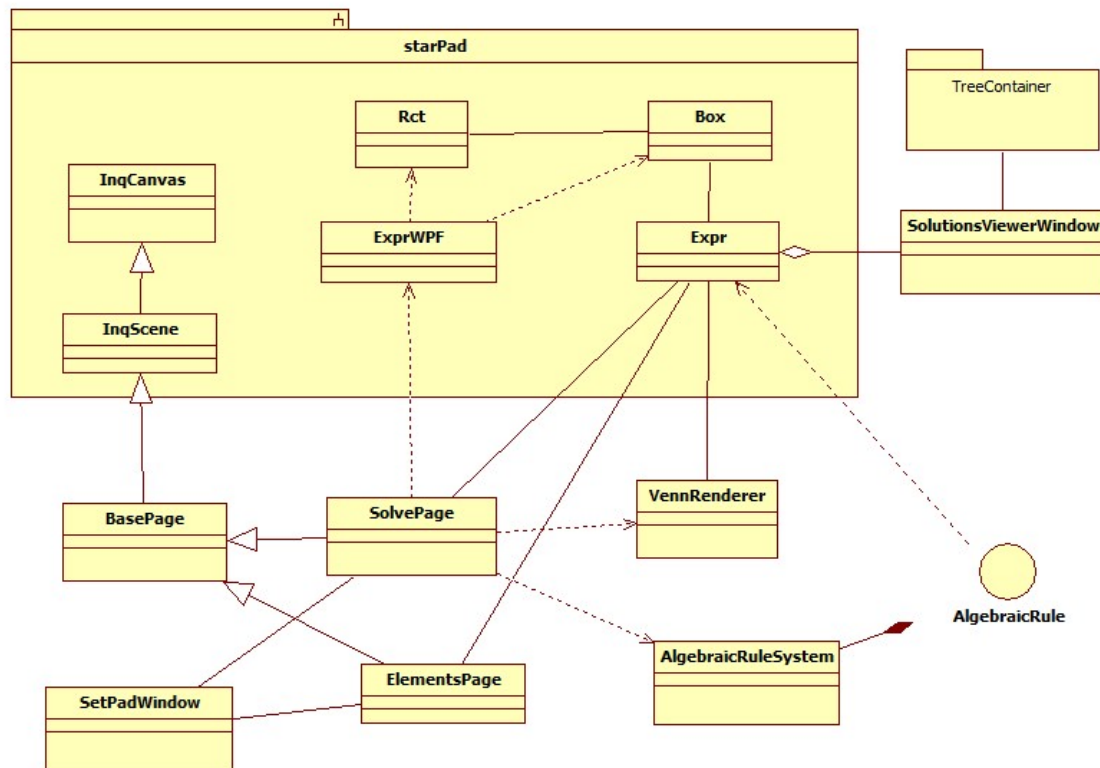


Figure 4.1: SetPad Software Architecture.

Section 4.1 Frameworks

SetPad is built off the starPad SDK 1.3 framework[10], built by the Microsoft Center for Research on Pen-Centric Computing at Brown University. StarPad is a general-purpose framework for Windows written in C# Language and .NET Platform 3.5, using the Windows Presentation Foundation (WPF) 3.5 framework that allows real-time sketch recognition and typesetting including mathematics.

This framework was chosen for its robustness in recognizing and typesetting handwritten mathematical expressions and symbols, as well as its modification flexibility to allow further recognitions and manipulation. SetPad itself is a C# WPF application that runs on any Windows XP or Windows 7 based device that also has pen and touch support.

SetPad heavily relies on starPad's data model classes for mathematical expressions called Expr and CompositeExpr, which are generic data classes that support containing other Expr classes to form a full expression tree. StarPad additionally provides several concrete implementations of Expr that SetPad uses including WellKnownSym to represent our set operators (e.g., intersection and union), and LetterSym that SetPad uses to represent set operands (e.g., set A or set B).

Most of SetPad's GUI is built off the starPad's InqScene object. InqScene is a special C# Canvas that supports pen and touch input and then processes them as ink stroke objects that starPad represents as its own Strok data model class.

Section 4.1.1 Modifications Made to starPad SDK Framework

To fully support the feature set of SetPad, a number of modifications had to be made to the starPad SDK Framework underlying implementation.

Firstly, the InqCanvas object that InqScene extends from was modified to allow multi-touch input to be handled differently than normal pen ink. Without the change, starPad was rendering any touch input like normal pen ink that was undesirable. The class was also extended so it could track multi-touch separately from pen input, and to expose the current number of multi-touch input points that are active at any point. This data was useful in gestures such as “pinch-to-zoom” handling that should only be processed when exactly two multi-touch points are active.

The Expr data model class itself required several changes as well. Most notably we altered the data model to include a unique identifier field called “id” that we could easily use to compare Expr objects for equality and comparison when doing rule logic. Moreover, we changed how starPad displays implicit parentheses rendering based on comments from the informal study session described in Section 5.1. Nearly all the participants in the user session responded that they desired the parentheses to always be drawn for easier understanding.

Most of SetPad’s rule system is based on the ability to properly recognize the associated Expr for any rendered expression that is touched. To do so, we first altered the ExprWPF class (which is responsible for rendering Expr models) to expose their rectangle bounds and associated Boxes class. Then in the Boxes model, we exposed the associated Expr member to be visible. With this information exposed, our application was then able to

take any touch input coordinate, calculate the associated ExprWPF based on its rectangle bounds, and then derive the associated Expr data model.

Lastly, several numerous modifications were made to starPad to change the precedence of recognition such that characters from the domain of set mathematics (i.e., intersection and union operators, null set, universal set, etc.) would be more likely to be recognized than non-set domain typesets.

Section 4.2 Components

SetPad is constructed of four main components: the GUI window and page classes for supporting set expression sketching and manipulating solutions, the Algebraic Rule System for processing the manipulation gestures from the GUI, the Venn diagram graph renderer for displaying set expressions, and the GUI window and page for displaying the Solution Viewer.

Section 4.2.1 Sketching and Solving GUI

The GUI components for sketching and manipulating expressions are comprised mainly of new “Page” classes that all share common inherited behavior from starPad’s base InqScene class. SetPad introduces several of these: the BasePage parent class, the SolvePage which is the main workspace for entering set expressions and manipulations, and the ElementsPage that allows users to input the literal elements that are contained within a set when in the element entry mode.

BasePage

The BasePage class contains all common functionality shared among SetPad's pages.

Notably, this page processes touch gestures as well as changes in touch positions to handle “pinch-to-zoom” gestures that are passed to the parent window to allow panning and scaling the workspace.

SolvePage

The SolvePage class is responsible for managing all input to the workspace when sketching and manipulating set expressions. It takes both pen and touch input, and either passes it to the underlying starPad canvas to recognize, typeset, and render, or passes it to the Algebraic Rule System for further handling. Furthermore, the SolvePage handles when a single expression is selected but not dragged by passing that data to the Venn diagram renderer component for drawing. Lastly, the SolvePage also manages the workspace expression and previous expressions list, such that if the user double taps a previous expression, it requests for the workspace to pan over and branch out to make that user's selected expression to become the new current expression.

ElementsPage

The ElementsPage class is a simple ink canvas associated to an Expr model from the SolvePage that collects ink input from the user, and if recognized as a element value (currently any numeric value), it adds it to a list of elements associated to the parent set Expr model. If the ElementsPage is reopened later by the user, those previous associated

elements are redrawn using starPad's expression renderer (vs. the user's original ink input.) Similarly, if the user opens an ElementsPage for a expression that is the result of two previously entered sets (e.g., if they manually entered elements in set A followed by set B, and then selected to open the ElementsPage for the expression $A \cap B$), SetPad will compute the proper associated elements based upon the Expr's associated set elements and the operator applied in the Expr model.

Section 4.2.2 Algebraic Rule System

All dragged gestures processed by SetPad are fed into a dynamic rule system called the AlgebraicRuleHandler. This class is initialized to construct an ordered list containing concrete implementations of AlgebraicRuleInterface for each of the set algebraic laws that SetPad supports. This interface is defined as:

```
bool AlgebraicRuleInterface::canProcess(  
    CompositeExpr firstTouchedParent, Expr firstExpr,  
    CompositeExpr lastTouchedParent, Expr lastExpr, bool applyChange)
```

When a touch gesture event is recognized by the starPad system, the SolvePage determines which Expr models were interacted with and passes that information to the AlgebraicRuleHandler for processing. It sends as input the Expr model that was touched first by the user (including its parent CompositeExpr), the second Expr that the user is currently touching or has released on (with its parent CompositeExpr), and lastly whether to apply the change (based if the touch gesture was released or is hovering over the expression.) The AlgebraicRuleHandler then passes this information to each

AlgebraicRuleInterface implementation in the ordered list, which in turn tries to process the input. If the rule determines it can process the input, it returns success.

To determine this, each rule implementation looks at exactly which Expr's models were touched in what order, and how they relate together in terms of their parent CompositeExpr's. For example, in the case of determining if the Commutative Law can apply, it is as simple as determining if the firstExpr's parent (firstTouchedParent) is equal to the lastExpr's parent (lastTouchedParent), as all Expr children under a single CompositeExpr parent share the same operator and thus can swap positions commutatively. If the change is going to be applied at that time, the software reconstructs the children for that parent CompositeExpr, but switches the positions in the list for firstExpr and lastExpr when rebuilding.

Similarly, when performing more complex law processing such as the Distributive Law, the software again looks at the Expr models that were touched and their parents. In this instance, SetPad checks that the lastExpr is an instance of a CompositeExpr, it has two or more children Expr, and that its operator is the opposite set operation from the firstTouchedParent's operator. (e.g., In the case: $A \cup (B \cap C)$, A would be the firstExpr, and $B \cap C$ would be the lastExpr that is a CompositeExpr where \cap is an opposite operation vs. firstTouchedParent's \cup .) If the Distributive Law change is applied, the firstTouchedParent is then reconstructed by changing its operator to the opposite operation, followed by rebuilding its list of children by adding a new child Expr for each child of the lastExpr where it is combined with the original firstExpr via the original set operation. (e.g., $A \cup (B \cap C)$ becomes $(A \cup B) \cap (A \cup C)$.)

All other remaining algebraic laws are processed in this manner, using logic that corresponds to each uniquely identifying single stroke gesture described in detail in Section 3.2. When their changes are applied, they also reconstruct the parent Expr by adding, reordering, or removing children Expr. Generally, the identity, idempotent, domination, and complement laws look for the firstExpr or lastExpr to be the proper expected set given the parent operator, and then one child is removed to simplify the expression.

Once any rule returns that it can process the input, the AlgebraicRuleHandler breaks out from any further processing and no other law is considered. This is an intentional design such that the interface would be simple and predictable to use, with just one gesture mapped to no more than one algebraic law. The order of the list determines which law takes precedent over the others if multiple matches were to occur.

Generally there are no conflicts based on the unique design of the gestures, but in the case of a few (e.g., Commutative Law and Distributive Law where both allow dragging an expression from the left side to the right for the same parent), we place precedence over the law that could be not performed via any alternative gesture. (e.g., With $A \cup (B \cap C)$, we can still perform a Commutative swap by starting and dragging the right expression $B \cap C$ onto set A , but the Distributive Law can only be recognized by dragging A over the right expression $B \cap C$ and thus has higher precedence.) This way, each and every algebraic law can at least be uniquely expressed by some single stroke gesture. The current ordering (in order from highest to lowest precedence) is as follows:

- Identity
- Complement

- Idempotent
- Involution
- Domination
- Absorption
- Distributive
- De Morgan
- Relative Complement Swap
- Commutative
- Associative

Section 4.2.3 Venn Diagram Renderer

The Venn diagram renderer component takes an Expr model as input, processes it, and then renders it on the parent Canvas if and only if the Expr contains a valid set math expression. It does this by first parsing the input Expr tree, building a stack of displayable set operands with their operators, followed by building the displayable graphic using C#'s CombinedGeometry display components. CombinedGeometry already supports several operations that correlated directly to the set operations including: Union, Intersect, Xor, and Exclude. Lastly, the renderer component adds standard form labels and rectangles to construct a legend to display to the user.

Section 4.2.4 Solution Viewer GUI

The Solution Viewer GUI is composed of a WPF Window containing a special custom tree rendering GUI component. This window can be opened from within SetPad to view the user's current progress on a problem, or can be ran as a separate process by an instructor to render a tree showing multiple student's solution attempts via reading a shared distributed file. The rendering of the solution tree is done via a slightly modified version of the TreeContainer WPF library [11], with each node in the tree associated to a starPad Expr object. The thickness of the lines connecting each node is adjusted to reflect the number of attempts (i.e., students) that went down the branch of the tree. Hovering over a node reveals detailed information including the algebraic law that was applied to get to that step.

CHAPTER 5: USER STUDY

Section 5.1 Informal Perceived Usefulness Study

For its initial evaluation, we performed an informal perceived usefulness evaluation of SetPad as an instructional tool. As part of the University of Central Florida's undergraduate computer science program, students must pass a foundation exam proving their understanding of several core computer science concepts. One portion of this exam is a set theory section that many students have historically struggled to solve correctly.

We held a study session for set problems prior to one of these exams that was open and announced to all undergraduate students taking the exam. We used SetPad along with a projector to display and demonstrate the solutions to 10 different set proof problems that we selected (half of which were from previous foundation exam questions, the other half generated using the tool and from textbooks), while allowing open feedback from the students during the session about the solution process and tool.

For this evaluation, SetPad was modified to display problem text at the top of the application for the sample problems, and large simple touch sensitive buttons were added to move on to the next problem when the solution was finished.

Set Pad: Problem 1: (Dec 18 2009) Prove the following for arbitrarily chosen sets A, B and C: $(B - A) \cup (C - A) = (B \cup C) - A$

Prev Next

$$(B - A) \cup (C - A)$$

$$(B - A) \cup (C - A)$$

$(B - A) \cup (C - A)$
 $(x \in B \wedge x \notin A) \vee (x \in C \wedge x \notin A)$

Figure 5.1: SetPad used as an instructional tool for a study session.

Afterwards, a questionnaire based on [4] (see Table 5.1) using a seven-point Likert scale (1 equals strongly disagree and 7 equals strongly agree) was given to each student asking them to evaluate the tool as both an instructional tool (Q1-Q5), as well as how useful they thought the tool would have been as a student in a discrete mathematics course (Q6-Q8). A total of 17 student questionnaires were collected from the study session.

Table 5.1: Questions presented to students in the study session post-questionnaire.

Perceived Usefulness as an Instructional Tool	
Q1	SetPad would have helped me learn the material in a Discrete Mathematics course when used by an instructor.
Q2	SetPad would help me understand how each step in a set proof is derived and what law is applied.
Q3	The displaying of real-time diagrams of the set expressions would help me understand set relationships better.
Q4	SetPad would help me better understand how elements are contained within and shared between sets.
Q5	Using SetPad in a study session has better prepared me for the Foundation Exam.
Perceived Usefulness as a Student Tool	
Q6	SetPad would have helped me learn the material in a collegiate Discrete Mathematics course when used as a student.
Q7	SetPad would enable me to complete my homework assignments more quickly.
Q8	The interactive multi-touch interface of SetPad would help me explore what laws are possible in transforming a set expression.

Figure 5.2 shows the mean responses from the questionnaire with the all responses generally positive toward SetPad. All but one of the students responded positively about the tool having benefits as both an instructor’s tool as well as a student’s tool. The lone response that did not agree with both still indicated that SetPad would be good to use as an instructor’s tool, but not as a student tool. That respondent thought that students would become “too dependent” on the tool. A few other students similarly remarked that they also worried SetPad could become a “crutch” for them as they worked on homework problems and ultimately exam problems.

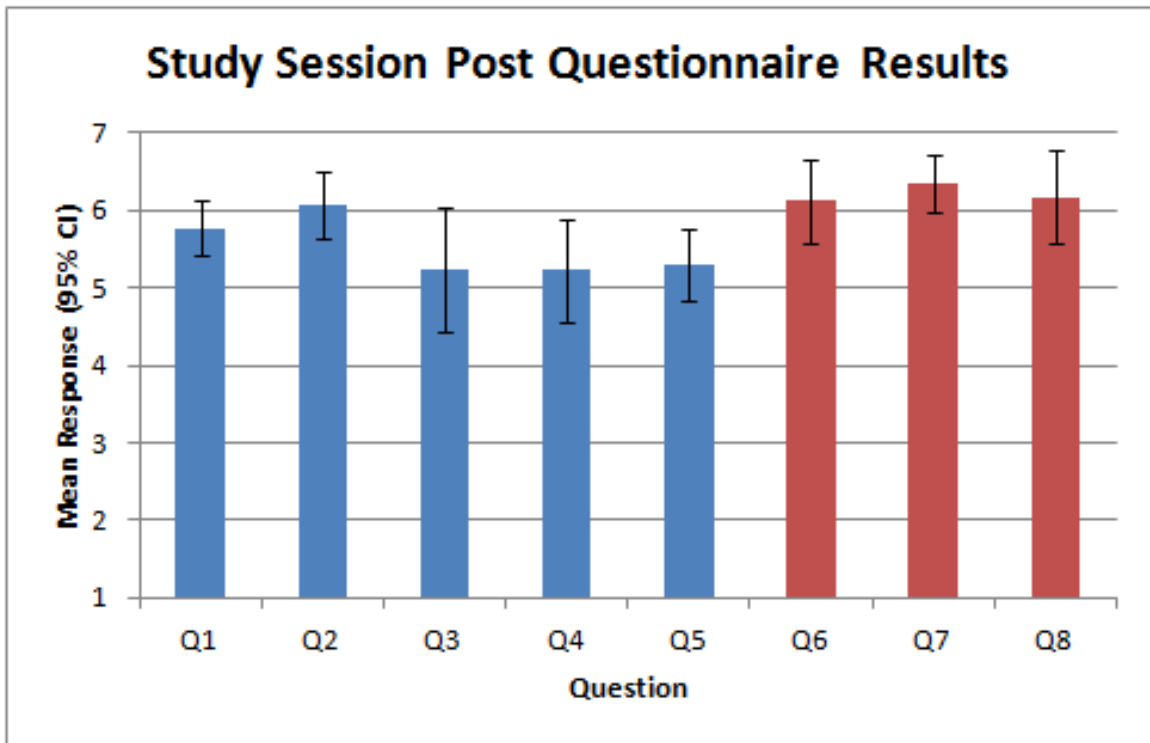


Figure 5.2: Mean results for study session post-questionnaire survey questions.

Nonetheless, most of the students commented that they liked the “interactivity” of solving set problems and the “playful set relations” one could do by exploring with the tool.

Moreover, many saw potential for the tool if used by instructors in a discrete mathematics course. Some students mentioned that they had trouble in their own discrete mathematics course in regards to reading the instructor’s handwriting on the whiteboard, as well as not remembering or hearing what law was applied to get to the next step as they struggled to keep up. Several of the students suggested we add a visual overview of the steps taken for a problem thus far, similar to the instructor’s “Solution Viewer” that was only briefly demonstrated. An overwhelming majority of the respondents asked for the tool to explicitly draw parenthesis around all sub expressions in the rendered output, which was later implemented.

Section 5.2 Pilot Test

Prior to the running a formal user study, we ran several pilot test trials using four different participants from the Interactive Systems & User Experience Lab. In the first trial, several observations were made based on the results. Notably, the experiment was originally setup to have the participants first sketch out the set expression in the problem, and then have them manipulate the expression to solve the proof. However, it was discovered very quickly that the underlying starPad framework sketch recognition engine was too inhibitive to those who had never worked with it before, as more than half the time of each trial problem was spent fighting to get the recognizer to properly recognize the starting expression. Because the focus of SetPad and the study was to measure the effectiveness of the new set manipulations and gestures (and not the existing recognition engine of starPad), we decided to instead have participants start out with pre-recognized expressions to start with and manipulate vs. sketching them out themselves. This proved to be a much more streamlined method in the next pilot tests and during the actual experiment. However, it was noted for future work to either replace the core recognizer or improve it by limiting the recognized characters to just the set domain.

Additionally, several other changes came out from observing the pilot test participants behavior that were made to improve the efficiency of the manipulations and gestures. Firstly, all manipulation gestures were changed to become bi-directional, i.e., being able to drag from right-to-left as well, not just left-to-right as the algebraic laws are explicitly written. Moreover, when performing operations like a reverse distribution, the software was modified to allow the common operand to lie anywhere in the sub expressions, not

just the first expression as the law is strictly written. This allowed users to avoid having to make several tedious commutative steps before applying the larger transformation.

Moreover, several of the users in the pilot test struggled with the initial implementation of the association gestures. After trying several methods, SetPad was changed so that dragging either parenthesis to another operand/operator in the associative expression simplified it by removing the parentheses grouping completely. This change led to a much simpler understanding of how the software reacted to associative changes and improved the efficiency of solving the set proof problems. Additionally, as the single parenthesis was sometimes difficult for users to touch and drag, SetPad was modified to allow dragging the entire grouped expression as well into the parent operator to remove the associative grouping.

Lastly during the pilot test, it was decided that all future participants would be required to have taken some level of discrete mathematics course previously, as students without prior experience do not have the knowledge to comfortably solve set theory problems for either the experiment or control tests.

Section 5.3 Experimental Study

To evaluate the effectiveness of SetPad in solving set proof problems, we conducted a formal user study. Specifically, we compared the performance of students solving five set proof problems using the SetPad tool vs. a control group who attempted to solve the same five problems using traditional pen and paper along with a “cheat sheet” of common set algebraic laws.

Section 5.3.1 Subjects and Apparatus

We recruited 20 student volunteers (17 male, 3 female) and divided them into the two groups (control and experiment) such that each group would have approximately the same level of experience and background in discrete mathematics and set theory. Participants ranged in age from 19 to 31 and were currently enrolled in either undergraduate or graduate programs of computer science, computer engineering, electrical engineering, or information technology. The experiment took approximately 0.5 to 1 hours to complete and each participant was paid 10 US dollars for their time.

Our experimental setup consisted of one multi-touch/digital pen capable laptop - an HP TouchSmart PC notebook running Windows 7. Only one person was needed to administer the experiment as a proctor.

Section 5.3.2 Experimental Task

We chose the five representative set proof problems from several collegiate discrete mathematics textbooks and course exams. We selected problems that could be solved via direct proof in order to have a consistent, measurable step-by-step solution and that would exercise most of the laws of set algebra. These problems are listed in Table 5.2.

Table 5.2: The five problems that study participants were asked to solve.

Representative Set Proof Problems	
Problem A	Prove: $(B - A) \cup (C - A) = (B \cup C) - A$
Problem B	Prove: $(A - C) \cap (C - B) = \emptyset$
Problem C	Prove: $A - (C \cup B) = (A - B) - C$
Problem D	Prove: $(A \cap B)$ and AB are disjoint. (Sets are disjoint if their intersection is \emptyset)
Problem E	Prove: $A - (A \cap \overline{B}) = A \cap B$

Section 5.3.3 Experimental Design and Procedure

We used a between subjects design with a control (pen and paper) and experimental (SetPad) group. Both experimental and control participants were first given a pre-questionnaire that gauged their knowledge of set theory and proofs, as well as their previous experience using pen-based and multi-touch interfaces.

The experimental group was first trained with SetPad for up to fifteen minutes by working through a number of manipulation exercises on given set expressions. These samples exercised each of the algebraic set laws that would be needed to solve the five set problems. When the participants finished with the training, they began the timed part of the study in which they had to solve the five set problems as quickly as possible. If the participant solved the problem, or was unable to solve the problem within five minutes, they moved on to the next problem. We recorded the time each participant spent as well as correctness for all five problems. If they were unable to correctly solve a problem within the five minute span, we recorded their time as the five minute max (i.e., 300 seconds). Afterwards, participants were given a questionnaire (shown in Table 5.5) using a seven-point Likert scale (1 equals strongly disagree and 7 equals strongly agree) asking them to evaluate the tool's effectiveness in solving the five problems and how effective they believed SetPad would be if it were to be used in a classroom.

Similarly, the control group was asked to solve the same five problems, but were asked to do so on paper using pen or pencil along with a special "cheat sheet" that contained all the algebraic laws that would be needed to solve the problems. Again, participants attempted to solve each problem, or if they were unable to solve it within five minutes,

they moved on to the next problem. We recorded the time each participant spent as well as correctness for all five problems. There was no post-questionnaire given to the control group participants.

Section 5.3.4 Results

From Figure 5.3 and Table 5.3, we observed that the participants who used SetPad spent significantly less time working on the problems than the control group, with the exception of Problem E.

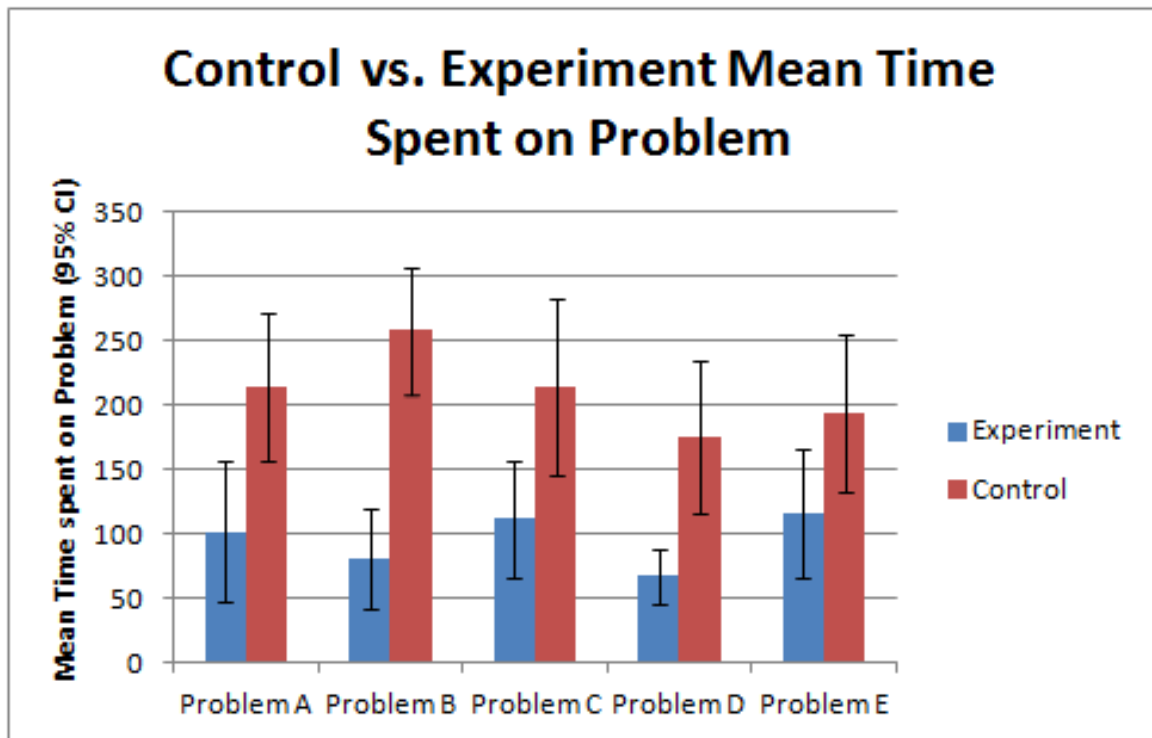


Figure 5.3: Results for mean time spent working on the problems.

Table 5.3: One way ANOVA where tool was the independent variable (pen and paper, or SetPad), and time on task was the dependent variable.

One way ANOVA Results	
Problem A	$F_{1,18} = 7.80, p < 0.05$
Problem B	$F_{1,18} = 31.09, p < 0.05$
Problem C	$F_{1,18} = 5.99, p < 0.05$
Problem D	$F_{1,18} = 11.06, p < 0.05$
Problem E	$F_{1,18} = 3.68, p = 0.07$

We also observed a significant difference in the participant’s abilities to correctly solve the problems between the two groups as shown in Figure 5.4. Despite both groups having similar background and experience with set problems, all but one experiment participant was able to correctly answer all five questions in the allotted time. The control group on the other hand only had a 50% success rate for the first three problems, fared slightly better with 80% completing Problem D, and 70% completing problem E. A Fisher’s exact test as shown in Table 5.4 shows that participants did a significantly better job at answering the problem correctly with SetPad for problems A and B and did no worse for problems C,D, and E.

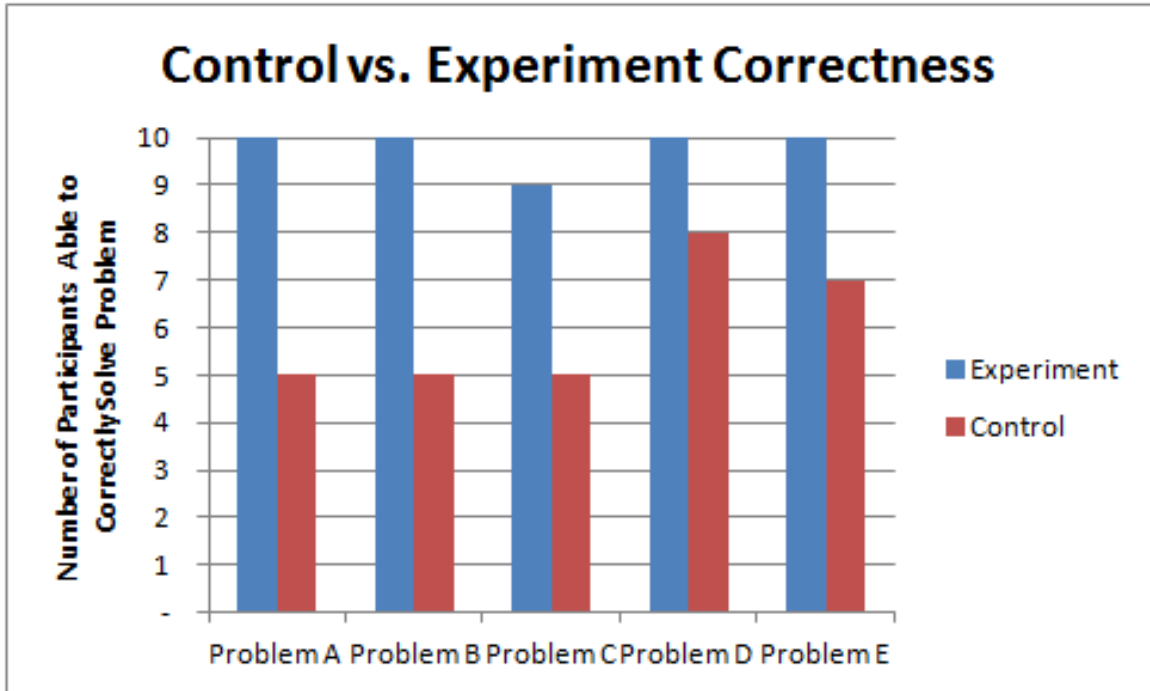


Figure 5.4: Number of participants able to correctly solve the problems.

Table 5.4: Fisher's exact test for experiment vs. control correctness results.

Fisher's Exact Test Results	
Problem A	$p < 0.05$
Problem B	$p < 0.05$
Problem C	$p = 0.14$
Problem D	$p = 0.47$
Problem E	$p = 0.21$

We believe that these results indicate that SetPad is at least as efficient, if not more, in solving set proof problems, and is more effective in helping students discover and find the solutions to complicated algebraic transformations. We witnessed that when the control group failed to solve a problem, it was either due to not knowing where to start (stuck on the first step), or was due to some immediate logic flaw in early steps that led them down a path where they never recovered. If they had a tool to give them immediate feedback, we believe they would have been able to see their mistakes and recover and solve the problem.

Survey Results

Following the timed trial, the 10 participants using SetPad then answered a questionnaire, based on [4] (see Table 5.5) using a seven-point Likert scale.

Table 5.5: Questions presented to participants in experiment post-questionnaire.
SetPad Effectiveness in the Experiment

Q1	SetPad made it easier to solve the set proof problems vs. if you had to solve them by hand with pen and paper.
Q2	The SetPad touch and drag interface was easy to use in manipulating the set expressions.
Q3	The real-time display of valid algebraic laws in SetPad was helpful in solving the problems.
Q4	SetPad was effective at visualizing the set relationships presented in the problems.
Q5	I had enough practice to adequately perform and solve the problems in the experiment.
Q6	The HP TouchSmart PC was appropriate for use in testing of SetPad.
SetPad Effectiveness in a Classroom	
Q7	The discrete mathematics course that I took previously adequately prepared me to solve these types of problems.
Q8	Using SetPad in a collegiate Discrete Mathematics course would have helped me learn and understand the material better.
Q9	SetPad would have helped me to complete my Discrete Mathematics coursework assignments more quickly and correctly.
Q10	SetPad would have allowed me to better explore and learn the set algebra laws it was available to use while taking a Discrete Mathematics course.
Q11	I would become too dependent on using a tool like SetPad in solving problems in a Discrete Mathematics course.

Figure 5.5 shows the mean responses from the questionnaire in regards to SetPad’s effectiveness during the experiment. We see that participants generally favored SetPad as an effective tool to solving the set problems, especially in regards to the “real-time display of algebraic laws.” Some students that had trouble with the “touch and drag interface” suggested we increase the spacing of the symbols to make it easier to select, and to allow the use of the stylus instead of the multi-touch interface. Note, for this experiment we had

our participants solely use the multi-touch interface for simplicity and consistency in the experiment, but SetPad does support the stylus pen to manipulate the expressions.

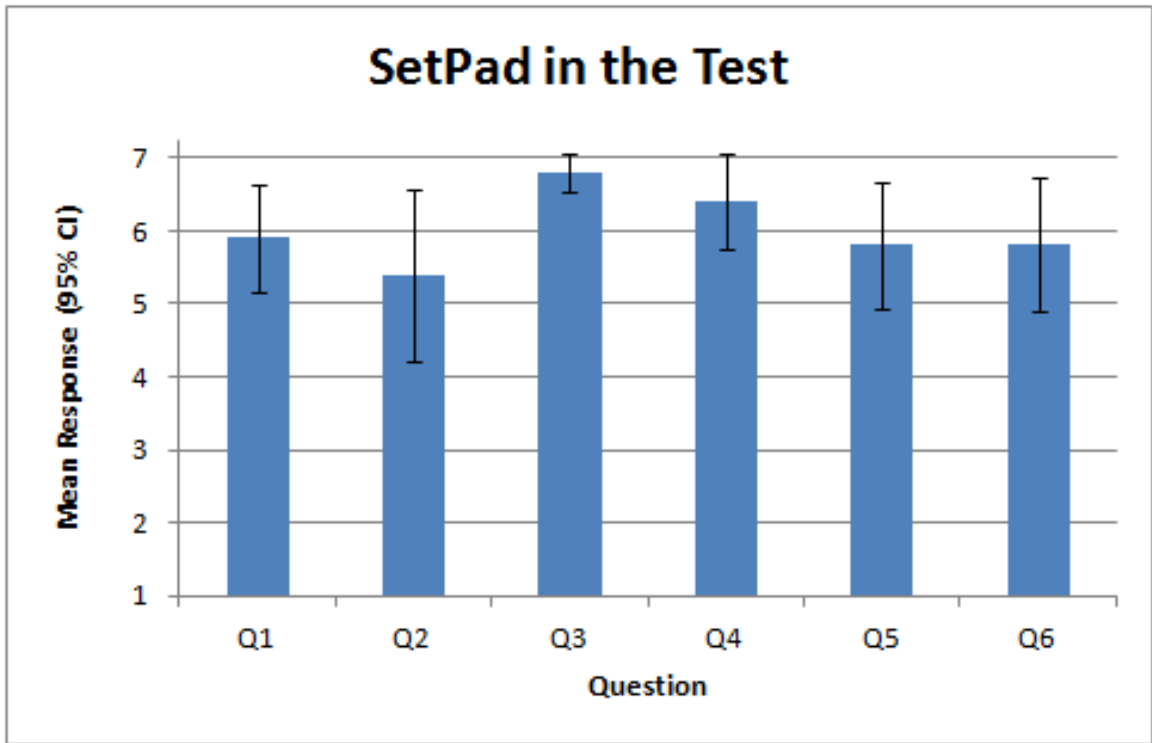


Figure 5.5: Mean results for post-questionnaire survey questions of SetPad’s effectiveness in the experiment.

Since students had the chance to use SetPad, we also asked them whether the tool would be useful in the classroom. Figure 5.6 shows the mean responses from the questionnaire questions that asked about their perception of SetPad’s effectiveness if used in a classroom.

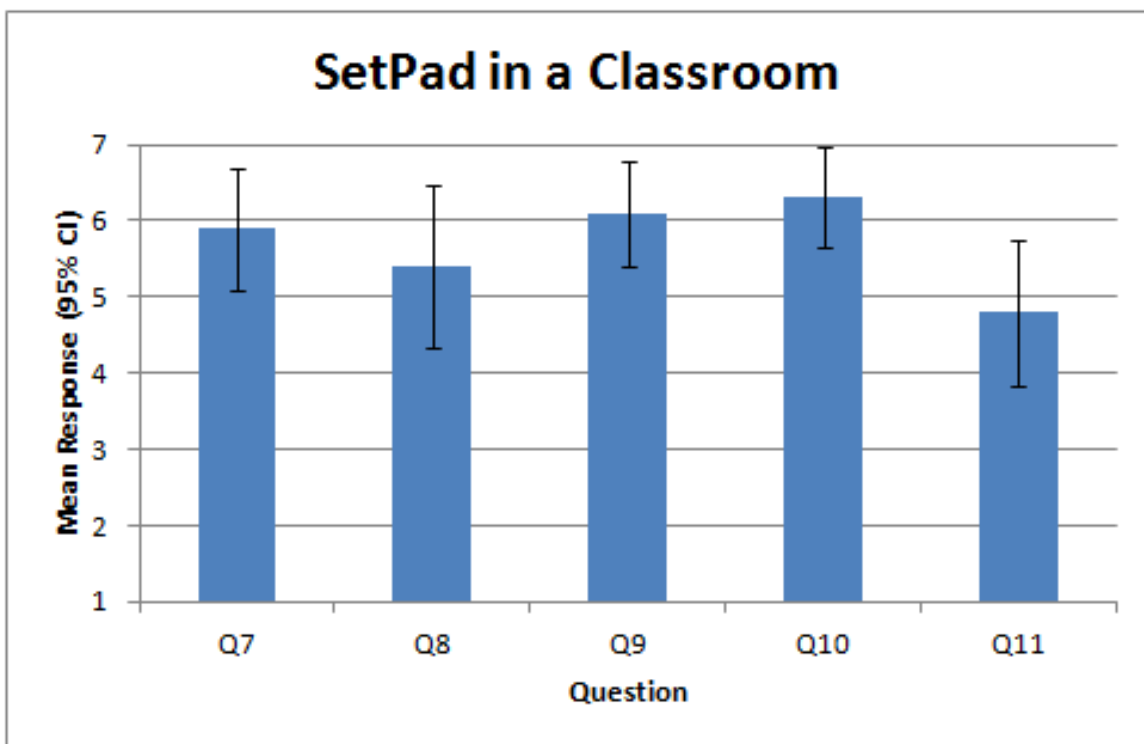


Figure 5.6: Mean results for post-questionnaire survey questions of SetPad's potential effectiveness in a classroom.

Again, participants showed general belief that SetPad would be effective in a classroom, particularly in exploring set laws and using it to aid in their homework. Unsurprisingly, there was some general concern that they might become too dependent on the tool, but opinions were mixed and some students suggested a balanced mix of using the tool and working problems by hand would be optimal.

CHAPTER 6: DISCUSSION, FUTURE WORK, AND CONCLUSION

Section 6.1 Discussion

One criticism of the tool is that SetPad simply addresses a single topic (set theory) from a single course (discrete mathematics.) We believe that most of the insights gained from this research would easily be extended to other applications and subjects. Many areas of mathematics (including traditional algebra, calculus, logic and inference problems, etc.) could easily benefit by presenting students with a digital sketch-based workspace that provides instant feedback as students solve problems in logical steps. Moreover, by being purely digital and step-based as such, the “Solution Viewer” concept could be easily implemented for those subjects as well. Further research is needed in expanding the concepts of SetPad into other branches of mathematics, perhaps even using the rules system that SetPad developed as a common general-purpose framework for future tools.

Lastly, during the user study we observed that students were able to quickly learn and pick up the gestures for solving set theory proof problems. With just a limited amount of training, their solution times were faster than traditional pen and paper, and they achieved a much higher success rate for correctly solving the problems. We believe that if deployed in an actual discrete mathematics course, SetPad could be effective in improving

student understanding and performance, as students would be exposed to even longer training and practice using the tool. A future user study that measured students' performance long term with and without the tool would help to support this claim.

Section 6.2 Uses For Instructors

Section 6.2.1 Whiteboard Replacement

In discrete mathematics courses, instructors typically work through set proof problems on a whiteboard or with transparencies. This method can lead to several problems when helping students understand the solution. Most notably, a whiteboard has limited space to sketch and solve the problem, and requires good handwriting skills so that students can follow the steps. Moreover, when an instructor moves from one step to the next, the law that was applied is usually only spoken aloud, and not written as part of the step. In that case, a student could easily mishear or entirely miss the law that was used for the step.

SetPad alleviates these issues when connected to a projector. By virtue of being a digital workspace, there is nearly unlimited room to sketch and branch out. Handwriting is a non-issue as once the problem is sketched, the system recognizes and typesets the problem for students to clearly read. As the instructor works on the next steps, students can see exactly what terms and sub-expressions are being impacted and changed. Lastly, the law that was applied for the step is displayed prominently above the new expression, such that students have time to read and take notes on what was done.

Section 6.2.2 Set Problem Generation

Because SetPad allows quick manipulation of existing set expressions, instructors can also use the tool to easily generate new problems for students. An instructor starts by sketching a complex expression, and then uses the tool to manipulate it into a different but equivalent form for students to solve in a proof or derivation. Alternatively, by using the Venn diagrams to compare two expressions, an instructor can quickly use the tool to see that these expressions are indeed equivalent.

Section 6.2.3 Grading Set Problems

Another potential use of SetPad is to aid in the grading of set proof problems by instructors and their assistants. As set proof problems may potentially have many correct solutions, it can be difficult to grade these problems consistently. SetPad via its "Solution Viewer" functionality can provide graders a quick view of the acceptable solutions for a problem, which they can use to more fairly grade the student's work. Additionally, if students were to use SetPad to solve the problems vs. pen and paper, the software itself could be extended to grade the students attempt and report the correctness of their solution directly back to the instructor.

Section 6.3 Future Work

SetPad is still a prototype application, and needs further iteration and testing to improve its effectiveness before becoming a full use educational tool deployed in a discrete mathematics classroom. Additionally, there are several features that SetPad could support

to better complement its coverage of the set theory domain of discrete mathematics. Specifically, the tool is heavily focused on direct solutions to set theory proofs (i.e., proving the equivalence of two expressions), and could be expanded with further features to help solve proofs via contradiction, etc.

Another future application of SetPad that was not explored yet is the potential for the tool to discover and illustrate common mistakes that students make. Traditionally with algebra and other step-by-step problem solving, there are many common misconceptions that many students make that lead them astray early on in the problem. SetPad could feasibly learn the common bad paths that a user makes, and notify him or her of those mistakes, possibly in real-time, before or as the mistakes are repeated.

Lastly, the “Solution Viewer” feature concept of the tool is still very preliminary and offers just a glimpse of the possibility of what can be accomplished if SetPad were setup in a distributed network of students. By collecting solution metrics digitally from students in this manner, there exists a plethora of analysis data and applications that could be presented to the educator to improve his or her teaching method and to identify trouble areas in student learning.

Section 6.4 Conclusions

We have presented SetPad, a new tool that lets both computer science students and instructors interactively sketch and explore set problems. Users are able to freely sketch a set expression, and then manipulate that expression using the pen or touch input using recognized gestures to apply the laws of set algebra. The tool provides instant visual

feedback to the user in real-time as they drag the touched expression around, indicating if and what algebraic law can be applied with the current gesture. SetPad allows students to easily convert between both notations of set expressions, as well as input elements that exist in the individual sets.

SetPad also provides functionality for instructors to observe in real-time the steps that students are taking in their solutions to better identify problem areas that need more coverage in the classroom. This feature displays a visual tree of every solution step that the students have tried, and indicates the number of students that have attempted that branch via the thickness of the lines connecting the nodes.

We also conducted two evaluations of SetPad: an informal study session for students taking a computer science foundation exam, and a formal user study experiment for students that have taken a discrete mathematics course. Based on our evaluation in those studies, we believe that the tool has the potential to help future students in understanding how sets relate in expressions in addition to the algebraic rules of sets. The evaluation results also indicate that there is strong potential for SetPad to be an effective complement to the typical pen and paper approach to learning to solve set problems, as it provides instant legible feedback to the user while manipulating the expressions.

Finally, we have identified a number of functional enhancements in which SetPad could be improved, several new similar applications in other subjects, and a potential future long term study to measure SetPad's effectiveness of improving student performance and learning in an actual classroom.

APPENDIX A: QUESTIONNAIRE GIVEN TO STUDY SESSION

PARTICIPANTS

SetPad Study Session Questionnaire
ISUE Lab, Computer Science Department
University of Central Florida

Based upon the presentation of SetPad, answer the following questions about how useful you think it would be.

- Respond to each question with whatever knowledge you have.
- There is no right or wrong answers.
- Be honest and realistic in your assessment.
- State your agreement based on the following scale:

1 2 3 4 5 6 7

Strongly Disagree |-----|-----|-----|-----|-----|-----|**Strongly Agree**

Section 1: Perceived Usefulness as an Instructional Tool

1. SetPad would have helped me learn the material in a Discrete Mathematics course when used by an instructor.

1 2 3 4 5 6 7

2. SetPad would help me understand how each step in a set proof is derived and what law is applied.

1 2 3 4 5 6 7

3. The displaying of real-time diagrams of the set expressions would help me understand set relationships better.

1 2 3 4 5 6 7

4. SetPad would help me better understand how elements are contained within and shared between sets.

1 2 3 4 5 6 7

5. Using SetPad in a study session has better prepared me for the Foundation Exam.

1 2 3 4 5 6 7

Section 2: Perceived Usefulness as an Student Tool

1. SetPad would have helped me learn the material in a collegiate Discrete Mathematics course when used as a student.

1 2 3 4 5 6 7

2. SetPad would enable me to complete my homework assignments more quickly.

1 2 3 4 5 6 7

3. The interactive multi-touch interface of SetPad would help me explore what laws are possible in transforming a set expression.

1 2 3 4 5 6 7

Section 3: Other Comments

1. What changes would you like to see made to SetPad to make it better?

APPENDIX B: PRE-QUESTIONNAIRE GIVEN TO ALL USER
STUDY PARTICIPANTS

Participant Number: _____

Studying the Effectiveness of SetPad for Solving Set Theory Proof Problems
ISUE Lab, Computer Science Department
University of Central Florida
Pre-Questionnaire

Gender (please circle one): Male / Female Date: _____

Age: _____ Major: _____

1. Do you have any experience using a digital pen interface? (e.g., Nintendo DS, pen tablets)
Yes No Not Sure
2. Do you have any experience using a multi-touch interface? (e.g., iPhone, iPad, etc.)
Yes No Not Sure
3. How long has it been since you have taken a discrete mathematics course (e.g., COT3100)?
Taking now Within 6 mos Within 1 yr Within 2yr 2+ yrs _____ Never Taken
4. Where do you rank your current general knowledge of set theory and set proofs?
Beginner Amateur Average Very good Expert
5. Which hand do you write with?
Right hand Left hand
6. Do you have any flexibility issues with your dominant arm & hand?
No Yes, a little Yes, a lot

APPENDIX C: POST-QUESTIONNAIRE GIVEN TO
EXPERIMENT PARTICIPANTS

Studying the Effectiveness of SetPad for Solving Set Theory Proof Problems
ISUE Lab, Computer Science Department
University of Central Florida
Post-Questionnaire

Based upon your use of SetPad in the experiment, please answer the following questions about your experiences.

- Respond to each question with whatever knowledge you have.
- There is no right or wrong answers.
- Be honest and realistic in your assessment.
- State your agreement based on the following scale:

1 2 3 4 5 6 7

Strongly Disagree |-----|-----|-----|-----|-----|-----|**Strongly Agree**

Section 1: SetPad in the Test

1. SetPad made it easier to solve the set proof problems vs. if I had to solve them by hand with pen and paper.

1 2 3 4 5 6 7

2. The SetPad touch and drag interface was easy to use in manipulating the set expressions.

1 2 3 4 5 6 7

3. The real-time display of valid algebraic laws in SetPad was helpful in solving the problems.

1 2 3 4 5 6 7

4. SetPad was effective at visualizing the set relationships presented in the problems

1 2 3 4 5 6 7

5. I had enough practice and training to adequately perform and solve the problems in the experiment.

1 2 3 4 5 6 7

6. The HP Touchsmart PC was appropriate for use in testing of "SetPad."

1 2 3 4 5 6 7

Section 2: SetPad in a Classroom

1. The discrete mathematics course that I took previously adequately prepared me to solve these types of problems.

1 2 3 4 5 6 7

2. Using SetPad in a discrete mathematics course would have helped me to learn and understand the material better.

1 2 3 4 5 6 7

3. SetPad would have helped me to complete my discrete mathematics coursework assignments more quickly and correctly.

1 2 3 4 5 6 7

4. SetPad would have allowed me to better learn which and when to use the set algebra laws that I could apply when taking a discrete mathematics course.

1 2 3 4 5 6 7

5. I would become too dependent on using a tool like SetPad in solving problems in a discrete mathematics course.

1 2 3 4 5 6 7

Section 3: SetPad To Study For The Foundation Exam

1. Have you taken the UCF CS Foundation Exam?
Yes No Not Sure
2. If so, did you correctly answer the problem on set theory from the exam?
Yes No Not Sure

3. SetPad would help or would have helped prepare me for the set theory problem in the UCF CS Foundation Exam.

1 2 3 4 5 6 7

Section 4: Other Comments

1. What changes would you like to see made to SetPad to make it better?

APPENDIX D: CHEAT SHEET GIVEN TO CONTROL
PARTICIPANTS

Laws of Set Theory

1. Law of Double Negation

$$\overline{\overline{A}} = A$$

2. DeMorgan's Laws

$$\overline{A \cup B} = \overline{A} \cap \overline{B}$$

$$\overline{A \cap B} = \overline{A} \cup \overline{B}$$

3. Commutative Laws

$$A \cup B = B \cup A$$

$$A \cap B = B \cap A$$

4. Associative Laws

$$A \cup (B \cup C) = (A \cup B) \cup C$$

$$A \cap (B \cap C) = (A \cap B) \cap C$$

5. Distributive Laws

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$$

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$$

6. Idempotent Laws

$$A \cup A = A$$

$$A \cap A = A$$

7. Identity Laws

$$A \cup \emptyset = A$$

$$A \cap \mathcal{U} = A$$

8. Inverse Laws

$$A \cup \overline{A} = \mathcal{U}$$

$$A \cap \overline{A} = \emptyset$$

9. Domination Laws

$$A \cup \mathcal{U} = \mathcal{U}$$

$$A \cap \emptyset = \emptyset$$

10. Absorption Laws

$$A \cup (A \cap B) = A$$

$$A \cap (A \cup B) = A$$

11. Definition of Relative Complement

$$A - B = A \cap \overline{B}$$

LIST OF REFERENCES

- [1] Jon Barwise and John Etchemendy. *Tarski's World: Version 4.0 for Macintosh* (*Center for the Study of Language and Information - Lecture Notes*). Center for the Study of Language and Information/SRI, 1993.
- [2] Jon Barwise and John Etchemendy. *Language, Proof and Logic*. Center for the Study of Language and Information, 2002.
- [3] Jared N. Bott and Joseph J. LaViola, Jr. A pen-based tool for visualizing vector mathematics. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, pages 103–110, 2010.
- [4] Fred D Davis. Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 13(3):319–340, 1989.
- [5] Cordelia Hall and John O'Donnell. *Discrete mathematics using a computer*. Springer-Verlag New York, Inc., New York, NY, USA, 2000.
- [6] Ken Hinckley, Michel Pahud, and Bill Buxton. Direct display interaction via simultaneous pen+ multi-touch input. *Information Display*, 41(May):537–540, 2010.
- [7] George Labahn, Edward Lank, Scott MacLean, Mirette Marzouk, and David Tausky. Mathbrush: A system for doing math on pen-based devices. In *The Eighth IAPR International Workshop on Document Analysis Systems*, pages 599–606, 2008.

- [8] Joseph J. LaViola, Jr. and Robert C. Zeleznik. Mathpad²: a system for the creation and exploration of mathematical sketches. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 432–440, 2004.
- [9] Microsoft: Math. Computer program, September 2011.
<http://www.microsoft.com/math>.
- [10] Timothy Miller. Microsoft center for research on pen-centric computing: starpad sdk. Computer program, July 2009. <http://pen.cs.brown.edu/starpad.html>.
- [11] Darrell Plank. A graph tree drawing control for wpf. Computer program, February 2009. <http://www.codeproject.com/KB/WPF/LayeredTreeDraw.aspx/>.
- [12] James F. Power, Thomas Whelan, and Susan Bergin. Teaching discrete structures: a systematic review of the literature. In *Proceedings of the 42nd ACM technical symposium on Computer science education*, SIGCSE '11, pages 275–280, 2011.
- [13] Regular Berry Software. Algebra touch, for ios. Computer program, August 2011.
<http://www.algebratouch.com/>.
- [14] Harold Thimbleby and W. Thimbleby. A novel gesture-based calculator and its design principles. In *Proceedings 19th. British Computer Society HCI Conference*, pages 27–32, 2005.
- [15] Robert Zeleznik, Andrew Bragdon, Ferdi Adeputra, and Hsu-Sheng Ko. Hands-on math: a page-based multi-touch and pen desktop for technical work and problem solving. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, UIST '10, pages 17–26, 2010.

- [16] Robert Zeleznik, Timothy Miller, Chuanjun Li, and Joseph J. Laviola, Jr.
Mathpaper: Mathematical sketching with fluid support for interactive computation.
In *SG '08: Proceedings of the 9th international symposium on Smart Graphics*, pages
20–32, 2008.