LogicPad: A Pen-Based Application for Visualization and Verification of Boolean Algebra

Bo Kang
Interactive Systems and User Experience Lab
University of Central Florida
Orlando, Florida 32816
bkang@cs.ucf.edu

Joseph J. LaViola Jr.
Interactive Systems and User Experience Lab
University of Central Florida
Orlando, Florida 32816
jjl@eecs.ucf.edu

ABSTRACT
We present LogicPad, a pen-based application for boolean algebra visualization that lets users manipulate boolean function representations through handwritten symbol and gesture recognition coupled with a drag-and-drop interface. We discuss LogicPad's user interface and the general algorithm used for verifying the equivalence of three different boolean function representations: boolean expressions, truth tables, and logic gate diagrams. We also conducted a short, informal user study evaluating LogicPad's user interface, visualization techniques, and overall performance. Results show that visualizations were generally well-liked and verification results matched user expectations.

Author Keywords
Pen-based User Interface, Sketch Understanding, Boolean Algebra, Logic Verification

ACM Classification Keywords
H.5.2 Information Interfaces and Presentation: User Interfaces—*Interaction Styles*; G.4 Mathematics of Computing: Mathematical Software—*User Interfaces*

General Terms
Design, Human Factors

INTRODUCTION
Boolean Algebra [8] is a fundamental concept in computer science and digital system design. Many problems in digital logic design and testing, artificial intelligence, and combinatorics can be expressed as a sequence of boolean operations and boolean variables. A boolean function can be represented as an expression, truth table, or logic gate diagram; the three different representations of the same boolean function are considered equivalent.

Traditional complex computer-aided design applications are important tools for designing logic circuits. Nevertheless, for novices, such as students who are learning the fundamentals of boolean algebra, simple and quick prototyping is important. Most students still find using pen and paper to quickly sketch out boolean expressions, truth tables, and logic gate diagrams to be useful. Therefore, we made LogicPad (see Figure 1), a new pen-based application that combines the ease-of-use of pen and paper with the automation of computer-aided design applications. LogicPad focuses on the relationship between three different representations of boolean functions. Users can quickly sketch a boolean function and visualize the function in each representation simultaneously. In addition, we provide a general method for verifying the equivalence of two boolean functions.

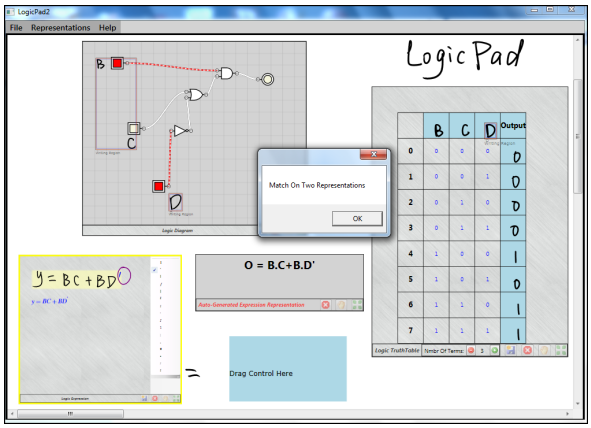

Figure 1. LogicPad's main window with three boolean function representations.

RELATED WORK
Computer aided design applications for undestanding logic circuits have been previously developed, such as LogicAid, which lets users develop boolean functions using representations such as truth tables, state tables, and state graphs [9]. LogicAid is similar in spirit to our work, but LogicAid does not make use of a pen-based interface.

Logic gate diagrams are well-suited for pen-based sketching environments. SketchREAD describes a multi-domain sketch recognition engine capable of recognizing hand-drawn diagram sketches [1]. Wais' user study explored critical sketch recognition user interface issues with logic gate diagrams [10]. Alvarado also analyzed how students created freely-drawn

IUI'12, February 14–17, 2012, Lisbon, Portugal.

logic gate diagrams in class over the course of a full semester [2]. However, these works use user interfaces that are entirely sketch-based, while our work uses a combination of sketching and a drag-and-drop interface.

Besides sketching diagrams, research in the recognition, evaluation, and visualization of mathematical expressions has also been conducted previously. MathPad^2 is a mathematical sketching tool for solving mathematics problems [5]. MathBrush is a mathematics recognition tool that supports mathematical transformations and problem solving through the use of a back-end Computer Algebra System [4]. Our work expands upon the notion of mathematical sketching, extending it by combining mathematical expressions and logic gate diagrams.

USER INTERFACE

LogicPad provides a sketching area for the user to write on as they normally would using pen and paper. Instead of sketching the various boolean function representations on a single canvas, a separate widget with its own canvas is created for each representation, which can be added to the sketching area.

Users write gestures on the sketching area in order to create widgets. For instance, drawing an 'E' will create a boolean expression widget. Truth table widgets and logic gate diagram widgets are created using 'T' and 'D', respectively. A scribble erase gesture is used to erase both ink and user interface elements, because it mimics natural erasing.

CONCEPTUAL MODEL

As previously discussed, boolean functions have three basic representations: truth tables, boolean expressions, and logic gate diagrams. The three representations in LogicPad provide visualization cues to help the user understand the boolean function they are working with.

Logic Gate Diagram Widget

With a sketching interface, the user needs to know the shape of each type of logic gate, which can be a cognitive burden for students who are still learning about boolean algebra. Therefore, we try to ease the burden by using a drag-and-drop radial menu [3] showing the different types of logic gates. Figure 2 shows a logic gate diagram widget with a radial menu showing the six basic logic gates.

Users can hold down the stylus on the tablet screen for 3 seconds in order to trigger the radial menu and drop a gate from it on the sketching area. Gates can also be freely moved around the sketching area. We provide a sketching interface for this widget; after gates are dragged and dropped on the canvas, a user can sketch a line to connect any two gates. If a stroke starts at one gate and ends at another, it is recognized as a wire and the stroke is removed and replaced with a wire.

A user can sketch a label near each gate. Label association is decided by checking whether an ink stroke falls within a threshold distance from a gate. Any strokes that fall within the threshold are analyzed by the Microsoft Ink Analyzer as typeset notations underneath strokes and used as the gate's label. In addition, after the user connects two gates with a wire, the widget animates current flowing through the wire.

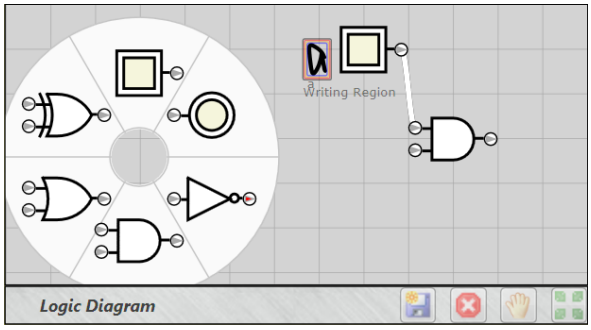

Figure 2. A logic gate diagram widget with a radial menu open showing the six kinds of logic gates: AND, OR, NOT, XOR, Input, and Output.

Handwritten Symbol	Boolean Operator	Example
$+$	OR	$A + B$
$\cdot$	AND	$A \cdot B$ or AB
$\prime$	NOT	A'
$\oplus$	XOR	$A \oplus B$

Table 1. Mapping between handwritten symbols and boolean operators.

Boolean Expression Widget

Since sketching mathematics is natural, LogicPad's boolean expression widget is a canvas that interprets the user's handwritten math and converts it to a typeset notation. After recognizing alphabetical letters, numbers and numerical operators using StarPad [11], four handwritten boolean operators can be interpreted by LogicPad, which are summarized in Table 1.

During expression construction, a user can circle strokes and select alternate recognition options from an alternates bar in case the default recognition does not match the user's expectation. In addition, expressions can be selected and moved around the widget.

Truth Table Widget

The truth table widget presents a table for the user to fill out with the expected output of the boolean function they are working with. The user can change the number of variables; each variable can be renamed by erasing the old name and writing a new one, then the handwritten symbol is analyzed by the Microsoft Ink Analyzer as the typeset notation underneath it. In comparison to pen and paper, LogicPad helps the user by creating a table with variables and the factorial combinations of their input values, which can be an onerous step when working with many variables. Thus, the user can concentrate on working out the output values for the truth table.

MAPPING

LogicPad can determine the boolean function each widget represents and minimize the function. The other two function representations can also be created from any type of widget.

Figure 3. Process for converting a widget in order to determine whether two widgets represent the same boolean function.

Verifying the equivalence of two boolean functions is essential to LogicPad, because it lets users compare the different representations, which can help them to understand boolean algebra and boolean function representations.

Each kind of widget stores data in a different format; when the user wants to see the widget's function in another representation, the widget converts its data to a parse tree that stores a variable name or boolean operator at each node. Logic gate diagram widgets use XML to represent the gates and wires, expression widgets use a binary tree to represent the mathematics, and truth table widgets use a tabular array to store the input values, output values, and variable names.

Looking at two boolean functions, it can be ambiguous whether they are equivalent. For instance, $F = X + 0$ and $F = X$ are equivalent, as are $F = XY' + XY$ and $F = X$. In order to determine the equivalence of two boolean functions, we apply the steps shown in Figure 3.

The tree rewriting phase always changes the structure of the parse tree. It includes five sub-steps. First, boolean identity laws are checked. For instance, the parse tree that represents $X + 0$ is reduced to X. Second, the XOR operator is substituted for its equivalent form. For example, the parse tree that represents $A \oplus B$ is converted to $AB' + A'B$. Third, the parser will make sure that all NOT operator nodes are located as deep in the tree as possible. Fourth, boolean functions are formulated as a sum-of-products expansion. Fifth, boolean complement laws are checked.

The goal of the tree segmentation phase is to convert the parse tree to an ordered array of 0-1 bitstrings. In this phase, the parse tree changes each sum-of-products expansion to an array-of-products expansion, and variable names are analyzed as a one or zero depending on the presence of a NOT operator parent node.The final phase is to minimize the boolean function. We use the Quine-McCluskey algorithm [6] to simplify the ordered array of bitstrings.

FEEDBACK

We use two approaches to verify the user's logic construction, both of which can be seen in Figure 1. Our first approach is an explicit feedback mechanism. When a user designs a boolean function using one type of representation, the other two can be generated from the representation menu.

Our second approach is a gesture-based mechanism which can be seen as an implicit feedback mechanism. A user can draw an equals sign on the sketching area between two widgets. When the equals sign is recognized, LogicPad will check if there is a widget on either side of the equals sign. If there is no widget on one side, then a rectangle will be drawn on the sketching area to give the user a hint to drag a widget to that area. Finally, LogicPad parses the two widgets and tells the user whether the boolean functions represented by the widgets are equivalent using a pop-up window.

INFORMAL USER STUDY

To gain insight into the perceived utility of LogicPad for domain users, we conducted an informal user study.

We recruited 6 participants (1 female and 5 males, aged 18-25, 2 left-handed) from the undergraduate electrical engineering department at the University of Central Florida. Participants all had basic knowledge of boolean algebra. The study was conducted in the lab, using an HP Compaq tc4400 12.1 inch tablet PC. We informed participants of minor bugs when possible, as our goal was to evaluate system design rather than implementation and recognition issues.

Users were asked to practice a set of tasks in order to get familiar with the system. The practice tasks consisted of creating and manipulating boolean expressions, truth tables and logic gate diagrams. Finally, users were asked to design two boolean functions and to verify the equivalence between corresponding representations.

After the practice phase, participants were required to solve a list of tasks using LogicPad. Pen and paper were also provided in case the participant needed to write down their thinking process. The tasks were divided into two categories. In the first category, one boolean function representation was provided and participants were then asked to sketch one of the other two corresponding representations of the same function and to verify their equivalence using LogicPad. In the second category, two boolean function representations were provided and participants were then required to manually determine if the representations were logically equivalent before using LogicPad to verify the result.

At the completion of the study (which lasted approximately 50 minutes), participants completed a questionnaire inquiring if the tool was useful for boolean function verification and if it was helpful for logic problem solving.

Observations

Triggering Widgets

Five out of six participants preferred our gestural method to add widgets. One participant argued that it would be better to give a option to select the widget from a menu. Two users were frustrated by the difficulty of triggering widgets by writing gestures due to incorrect handwriting recognition results obtained from the recognizer.

Diagram User Widget

All participants preferred the method for triggering the radial menu, noting that it reduced the learning time and augmented the retention time in drawing logic gates. Only one participant said that it would be better to make the radial menu visible longer than the given time interval.

All became accustomed to the drag-and-drop interface of applying the gate to the canvas. One participant suggested that it may be better to simply tap each gate on the radial menu

to make the gate visible in the widget. From our observations, we believed that other participants would have the same preference. After participants triggered the radial menu, their first instinct was to use the stylus to target one specific gate and wait for response from the system. We also found one mode-switching issue. For instance, some users would unintentionally trigger the radial menu when trying to change the position of logic gates already present in the sketching area. We conjecture that using a mode inference approach [7] could reduce this ambiguity.

Expression User Widget

Four out of six participants found it easy to manipulate. However, one participant reported being frustrated by wrong recognition results when writing expressions. For example, when he wrote apostrophes as negation operators, they were always falsely recognized as another notation, requiring him to select an alternative recognition result. He recommended to use a hat symbol as the negation operator.

Truth Table User Widget

All of the participants appreciated not having to draw the entire truth table. One participant said, "I like how I could add variables easily with the menu at the bottom." Another participant commented that it would be better to initialize all of the output values to zero first and then scribble out non-zero values, replacing them with ones.

Feedback

All participants preferred our two approaches, as both of them provided intuitive ways to visualize and comprehend boolean functions.

Overall, every participant preferred having separate widgets, and one of them commented that it was a natural way to represent boolean functions. Another user commented "I did like LogicPad but I think you should be able to have the choice on if you want separate widgets or not." All participants who used LogicPad thought that it was a good verification tool for problem solving, but two commented that it would be better if LogicPad was faster and more reliable.

CONCLUSION AND FUTURE WORK

We have presented LogicPad, a pen-based tool that support boolean function visualization and verification. It provides different sketching areas to write boolean function representations and parses user's logic representations in order to generate other boolean function representations. In the logic gate diagram widget, we illustrated one mixed sketch-drag-and-drop approach. We also conducted an informal study, where we found that our boolean function knowledge representation approach was well liked by the participants. We also found that participants wanted tools to help them to verify their problem solving results in general. In the future, we plan to continue to explore under what circumstances should a sketching interface or drag-and-drop interface be chosen, how LogicPad could be used to aid students, and understanding LogicPad's pedegogical implications.

ACKNOWLEDGMENTS

This work is supported in part by NSF CAREER award IIS-0845921 and NSF awards IIS-0856045 and CCF-1012056. We would also like to thank the members of the ISUE lab for their support and the anonymous reviewers for their useful comments and feedback.

REFERENCES

1. Alvarado, C., and Davis, R. Sketchread: A multi-domain sketch recognition engine. In *Proc. of the 17th annual ACM symposium on User interface software and technology* (Santa Fe, New Mexico, USA, 2004), 23–32.
2. Alvarado, C., and Lazzareschi, M. Properties of real-world digital logic diagrams. In *Proceedings of the First International Workshop on Pen-Based Learning Technologies*, PLT '07, IEEE Computer Society (Washington, DC, USA, 2007), 1–6.
3. Callahan, J., Hopkins, D., Weiser, M., and Shneiderman, B. An empirical comparison of pie vs. linear menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '88, ACM (New York, NY, USA, 1988), 95–100.
4. Labahn, G., Lank, E., MacLean, S., Marzouk, M., and Tausky, D. Mathbrush: A system for doing math on pen-based devices. In *Document Analysis Systems, 2008. DAS '08. The Eighth IAPR International Workshop on* (sept. 2008), 599 –606.
5. LaViola, Jr., J. J., and Zeleznik, R. C. Mathpad^2: a system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2004 Papers*, SIGGRAPH '04, ACM (New York, NY, USA, 2004), 432–440.
6. McCluskey, E. J., and Schorr, H. Minimization of boolean functions. *Bell System Technical Journal 35*, 5 (November 1956), 1417–1444.
7. Negulescu, M., Ruiz, J., and Lank, E. Exploring usability and learnability of mode inferencing in pen/tablet interfaces. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*, SBIM '10, Eurographics Association (Aire-la-Ville, Switzerland, Switzerland, 2010), 87–94.
8. Nelson, V. P., Nagle, H. T., Carroll, B. D., and Irwin, J. D. *Digital logic circuit analysis and design*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1995.
9. Roth, C. *Logicaid, CAD Software for Logic Design, Version 3.0 for Windows: Users Guide and Reference Manual*. Cengage Learning, 1995.
10. Wais, P., Wolin, A., and Alvarado, C. Designing a sketch recognition front-end: User perception of interface elements. In *SBM*, M. van de Panne and E. Saund, Eds., Eurographics Association (2007), 99–106.
11. Zeleznik, R., Miller, T., Li, C., and Laviola, Jr., J. J. Mathpaper: Mathematical sketching with fluid support for interactive computation. In *Proceedings of the 9th international symposium on Smart Graphics*, SG '08, Springer-Verlag (Berlin, Heidelberg, 2008), 20–32.