

# **Developing Handwriting-based Intelligent Tutors to Enhance Mathematics Learning**

**Lisa Anthony**

October 9, 2008  
CMU-HCII-08-105

Human-Computer Interaction Institute  
School of Computer Science  
Carnegie Mellon University  
Pittsburgh, PA 15213

**Thesis Committee:**  
Kenneth R. Koedinger, co-chair  
Jie Yang, co-chair  
Jennifer Mankoff  
Tom Mitchell  
Mark Gross

*Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy*

© 2008 Lisa Anthony

This research is supported by grants from the National Science Foundation (NSF Award #SBE-03554420) and the Pittsburgh Science of Learning Center. The author is partially supported by an NSF Graduate Research Fellowship. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NSF or PSLC.

**Keywords:** Handwriting recognition, recognition accuracy, recognition evaluation, writer-independent training, average-rank sort, equation entry, mathematics, algebra, intelligent tutoring systems, equation solving, handwritten mathematics, math learning, algebra learning, handwriting input, Cognitive Tutors, worked examples, math interfaces, human-computer interaction.

## Abstract

Mathematics is a topic in American education in which students lag behind their international peers, yet it is a key building block for high-performing careers in science, computers, and engineering. Intelligent tutoring systems have been helping to narrow this gap by providing students with opportunities to practice problem-solving and receive detailed feedback along the way, letting them work at their own pace and practice specific concepts. Prior to this work, intelligent tutors for math have been shown to improve student performance one standard-deviation above traditional classroom instruction [35]. This dissertation explores ways to improve this effect via the use of alternative input modalities, specifically: handwriting input, and investigates the impact on learning in the domain of algebra equation solving.

This dissertation shows that handwriting provides *usability* benefits in that speed of entry increases, user error decreases, and user satisfaction increases. Furthermore, it shows that handwriting may also provide *learning* benefits: students solving the same problems by handwriting as others who are typing experience a faster learning rate. Specific math advantages of using handwriting are: a reduction in extraneous cognitive load due to the affordance of handwriting for more direct manipulation, and improved support for the two-dimensional spatial information which is inherently meaningful in mathematics (*e.g.*, vertical fraction notation). This dissertation investigates these factors and their impact.

One concern with the use of handwriting in intelligent tutoring systems, however, is that recognition technology is not perfect. To the extent that the system cannot be confident of correctly recognizing what the student is writing, it cannot identify tutoring opportunities and provide detailed, step-targeted feedback. Therefore, a trade-off is clear between the difficulty of improving recognition accuracy and the need to support step-targeted feedback. One strategy to address this trade-off is using a type of instruction based on *worked examples*, which provide a sort of *feed-forward* to guide learners. A second strategy is to investigate technical approaches to improving handwriting recognition accuracy. This dissertation explores two methods of enhancing baseline recognition: training the recognition engine on a data corpus of student writing in order to maximize writer-independent recognition accuracy; and making use of domain-specific context information on the fly to refine the recognition hypotheses.

The approach taken in this dissertation includes technical development, pedagogical development, and user studies. Topics addressed include what the advantages of using handwriting are, how the above factors contribute to these advantages, and how these advantages can be leveraged in real tutoring systems. Reasonable writer-independent handwriting recognition rates can be achieved by *a priori* data collection and training, and these can be even further improved via the addition of domain-context information. Furthermore, a realistic tutoring interaction paradigm can be achieved through these methods, in spite of imperfect raw recognition accuracy. This dissertation leaves the door open to continued work on basic recognition technology which can improve the achievements reported here even further.

*for my mother*

## Acknowledgments

Writing this section of my dissertation turned out to be the hardest part. It is usually expected to be a sentimental and inspirational message, which is quite different from the kind of writing in the rest of the document. There have been so many people who have guided me, provided for me, and helped me along the way to my doctorate. If in any part of talking about and presenting this work I have implied otherwise, that is my error.

First thanks of course go to both of my advisors, Ken Koedinger and Jie Yang, who have both helped focus my work, provided feedback and helped me brainstorm on both implementation and data analysis throughout the process. I thank them for always having faith in the value of the work I was doing, even when it seemed like no one else did. It was a productive and enjoyable “n” years working with you. I would also like to thank my committee members, Mark Gross, Jen Mankoff, and Tom Mitchell, who have never hesitated to meet with me when I needed their specific expertise or advice along the way. I extend a special thank you to Albert Corbett, my advisor in the first two years on another project than that presented here: thanks for being flexible when I wanted to pursue my own interests.

The entire faculty of the Human-Computer Interaction Institute has been supportive and available when I needed advice, either personal or professional, at various times throughout my graduate school career. Even when it seemed like everyone was rushing to meet deadlines and handle their many responsibilities, the faculty would always take time out for any student. Jodi Forlizzi, Sara Kiesler, Bob Kraut, Chris Neuwirth, Carolyn Rosé, and especially Scott Hudson deserve special mention for their dedication and attention. It meant a lot to a graduate student who was sometimes struggling to find her place, in the field and in life.

Faculty further afield have also provided helpful input, feedback, guidance and support, especially Richard Zanibbi, energetic maintainer of FFES who never failed to answer my questions about FFES, how to get it to work with cygwin instead of native \*nix, etc.; Sharon Oviatt, who was always more than happy to share her long-time expertise in multimodal interfaces; and Chris Atkeson, Shelley Evenson, and Susan Fussell, who loaned our project much-needed and appreciated equipment.

My fellow graduate students in the HCII form a wonderful, cohesive community that prizes interdisciplinary collaboration above cutthroat competition. From reading groups, lunch seminars, and pot-lucks, to piloting experiments or just sitting around brainstorming and chatting, our community was extremely well-read, diverse, eclectic and interesting. I couldn’t imagine a better group of folks to call my fellow graduate students. Specifically I mention Sonya Allin, Anupriya Ankolekar, Ryan Baker, Aruna Balakrishnan, Moira Burke, Laura Dabbish, Scott Davidoff, Matt

Easterday, Adam Fass, Darren Gergle, Elsa Golden, Amy Hurst, Ian Li, Ido Roll, Peter Scupelli, Karen Tang, Cristen Torrey, Angela Wagner, Erin Walker, Jake Wobbrock, Jeff Wong, and Ruth Wylie; and my own classmates, Aaron Bauer and Andy Ko; for their own individual contributions to my time at the HCII, including, among other things, being pilot subjects!

Through my advisor Jie Yang, I was fortunate to be involved in both the Interactive Systems Lab and the Computer Vision and Pattern Recognition reading group. I met many talented students that deserve thanks, especially Jiazhui Ou and Datong Chen for the time they spent helping me set up recording equipment or digitizing data.

Through my advisor Ken Koedinger, I was also fortunate to be involved in the Pittsburgh Science of Learning Center since its inception. While I thank the PSLC Executive Committee for providing my dissertation project with funding, the PSLC also brought together many bright and curious learning science researchers to discuss the future of the field and how best to incorporate technology into the classroom environment. I especially thank the PSLC staff including Michael Bett, Ido Jamar, Alida Skogsholm, and Kyle Cunningham for their support and hard work keeping the PSLC and DataShop running smoothly, and answering any question I had. Other PSLC-affiliated people who contributed in content or other support to my dissertation project include Noboru Matsuda, Albert Corbett, and Vincent Aleven. Thanks!

I must of course mention the HCII administrative staff, who are such a wonderful group of people. I thank them heartily for their tireless hours and cheerful attitudes, especially Jo Bodnar and Queenie Kravitz. Thanks for all your hard work and well-wishes.

When our project was lucky enough to have research assistants, we had terrific ones. Andrea Knight deserves mention for conducting some of the Math Input Study sessions. Thomas Bolster joined us for a summer and ran the Lab Learning Study sessions. Keisha How also joined us for a summer and implemented and ran the Microsoft TabletPC recognizer tests. Thanks to all three of you for your dedication and excellent work.

In running the Cognitive Tutor Study, we worked in two different Pittsburgh-area schools. I must thank the teachers who volunteered their classrooms to participate, but, due to anonymity concerns, I cannot name them here. Suffice it to say that their commitment to teaching and furthering knowledge in the field of education is a credit to them. Thanks for allowing us access to your classrooms in the name of science.

Thanks to Richard Zanibbi and Ernesto Tapia for allowing us to use their handwriting recognition systems throughout my dissertation. Thanks to Carnegie Learning and especially Frank Baker, Jon Steinhart, and Steve Ritter for allowing us to use their Cognitive Tutor and answering questions about the code and curriculum.

An absolutely critical person to my success and achievement in the field of computer science and human-computer interaction today is William Regli, my undergraduate advisor and mentor for four years at Drexel University. His encouragement and the opportunities he created for his students sparked my imagination and ambition. I have no doubt in my mind that I would not be where I am today if I had not been an undergraduate working in his research lab. A special mention also goes to his wife, Susan Harkness Regli. Both of them opened their home and hearts to me

and, when things got rough, both helped me remember and re-invest in my goals.

Of course, work and research are only one part of life, even as a graduate student, and I would not have enjoyed my time in Pittsburgh nearly so much as I did without my close friends, some of whom were at the HCII themselves, and others who were in other programs or even just “real people” with “real jobs.” Thank you to Amy, Elsa, Marty, Angela, Carson, Allyson, Dave, Aaron, and Becky, for all the fun hours and true friendship that you demonstrated to me over and over. I can only hope to have been half as good of a friend to you as you each were to me. Amy and Elsa, I especially want to thank you for being two of the strongest, most independent, fun, smart and talented women that I know. I’m fairly certain that the three of us together could conquer the world. Thanks for being my best friends through thick and thin.

Though my family was far away and living their own busy lives, I know they never stopped caring or believing in me, especially my mom and two sisters. I know I made them proud on the day I completed my PhD; what they maybe don’t know is how proud I am of each of them for going out and achieving their dreams. To my father: thank you for giving me the (somewhat nerdy) gift of a love of technology; I’m glad we can be friends. To my dearest friend, Vera Zaychik Moffitt: I am so proud of you; your strength and matter-of-factness lent me mine. Let’s continue to travel the world together!

Last but not least, I want to thank and express my utmost appreciation and love for the person who shares my life and my heart: Isaac Simmons. He helped me through the last year of my graduate work, which seemed like the longest year of my life. He was always there to help me brainstorm about work and listen to my half-baked ideas, but also to remind me to stop working and take some time for myself once in a while. I truly love you and hope we are part of each other’s lives for a long long time.

# Contents

<b>1</b>	<b>Introduction</b>	<b>19</b>
1.1	Motivation . . . . .	19
1.1.1	Limitations of Typing in Intelligent Tutors for Math . . . . .	19
1.1.2	Limitations of Handwriting Recognition . . . . .	21
1.2	Concept . . . . .	22
1.3	Approach . . . . .	23
1.4	Document Organization . . . . .	25
<b>2</b>	<b>Related Work</b>	<b>26</b>
2.1	Intelligent Tutoring Systems and Cognitive Tutors . . . . .	26
2.2	Learning Science and Educational Technology . . . . .	27
2.2.1	Worked Examples as Instructional Interventions . . . . .	29
2.2.2	Ways to Measure Learning . . . . .	29
2.3	Handwriting Recognition Techniques and Systems . . . . .	31
2.3.1	Neural Networks . . . . .	31
2.3.2	Support Vector Machines . . . . .	32
2.3.3	Hidden Markov Models . . . . .	32
2.3.4	Pen-Based Handwriting Recognition Performance . . . . .	32
2.3.5	Handwriting and Child-Computer Interaction . . . . .	34
2.4	Interfaces for the Math Domain . . . . .	35
2.4.1	Traditional Input Modalities . . . . .	35
2.4.2	Handwriting Input for Math . . . . .	37
2.5	Methods and Tools Used in this Dissertation . . . . .	38
2.5.1	Wizard-of-Oz . . . . .	38
2.5.2	Cross-validation . . . . .	38
2.5.3	Cognitive Load Self-Report . . . . .	38
2.5.4	Collaborative Information Retrieval: Ranking Fusion . . . . .	39
2.5.5	Freehand Formula Entry System . . . . .	39
2.5.6	Cognitive Tutor Algebra . . . . .	40
2.6	Glossary of Terminology . . . . .	41

<b>CONTENTS</b>	<b>9</b>
<b>3 Handwriting Helps: Theory</b>	<b>43</b>
3.1 Usability and Handwriting . . . . .	43
3.2 Pedagogical and User-Focused Factors . . . . .	44
3.2.1 Cognitive Load . . . . .	44
3.2.2 Spatial Characteristics of Math . . . . .	45
3.2.3 Fluency and Transfer to Paper . . . . .	45
3.3 Bridging Pegagogy and Technology . . . . .	46
3.4 Proving the Hypotheses . . . . .	47
3.4.1 Usability Measures . . . . .	47
3.4.2 Pedagogical Measures . . . . .	48
<b>4 Handwriting Helps: Foundational Studies</b>	<b>49</b>
4.1 Study 1: The Math Input Study . . . . .	49
4.1.1 Experimental Design . . . . .	51
4.1.2 Results and Discussion . . . . .	54
4.1.3 Conclusions . . . . .	61
4.2 Study 2: The Lab Learning Study . . . . .	61
4.2.1 Experimental Design . . . . .	61
4.2.2 Results and Discussion . . . . .	64
4.2.3 Conclusions . . . . .	71
4.3 Study 3: The Cognitive Tutor Study . . . . .	72
4.3.1 Experimental Design . . . . .	73
4.3.2 Results and Discussion . . . . .	79
4.3.3 Conclusions . . . . .	85
4.4 General Conclusions from the Three Studies . . . . .	86
<b>5 Improving Recognition: Baseline Accuracy</b>	<b>88</b>
5.1 Choosing a Recognition Engine: Case Studies . . . . .	88
5.1.1 Freehand Formula Entry System . . . . .	89
5.1.2 JMathNotes . . . . .	89
5.1.3 Microsoft TabletPC Recognizer . . . . .	90
5.2 Baseline Handwriting Recognition Accuracy . . . . .	90
5.2.1 Domain-Specific <i>vs</i> Domain-General Use . . . . .	91
5.2.2 Writer-Dependent <i>vs</i> Writer-Independent Training . . . . .	92
5.2.3 Symbol-Level <i>vs</i> Equation-Level Testing . . . . .	93
5.2.4 The Algebra Learner Corpus . . . . .	93
5.2.5 The Evaluation Method . . . . .	94
5.2.6 Accuracy Results for Each Recognizer . . . . .	95
5.2.7 Summary of Results and Discussion . . . . .	101
5.3 General Conclusions . . . . .	103

<b>CONTENTS</b>	<b>10</b>
<b>6 Improving Recognition: Context</b>	<b>104</b>
6.1 Adding Domain-Specific Information . . . . .	104
6.1.1 Working with the Tutor Information . . . . .	105
6.1.2 Average-Rank Sort . . . . .	107
6.2 Context-Enhanced Recognition Accuracy . . . . .	110
6.2.1 Tutor Testbed . . . . .	111
6.2.2 Iterative Algorithm Tuning . . . . .	113
6.2.3 Choosing the Test Problems . . . . .	113
6.2.4 Evaluation Results . . . . .	115
6.3 Limitations . . . . .	124
6.4 General Conclusions . . . . .	125
<b>7 Interaction Case Studies</b>	<b>126</b>
7.1 Interaction Scenario . . . . .	127
7.2 Interaction Flow . . . . .	139
7.3 Errors and Error Repair Strategies . . . . .	140
<b>8 Conclusion</b>	<b>142</b>
8.1 Discussion and Summary . . . . .	142
8.2 Contributions . . . . .	143
8.3 Future Work . . . . .	144
8.3.1 Further Pedagogical Explorations . . . . .	144
8.3.2 Further Technical Explorations . . . . .	145
8.4 Final Remarks . . . . .	146
<b>Bibliography</b>	<b>147</b>
<b>Appendices</b>	<b>161</b>
<b>A Character Set Used</b>	<b>161</b>
<b>B Study 1 Materials: The Math Input Study</b>	<b>162</b>
B.1 Full List of Equations Used in the Math Input Study . . . . .	162
B.2 Math Symbols Test . . . . .	163
B.3 Pre-Session Questionnaire . . . . .	164
B.4 Post-Session Questionnaire . . . . .	166
B.5 Demographics Questionnaire . . . . .	167
<b>C Study 2 Materials: The Lab Learning Study</b>	<b>168</b>
C.1 Full List of Equations Used in the Lab Learning Study . . . . .	168
C.1.1 Equations Used During Copying Phase . . . . .	168

<i>CONTENTS</i>	11
C.1.2 Examples and Problems Used During Learning Phase . . . . .	171
C.2 Pre-Session Training Handout . . . . .	174
C.3 Test A . . . . .	180
C.4 Test B . . . . .	180
C.5 Post-Session Questionnaire . . . . .	181
<b>D Study 3 Materials: The Cognitive Tutor Study</b>	<b>184</b>
D.1 Demographics . . . . .	184
D.2 Test A . . . . .	185
D.3 Test B . . . . .	186
D.4 Test C . . . . .	187
D.5 Cognitive Load Questionnaire . . . . .	188
<b>E Set of Test Problems Used in Recognition Experiments</b>	<b>189</b>

# List of Figures

1.1	A screenshot of a proposed tutoring system for algebra equation solving that allows students to enter their solutions via handwriting in an unconstrained problem-solving space. Different versions of this prototype were used throughout this work.	23
2.1	A screenshot of the Cognitive Tutor interface for an algebra unit involving formulating the relationship between two variables. . . . .	27
2.2	Sample worked examples from two different domains. Note the differences in level of detail. . . . .	30
2.3	Examples of user interfaces for the most common computer tools for mathematics. Clockwise from upper left: Maple, Matlab, Mathematica, MathType. . . . .	36
4.1	Screenshots of the interfaces used in the Math Input Study. From top to bottom: the typing condition using Microsoft Equation Editor, the handwriting condition, and the speaking condition. The handwriting-plus-speaking, or multimodal, condition looked like the handwriting condition from the user's perspective; the speech recorder was running in the background. . . . .	52
4.2	Experimental stimuli as users saw them. . . . .	54
4.3	Mean time in seconds per equation by condition. Error bars indicate standard error. . . . .	58
4.4	Mean number of errors made per equation by condition. Error bars indicate standard error. . . . .	59
4.5	Pre-session and post-session questionnaire rankings of each condition on a five-point Likert scale. The pre-session questionnaire did not include a question about the multimodal condition. Error bars indicate standard error. . . . .	60
4.6	Screenshots of the interface used in the Lab Learning Study. The typing condition is shown on top, and the handwriting condition is shown below it. The multimodal interface looked identical to the handwriting condition, but a background process was also recording the student's voice. . . . .	63
4.7	Samples of equations and problems from each phase of the experiment. . . . .	64
4.8	Mean time per problem by condition crossed with appearance of fractions in the <i>learning</i> phase for both copying examples and solving problems. Error bars indicate standard error. . . . .	67

4.9	Histogram of user responses rating their <i>favorite</i> input modality grouped by the modality they used during the learning phase of the session. . . . .	68
4.10	A screenshot of the CogTutor-NoExamples-StepFeedback condition (control condition) in the Cognitive Tutor Study. . . . .	75
4.11	A screenshot of the CogTutor-Examples-StepFeedback condition in the Cognitive Tutor Study. . . . .	76
4.12	A screenshot of the CogTutor-Examples-AnswerFeedback condition in the Cognitive Tutor Study. . . . .	77
4.13	A screenshot of the Handwriting-Examples-AnswerFeedback condition in the Cognitive Tutor Study. . . . .	78
4.14	Estimated marginal means of learning gains as measured from pre-test to retention test condition in the Cognitive Tutor Study. The key finding is that the handwriting condition (Handwriting-Examples-AnswerFeedback) is only marginally significantly better than the control (CogTutor-NoExamples-StepFeedback). Error bars indicate standard error. . . . .	82
5.1	An example of two ways of writing the symbol '4', either with one continuous stroke or two separate strokes. . . . .	92
5.2	Baseline accuracy results for the Freehand Formula Entry System. . . . .	98
5.3	Baseline accuracy results for JMathNotes. . . . .	100
6.1	The type of information received from the tutor: the set of possible correct options for the current step (the step following $2x + 10 = 30$ ). . . . .	106
6.2	The alignment problem prevents comparing the tutor information to the recognizer information directly. Because the recognizer can make errors in stroke grouping, the tutor does not know which symbol (in order from right to left) to consider. In this example, stroke grouping errors have occurred on the '4', which has been split into two groups, and on the '-1' symbols, which have been combined into one group. When the recognizer asks the tutor for the sixth character, the tutor returns '8' but the recognizer is already looking at the '='. In this case, the tutor information would not help, and may in fact harm, recognition. . . . .	107
6.3	Converting the tutor information, provided as a set of possible correct equations, into a "bag of words" rank-list involves the three steps shown above. The symbols in each of the equations are jumbled together and re-sorted by frequency. Symbols with the same frequency are assigned the same rank, and symbols in the vocabulary but not in any of the possible equations are assigned one more than the maximum rank. . . . .	108
6.4	The step-by-step algorithm which takes the recognition results and the tutor information and combines them to better interpret students' written input. . . . .	109

6.5	System architecture diagram of the prototype tutoring system. For the experiments reported in this chapter, the “Interface wrapper” component is replaced with a batch testing program that reads handwritten strokes from corpus files and feeds them into the recognizer. . . . .	112
6.6	The raw accuracy of the combined system at varying weights on the tutor and recognizer during the average-rank sorting process. The best improvement over the recognizer alone was seen at ( $W_T = 0.40$ , $W_R = 0.60$ ) pair during the average-ranking process, with an accuracy of 72.5%. Performance of the recognizer alone ( $W_T = 0.00$ , $W_R = 1.00$ ) is 66.3%. . . . .	116
6.7	Performance of the system on the error identification problem. . . . .	120
6.8	The ROC curve showing effect of applying different thresholds on identifying the error. Each line is a different ( $W_T$ , $W_R$ ) pair. . . . .	121
7.1	The proposed interface for a tutor using Cognitive Tutor Algebra as its base and allowing students to enter the problem-solving process via handwriting input. The worked example being used as reference by the student appears on the lefthand side of the screen. The handwriting input space is a blank, unconstrained input space. The text box for the student’s final answer is on the bottom right of the screen, next to the “Check My Answer” button, which launches a tutoring intervention if the typed final answer is incorrect. . . . .	127
7.2	The scenario begins with the (fictional) student beginning a new lesson on solving linear equations with variables on both sides ( $ax + b = cx + d$ ). The lesson begins with a review of a simpler problem type, $ax + b = c$ . The student is given an example of the first type of problem she will see, and is instructed to copy it out while thinking critically about the steps involved. . . . .	128
7.3	The student has copied out the example as instructed. When she clicks the “Check My Answer” button, the system will check that she has actually copied the example and done it successfully (by checking the final, typed-in answer). In this case, she has, so she will be allowed to move on. . . . .	129
7.4	After moving on, the student is given a new problem, analogous to the problem shown in the example that she has just copied, to solve on her own. The example remains onscreen to <b>scaffold</b> her problem-solving experience. The problem and the example are of the same type (e.g., $ax + b = c$ ), but may have different surface forms. . . . .	130
7.5	The student is solving the problem given to her, by referring to the example onscreen.	131
7.6	The student has completed solving the problem and types in her final answer. When she clicks the “Check My Answer” button, the system will check that she has actually solved the problem (by checking the handwriting space for input) and done it successfully (by checking the final, typed-in answer). In this case, she has, so she will be allowed to move on. . . . .	132

- 7.7 In an alternative scenario, the student has completed solving the problem and types in her final answer. When she clicks the “Check My Answer” button, the system will check that she has actually solved the problem (by checking the handwriting space for input) and done it successfully (by checking the final, typed-in answer). In this case, she has *not* solved the problem successfully: she has forgotten the negative sign when transcribing “-1308” in the third step, so the tutoring intervention begins. . . . . 133
- 7.8 The system launches recognition of the handwriting input space containing the student’s complete solution. The system first extracts the strokes belonging to each step by finding baselines for each line or step of the problem and grouping strokes within steps. These strokes are then iteratively fed into the recognizer; as each character is recognized, the tutor context information about the set of correct options for each step is considered. Once the problem has been completely recognized, the system attempts to determine on which step the student’s error occurred by calculating the deviation of the recognized steps from the correct options and choosing the maximum-deviation step as the most likely to contain the error. These are background processing steps and are *not* shown to the user, but are included here for illustrative purposes. . . . . 134
- 7.9 Once the system has identified a step as the most likely to contain the error, it highlights the strokes associated with that step and prompts the student to revisit her solution beginning with that step. An alternative, shown in the next figure, is to ask the student to verify that the recognition result matches what she had written and commence tutoring once any ambiguity is resolved. . . . . 135
- 7.10 An alternative prompting style to the one presented in the previous figure, which is agnostic about what the student wrote, is the one presented here. The system asks the student to verify that the recognition result matches what she had written and commences tutoring once any ambiguity is resolved. An advantage of using this method is that the system exposes its interpretation to the user immediately, cutting off any error spirals which could occur in the previous prompting style. However, this very exposure could disrupt the student’s learning process. . . . . 136
- 7.11 Following the prompting style from the previous figure, if the student indicates that the recognition result is not correct (*i.e.*, it does not match her intended input), the system brings up a text box in which the student can enter her input for that step unambiguously. If it is an error, tutoring commences as per Cognitive Tutor methods. If it is not an error, the recognizer can either iteratively attempt to identify another error step, or turn this problem into a worked example by providing the solution for the student to study, depending on its error identification confidence. . 137



# List of Tables

4.1	All user studies performed to support this dissertation. Note that although all studies include a handwriting input modality, none use real-time recognition or provide feedback to the participants as to what specifically the system thinks was written. . . . .	50
4.2	Means tables for all measures reported from the Math Input Study. . . . .	55
4.3	Samples of user input in the four conditions of the Math Input Study. Note that the typing sample contains three errors: the use of '/' instead of ' ' and '1' instead of 'x' (twice). In the multimodal entry, the point at which the user substituted which quantifier symbol was used was not considered an error. . . . .	57
4.4	Distribution of errors per equation by condition. . . . .	59
4.5	Means tables for all measures reported from the Lab Learning Study. . . . .	65
4.6	List of all qualitative comments by students in answer to the question: "Did you like the modality you used during problem-solving? Why or why not?" . . . . .	69
4.7	Results of bivariate correlations between performance during training and on each of the two tests given during the study (pre-test and post-test). The correlation results are grouped by condition. † indicates significance at the 0.05 level; ‡ indicates marginal significance at the 0.10 level. . . . .	71
4.8	Experimental design matrix showing the full cross of the three experimental factors; the shaded cells indicate conditions in the Cognitive Tutor Study. . . . .	74
4.9	Means tables for all measures reported from the Cognitive Tutor Study. . . . .	80
4.10	Cross tabulated totals of responses of students to the question: "What was the source of the mental effort?" . . . . .	84
4.11	Results of bivariate correlations between performance during training (full credit only) and on each of the three tests given during the study (pre-test, post-test, and retention test). The correlation results are grouped by condition. † indicates significance at the 0.05 level; ‡ indicates marginal significance at the 0.10 level. . . . .	85
5.1	The set of all math symbols used throughout this dissertation. . . . .	93

5.2	Summary of baseline accuracy results for the three recognizers tested in this dissertation. Numbers in parentheses indicate the number of samples per symbol per user that yielded that accuracy value. Microsoft's TabletPC recognizer could not be trained to a writer-dependent model at the time of these experiments. Symbol accuracy is the cumulative match score over all symbols tested. Equation accuracy is average score over all equations, computed via normalized Levenshtein string distance. . . . .	95
5.3	Means tables for baseline accuracy measures reported for the Freehand Formula Entry System and JMathNotes. . . . .	96
5.4	Histogram table of all writer-dependent results for both the Freehand Formula Entry System and JMathNotes. . . . .	97
5.5	Writer-independent recognition results for the Microsoft TabletPC recognizer. . . .	101
5.6	Summary of writer-independent recognition results for the three case studies. . . .	102
6.1	Types of conceptual or other problem-solving errors represented in the corpus of problem-solving examples from the Lab Learning Study's learning phase, grouped by similarity. The frequency column sums to more than 73 because some problems contained examples of multiple error types. . . . .	114
6.2	Means tables for accuracy and other performance metrics reported in this chapter. .	117
6.3	Performance on the error identification task. Standard deviations are not included because these are raw proportions across all folds/data. . . . .	118
6.4	Means table for deviation from correct of the recognition result on all error steps vs on all non-error steps. $N$ is the number of steps. . . . .	122

# **Chapter 1**

## **Introduction**

This dissertation presents the results of explorations into the adaptation of handwriting input for the interfaces of intelligent tutoring systems, specifically for high school algebra. The approach taken in this dissertation includes technical development, pedagogical development, and user studies. Topics addressed include what the advantages of using handwriting are in terms of learning and usability, what factors contribute to these advantages, and how these advantages can be leveraged in real tutoring systems. Case studies of three unique recognizers evaluate their handwriting recognition accuracy for this domain. Reasonable writer-independent handwriting recognition rates can be achieved by *a priori* data collection and training, and these can be even further improved via the addition of domain-context information. Furthermore, a realistic tutoring interaction paradigm can be achieved through the methods demonstrated by this dissertation, in spite of imperfect raw recognition accuracy.

### **1.1 Motivation**

This work is motivated symmetrically along two dimensions: the pedagogical needs of students working with intelligent tutoring systems in the classroom, and the technological needs of enhancing and improving handwriting recognition for use in real-world applications. Intelligent tutoring systems are becoming much more common tools for students to use in the classroom, and it is imperative that these systems are able to provide the most seamless and natural learning environments for the students as possible. Handwriting recognition is not perfect, but this dissertation shows that it can be improved for use in a learning application via certain domain-specific techniques.

#### **1.1.1 Limitations of Typing in Intelligent Tutors for Math**

Mathematics training is essential for participation in science and engineering careers. American high school students have a poorer mastery of basic math concepts than their counterparts in most

other leading industrialized nations, as found by the Programme for International Student Assessment (PISA) [51]. There are many theories explaining why U.S. students lag behind their peers abroad in math and other science subjects (*e.g.*, [17, 87]), including teaching style. Other reasons include a shortage of teachers in general, high rates of teacher turnover, and a lack of qualified teachers (*c.f.*, [66, 67]). For example, many teachers teach math without being certified in the subject [129].

These types of issues may be at least partially addressed by supplementing some classroom instruction with one-on-one tutoring. Bloom found that the best human tutors can raise the grade of a *C* student to an *A*, known as the “two-sigma effect” [19]. However, it is clearly not feasible from either a financial or human resources perspective to provide every student in America with an expert human tutor. A potential solution to the scarceness of qualified human teachers and tutors is to use intelligent software math tutors. An intelligent tutoring system is educational software containing an artificial intelligence component. The software monitors the student as she works at her own pace, and tailors feedback and step-by-step hints along the way. By collecting information on a particular student’s performance, the software can make inferences about her strengths and weaknesses, and can tailor the curriculum to address her needs. Although tutoring system have been shown to be quite effective, raising the grade of a *C* student to a *B* [2, 38], they are still not at the level of improvement the best human tutors can provide, which is treated as the *de facto* “gold standard” of the intelligent tutoring systems community.

One area in which tutoring systems may be improved is with respect to the interface they provide to students for problem-solving. Most systems use keyboard- and mouse-based windows-icons-menus-pointing (WIMP) interfaces. Such interfaces are not well-suited for math tutoring systems. These interfaces impose *extraneous cognitive load* [134] on the student, because representing and manipulating two-dimensional mathematics equations can be cumbersome in a typing interface. This cognitive load is extraneous (rather than germane) because using and learning the *interface* is (and should be) separable from the *math* concepts being learned. A more natural interface that can more directly support the standard notations for the mathematics that the student is learning could reduce extraneous cognitive load and therefore yield increased learning (*c.f.*, [134]). Furthermore, young children may be a particularly good audience for handwriting-based interfaces, even without considering learning. Recent studies have shown that children experience difficulties with the standard QWERTY keyboard, making text entry laborious and causing them to lose their train of thought—a sign of high cognitive load [119]—even given the rise in computer use by children. There is also some evidence that children may write more fluently when using a handwriting-based interface than a standard keyboard-and-mouse interface when entering unconstrained text [120].

Anecdotally, teachers say that students have difficulty moving from the computer tutor to working on paper. A teacher might see a student solving a problem on the computer with no trouble, but then see that same student unable to solve a similar problem on his own on paper. The WIMP interface may act as a crutch. Even with pedagogical *scaffolding* [28], the knowledge students acquire may become most strongly activated by (or linked to) the visual cues of the interface, making

it difficult for them to access their conceptual knowledge without those cues. In this case, students may not be engaging in deep learning, and their knowledge is imperfect, making transfer to new skills or different situations difficult or impossible.

This dissertation uses handwriting input in these tutors, mimicking the paper-based learning experience in that it is unconstrained. It is not an assumption of this dissertation that that students will always work on paper and so interfaces must be modeled based on paper affordances, since it seems that more and more societies are moving away from being paper-based; but rather, this dissertation posits that current keyboard-based interfaces for math tutoring systems constrain student problem solving too closely.

The use of handwriting interfaces has particular pedagogical advantages in the domain of learning environments, especially for the mathematics domain. Studies conducted as part of this dissertation find that handwriting input for math is faster than in typing interfaces. The efficiency of a handwriting interface for a math tutor allows students to complete more problems in the same amount of time (*c.f.*, [56]). Second, the use of handwriting rather than a menu-based typing interface may result in a reduction of extraneous cognitive load on students during learning. Extraneous cognitive load (*c.f.*, [134]), in this context, can be thought of as a measure of how much mental overhead is experienced as a result of interface-related tasks while one is also trying to learn a mathematical concept. Additionally, students may prefer handwriting, especially if it makes the problem-solving process easier or more natural for them, which leads to increased engagement during tutoring (*c.f.*, [47]). Finally, in mathematics, the spatial relationships among symbols have inherent meaning, even more so than in other forms of writing. For example, the spatial placement of the  $x$  in the following expressions significantly changes the meaning:  $2x$  vs  $2^x$ . Handwriting is a much more flexible and robust modality for representing and manipulating such spatial relationships, which become more prevalent as students advance in math training. This dissertation explores these areas in more depth in order to establish a theoretic foundation on how to achieve better learning gains using an appropriate interface.

### 1.1.2 Limitations of Handwriting Recognition

Most intelligent tutoring systems rely on standard keyboard- and mouse-based graphical user interfaces (GUIs); however, the reasons tend to be technological rather than pedagogical. Handwriting recognition is often seen as being in its early stages of development, too inaccurate for use with real users. In addition, developers of intelligent tutoring systems are typically not experts on current handwriting recognition technology. Handwriting recognition systems can range from entirely opaque (black-box) and non-customizable implementations such as the Microsoft TabletPC recognizer<sup>1</sup>, to entirely open and customizable, but undocumented, systems such as the Freehand Formula Entry System (FFES) [131]. Both extremes make handwriting recognition technology somewhat inaccessible to non-experts. One potential outcome of disseminating this dissertation's

---

<sup>1</sup><http://msdn.microsoft.com/en-us/library/aa510941.aspx>

findings in favor of using handwriting input in learning applications is that handwriting recognition experts and intelligent tutoring systems experts may more highly prioritize working together toward making it easier to incorporate such tools into real user interfaces.

In order to get maximum benefit from the automated instruction provided by an intelligent tutor, student entries must be interpreted by the computer tutor in order for it to be able to offer instructional feedback. Handwriting recognition technologies have been studied since the 1950s (*e.g.*, [6, 97]). Although they have advanced significantly since the first systems, they are still far from 100% recognition accuracy. For some applications, small recognition imperfections may not be critical. However, for students learning new concepts, a system making errors recognizing their input, and then presenting these errors to the student, introduces new problems. Requiring the students to correct the system simply moves the extraneous cognitive load from learning interface menus to monitoring recognition performance. How accurate the recognition has to be in order to successfully interpret student input and provide adequate instructional feedback is an open question partially addressed by this dissertation. The tutor should not provide inaccurate instructional feedback, which would have potentially serious learning consequences. This work investigates two methods to address this concern. First, training the recognition engine in advance on a data corpus of samples from a student population helps by increasing baseline writer-independent accuracy. Second, the handwriting engine can be adapted to utilize alternate sources of information from the problem-solving context of the tutoring system in order to more accurately interpret student input.

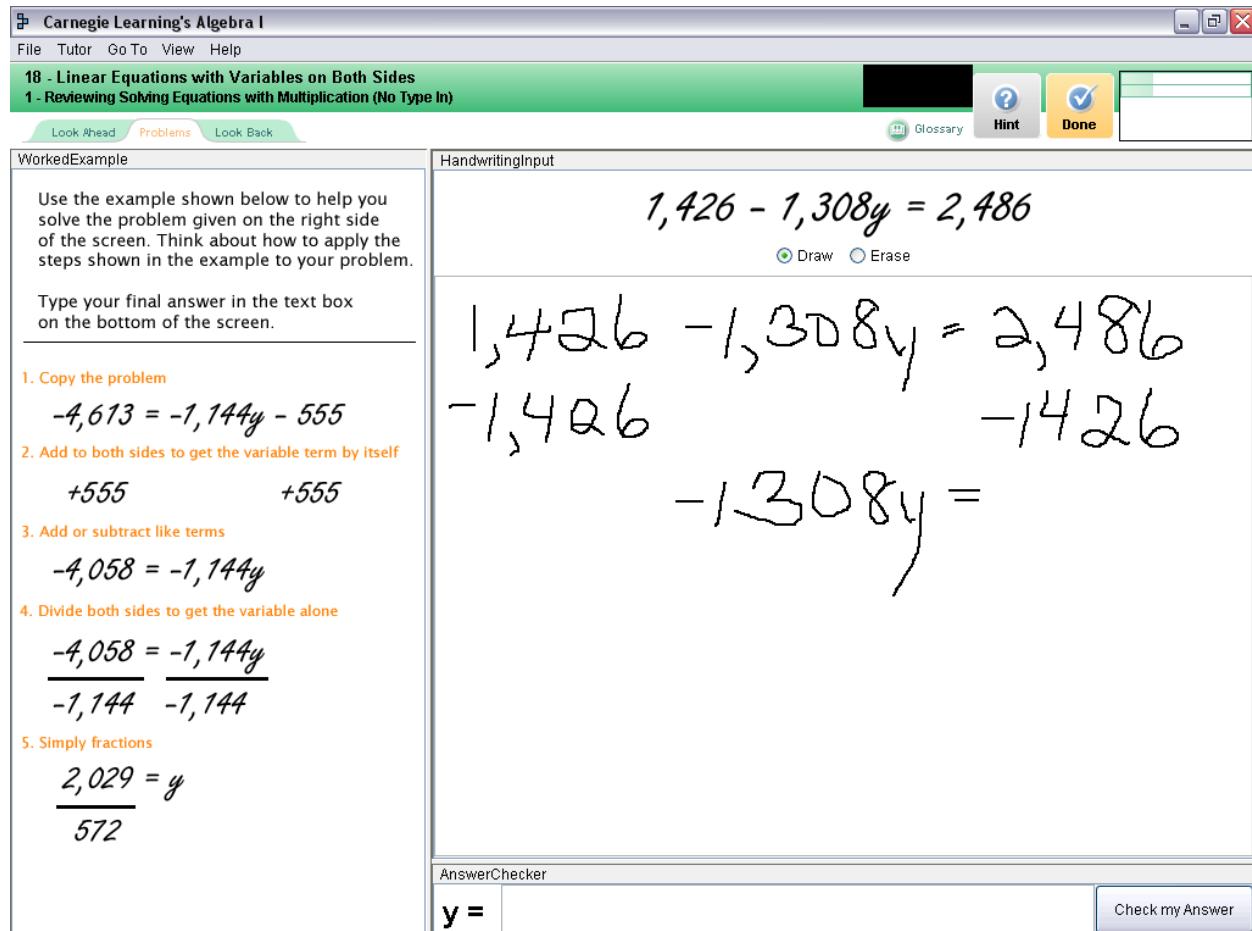
In summary, although current methods for interpretation of handwritten equations may in fact not be adequate for classroom use, the methods explored in this dissertation yield useful results and techniques for the successful incorporation of handwriting recognition into computer tutors.

## 1.2 Concept

The thesis statement of this dissertation is as follows:

*The use of handwriting interfaces in intelligent mathematics tutoring software can yield higher learning gains in students through lower cognitive load than the use of standard typing interfaces. An important part of achieving this effect is increasing recognition accuracy to a level sufficient for adequate instructional feedback.*

In order to investigate this thesis statement, a testbed is used consisting of an intelligent tutor for math learning that allows students to enter their solutions in an unconstrained problem-solving space via handwriting input. A screenshot of this prototype for the algebra equation-solving domain is shown in Figure 1.1. The interface components are explained in more detail in Chapter 7. This system has been evaluated both in the laboratory and in the classroom on measures of usability, learning, and cognitive load.



**Figure 1.1:** A screenshot of a proposed tutoring system for algebra equation solving that allows students to enter their solutions via handwriting in an unconstrained problem-solving space. Different versions of this prototype were used throughout this work.

## 1.3 Approach

The research approach taken in this work encompasses the following four goals:

- **Evaluate performance of novice users entering mathematical equations using the handwriting modality.** This dissertation advances learning theory by examining (a) how handwriting affects student interactions with intelligent tutoring systems with respect to **speed**, **errors**, and **engagement**, and (b) what advantages the handwriting modality provides to learning applications with respect to decreasing **cognitive load** and increasing **learning gains**.

- **Evaluate performance of existing handwriting recognition technologies for equation entry.** This dissertation explores (a) how handwriting engines perform with respect to recognition errors during equation entry, and (b) how it is possible to improve recognition accuracy via *a priori* writer-independent training and on-the-fly consideration of domain-specific context information.
- **Develop handwriting interfaces to support intelligent tutoring systems for mathematics.** Technical products of this dissertation include (a) a multimodal system using handwriting to enhance math tutors, and (b) co-recognition algorithms using domain-specific context information to enhance the robustness of the system.
- **Evaluate potential handwriting-based math tutors in *in vivo* experiments.** Iterations of the developed handwriting-based math tutor are evaluated for different learning tasks, such as solving algebraic equations, with high school students in the Pittsburgh Science of Learning Center’s LearnLab environment. The final product consists of design and educational guidelines toward applying such interfaces to a handwriting-based tutor for algebra equation solving that may use the recognition enhancements developed in this dissertation.

Several systems already exist for handwriting-based mathematics input both online and offline (see § 2.3.4 for more details), but are not widely available to most novices. This dissertation (1) develops an interface for an intelligent tutoring system for beginning algebra equation solving that will allow middle and high school students to use handwriting input to solve equations on the computer, (2) investigates potential learning gains in the use of handwriting interfaces for the intelligent math tutor, and (3) advances handwriting recognition technology via machine learning techniques to improve accuracy to a level suitable for use by students in a learning situation.

Existing recognition technologies that were relatively mature were adapted for this dissertation, rather than implemented from scratch. Their design and optimization have been the subject of research for decades. In order for such technologies to become *usable* in actual interfaces for real users, they must be *incorporated* into interfaces for real users. The ways in which such interfaces can provide advantages for users, as well as the ways in which such interfaces can be adapted for users, are studied in this dissertation. The focus of this work is on the intersection of the fields of educational technology and handwriting recognition, with respect to the advantages both can give to the other. The component handwriting recognition engine used in this work is FFES [131, 149], although others were explored (discussed in § 5.1). In addition, intelligent tutoring systems are already highly effective learning environments. Existing successes from past research on intelligent tutoring systems are leveraged in the exploration of ways to continue to improve them. The intelligent tutoring system used is Carnegie Learning’s Cognitive Tutor for Algebra<sup>2</sup> (*e.g.*, [5, 37]). Details on each of these systems are given in § 2.5.5 and § 2.5.6, respectively.

---

<sup>2</sup><http://www.carnegielearning.com>

## 1.4 Document Organization

This dissertation is organized into the following chapters. Chapter 2 surveys the related work in the areas of handwriting interfaces, especially for mathematics, intelligent tutoring systems and pedagogical theory, and some machine learning data-driven classification approaches. Definitions of key terminology used in this dissertation are given in § 2.6. The dissertation then establishes the ways in which handwriting input can help in the domain of intelligent tutoring systems for mathematics, especially algebra, by first discussing the theoretical contributors in Chapter 3, and then by describing in Chapter 4 three foundational studies that were conducted. The dissertation next discusses the technical methods used to enhance handwriting recognition accuracy for use in a real application with real students learning algebra, by first describing the process of choosing a recognition engine and establishing baseline writer-independent accuracy rates in Chapter 5, and then by describing the process of incorporating domain-specific context information into the recognition process in Chapter 6. Chapter 7 lays out a set of interaction scenarios that could be realized because of the results presented in this dissertation. Finally, Chapter 8 presents the conclusions of this dissertation, contributions and limitations, and outlines areas for future work.

# Chapter 2

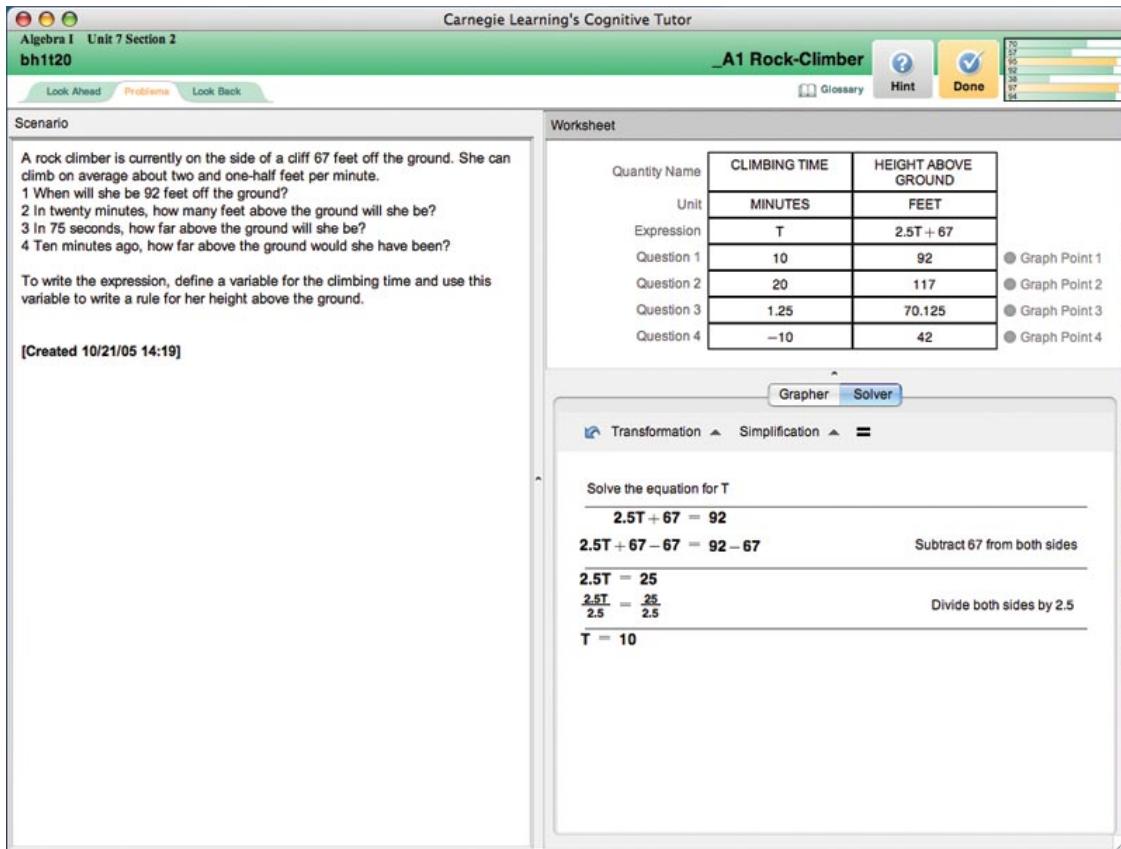
## Related Work

### 2.1 Intelligent Tutoring Systems and Cognitive Tutors

Intelligent tutoring environments for problem solving have proven to be highly effective learning tools [5, 142]. Many of these environments present complex, multi-step problems and provide the individualized support that students need to complete them: step-by-step feedback and context-specific problem-solving advice. They are two or three times as effective as typical human tutors, but only half as effective as the best human tutors [35], who can improve student learning by two standard deviations [19].

Cognitive Tutors are a specific class of intelligent tutoring systems that are designed based on cognitive psychology theory and methods and that pose authentic problems to students and emphasize *learning-by-doing* [5]. Each Cognitive Tutor is constructed around a cognitive model of the knowledge students are acquiring, and can provide step-by-step feedback and help as students work. They have been created for a variety of learning domains, including algebra, geometry, foreign languages, chemistry, computer programming and more. Cognitive Tutors for mathematics are in use in over 2,600 schools in the United States. A screenshot of a typical Cognitive Tutor interface for an algebra unit is shown in Figure 2.1, showing important interface components such as the worksheet and equation solver tool.

Cognitive Tutors, and other intelligent tutoring systems, are beginning to explore more natural interfaces such as natural language processing of typed input (*e.g.*, [3, 53]), spoken dialogues with conversational agents (*e.g.*, [16, 60, 82]), and animated characters with gesture-based interfaces (*e.g.*, [101]). Most systems still currently rely on standard windows-menu-icon-pointing (WIMP) interfaces. The prevalence of WIMP interfaces is due in part to the fact that the technology available to most students in the classroom is limited to keyboard-and-mouse—this situation is changing however, as students receive PDAs or TabletPCs in the classroom [68, 147]. In addition, research into handwriting recognition technology has not emphasized making recognizers easy to use and to adapt for new domains by non-experts, and recognition systems are often inaccessible or opaque to anyone but the system’s own developers. However, there is reason to expect that the use of hand-



**Figure 2.1:** A screenshot of the Cognitive Tutor interface for an algebra unit involving formulating the relationship between two variables.

writing interfaces could have particular advantage in the domain of math learning environments, as this dissertation establishes.

## 2.2 Learning Science and Educational Technology

The learning sciences comprise an interdisciplinary field borrowing from the traditions of cognitive science, computer science, psychology, education, neuroscience, and social science to study how people learn. In addition, the learning sciences and the related field of educational technology are concerned with designing and implementing learning innovations. The science of learning re-invented the fact- and procedure-based educational model of the early twentieth century in favor of an educational system based more on deeper conceptual understanding and critical thinking [126].

The *Cambridge Handbook of the Learning Sciences* [126] outlines the following five thrusts of

modern learning science research and emphases in learning and teaching:

- Deeper conceptual understanding,
- Focusing on the process of learning (not just how to teach),
- Creating complex and rich learning environments,
- Building on a learner's prior knowledge, and
- Encouraging student reflection on knowledge and concepts.

These five thrusts represent a shift in thinking away from exclusive *drill-and-practice* types of instruction; in drill-and-practice instruction, students are asked to repeatedly review previously learned concepts until they have reached a predetermined level of mastery [126]. Instead, a new conceptualization of learning evolved; rather than characterizing learning as a process of forming connections between stimuli and response [23], scientists began emphasizing more contextual, conceptual and applied types of instruction. It was found that:

...children retain material better, and are able to generalize it to a broader range of contexts, when they learn deep knowledge rather than surface knowledge, and when they learn how to use that knowledge in real-world and social contexts [126].

The learning sciences owe much of the methodologies and tools for testing theories of learning and learning innovations to the field of cognitive science (*c.f.*, [23]), which emerged in the mid-1950s and drew together many different disciplines such as anthropology, linguistics, computer science, neuroscience, philosophy and psychology to formulate foundations of learning theories, anchored within models of the brain and memory.

Today the learning sciences emphasize “learning with understanding” [23] and *active learning* [21], which is a philosophy of instruction that places the responsibility of learning on the learner. Active learning can be encouraged in many ways, including role-playing, debate or class discussion, cooperative learning, peer tutoring [145], question-asking [7], and studying examples [135]. A further component of active learning is *metacognition*. Students can be metacognitively aware of their own cognitive processes, and in the case of learning, can have knowledge about what they know, how difficult something is for them to learn, and to what degree they understand something [50]. Learning science and intelligent tutoring systems attempt to support metacognition, even going so far as to tutor it through, for example, requiring students to engage in self-explanation (*e.g.*, [2, 31]).

A self-explanation is a meaningful and correct explanation of a step in the students own words [31]. When students engage in self-explanation, they tend to develop a deeper understanding of the material. Novices tend to match surface features of a problem, like diagrams and problem statement wording, with those in a worked-out example. In contrast, experts use the principles and deep structure, that is, conceptual knowledge that generalizes across problems, as criteria for matching a worked-out example to a problem [141].

### 2.2.1 Worked Examples as Instructional Interventions

Actively studying worked examples is a type of active learning [135]. A worked example is a step-by-step demonstration of how to perform a task or how to solve a problem [33]. Sample worked examples are shown in Figure 2.2, from geometry (a) and from physics (b). In 1985, Sweller proposed studying worked examples as an alternative to problem solving in order to limit cognitive load caused by mental search [135]. Studying such worked examples have been shown to be an effective strategy for teaching problem-solving skills, because such examples reveal the mental models of problem-solving experts to the novices studying the examples, who may have misconceptions and false models [93].

Many studies have established the benefit of using worked examples as a supplement to problem-solving instruction, including [34, 104, 123, 140], although there is no consensus on how and when to provide them or what they should look like [128]. Much of the benefit of worked examples requires that students engage in metacognitive self-explanation of the solution steps listed in the example [123].

This dissertation uses worked examples alongside problem-solving tasks as a means of providing students more information when detailed feedback is unavailable.

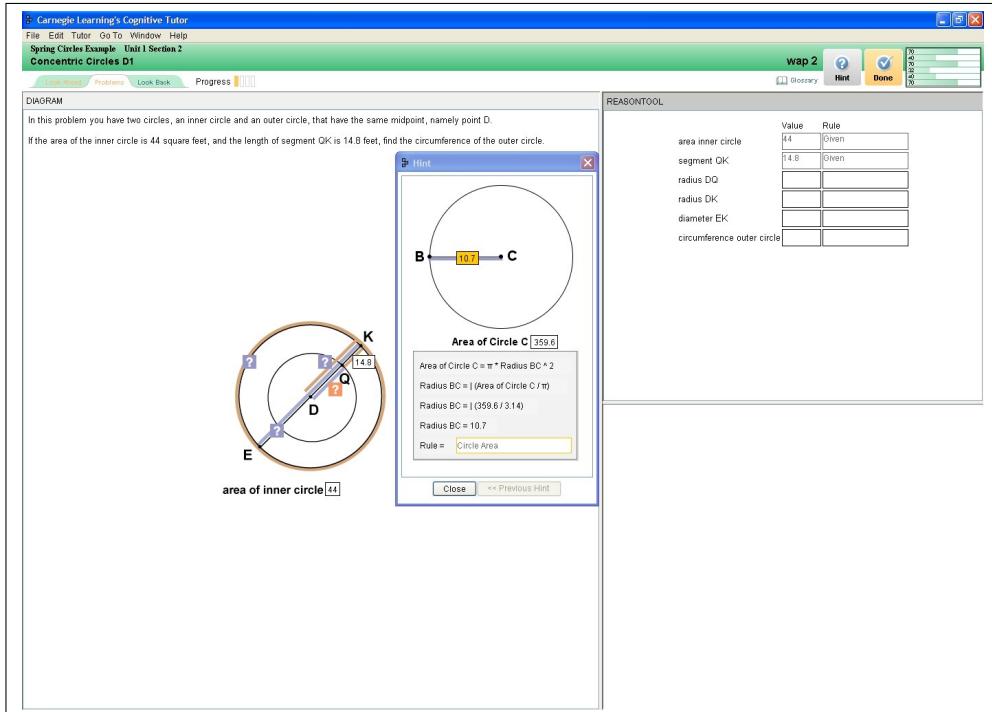
### 2.2.2 Ways to Measure Learning

Learning methods are often compared to one another through controlled experiments, in which students are assigned randomly to one of several instructional methods. Another common method is through quasi-experimental studies, in which students are not assigned individually to conditions, but rather entire classrooms or all students taught by one teacher might be randomly assigned as a group, for practical reasons. The learning method with the most improved learning will be considered “best,” but how does one measure improved learning? Learning gains are measured through *assessments*, usually in the form of a pre-test to assess pre-existing knowledge and a post-test to assess differences in the levels of knowledge after a study is complete.

The Pittsburgh Science of Learning Center (PSLC) is pioneering a deeper way of measuring learning called **robust learning** [99]. Three crucial components that make learning robust are *long-term retention*, *far transfer*, and *accelerated future learning*.

So-called “normal learning” [99] is typically measured via immediate (as in, immediately following instruction) post-tests containing items that are **isomorphic** to the items in instruction. Isomorphic in this case means that the items have similar form, but may have different content. In addition, these tests can include *near-transfer* items.

The concept of *transfer* refers to the application of a skill learned in one situation to a different but similar situation [130]. For example, in algebra equation-solving instruction, students might first be taught problems such as  $4x + 3 = 9$ . Problems that were similar, such as  $6x - 7 = -4$ , would be considered isomorphic; problems involving slightly harder skills, such as  $5 + \frac{2}{7}x = 29$ , would be considered near transfer. Examples of far (or farther) transfer items might be: (a) more



(a) A sample of a worked example in the *geometry* domain. Figure reprinted courtesy of Salden et al. [124] and the Geometry Self-Explanation Tutor Project.

### Problem Statement

In the circuit shown,  $R_1 = 2.0 \text{ ohm}$ ,  $L_1 = 6.0 \text{ H}$ , and  $V_b = 12.0 \text{ V}$ . The switch is closed. When the current through the resistor reaches  $4.0 \text{ A}$ , what is the energy stored in the inductor?

### Solution

**Variables**

- $T_0$  = switch closed
- $T_1$  = current at  $4.0 \text{ A}$
- $L_1$  = inductance of  $L_1$
- $I_{R1}$  = Current through  $R_1$  at time  $T_1$
- $I_{L1}$  = Current through  $L_1$  at time  $T_1$
- $U_{L1}$  = Energy stored in  $L_1$  at time  $T_1$

**Equations**

Equation	Source
1. $L_1 = 6.0 \text{ H}$	Given
2. $I_{R1} = 4.0 \text{ A}$	Given
3. $I_{R1} = I_{L1}$	Kirchoff's Junction Rule
4. $U_{L1} = 0.5 * L_1 * I_{L1}^2$	Energy stored in an Inductor
5. $U_{L1} = 0.5 * 6.0 \text{ H} * 4.0 \text{ A}$	4,3,2,1
6. $U_{L1} = 48 \text{ J}$	5

(b) A sample of a worked example in the *physics* domain. Figure reprinted courtesy of Ringenberg et al. [123] and the Andes Physics Tutor Project.

**Figure 2.2:** Sample worked examples from two different domains. Note the differences in level of detail.

difficult problems that have additional features, such as  $2x - 7 = -5x + 9$ ; or (b) a conceptual format assessing knowledge of features, such as, “Which of the following does not belong and why? a.  $3 + 4x$  b.  $3 + (-4x)$  c.  $4x - 3$  d.  $4x + 3$ .”

A further measure of learning might be to determine how long or how strongly students retain the knowledge they acquired during certain instruction. A **retention** test can be administered after a delay to measure how well students remember or can reconstruct acquired skills. There are no rules about the length of this delay, but a reasonable long-term retention interval (the time between the end of instruction and the test) should be as least as long as the time of the instruction (the time between the beginning and end of the instructional period of the study) [99]. Differences between control and treatment instruction tend to be harder to detect at longer retention intervals, but the longer the interval at which a difference is detected, the greater the evidence of the treatment leading to long-term retention [99].

In the learning-oriented studies presented in this dissertation, learning is measured via normal post-tests, near-transfer items, and long-term retention tests.

## 2.3 Handwriting Recognition Techniques and Systems

Handwriting recognition has been an active area of research since the late 1950s and 1960s (*e.g.*, [25, 41, 54, 97]), even for mathematics (*e.g.*, [6]). Techniques for the recognition of handwritten mathematics range from the recognition of a page of notes after it has already been written (offline) (*e.g.*, [48, 78, 94]), to the recognition of a user’s handwriting even while she is in the process of writing (online) (*e.g.*, [18, 40]). Approaches to the pen-based handwriting recognition problem have included statistical classifiers, support vector machines (SVMs), clustering, nearest neighbor algorithms, Bayesian networks, fuzzy logic, decision trees, dynamic programming, Hidden Markov Models (HMMs), neural networks, expert systems, hardware solutions, and combinations of these techniques [62, 73]. A brief discussion of several of the most relevant techniques used in handwriting recognition systems is provided here; for excellent surveys of the field, see [29, 73, 139].

### 2.3.1 Neural Networks

Neural networks are learning algorithms that are modeled after neurobiological systems [95]. They consist of a network of nodes that can accept real-valued inputs and produce real-valued outputs. The connections between nodes in a neural network are associated with activation weights which determine the effect of that connection. Neural networks with only one input layer and one output layer are limited to the representation of linear functions. However, neural networks can be extended to contain hidden layers of nodes which provide the ability to learn a much larger set of functions. Due to this feature of neural networks, the weights learned are not usually interpretable by humans. A common algorithm for training neural networks is called *back-propagation*. This

algorithm feeds a training instance forward through the network and then calculates the errors backward through the network to use in readjusting the weights. This process iterates many times through the training set until a termination condition is reached. Although it may take a long time to train a neural network, evaluating a new instance is relatively much faster. An example of a handwriting system that uses neural networks for recognition is [148].

### 2.3.2 Support Vector Machines

SVMs [43, 143] are a method of classification that consider two input datasets (*i.e.*, clusters) as vectors in an  $n$ -dimensional space for which a plane can be constructed that attempts to maximize the margin between the two data sets. The margin is defined as the distance between the data set and the plane. Samples from each class that lie on the margin are called the support vectors. SVMs are typically binary classifiers, that is, distinguishing between two possible clusters only. However, multiple SVMs can be combined to achieve a multiclass classifier by creating several one-*vs*-rest classifiers, one for each class of interest. An example of a handwriting system that uses SVMs for recognition is [137].

### 2.3.3 Hidden Markov Models

HMMs can use information about transition probabilities in sequential data to augment classification based on observation probabilities. HMMs can answer questions such as the following: what is the probability of a given sequence of observations given a model, and what is the sequence of true states that best explains the set of observations [111]. Words are sequential streams of characters and therefore an HMM can take advantage of within-word context to enhance recognition accuracy in handwriting, although this is more difficult in mathematics.

For fully-observable data, learning an HMM is trivial: one simply counts the occurrences of each observation, state and transition. To classify a pattern, HMMs use a dynamic programming technique known as the Viterbi algorithm [144] to find the most likely explanation of a series of observations. The algorithm iterates over the sequence of observations, one for each state, storing the highest probability out of all possible ways to have reached each possible value of this state from the previous state. Once it reaches the final state, it then backtracks and builds the path that would have generated the most likely assignment for the entire sequence.

An example of a handwriting system that uses HMMs for recognition is [30].

### 2.3.4 Pen-Based Handwriting Recognition Performance

As discussed above, a broad range of approaches have been used to address the pen-based handwriting recognition problem. These approaches present different speed, accuracy, and memory demands and tradeoffs, but none of them significantly outperforms all others in every respect [62].

Vocabulary size often dictates the type of approach used. Large vocabularies increase the difficulty of the recognition task because there can be more similar pairs of words in the dictionary as its size increases. Typically, modern approaches to handwriting recognition using neural networks for such large vocabularies are not frequently used as standalone classifiers, but as part of hybrid approaches, where they are used to estimate *a priori* class probabilities or grapheme probabilities. Recently, HMMs have become the dominant approach to automatic speech and pen recognition due to their power in representation [73].

A technical limitation of recognition technologies such as handwriting is that recognition accuracies are not perfect. LaLomia [74] provided an argument and evidence that adults will tolerate accuracy rates in handwriting recognition for a variety of tasks only as low as 97%.<sup>1</sup> It is only through writer-dependent recognition that current systems get even close to achieving this high level of accuracy. It is difficult to quote a state-of-the-art handwriting recognition accuracy rate because few rigorous evaluations have been done from a usability perspective on handwriting recognizers for any domain, a weakness identified early on in the literature [57] but never pursued. Typically developers report accuracy numbers without much context or detail. Many of the evaluations that do exist are now out-of-date (*e.g.*, [88, 125]), as recognition technology has continued to advance over the past decade or so. Handwriting recognition systems for math are especially lacking in formal evaluations. MathPad<sup>2</sup> is one of the few recent systems to perform a complete user study designed to gauge factors such as user performance, satisfaction, ease-of-use, and learnability along with recognition engine performance [77]. Still, the study reported in that paper involved only seven users and did not report statistical significance of findings because it had only one condition, although more rigorous studies are planned. Other recent studies have similarly small numbers of users or report system accuracy as a footnote in the context of a larger discussion of a new algorithm or technique (*e.g.*, [131]).

The Lipi Toolkit is a project to provide tools to allow data processing and annotation, and adapting recognizer engines to use in applications [89]. It is in the early stages of development and therefore can currently only support a limited set of recognition algorithms and isolated characters (with bounding boxes, for instance). A formal set of evaluations on an HMM recognizer is reported in [84], but the domain is offline handwriting recognition of cursive handwriting. Its methodology can be informative but must be extended in order to apply to online (real-time) recognition applications.

User-centered design requires that both halves of the equation be considered when developing an application to use handwriting input: both the accuracy of the system itself and how a user reacts to and interacts with the system. Frankish et al. [52] explored the relationship between recognition accuracy and user satisfaction and found that it was highly task-dependent: some tasks (such as a form-filling task) were rated as very suitable for pen-based input no matter what the recognition accuracy level was, whereas others (such as a diary task) were only rated highly when accuracy was also high. The type of recognition supported may have also impacted these results;

---

<sup>1</sup>Note that human recognition rates are around 96.8% [125] and see § 2.3.5 for acceptable accuracy with children.

the system only accepted isolated characters printed within boundary boxes. As newer, more natural methods of handwriting input become available, it is important to re-evaluate them from a usability perspective.

Some handwriting recognition researchers have implied that the burden is on the user to “adapt” her handwriting style as she learns the idiosyncrasies of the particular recognizer (*c.f.*, [52]). In contrast, the foundational approach to this dissertation work is that the user should never have to adapt to the system, but it should instead be the other way around. In fact, researchers have pointed out that errors in handwriting input tend to be constant over time [88], implying that users will not reliably adapt to specific recognizers.

### 2.3.5 Handwriting and Child-Computer Interaction

While higher recognition accuracies may manifest in certain domains such as beginning algebra equation solving due to the limited symbol set and grammar used, the fact that middle and high school students are the target audience of this work may itself hurt performance. Read et al.<sup>2</sup> have done extensive research on the area of designing, developing, and deploying user interfaces that utilize handwriting or pen-based input for children. Much of the work has informed this dissertation and the most relevant aspects are described here.

While work with adults has found that 97% accuracy is considered “acceptable” by users of a system with handwriting input, Read found that 91% may be closer to the truth for children [117]. Though the study in [117] was quite limited, its results indicates a trend in a direction that could be useful for designing these interfaces for students: knowing that lower accuracy may be acceptable may help system designers be more confident about including handwriting input in their systems for kids. Reasons for this difference in acceptability of errors include the fact that children find handwriting input to be very appealing and engaging, thus increasing their overall tolerance for the system making errors [112].

Children, and therefore students, are a specific population of users that share much in common with their adult counterparts, but who also have their own special usability requirements and purposes for using computers or other technology. Druin and Solomon studied the types of requirements children bring to the table and what features they want in a product, such as “honesty, curiosity, repetition, and control” [44]. It can be extremely difficult, if not impossible in some cases, for recognition-based interfaces to provide these features. Honesty and control are particularly difficult, due to the ambiguous nature of the interface, which may provide conflicting feedback to the children using the system. However, studies show that children experience difficulties with the standard QWERTY keyboard, making text entry laborious and causing them to lose their train of thought [119]. There also is some evidence that children may write more fluently when using a handwriting-based interface than a standard keyboard-and-mouse interface when entering unconstrained text [120]. Deciding what type of interface to provide for a given application depends on the trade-off between the pros and cons of each input modality, based on these findings.

---

<sup>2</sup><http://www.chici.org/>

While observational studies of children using pen-based input and handwriting recognizers have been undertaken that determined some specific types of user errors (*e.g.*, [115]), the studies were in the domain of creative writing rather than math. In addition, these studies have used a paradigm in which the input device and display screen are separate, requiring the student to look at one *or* the other. In the studies reported in this dissertation, onscreen input was used in order to eliminate many of these problems, although some still persist and others arise. Finally, while these studies were conducted in the classroom, they did not address the topic of learning, only entering input on the computer; their findings may shed light on the domain addressed in this dissertation but it is not clear to what degree the results will generalize.

## 2.4 Interfaces for the Math Domain

### 2.4.1 Traditional Input Modalities

The tools currently available for entering and manipulating mathematics on the computer are sharply divided between expert mathematics users and novices. For the purpose of this dissertation, novice users are defined as those who are still learning mathematical concepts and notations, or who need to use or include mathematics within text documents, but who are not professional mathematicians. Entering algebraic equations is the focus of this dissertation, but similar limitations exist for other, more complex mathematical operations. Current standard interfaces for entering mathematical equations into computers are largely limited to keyboard- and mouse-based interfaces. Mathematics tools that use a typing interface often require the user to become an expert at a new programming language (*e.g.*, MapleSoft’s Maple<sup>3</sup>, The MathWorks’ Matlab<sup>4</sup>, and Wolfram Research’s Mathematica<sup>5</sup>). These programs have a large learning curve, even for mathematics experts, and therefore can be not only difficult and inaccessible for novices but also slow for experts to use. Furthermore, handwritten or typeset mathematics often appears in higher-dimensional layouts, enabling the representation of, for example, both superscripts and subscripts. Figure 2.3 shows examples of the interface of each of the most commonly used math input tools currently available. Most computer interfaces are optimized for entering linear text [131]. Linear input methods might inhibit mathematical thinking and visualization, especially for some learning tasks.

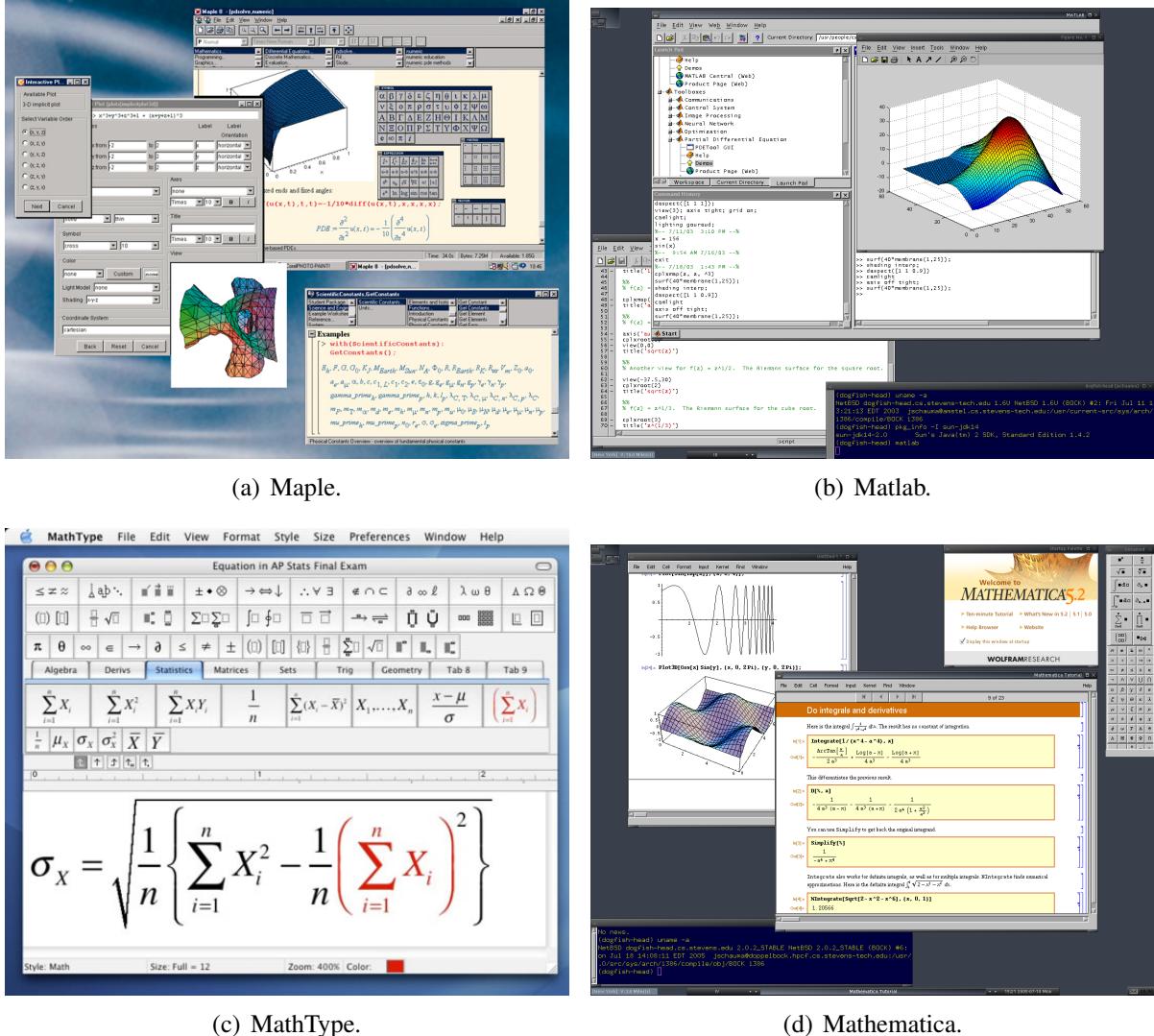
Mathematics interfaces that do not require users to linearize their input are called template-based editors, which force users to select pre-defined mathematical structure templates (*e.g.*, fractions, superscripts, subscripts) from a menu or toolbar and then fill in the templates with numbers and operators by typing on the keyboard. Users can construct a representation of higher-dimensional mathematics, but must do so in a top-down manner, making later structural changes difficult [131]. The most commonly accessible such tool for novices is the Equation Editor in

---

<sup>3</sup><http://www.maplesoft.com>

<sup>4</sup><http://www.mathworks.com>

<sup>5</sup><http://www.wolfram.com>



**Figure 2.3:** Examples of user interfaces for the most common computer tools for mathematics. Clockwise from upper left: Maple, Matlab, Mathematica, MathType.

cluded in Microsoft Office; the professional version of this equation editor is MathType<sup>6</sup>. Worthy of note is that Microsoft has an extension to the Equation Editor for the TabletPC that allows handwritten input<sup>7</sup>. However, because it is not customizable by the end-user or an application developer, it cannot be easily adapted to new domains such as math learning, making it suboptimal for use in research into new handwriting recognition applications beyond the original goals of EquationWrite.

### 2.4.2 Handwriting Input for Math

Unlike such computer-aided math input tools, writing math allows the use of paper-based mathematical notations simply and directly. It is therefore natural and convenient for users to communicate with computers this way [20]. Because pen-based input can use traditional paper-based notations, it may be more suited for entering mathematics on computers. However, pen-based interfaces for mathematics are not widely available. In the past, computer recognition of mathematics has been limited to the recognition of a printed page of mathematics, but now with the prevalence of TabletPCs and PDAs (personal digital assistants) which offer handwriting as a main mode of input, *online* (that is, while in the act of writing) human handwriting recognition is becoming more important.

Several research and commercial systems do exist that allow users to input and/or edit mathematical expressions via handwriting input. MathPad<sup>2</sup> [76] is among the most robust and complex. In MathPad<sup>2</sup>, users can write out mathematics equations and the system animates the physical relationships given by these equations, for example, to animate a pendulum or oscillating sine curve. Other systems such as xThink's MathJournal<sup>8</sup> allow the sketching and writing of mathematics, but rely on in-context menus to allow users to perform manipulations. In addition, even traditional keyboard-based math software such as Microsoft's Equation Editor [106] are now offering handwriting-based input, although limited in the amount of the equation that can be written or in what can be done as far as manipulation of the equation once it is input. Finally, Littin's recognition and parsing system [83], Natural Log [91], inftyEditor [70], FFES [131], and JMathNotes [137] are simple equation entry/editing programs without the added benefit of sketching or graphing.

The added-value of the work of this dissertation on handwriting input for tutoring software is that the focus is on *learning* mathematics. Most other systems focus only on letting users input mathematics. They do not provide a structured approach to learning how to perform mathematical operations conceptually; they assume their users already know how. There is at least one system that does consider education: Jumping Minds' Practice series<sup>9</sup>. The Jumping Minds series has a simple interface which use an instructional paradigm similar to drill-and-practice. Simple prob-

<sup>6</sup><http://www.dessci.com>

<sup>7</sup><http://www.microsoft.com/windowsxp/downloads/tabletpc/educationpack/overview4.mspx>

<sup>8</sup><http://www.xthink.com/MathJournal.html>

<sup>9</sup><http://www.jumpingminds.com>

lems such as beginning arithmetic are provided to students one after another with feedback only of the type “Correct!” or “Try again!” Oviatt has investigated the use of pen-based input for geometry learning [102]. However, in neither of these works is there tailored feedback or a model of student learning, both of which are significant contributors to the advantage of Cognitive Tutors (*c.f.*, [5]).

## 2.5 Methods and Tools Used in this Dissertation

### 2.5.1 Wizard-of-Oz

The Wizard-of-Oz method can be used to evaluate an interface or interaction afforded by a technology for which the computer component would be costly or difficult to develop. In this technique, a human takes the place of the computer (usually this fact is unknown to the user), and performs whatever computation is needed. For instance, in recognition systems, a Wizard-of-Oz set-up would involve a human performing recognition rather than the computer. The user’s input would be sent to the Wizard, who would then send back the “computed” results. To improve speed of response, some level of automation can be provided to aid the Wizard. This technique was first described in [132], though usually attributed to [59]. It was not referred to by the “Wizard-of-Oz” moniker until [71]. See [113] for a detailed history and summary of this technique.

### 2.5.2 Cross-validation

In machine learning experiments, cross-validation is a method used in order to prevent over-training to the training set. There are several different types of cross-validation (CV), including repeated random sub-sampling CV,  $k$ -fold CV, and leave-one-out CV. The type used throughout this dissertation is  $k$ -fold CV, in which the complete dataset is broken into  $k$  segments, usually five or ten, called “folds.” Then, each fold is used as a testing set iteratively; in each case, the remaining folds are all used together to build the training set. In this dissertation, the data for a particular user are all grouped together in one fold, rather than allowing them to be split across several folds.  $k$ -fold CV has the benefit over other methods that that all observations are used for both training and validation, and each observation is used for validation exactly once. The cross-validation method was first described in the field of statistical analysis in [55].

### 2.5.3 Cognitive Load Self-Report

Cognitive load [134] can be a difficult quantity to directly measure because interpreting a given level of cognitive load depends on the context of its associated performance level [105].

According to [104], the more human-centered concept of *mental effort* can be used as an “index” of cognitive load, and can be measured both via objective techniques and subjective techniques. Objective techniques include physiological parameters such as heart-rate variability, pupillary dilation, blink rate, or galvanic skin response (GSR) [105]. Such physiological measurements

can be invasive and costly to implement, especially during a classroom study. There is a question as to whether these invasive measurement techniques might not themselves interfere and impose their own mental effort. Self-report has been shown to correlate with physiological measures of cognitive load [104], and is a much less invasive and costly measurement technique. In self-report, participants are periodically asked to answer a question, usually on a Likert scale [81], during a task, about their perceived level of cognitive load or mental effort. Much work has been done establishing the validity of such self-report metrics in capturing a representation of participants' mental effort or cognitive load (*e.g.*, [58, 98]).

### 2.5.4 Collaborative Information Retrieval: Ranking Fusion

The problem of how to fuse two independently ranked lists emerges in many domains, especially collaborative information retrieval and search. In metasearch, for example, multiple search engines perform a query toward a particular information-seeking goal and must then combine the results they have obtained into one coherent list. The question is of how best to order the results in the final list when the original lists: (a) may not have the same number of elements, (b) may have elements that do not appear in both lists, and (c) may have widely different ranking schemas resulting in the elements appearing at very different positions in the two lists (*c.f.*, [122]).

Borda count is the most commonly used method, as well as the simplest and fastest in terms of processing speed [45], to address the problem of ranking fusion. The method was originally proposed in the 1700s by a French mathematician and political scientist named Jean-Charles de Borda, as a voting method to decide elections [22]. In this method, each voter ranks each of several candidates by order of preference. The candidates are given a certain number of points based on their position in each list, and usually the candidate with the most points is the winner.

In the method as applied to the search and information retrieval domain, the voters are two sources of ranked lists such as search engines. The candidates are the elements of the rank-list from each search engine. If the raw ranks are summed, the candidate with the least amount of points will be the winner, or best match. A weighted Borda count can apply different weights to the ranks from each source, if, for example, one search engine is more trusted than the other [42]. In the case of this dissertation, the handwriting recognizer is one “voter” and the other is the tutor’s knowledge model. Both voters use their own information sources, the handwritten strokes and the problem-solving operations and answers, respectively, to construct their own rank-lists. Then a weighted Borda count is used to combine these lists into one aggregated list. More detail on this procedure and how it is used in this dissertation can be found in § 6.1.2.

### 2.5.5 Freehand Formula Entry System

For handwriting recognition, the system primarily used in this work is the Freehand Formula Entry System (FFES) [131]. However, several handwriting recognition systems were tested prior to se-

lecting FFES, including JMathNotes [137] and Microsoft’s TabletPC recognizer<sup>10</sup>. FFES achieved a much higher base accuracy rate than the others on a corpus of test data for the target population and application of this dissertation. Chapter 5 discusses the comparison of these recognition systems in further detail.

FFES is a pen-based equation editor written in C++. FFES recognizes mathematical equations via two components: character recognition (California Interface Tools, or CIT) [131], and mathematical expression parsing (DRACULAE) [149]. The main advantage of this handwriting recognizer is that it is easily trainable to whatever subset of the entire symbol set is needed; for instance, one can keep only mathematical symbols and numbers that are needed in the beginning algebra domain. By narrowing the symbol set, higher recognition accuracy rates can be achieved because there are fewer possible character classifications from which to choose. In addition, this system includes a built-in spatially-based mathematical expression parser (DRACULAE). Another advantage of this system is that its source code is available under the Gnu Public License (GPL), allowing changes to be made to the recognition algorithm. FFES has reported character recognition accuracy rates of about 77%, for both expert and novice users who had not trained the system to their style of writing. With training, FFES can yield accuracy rates as high as 95% [131].

### 2.5.6 Cognitive Tutor Algebra

Cognitive Tutors are distributed by Carnegie Learning, Inc.<sup>11</sup> Several full-year curricula are offered, including Algebra I, Geometry, and Bridge to Algebra (pre-algebra). Cognitive Tutors have been designed based on cognitive psychology theory and methods [5]. The primary instructional paradigm is problem solving. In Cognitive Tutor Algebra, students solve problems set in realistic contexts (see Figure 2.1). They are given word problems in which they represent the situation algebraically in the worksheet, graph the functions, and solve equations with a symbol manipulation tool.

In this dissertation, Cognitive Tutors are used as the intelligent tutor foundation; a handwriting interface is added to already-existing algebra equation-solving lessons. There are several advantages to using this system. First, its curriculum and implementation have been previously developed and field-tested extensively. Second, Cognitive Tutors exist for a variety of learning domains including algebra, geometry, foreign languages, chemistry, computer programming and more, which provides possibilities for generalization of these techniques to other domains in which handwriting may be advantageous. Third, Cognitive Tutors for mathematics are in use in about 2,600 schools in the United States, and therefore the results of this research have the potential to be disseminated on a large scale in real classrooms and to improve math learning in students all over the country.

---

<sup>10</sup><http://msdn.microsoft.com/en-us/library/aa510941.aspx>

<sup>11</sup><http://www.carnegylearning.com>

## 2.6 Glossary of Terminology

This list of terminology defines concepts used in this dissertation which may be less well-known outside of their natal field. Throughout this dissertation, glossary terms will be indicated by **bold text**, pointing as a reference back to this location.

**Alignment:** In multimodal interfaces, the problem of temporally or spatially aligning two distinct streams of input in order to extract semantics or improve recognition.

**Answer-level feedback:** In intelligent tutoring systems, instructional feedback only on a student's final answer, without reference to the actual problem-solving steps that the student performed.

**Cognitive load:** In learning science, amount of mental effort or work involved in a particular activity. Load can be *germane*, meaning *a propos* and critical to the task, or *extraneous*, meaning irrelevant and potentially interfering.

**Cross-validation:** In machine learning, a method of selecting from data to create training sets such that all observations are used for both training and testing, and each observation is used for validation exactly once. Overall performance is reported by averaging across the “folds.” This prevents overfitting to a specific training set.

**Enrollment:** In speech or handwriting recognition, the training period during which a user trains the recognition system to his specific style of speech or writing.

**Isomorphic:** In learning science, two problems are isomorphic if they have the same surface structure, and therefore require similar or identical skills in order to solve them. For example, the problems  $5 + 2x = 3$  and  $8 = 3 - 4x$  are isomorphic forms of the  $ax + b = c$  problem type.

**Likert scale:** In social science, a method of quantitatively assessing a subjective variable, by asking to participants to rate their level of agreement with a statement. The rating scale is typically arranged from least agreement to most agreement, with five to nine discrete values.

**Mastery:** In learning science and intelligent tutoring systems, the point at which a student demonstrates a predetermined level of proficiency on certain material.

**N-best list:** In handwriting recognition, a list sorted by confidence of the top  $N$  candidates for a given set of stroke(s) as returned by the system.

**Rank-list:** In information retrieval, a list of the top  $N$  candidates, sorted by the degree to which they match the search criteria.

**Retention:** In learning science, the degree to which students retain knowledge after the instructional period has ended. An important measure of robust learning, retention is often measured via a delayed post-test.

**Robust learning:** In learning science, and as defined by the Pittsburgh Science of Learning Center (PSLC), a type of learning which is either retained for long periods, transfers to novel situations, or aids future learning.

**ROC curve:** A graph of the relationship between the sensitivity or recall rate (true positive rate) vs (1 - the specificity) or precision rate (false positive rate) of some binary classifier, e.g., a threshold value.

**Scaffolding:** In learning science, tutoring aids that help students successfully arrive at the solution to a problem. Typically, these are *faded* with time so that students are eventually solving problems on their own.

**Step-targeted feedback:** In intelligent tutoring systems, instructional feedback on the specific problem-solving process the student performed, with a focus on individual steps; this feedback may be immediate (after each step is executed) or delayed (at the end of an entire problem).

**Transfer:** In learning science, the ability for students to apply knowledge they learn in one situation or domain to another they encounter later.

**Wizard-of-Oz:** In user testing, a study in which part or all of a complex technical component is simulated by a human for the purpose of evaluating the interaction made possible if the technical component were available.

**Worked example:** In learning science, a worked-out problem solution provided to a student as an example of the conceptual problem-solving steps needed to solve a particular type of problem.

**Writer-dependent recognition:** In handwriting recognition, recognition in which the recognizer has been trained only on the same user on which it is being tested.

**Writer-independent recognition:** In handwriting recognition, recognition in which the users in the recognizer's training set do not intersect with the users in the testing set.

# Chapter 3

## Handwriting Helps: Theory from Learning Science and Human-Computer Interaction

The foundational approach this dissertation takes is to establish the ways in which handwriting input can provide benefits for students in intelligent tutoring systems, in order to properly motivate research into ways to effectively incorporate handwriting input into ITS for math and improve handwriting recognition accuracy for this application domain. In service of this approach, hypotheses were formulated as to how the benefits of handwriting would manifest and what factors would be causing these advantages, from both a usability and a pedagogical perspective. This chapter describes these theoretical foundations in terms of motivating the use of handwriting input in tutoring systems for math. The chapter concludes with pointers to the places in this dissertation that directly address each factor hypothesized to contribute to handwriting input's benefits for learning math on the computer.

### 3.1 Usability and Handwriting

Given prior studies where typing was found to be faster than handwriting (*e.g.*, [24, 69]), one might ask why handwriting would ever be used instead of typing. In point of fact, studies favoring typing over handwriting with respect to speed have focused on entering paragraphs of English text and may not apply to equation entry. Standard keyboards do not allow users to easily type complex mathematical expressions such as fractions, exponents or special symbols like  $\sum$  and  $\sqrt{}$ . It is possible that for *simple* linear equations, the keyboard may be faster.

Although some systems that can recognize handwritten equations have reported evaluations (*e.g.*, [52, 77, 88, 125, 131]), none of them have reported an evaluation of the handwriting *modality* from a usability perspective, de-coupled from *recognition* limitations with respect to accuracy and correction of errors. A foundational assumption of this work is that usability and user preference concerns are critical to the success of a system. However, evaluating usability of a modality in the company of a system's recognition errors measures only the usability of that particular system

with its particular idiosyncratic recognition behavior. The usability of handwriting input itself has not been established prior to this dissertation. Without such motivation, further effort to develop handwriting recognition for user domains might be superfluous. For novice users of math input tools, such as middle and high school students, this dissertation posits and proves the hypothesis that handwriting input should be faster and more natural for them due to its similarity to the familiar modality of paper.

## 3.2 Pedagogical and User-Focused Factors

Pedagogical theory grounds the approach taken in this work. This work hypothesizes, and then explores, how several factors could contribute to pedagogical advantages that handwriting interfaces may have in the domain of learning environments, especially for the mathematics domain.

### 3.2.1 Cognitive Load

One factor which might contribute to handwriting's advantages for learning is an expected reduction in **cognitive load** due to the use of handwriting rather than a menu-based typing interface. Extraneous cognitive load (*c.f.*, [134]) can be thought of as a measure of how much mental overhead students experience as a result of interface-related tasks while they are also trying to learn a mathematical concept. That is, extraneous cognitive load interferes with the *learning event*.

Although intelligent tutors for math have improved with respect to pedagogical style and overall effectiveness over the last 15 years (*e.g.*, [38]), their interfaces have remained more or less the same: keyboard-and-mouse windows-icons-menus-pointing (WIMP) interfaces. *Output* modality contrasts have been studied with respect to learning, including the use of animations, diagrams and talking heads (*e.g.*, [60, 92], but the literature has been silent on the effects of *input* modality on learning.<sup>1</sup> In designing such interfaces for online learning and tutoring, it is important to consider what aspects of using the software are directly relevant to the learning event and what aspects are extraneous. The output modality of the student is most likely extraneous to the learning event; that is, the actual method of *outputting* the steps of a problem-solving process is irrelevant to *learning* the problem-solving process. However, inherent in various output modalities are the amount of attention, time and extraneous load spent in performing the cognitive, perceptual and motor processes associated with generating that output. These output processes are irrelevant to the cognitive and perceptual processes associated with solving the problem, and as such distract the user from the learning event. To the extent that certain modalities require less time, and therefore attention, the user experiences less distraction from the learning event, and vice versa.

For example, in current Cognitive Tutors, students are required to search for the appropriate command operation in a set of menus and submenus (*e.g.*, “Combine like terms”) in order

---

<sup>1</sup>Note that input modality here refers to the modality of generation by the student, and the output modality is the modality presented to the student by the system.

to perform the next step of the equation's solution. This requires the student to either memorize or search the menus after every problem-solving step. Issues of cognitive load caused by such resource-consuming interfaces may interfere with learning of the goal concepts. This dissertation hypothesizes that a handwriting-based interface that allows students to directly represent and manipulate equations, via standard mathematical notations, induces less cognitive overhead for the students and interferes less with the primary learning event. This hypothesis can be measured by self-reports and tests.

### 3.2.2 Spatial Characteristics of Math

Another factor that may contribute to pedagogical advantages of handwriting input is that in mathematics, the spatial relationships among symbols have inherent meaning. For instance, the spatial placement of the  $x$  in these two expressions changes the meaning of the expression significantly:  $2^x$  vs  $2x$ . Handwriting is a much more flexible and robust modality for representing and manipulating such spatial relationships, which become more prevalent as students advance in math training to calculus and beyond. Two-dimensional mathematics can be difficult to represent and manipulate in text-based interfaces, involving menu-based markups or special characters. Handwriting provides affordances for annotations and nonlinear input much more naturally and easily than typing, from a user-centered perspective. This dissertation finds support for the hypothesis that the appearance of non-keyboard math symbols, or even nonlinear notations such as fractions, magnifies the negative impact of typing interfaces compared to handwriting ones, especially in terms of input speed and efficiency. Increased input efficiency will allow students using handwriting to cover more material in the same amount of time, achieving more advanced curricular goals than their typing counterparts.

### 3.2.3 Fluency and Transfer to Paper

Modality fluency and familiarity is another factor that might contribute to handwriting's advantages for learning. Students practice in the classroom, do homework, and take tests on paper using handwriting; this modality may become more *fluent* for them than typing for students when solving algebra equations. An interface that can take advantage of this fluency should allow a higher degree of **transfer** and cause the tutoring system to overpredict student performance after achieving **mastery** in a lesson less than a typing interface for the same lesson. Anecdotally, teachers have said that students do have trouble moving from the computer interface to paper, meaning that the tutor may overpredict student capabilities. A tutoring interface that better predicts students' performance when working on their own is important to ensuring accurate assessment of student skills, and to ensuring that the tutoring system is actually helping the students. This dissertation hypothesizes that, due to its similarity in both look and feel to paper, tutors that allow handwriting input will better predict student mastery levels, as measured by performance on classroom tests, than tutors that use typing input.

### 3.3 Bridging Pegagogy and Technology

The main research pillars of this dissertation are:

1. that recognition accuracy can be improved “enough” to be usable by students by taking advantage of domain-specific knowledge, and
2. that less than 100% recognition accuracy will be “enough” in the tutoring domain because the instructional paradigm can be designed to rely less heavily on step-by-step feedback.

These hypotheses focus strongly on creating a bridge between the world of pedagogy and the world of recognition technology. Each half of the equation can capitalize upon elements of the other to overcome its own weaknesses and vice versa. The interplay between recognition accuracy and instructional feedback is the hub around which this dissertation centers.

Recognition accuracy rates vary between systems, but are usually better for domain-specific vocabularies and applications [86, 108] or for **writer-dependent** systems in which the recognizer has been trained on the writing of the user using it [131]. In the math tutoring domain, the vocabulary is small (only 22 symbols are used throughout this dissertation), and domain-specific context information is available. However, training handwriting recognition engines usually involves a large upfront time commitment during which the user inputs many (20 or more) examples of each character the recognizer is to understand. In a classroom setting, teachers are resistant to spending any classroom time on non-learning objectives in order to allow students to train the handwriting system. Therefore, it is imperative that the system maximize recognition accuracy while minimizing upfront training cost for individual users.

The type of instructional feedback a tutoring system can provide is dependent on the level of accuracy or confidence the system has about its interpretation of student input. If the system is very unconfident about its interpretation (*i.e.*, it is known to be highly inaccurate), it may only be able to provide feedback at the most abstract level—whether or not the final answer is correct (**answer-level feedback**). If it is very confident, it may be able to provide more detailed feedback at a lower level of granularity, for instance, **step-targeted feedback**. Thus, the accuracy of the recognition is related to the level of feedback the system can provide. Before this work was undertaken, it was not clear what level of feedback would be required for students to succeed in the math tutoring domain. Through the course of this dissertation it was found that the use of **worked examples**, in which students study or copy complete problem solutions in addition to solving problems, helps mitigate the criticality of step-targeted feedback. In addition, because in this domain the final answers (*e.g.*,  $x = -4$ ) tend to be short and simple, if needed students can type them into the interface after entering their solution process via handwriting; typing the last step completely eliminates ambiguity and allows answer-level feedback to be perfectly accurate.

The learning sciences literature has not yet come to a consensus on the benefits of including worked examples in intelligent tutoring systems, or even how and when to provide them, or what they should look like [128]. This dissertation provides valuable evidence in favor of annotated worked examples interspersed with problem-solving, in the company of step-targeted feedback.

Prior literature has shown that, in a comparison of step-targeted *vs* answer-level feedback, students perform better with the former. In the LISP programming tutor study of [36], in the only condition that had only answer-level feedback, students took longest to complete the lesson and performed worst on post-test quizzes. That study was not done in the context of worked examples, however, but in straight problem solving, which may be where the critical difference lies. Trafton and Reiser [140], in the same LISP programming tutor curriculum, demonstrated large learning benefits for relevant worked examples interleaved with problem solving and did not give step-targeted feedback. The paradigm used in this dissertation is to study a relevant worked example just before solving a problem, as a means to provide a level of *feed-forward* help that may be able to compensate for less step-targeted *feedback*. The outcome of using this method will provide another datapoint in favor of worked examples in tutoring systems and may begin to focus the design of consistent instructional paradigms based on worked examples.

The learning and technology components of this work are highly intertwined. The relationship between these components is examined, and alternative methods of instruction, such as worked examples, are explored, that may help the resulting system become more than the sum of its parts.

## 3.4 Proving the Hypotheses

To summarize, the factors this dissertation posits as the source of handwriting input's benefits for online math learning are as follows:

1. Speed of input and time on task
2. User errors
3. User preferences
4. Reduced cognitive load via unconstrained input
5. Better support for the spatial characteristics of math notation
6. Better transfer to paper and tutor predictiveness

Each of these factors is explored in the foundational user studies described in Chapter 4. For quick reference, cross references to the specific findings that address each one from each study are listed here.

### 3.4.1 Usability Measures

**Speed of input and time on task.** Measured via computer logs of student input and tutoring sessions, on both a total-time-spent scale and an individual-equation (or problem) scale. See § 4.1, § 4.2, and § 4.3 for how this is addressed in each of the studies.

**User errors.** Measured via computer logs of student input and tutoring sessions. In some studies it was measured via human coding of video data showing student input (§ 4.1). In those cases, an “error” was defined as when the user submitted a completed equation with an incorrect character, or when the user acknowledged having made an error by correcting something previously entered (*e.g.*, scratching out, overwriting). In other studies, it was measured by virtue of the tutoring system logs indicating on which steps students made a mathematical or conceptual error (§ 4.3), or if the students entered an incorrect final answer (§ 4.2 and § 4.3).

**User preferences.** Measured via Likert scale questionnaires (§ 4.1) or open-ended survey questions (§ 4.2). Students were asked to indicate the degree to which each input modality felt “natural” for entering math (§ 4.1), or which modality they liked best and why (§ 4.2), and whether they would want to use any of the modalities again for math (§ 4.2).

### 3.4.2 Pedagogical Measures

**Reduced cognitive load via unconstrained input.** Measured via self-report on a Likert scale of mental effort in § 4.3. Students were asked to indicate the degree of mental effort they felt during the study and how this compared to normal use of the Cognitive Tutor in their classroom. These self-report questions were modeled on the same questions used to measure cognitive load in [103].

**Better support for the spatial characteristics of math notation.** Measured by item analysis on individual equations or problems students entered via different modalities; some items contained non-keyboard characters such as  $\sum$  and  $\sqrt{}$  (§ 4.1), others contained fractions, a common nonlinear math notation (§ 4.2). Interactions between the occurrence of these types of math notations and other measures such as input speed and errors were analyzed.

**Better transfer to paper and tutor predictiveness.** Measured via correlation of performance during the training session (*e.g.*, tutor use) and performance on the tests (§ 4.2 and § 4.3). Performance during training is defined as the proportion of training problems solved correctly on the first try; this is analogous to a test-taking environment in which students only have one try to solve a problem. The correlation between the two reveals the degree to which a tutoring environment effectively predicts student performance when solving problems on their own *vs* with the tutoring hints and **scaffolding** available.

# Chapter 4

## Handwriting Helps: Foundational Studies

In order to begin exploring ways in which handwriting input has advantages for students inputting and *learning* mathematics, several foundational studies were conducted. The first matter of interest was whether or not the use of handwriting input provided any usability benefits—after all, students are users, too, and it is important to ensure the most usable interaction possible so as not to interfere with the learning process. After establishing that handwriting did in fact have benefits in terms of usability, showing how and to what degree handwriting input leads to improved learning was next.

Table 4.1 enumerates each study performed in this dissertation and gives a summary of the important experimental design variables. The following sections describe each study in detail. None of the user studies reported in this dissertation were conducted with a prototype where handwriting recognition was used to respond to the user. In all cases, the system allowed handwriting input, but instructional feedback was either not provided or was only provided on a portion of the input that was typed, for instance, the final answer of a problem-solving solution. Following these studies, which establish the benefits of handwriting *input* for learning and the necessity for more detailed feedback than one can provide without recognition, technical improvements were undertaken on a recognition system to improve accuracy for the purpose of provided detailed feedback to students, described in Chapters 5 and 6. In Chapter 7, a system is described that can make use of such improvements in order to provide a natural interaction flow for the students; the implementation and evaluation of such a system is left to future work.

### 4.1 Study 1: The Math Input Study

The research questions addressed by this study<sup>1</sup> include:

- Which of the most common desktop input modalities is the fastest or least error-prone when entering mathematics on the computer?

---

<sup>1</sup>This section is partly based on content from the following publications: [8, 9].

**Table 4.1:** All user studies performed to support this dissertation. Note that although all studies include a handwriting input modality, none use real-time recognition or provide feedback to the participants as to what specifically the system thinks was written.

Study Name	Type of Study	Number of Participants	Type of Participants	Conditions	Number of Sessions (Length in minutes)	Summary of User Task(s)	Measures	Main Result(s)
Math Input Study	Within-subjects; Lab	48	College students	Typing (MSEE); Handwriting; Speaking; Handwriting-plus-speaking	1 (45)	Copying 36 given equations of calculus level	Speed of input; User errors in input; Likert scale questionnaire of preferences	Typing is 3x slower than handwriting. Users rate handwriting more highly in terms of naturalness.
Laboratory Learning Study	Within-and between-subjects; Lab	48	Middle and high school students	Typing; Handwriting; Handwriting-plus-speaking	1 (150)	Copying 85 given equations of algebra level; Solving 9 problems of algebra level	Speed of input; user errors in input; Total time and time per problem; Change in score from pre-test to post-test	Typing is 2x slower than handwriting. No learning difference in spite of time gain. Students experience better transfer to paper in handwriting conditions.
Cognitive Tutor Study	Between-subjects; Classroom	76 from eight classrooms in two schools	High school students	Cognitive Tutor (control); Cognitive Tutor plus examples; Cognitive Tutor plus examples minus step-targeted feedback; Cognitive Tutor plus examples minus step-targeted feedback using handwriting input	2 or 3 (150)	Completing an algebra unit of the curriculum	Change in score from pre-test to post-test to follow-up retention test; Self-report of cognitive load; Amount of time spent in tutoring lesson	Worked examples add value to Cognitive Tutor. Lack of step-targeted feedback is harmful. Handwriting is good but not good enough to ignore step-targeted feedback.

- Do these effects change significantly as the mathematics being entered increases in complexity?
- Which modality do users rate most highly as being natural for entering mathematics on the computer?

Overall, this study found that handwriting was three times faster for entering calculus-level equations on the computer than typing using a template-based editor, and this speed impact increased as equations got more complex (namely, as characters not on the keyboard were included). In addition, user errors were three times higher when typing than when writing for entering math on the computer. Finally, users rated the handwriting modality as the most natural, suitable modality for entering math on the computer out of the ones they used during this study.

#### 4.1.1 Experimental Design

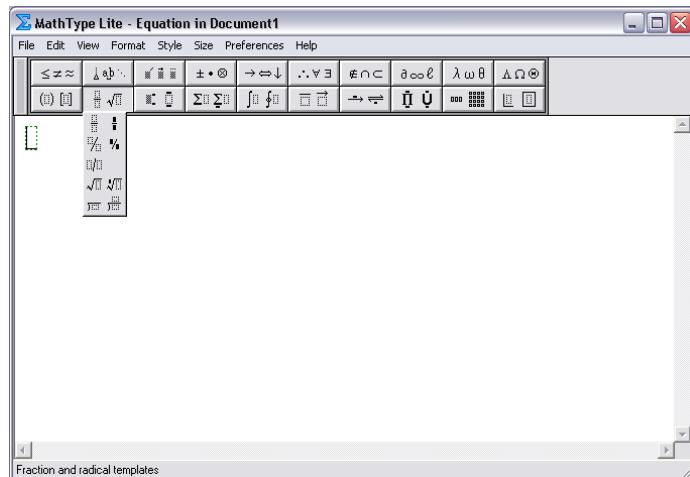
This study was a within-subjects laboratory study in which users were asked to enter given mathematical equations on the computer in several modalities. Prior literature had indicated that handwriting was not any faster or less error-prone than typing [24], but that research was in the domain of writing natural English. The hypothesis of this study was that studying handwriting for mathematics would yield different results.

In this study, users were asked to enter mathematical equations of varying complexity using four different modalities: (1) traditional keyboard-and-mouse (*typing*) using Microsoft Equation Editor (MSEE), (2) pen-based handwriting entry (*handwriting*), (3) speech entry (*speaking*), and (4) handwriting-plus-speech (*multimodal*). MSEE was chosen as a representative tool for novice users because it is in wide use and is a prime example of a common interface for mathematics. There was no automatic handwriting or speech recognition in this (or any) study; users simply input the equations and did not get feedback about computer recognition of their input. Figure 4.1 shows the interfaces used in the study as users saw them.

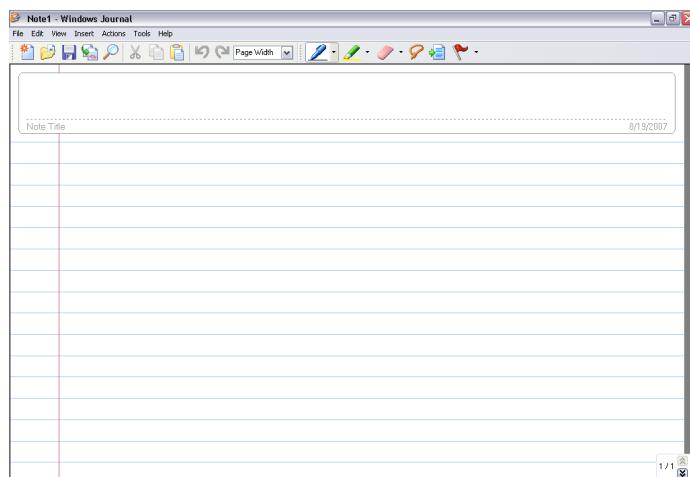
Pairing handwriting and speaking may not immediately seem like a natural choice. A multimodal input method combining handwriting and speech was included because such a combination might enhance computer-based recognition of equations (*c.f.*, [100]) and could aid user cognition. Research has shown that people speak in an “inner voice” (subvocalization) while reading or writing [85]. Several users during the sessions, in the speaking-only condition, wrote in the air with their hands while speaking the equation out loud. Exploring the pairing of these two modalities may be important to understanding how to support user cognition during handwriting input on the computer.

#### Participants

Forty-eight paid participants (27 male, 21 female), graduate or undergraduate students at Carnegie Mellon, answered an ad to participate in this study. All participants were fluent English speakers



(a) Typing Condition.



(b) Handwriting Condition.



(c) Speaking Condition.

**Figure 4.1:** Screenshots of the interfaces used in the Math Input Study. From top to bottom: the typing condition using Microsoft Equation Editor, the handwriting condition, and the speaking condition. The handwriting-plus-speaking, or multimodal, condition looked like the handwriting condition from the user's perspective; the speech recorder was running in the background.

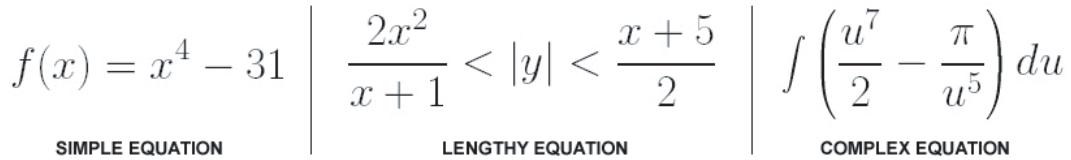
with unaccented speech. No effects of age or ethnicity were seen in exploratory data analysis, so these variable were excluded from further analyses. Most (33) had no experience with MSEE before the study. Of those who knew of it or had used it, only two classified themselves as knowing it “very well.”

### Procedure

The experiment was a within-subjects design in which participants came to the lab for a 45-minute session and entered mathematical equations on a TabletPC in four different conditions. There was a list of 36 equations (nine per condition) which remained constant for all participants; order of presenting each condition was counterbalanced across all possible orderings. Before the session, participants took a math symbols recognition test to ensure that all users would be able to speak the name of each symbol they would encounter. Participants answered a questionnaire before the session in which they rated their pre-existing preferences for each condition. Before performing each condition, participants were instructed as to how to enter equations in that condition. For instance, in the handwriting condition, the experimenter explained that the stylus could be used like a regular pen on paper. The experimenter did not tell the participants in what format to write the math, or how to find certain symbols or express themselves. Participants were given a five-minute practice period before the typing condition to familiarize themselves with the MSEE toolbar. This toolbar provides menus to allow users to enter special symbols, fractions, exponents, etc. During this time, participants explored on their own with no feedback or input from the experimenter. There was no exploratory period for the other three conditions; to account for interface learning effects, the first two equations in each condition were considered practice and were not included in the analyses. When participants finished all four conditions, they answered a questionnaire again rating their preferences for entering equations in each condition. All materials for this study are given in Appendix B.

### Stimuli Design

The experimental stimuli (36 equations) were designed with two factors in mind: (1) the number of characters in the equation, and (2) the number of “complex” symbols appearing in the equation such as fractions, exponents, special symbols, and so on. Figure 4.2 shows three sample equations in increasing complexity from left to right. The first equation has 10 characters and has no special symbols that do not appear on the keyboard. The second equation has 17 characters and also no non-keyboard symbols. The third equation has 14 characters, two of which are special symbols. Both factors should have an effect on user performance. Increased length should increase time because additional characters in any modality would require more time to enter. Adding symbols that do not appear on the keyboard, such as  $\sum$  and  $\sqrt{ }$ , should only have a significant effect in the typing condition, because special symbols are no more difficult than normal symbols when speaking or writing. The length of each equation ranged from 10 to 18 characters. All 36 equations are listed in full in Appendix B.1.



**Figure 4.2:** Experimental stimuli as users saw them.

## Measures

The data from each session were collected at 30 frames per second by capturing the screen output and audio on a DV recorder. The videotape was later analyzed by a single coder to extract the number of errors each participant made while entering each equation in each condition. An “error” was defined as when the user submitted a completed equation with an incorrect character or acknowledged having made an error by correcting something previously entered. Time for each participant to enter each equation in each condition was logged. User preference questionnaires rating each modality were administered both before and after the session, consisting of a five-point Likert scale from “least suitable or natural” to “most suitable or natural” (see Appendices B.3 and B.4 for the exact wording and form of the questionnaires).

### 4.1.2 Results and Discussion

All means for quantitative measures reported for the Math Input Study are given in Table 4.2. Individual tables are referenced in each relevant section that follows.

#### Qualitative Results

Table 4.3 shows examples of an equation from each condition and a randomly chosen user’s response to that equation. In the typing condition, students typically did utilize the template menus provided by MSEE in order to construct their equations. However, in the example given in Table 4.3 for the typing condition, that user did not use the template for the absolute value symbol. The user could not find the appropriate symbol on the keyboard, substituting ‘/’ instead. The ‘/’ and the ‘1’ instead of ‘x’ were counted as errors. The equation shown for the typing condition took the longest to complete out of all equations, at 209 seconds for that user. In the multimodal condition example, the use of ‘()’ quantifiers instead of ‘[]’ was not counted as an error, as they are syntactically and semantically equivalent quantifiers.

The speech utterances of students speaking math in this study were very interesting sources of data. Although a detailed analysis was out of scope for this work, it is worth mentioning some

**Table 4.2:** Means tables for all measures reported from the Math Input Study.(a) Means table of time per equation in seconds.  $N$  is the number of equations.

Condition	Mean	$N$	StdErr
Typing	46.193	315	0.636
Handwriting	15.688	333	0.597
Speaking	13.958	313	0.625
Multimodal	19.513	325	0.608

(b) Means table of total errors per equation.  $N$  is the number of equations.

Condition	Mean	$N$	StdErr
Typing	1.769	316	0.106
Handwriting	0.589	332	0.105
Speaking	0.658	311	0.101
Multimodal	1.395	326	0.102

(c) Means table of interaction between appearance of non-keyboard characters (e.g., “Complex”) and condition with respect to time per equation in seconds.  $N$  is the number of equations.

Condition	$N$	Simple		Complex		
		Mean	StdErr	$N$	Mean	StdErr
Typing	232	39.09	0.687	83	51.98	1.196
Handwriting	238	14.50	0.677	95	16.85	1.076
Speaking	224	12.59	0.703	89	15.31	1.123
Multimodal	233	17.54	0.687	92	21.35	1.108

(d) Means table of Likert scale ratings of user preferences for each condition. Higher numbers correspond to a better rating.  $N$  is the number of participants.

Condition	$N$	Pre-Session		Post-Session	
		Mean	StdErr	Mean	StdErr
Typing	48	4.10	7.48	3.33	9.42
Handwriting	48	4.46	11.64	4.75	3.67
Speaking	48	3.81	6.51	3.33	6.72
Multimodal	48	n/a	n/a	4.00	7.83

high-level qualitative notes here.<sup>2</sup> The length of student utterances in speech was affected by the appearance of non-keyboard characters such as  $\sum$  and  $\sqrt{}$ . The occurrence of these characters appeared to prompt more phrases such as “uh,” “um” and so on. The average utterance length across all equations (not accounting for number of characters in the equation) was 17.6 words (including conversational phrases such as “uh” or self-corrections such as “oops”), with a range of nine to 42 words per utterance. The number of pauses in spoken utterances between the speech-only and multimodal conditions did differ significantly, however. This difference is likely due to the effect of synchronization of speech and writing in the multimodal condition. The overall mean number of pauses per equation was 3.0, with a range of zero pauses to 13 pauses per equation (speech-only *mean*: 2.76, *stdev*: 1.49; multimodal *mean*: 3.6, *stdev*: 1.46).

Finally, ambiguity control in speech was inconsistent, even within users. The examples of speech shown in Table 4.3 show that participants were not generally very precise in their speech with respect to ambiguity of expression. For instance, they often left out phrases such as “quantity of,” especially in complex equations where it might become difficult to keep all of the open quantities in short term memory. In comparing users’ speech to the typeset equations with respect to parenthetical markers, it seemed that participants were most likely to omit parentheticals in the speech modality, but were also more likely to add in their own, different parentheticals in those conditions containing speech (speech-only and multimodal). In addition, they also tended to add more parentheticals in the typing condition. This may be due to the fact that some of the participants linearized the typeset version of the equation while typing it, thus requiring the addition of parentheses to be interpretably correct. These ambiguities are not counted in general as errors throughout this dissertation, as they are naturally occurring speech patterns. For teaching purposes, tutoring systems can correct students if the teacher desires them to be mathematically precise in speech, but the system must be able to recognize the common patterns of speech in general.

## Speed

Means of time per equation in seconds by condition are shown in Table 4.2(a). The typing condition was three times slower than the others, including handwriting, and this difference was significant. A univariate ANOVA on time per equation was conducted considering the following factors: (1) participant as a random factor to account for the correlations between datapoints, (2) input condition and appearance of non-keyboard characters in each equation as fixed factors, and (3) the number of characters in each equation as a continuous covariate. This analysis yielded a significant interaction between condition and appearance of non-keyboard characters ( $F_{3,1241} = 13.53, p < 0.05$ ), and a significant main effect of the number of characters in the equation ( $F_{1,1241} = 39.35, p < 0.05$ ). Estimated marginal means of time per equation given these two factors are shown in Table 4.2(c). Longer equations took more time to enter. The typing condition experienced a much larger slowdown due to appearance of non-keyboard characters than the other three conditions, which follows intuitively from the nature of the factor itself. Writing or saying

<sup>2</sup>Detailed statistics on these results can be found in [9].

**Table 4.3:** Samples of user input in the four conditions of the Math Input Study. Note that the typing sample contains three errors: the use of '/' instead of '|' and '1' instead of 'x' (twice). In the multimodal entry, the point at which the user substituted which quantifier symbol was used was not considered an error.

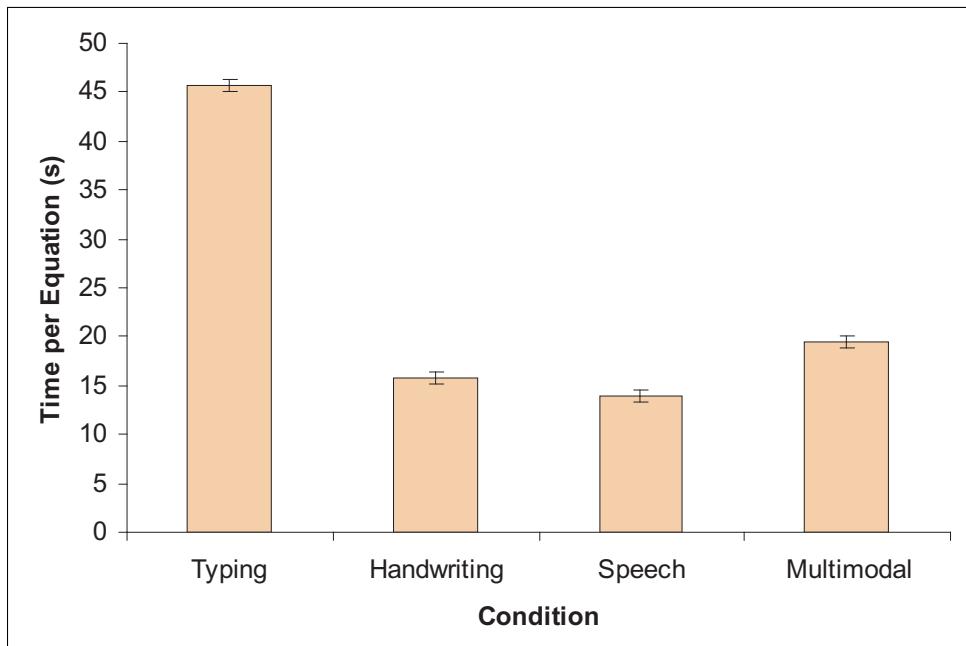
Condition	Typeset Version	Example of User Input
Typing	$\frac{1}{ x +1} - \frac{x^2}{2} \leq y$	$\frac{1}{/1/+1} - \frac{x^2}{2} \leq y$
Handwriting	$f(x) = 5(y_2 - y_1)$	$f(x) = 5(y_2 - y_1)$
Speaking	$\frac{y-4}{y^2-5y+4} = 9$	“y minus four over y squared minus five y plus four equals nine”
Multimodal	$\sum [c_k^2 - 2c_k - 10]$	$\sum (c_k^2 - 2c_k - 10)$ “sum of c subscript k squared minus two c subscript k minus ten close parentheses”

a character that does not appear on the keyboard but is a common mathematical notation is no more difficult than writing or saying actual keyboard characters. A planned contrast comparing the typing condition to the other three conditions showed a significant difference ( $t(1241) = 34.91$ ,  $p < 0.05$ ), typing being slower than the others. Figure 4.3 shows a graph of the mean time in seconds for each condition.

## User Errors

Means of errors per equation by condition are shown in Table 4.2(b). The typing condition had three times as many errors as the handwriting condition, and this difference was significant.

A univariate ANOVA on errors per equation was conducted with the same factors as above. A significant three-way interaction was found between condition, length of the equation, and appearance of non-keyboard characters ( $F_{4,1222} = 2.39$ ,  $p < 0.05$ ), which implies that the length of the equation alters the relationship between condition and appearance of non-keyboard characters with respect to errors per equation. Students are more likely to make many errors (*i.e.*, more than the length of the equation) on equations of any length when typing than when handwriting or speaking, where the number of errors increases linearly with length of equation. Because there were no significant two-order interactions, however, here the main effects of each variable are the focus. Significant main effects of all three factors were found: condition ( $F_{3,1232} = 35.33$ ,  $p < 0.05$ ), the appearance of non-keyboard characters ( $F_{1,1232} = 13.61$ ,  $p < 0.05$ ), and the length of the equation

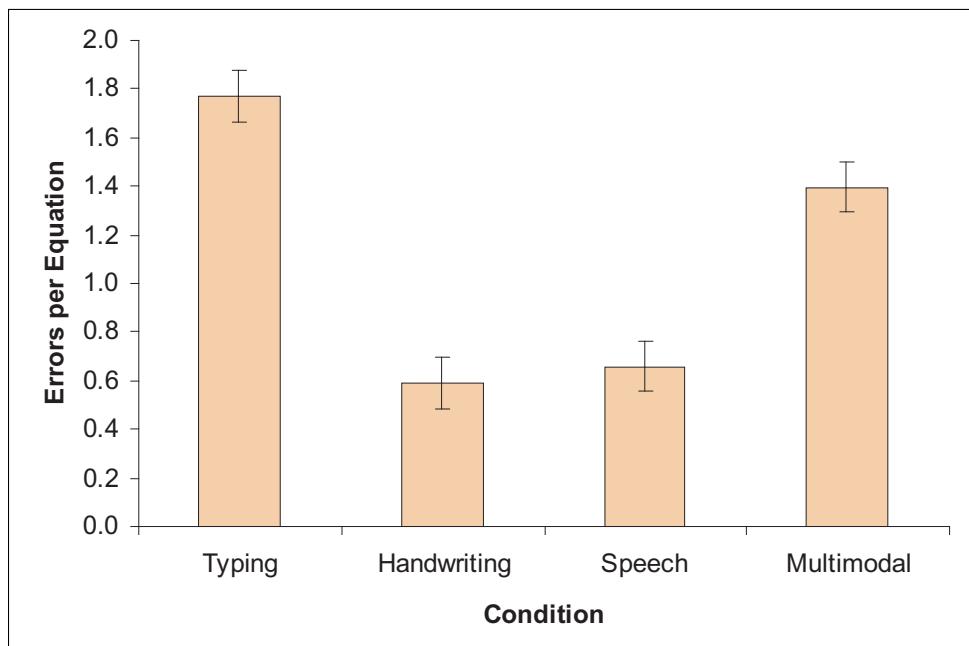


**Figure 4.3:** Mean time in seconds per equation by condition. Error bars indicate standard error.

( $F_{1,1232} = 4.95, p < 0.05$ ). Longer equations and appearance of non-keyboard characters tend to result in more errors.

A planned contrast comparing typing to the other three conditions yielded a significant difference ( $t(1232) = 8.17, p < 0.05$ ), typing having more errors per equation. Figure 4.4 graphs the expected mean number of errors per equation for each condition. The pattern in this graph matches the pattern of speed by condition shown in Figure 4.3. The relationship between condition and errors helps to explain why some conditions are slower than others. It is likely that the fact that the typing condition had the most errors per equation is an artifact of the input device itself. Because the keys are close together on a keyboard, people often make typographical errors and must use the backspace key to correct them, whereas in the other conditions errors are more likely to be cognitive (although slips of the pen are possible). The multimodal condition has about as many errors as the speaking condition plus the handwriting condition combined because users did both modalities and could make independent errors in each of them.

Table 4.4 gives a summary of the distribution of errors per equation across the four conditions. Errors occurred on 46% of all equations. The speaking condition was the least error-prone with errors on only 29% of equations performed in this condition, and a maximum of five errors on any one equation (compared to 55% in the typing condition, with a maximum of 15 errors). However, participants rated speaking lowest in spite of its higher accuracy. They said later in informal interviews that they did not like the lack of feedback in the speaking condition—with no visual



**Figure 4.4:** Mean number of errors made per equation by condition. Error bars indicate standard error.

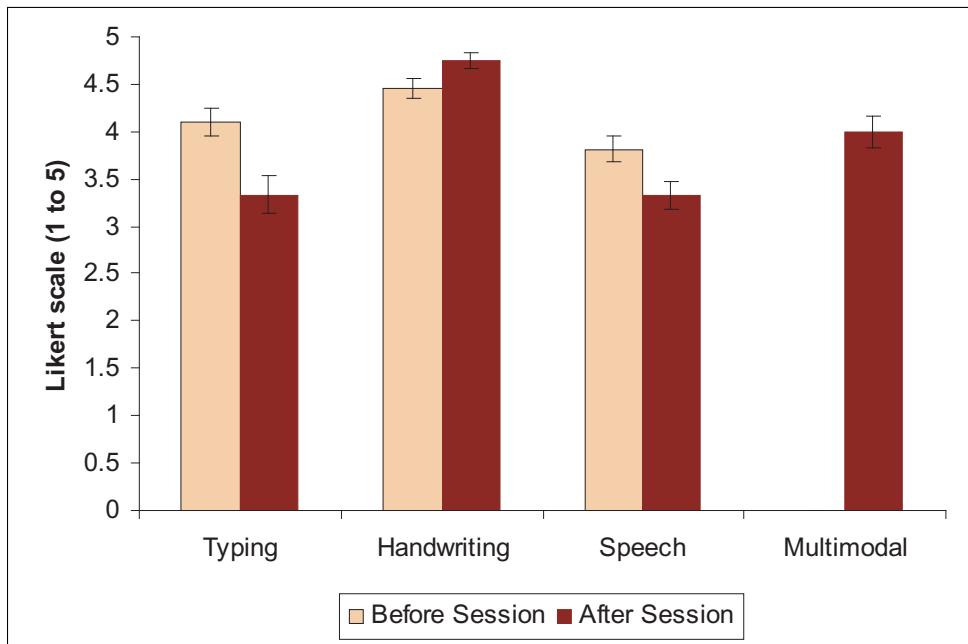
**Table 4.4:** Distribution of errors per equation by condition.

Condition	% of Equations with No Errors	% of Equations with One Error	% of Equations with Multiple Errors (Max Errors)
Typing	45%	23%	32% (15)
Handwriting	58%	28%	14% (5)
Speaking	71%	20%	9% (5)
Multimodal	45%	24%	31% (11)

reminder of what they had said, they did not feel confident in that modality.

### User Preferences

Users were asked both before and after the session to rate the “suitability” and “naturalness” of each modality in the session for entering mathematics on computers (see Appendix B.3 for the exact questions asked). Users rated each modality’s suitability on a five-point **Likert scale**. Means of these ratings by condition are shown in Table 4.2(d). The pre-session questionnaire did not ask a question about the multimodal condition because it was not a condition that participants were



**Figure 4.5:** Pre-session and post-session questionnaire rankings of each condition on a five-point Likert scale. The pre-session questionnaire did not include a question about the multimodal condition. Error bars indicate standard error.

likely to have encountered before the study. The conditions were rated equally well before the session, indicating no pre-existing biases. After the session, the handwriting condition was rated more highly than the typing condition, and this difference was significant. On the pre-test questionnaire, there was no significant difference between the typing condition and the handwriting condition ( $t(47) = -1.61$ , *n.s.*). However, on the post-test questionnaire, users rated the handwriting condition higher than the typing condition ( $t(47) = 4.49$ ,  $p < 0.05$ ). Figure 4.5 shows a graph of the overall mean pre-session and post-session questionnaire ratings of the four conditions in the Math Input Study.

Users were not told how to perform the multimodal condition, meaning they could either use both modalities in parallel or perform one and then the other. Most users (77%) chose to use the two modalities in parallel for each equation. Of those who did *not* use the modes in parallel, they were split evenly between starting with handwriting or starting with speech. The way users performed the multimodal condition did not significantly affect their rating of that condition post-session ( $F_{2,45} = 0.34$ , *n.s.*).

### 4.1.3 Conclusions

This study found that handwriting was faster, less error-prone and more well-liked by users for entering math on the computer than a common typing interface. The increased efficiency of a handwriting interface for a math tutor would allow students to accomplish more problems in the same amount of time, and the fact that students prefer handwriting might lead to increased engagement during tutoring (*c.f.*, [47]). As a result, this study established that, even when not considering learning as a task, math input via handwriting is more usable than typical typing interfaces for math.

## 4.2 Study 2: The Lab Learning Study

The Math Input Study established the usability advantages of the handwriting modality for entering math on the computer. The next step was to determine if these effects applied for the target population of algebra learners, and also to begin to collect evidence for the type and degree of learning improvements handwriting input might engender.

The research questions addressed by this study<sup>3</sup> include:

- Do students experience differences in learning due to the modality in which they generate their answers?
- Do the results reported in § 4.1, with college students entering calculus expressions with complex symbols often not found on keyboards, generalize to a younger population and simpler equations that can be typed easily?

Overall, the Lab Learning Study did in fact show that the usability advantages generalized to students performing a learning task with simpler equations. Students were twice as fast to complete the tutoring lesson in handwriting than in typing, but experienced no significant difference in learning—the extra time students spent in the typing condition did not help their learning. In addition, students overwhelmingly chose handwriting as their favorite of the modalities that they tried during this study.

### 4.2.1 Experimental Design

The study was a laboratory experiment with both within-subjects and between-subjects phases, described in detail below, with local middle and high school students. Three modalities were included in this study: *typing*, in which students typed out the solution in a blank text box (not MSEE); *handwriting*, in which students wrote the solution using a stylus in a blank space on the screen; and *multimodal*, identical to handwriting but students were also asked to speak aloud

---

<sup>3</sup>This section is partly based on content from the following publications: [12, 13].

the steps to the problems they were solving. This condition was included based on prior work finding that spoken self-explanations are more effective for learning than written or typed ones [63]. Speaking-only was not included in this study because of its extremely low user satisfaction rating in the Math Input Study. Examples of the interfaces the participants used in each condition are shown in Figure 4.6.

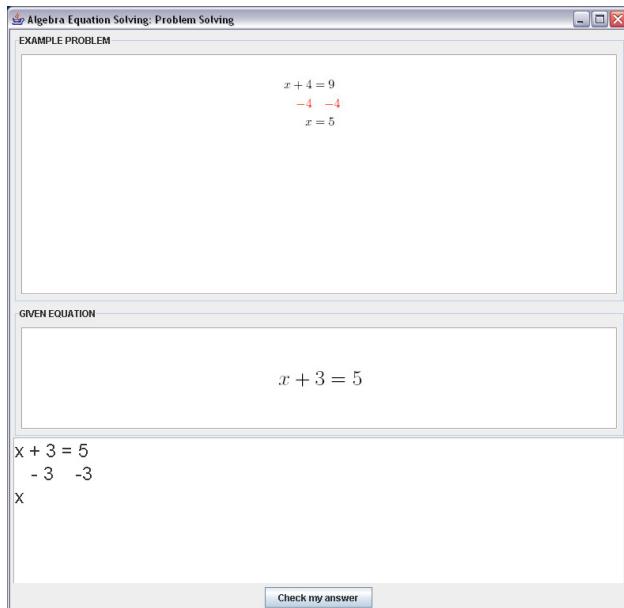
## Participants

In this study, 48 middle and high school students participated. Ten students were dismissed due to technical difficulties with the experiment software or due to scoring 100% on the pre-test measuring algebra skills. Of the remaining 38 students, 19 participants were female and 19 were male. They ranged from sixth to tenth grades (age *range*: 11 to 17, *mean*: 13.5 yrs). They were all paid participants who responded to a newspaper ad offering help in algebra. Most students had not used handwriting input on the computer before and two-thirds claimed to be very comfortable with typing. In spite of the wide range of ages and grades, most students were at about the same level of algebra skills, based on the pre-test. No effects of ethnicity, gender, age or other demographic data were seen during exploratory data analysis, so they are excluded from further analyses here.

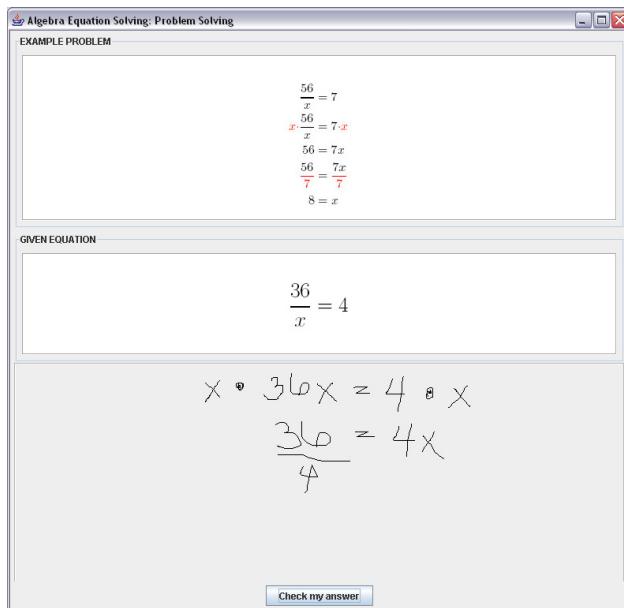
## Procedure

All students came to a research laboratory at the university for about two and a half hours. The session had two main phases. Before each phase, a training handout was distributed and the experimenter reviewed this with each participant to ensure he or she understood the task and the interface. During the first phase, students copied given equations of beginning algebra level in all three conditions; an example of these equations is given in Figure 4.7(a). In the second phase, following a brief pre-test to gauge prior algebra knowledge, students solved beginning algebra equations in only one of the three conditions. The equations students saw in this part of the session were simpler than in the copying phase (see Figure 4.7(b)). All equations given in this study during both phases are given in Appendix C.1. During the problem-solving phase, students alternated copying a non-annotated **worked example** (see Figure 4.7(c)) and then solving an analogous equation while referring to the example. This instructional paradigm was modeled after [140] and was chosen because step-by-step feedback was not provided during problem-solving due to technological constraints of recognizing handwritten input at this stage. The example was intended to provide a kind of step-by-step *feed-forward* to aid students.

Feedback on student answers was given via the **Wizard-of-Oz** paradigm. When students completed a problem their answer was sent to an experimenter (Wizard) at a separate computer for answer verification; the experimenter responded only “Yes” or “No” based on the student’s final answer. This response was then shown on the student’s screen. Students were not given specific reasons why their answers were incorrect. To prevent students from becoming stuck, after three incorrect attempts, the program automatically displayed the correct solution; the students copied it and moved on.



(a) The typing condition.



(b) The handwriting condition.

**Figure 4.6:** Screenshots of the interface used in the Lab Learning Study. The typing condition is shown on top, and the handwriting condition is shown below it. The multimodal interface looked identical to the handwriting condition, but a background process was also recording the student's voice.

$\frac{x}{5} + 4 = 7$	$\frac{28 - 2y}{9 - (4y + c)} = 6y + 4$	$\begin{aligned} \frac{56}{x} &= 7 \\ x \cdot \frac{56}{x} &= 7 \cdot x \\ 56 &= 7x \\ \frac{56}{7} &= \frac{7x}{7} \\ 8 &= x \end{aligned}$
(a) Copying Phase.	(b) Learning Phase.	(c) Worked Example.

**Figure 4.7:** Samples of equations and problems from each phase of the experiment.

This study controlled for content rather than time. When the students had completed all nine problems in the curriculum, they took the paper post-test. Two test forms (A and B) were designed with problems **isomorphic** to those in the session; students were randomly assigned each of the forms as either the pre-test or the post-test. After the post-test, students filled out a questionnaire about their satisfaction with their experiences in the session, their prior math and computer skills, and what their favorite modality was in the study. All materials for this study are given in Appendix C.

## Measures

Dependent variables in this study were different in each phase of the session. In the copying phase, they included the time it took students to copy each equation and the number of errors they made during copying. In the solving phase, the dependent variables included the total time it took students to complete all problems; the time it took them to solve each problem or copy each example; the number of attempts it took them during training to either get the answer correct or move on (up to a maximum of three); and the change in score from pre-test to post-test. General dependent variables included the responses on the user satisfaction questionnaire about the three modalities they had tried.

### 4.2.2 Results and Discussion

All means for quantitative measures reported for the Lab Learning Study are given in Table 4.5. Individual tables are referenced in each relevant section that follows.

**Table 4.5:** Means tables for all measures reported from the Lab Learning Study.(a) Means table of time per equation during copying phase.  $N$  is the number of participants.

Condition	Mean	$N$	StdErr
Typing	33.509	38	1.790
Handwriting	19.030	38	0.954
Multimodal	18.196	38	1.065

(b) Means table of total time on task during learning phase.  $N$  is the number of participants.

Condition	Mean	$N$	StdErr
Typing	19.757	12	1.940
Handwriting	13.568	13	2.395
Multimodal	14.199	14	1.907

(c) Means table of interaction between appearance of fractions and condition with respect to time per equation during the copying phase.  $N$  is the number of participants.

Condition	$N$	No Fractions		Fractions	
		Mean	StdErr	Mean	StdErr
Typing	26	31.22	1.78	42.66	3.198
Handwriting	26	19.01	0.93	19.94	0.907
Multimodal	26	19.55	1.41	16.84	1.041

(d) Means table of interaction between appearance of fractions and condition with respect to time per problem during the learning phase.  $N$  is the number of participants.

Condition	$N$	No Fractions		Fractions	
		Mean	StdErr	Mean	StdErr
Typing	12	73.98	5.79	110.61	9.46
Handwriting	13	41.76	5.56	57.71	9.09
Multimodal	14	43.78	5.36	58.03	8.76

(e) Means table of test scores and gains from pre-test to post-test.  $N$  is the number of participants.

Condition	$N$	Pre-Test		Post-Test		(Post - Pre) Gain	
		Mean	StdErr	Mean	StdErr	Mean	StdErr
Typing	12	53.69	6.70	66.31	7.01	12.62	5.10
Handwriting	13	51.83	6.44	65.75	6.73	13.92	4.90
Multimodal	13	60.22	6.44	69.01	6.73	8.79	4.90

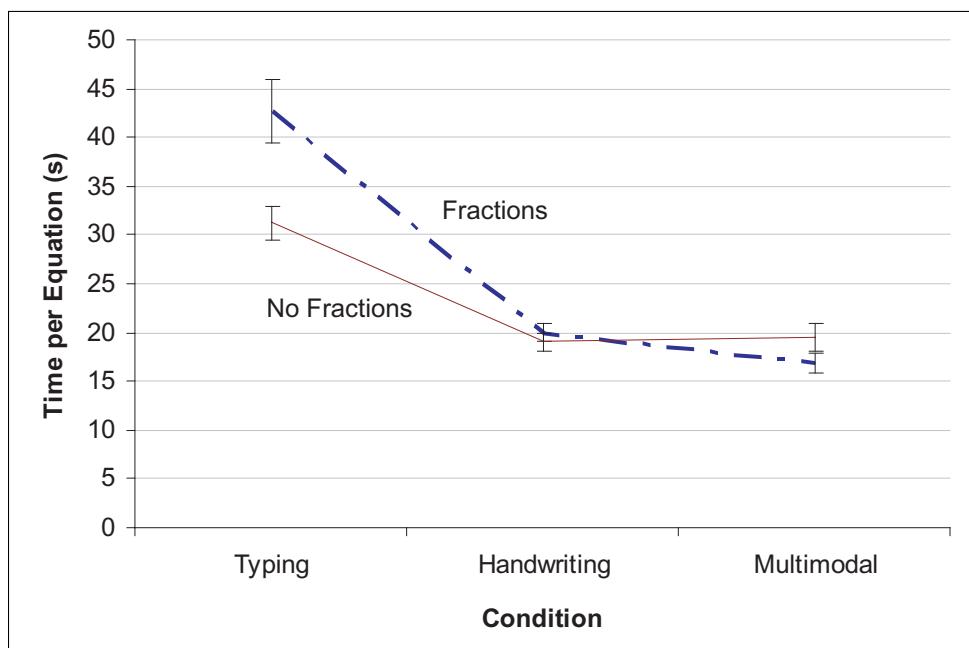
### Time on Task

The time students took to complete both the copying phase and the learning phase was measured. Means of time per equation by condition during the copying phase are shown in Table 4.5(a), and means of total time on task by condition during the learning phase are shown in Table 4.5(b). During both the copying and the learning phases, students took about twice as long to enter equations in typing than in the other two conditions, and this difference was significant.

In a repeated measures analysis of the within-subjects factor of condition with the dependent variable of average time per equation in the copying phase, a significant main effect of condition was found ( $F_{2,74} = 49.60, p < 0.05$ ), with a planned contrast showing typing as the slowest condition ( $F_{1,37} = 58.49, p < 0.05$ ). In a univariate ANOVA on total time to solve the problems with condition as a fixed factor in the learning phase, a significant main effect of condition was found ( $F_{2,35} = 11.05, p < 0.05$ ) in which typing was the slowest ( $t(35) = 12.31, p < 0.05$ ). These results replicate the findings from the Math Input Study that showed handwriting was the faster modality when compared to typing; the effect is smaller in this case (two times faster here *vs* three times faster in the Math Input Study), which may be due to the complexity of the interface used in the prior study (Microsoft Equation Editor). The time speed-up during the learning phase is not a direct measure of learning gain, but has important implications for learning in that students who can get through problems more quickly by virtue of a more natural interface can therefore advance further in the curriculum than if they had been typing.

Prior to this study, it was hypothesized that equations with nonlinear elements such as fractions would impact the time taken and/or learning experienced for the typing modality and not the handwriting modality. In both the learning and the copying phases, students took much longer to enter equations with fractions when typing, but not when handwriting. Estimated marginal means, given the factors of condition and presence of fractions, of time per equation in the copying phase are given in Table 4.5(c), and of time per problem in the learning phase are given in Table 4.5(d). In the copying phase, about 40% of the equations contained fractions. A separate repeated measures analysis on time per equation, using the 26 students for whom data was available on what problems contained fractions in the copying phase, revealed a significant interaction between the two within-subjects factors of condition and appearance of fractions ( $F_{1,25} = 4.76, p < 0.05$ ). In the learning phase, half of the problems contained fractions and half did not. A repeated measures analysis of the average time students took per problem to solve problems with fractions *vs* without fractions revealed a significant interaction between the between-subjects factor of condition and the within-subjects factor of appearance of fractions ( $F_{2,36} = 5.25, p < 0.05$ ).

Figure 4.8 shows the interaction plot of appearance of fractions and input condition for the learning phase. The typing condition is slowed down more by the appearance of fractions, whereas there is no difference in the other two conditions when fractions appear *vs* when they do not. This interaction between speed and complexity of the math replicates the findings in the Math Input Study that showed a similar interaction based on appearance of non-keyboard characters. The result reported in this study is more robust in that the interaction is based on spatial characteristics



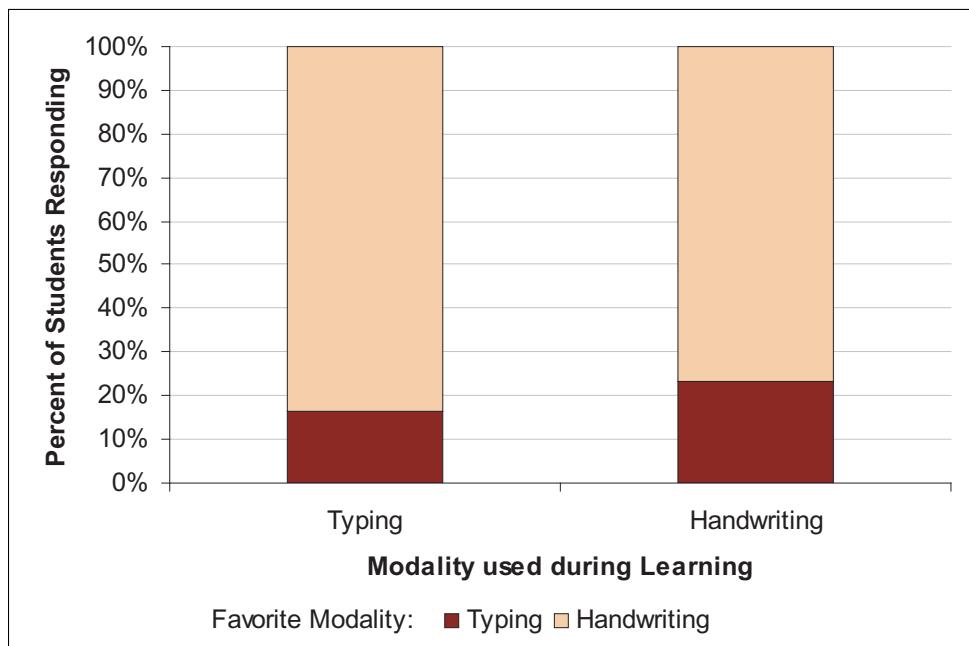
**Figure 4.8:** Mean time per problem by condition crossed with appearance of fractions in the *learning* phase for both copying examples and solving problems. Error bars indicate standard error.

of the math rather than simply what is easily typeable on a keyboard. This result implies that the speed benefits of handwriting input will magnify as students progress to more complex math such as polynomial algebra or calculus, which contain high frequencies of fractions, exponents, etc. Again, students may be able to progress further more quickly by virtue of handwriting's faster, more natural support for these notations.

### User Preferences

All students were exposed to all three conditions during the copying phase and then used only one modality during the learning phase. All qualitative comments in response to the question: “Did you like the modality you used during problem-solving? Why or why not?” are included in Table 4.6. Those students that did not answer the question or who selected the wrong modality (*i.e.*, it did not match the one they actually used during learning) have been excluded ( $n = 10$ ). Most students indicated that they did indeed like the method they used during the learning phase. However, none of the handwriting students indicated that they did not like this method at all. Note the student in the typing condition who said he or she liked the method, but “handwriting and speech made it even better.”

Students showed a strong preference for handwriting. Out of 38 total students, only 21% said typing was their favorite method, while over 78% preferred one of the methods with handwriting.



**Figure 4.9:** Histogram of user responses rating their *favorite* input modality grouped by the modality they used during the learning phase of the session.

As shown in Figure 4.9, this difference was not based on a bias to prefer the method used during learning (a Pearson Chi-Square test of independence reveals no significant association between the method used during learning and the method chosen as the favorite;  $\chi^2(4, N = 38) = 1.802, n.s.$ ). This lack of association implies little or no novelty effect of handwriting occurred.

### Learning Gains and Learning Efficiency

Means of test scores on both tests by condition, including gain scores from pre-test to post-test, are shown in Table 4.5(e). Gains are roughly equal across all three conditions; no significant difference was found. Despite taking about half the time during the learning phase, the handwriting students learned just as much as the typing students.

In a univariate ANOVA on learning gain from pre-test to post-test with condition as a fixed factor, there was no significant difference ( $F_{2,35} = 0.293, n.s.$ ). Even though students in the handwriting condition spent much less time to complete the same exercises in the instruction, they still learned just as much (*mean*: 11.75%, *stdev*: 17.34).

Although learning gains appeared to be of the same magnitude based on pre- to post-test scores, the fact that the time spent per condition was so different suggests that perhaps handwriting was a more *efficient* learning modality than typing. The concept of *learning efficiency* has been used in, for example, [123], to explore how students may be able to achieve similar levels of **mastery**

**Table 4.6:** List of all qualitative comments by students in answer to the question: “Did you like the modality you used during problem-solving? Why or why not?”

Learning Modality	Liked it?	Why or why not?
Typing	Yes	<p>“Because I don’t like using the handwriting on computers”</p> <p>“Because it helped me to see the answer and figure out [when] it’s done.”</p> <p>“Because it was nice”</p> <p>“Because it was easier and more fun”</p> <p>“I liked it because it was an easier way of doing the problems and it was more fun.”</p> <p>“[It] seemed easier although handwriting and speech made it even better”</p> <p>“It was easier than writing it out and saying everything”</p> <p>“It was easy and understandable. I’ve done things like this before.”</p> <p>“It was easy to use and understand”</p> <p>“It was [good] and easier by [computer]”</p>
	No	<p>“It took [too] long”</p> <p>“It took too long and was hard to get everything where I wanted”</p>
	(blank)	“[I’m] not really sure if I liked it or not”
Handwriting	Yes	<p>“Because I knew what I was writing down”</p> <p>“Because it is easier than typing”</p> <p>“Because [it’s] easy”</p> <p>“Because the Handwriting is more natural than speech or typing. It’s hard to write + speak.”</p> <p>“I thought it was a little bit helpful”</p> <p>“It was better than typing”</p> <p>“It was fairly easy to just write the problem out rather than type it”</p> <p>“It was faster than typing”</p> <p>“It was fun”</p> <p>“It was [just] like using pen and paper”</p> <p>“Yes, because it is how I’m used to doing problems in math class, by writing them out. It was harder [than] writing on paper though.”</p>
	No	<i>No students indicated this choice.</i>
Multimodal	Yes	<p>“Because I could hear my work and it is [easier] if I hear it.”</p> <p>“Because I like computers”</p> <p>“Because it was it was [easier] to [write] it than it was to type it.”</p> <p>“I could write it down and think about the problems so I thought I did better than just writing”</p> <p>“I enjoyed writing free hand and it helped”</p> <p>“I liked using a pen better and typing.”</p> <p>“I think it helped me.”</p> <p>“It helped me understand the problem more”</p> <p>“It made it easier to think it out when I said it while doing it.”</p> <p>“[It’s] easier to understand when you talk through the problems”</p>
	No	<p>“I kept forgetting that I had to use speech too”</p> <p>“My handwriting looked awful”</p>

but do fewer problems. In this study, students all did the same number of problems, but the time spent per session was significantly different by modality. If time in each modality is significantly different, it follows that learning *rate per hour* should be significantly different. The pre- to post-test gain per hour of instruction was extrapolated for each modality based on the average time over all three conditions per session for the learning phase (31 minutes). Handwriting performed the best at about 10 percentage points of gain per hour, multimodal was at about eight points per gain per hour, and typing was worst at about five points per gain per hour. Although there is a trend in favor of handwriting, there was no significant difference. The learning measurement of raw pre-test to post-test gain is relatively coarse, and the large standard deviation indicates that individual differences may have overshadowed the effects of condition in this study for learning benefits. Even so, handwriting had significant benefits in terms of other measures, such as time on task and transfer to paper, which may impact student performance.

### Transfer to Paper

One hypothesized advantage of using the handwriting modality is that handwriting will allow a greater degree of **transfer** to paper than using typing interfaces. In this study, level of transfer in each condition was assessed by correlating the pre-test score and post-test score with performance during training to see if there was any difference—if the tutor predicts performance effectively, there should be a significant correlation. It was hypothesized that, in the cases in which there was a *modality switch* (*i.e.*, writing on the pre-test to typing in the interface to writing on the post-test), there should be a lower correlation of performance during training to performance on the tests. A bivariate correlation was run on the proportion of problems solved correctly on the first try during training and the *pre-test* score, grouped by condition; the contrasts are shown in Table 4.7. The Pearson correlation for the typing condition was not statistically significant ( $r = 0.343, n.s.$ ), whereas for the two handwriting conditions, there was a significant, strong correlation ( $r = 0.613, p < 0.05$  for handwriting;  $r = 0.614, p < 0.05$  for multimodal). A separate bivariate correlation of training performance and the *post-test* score was conducted. The Pearson correlation for the typing condition in this case was again not statistically significant ( $r = 0.320, n.s.$ ), whereas for the two handwriting conditions, there was a significant, strong correlation ( $r = 0.708, p < 0.05$  for handwriting;  $r = 0.553, p < 0.05$  for multimodal). These results support the conclusion that handwriting affords students a higher degree of transfer to paper, likely because it does not involve a modality switch from training to testing. Performance during testing more closely matches performance during training when the modality of testing is similar to that of training (or vice versa), leading to stronger assessment power.

### Handwriting plus Speaking, or Multimodal Input of Math

Throughout this dissertation, the focus is primarily on contrasting typing with handwriting. However, multimodal input illustrated an interesting trade-off between speed and learning that did not

**Table 4.7:** Results of bivariate correlations between performance during training and on each of the two tests given during the study (pre-test and post-test). The correlation results are grouped by condition. † indicates significance at the 0.05 level; ‡ indicates marginal significance at the 0.10 level.

Performance during:			
Training	N	Pre-Test	Post-Test
Typing	12	.343	.320
Handwriting	13	.613†	.708†
Multimodal	13	.614†	.553†

manifest in the other two conditions. While the multimodal students were about as fast as the handwriting students, they did not learn as much, based on the time-adjusted score. (They did learn more than the typing students.) It appears from these preliminary data that the **cognitive load** of speaking while also writing, in terms of the extra cognitive processing needed, somehow interferes with learning the goal concept. Teachers frequently speak and write math while lecturing at the blackboard, yet it seems that students are not comfortable with generating solutions in both modalities at once. Prior literature has demonstrated that students learn better when they self-explain [31], and even further, that they learn better when their self-explanations are spoken rather than typed [63]. The multimodal input results presented here indicate that the learning benefits for students self-explaining aloud do not manifest in a “shadowing” paradigm in which students simply state aloud the problem-solving steps they perform; and are consistent with findings in the literature that paraphrasing is not as good as independent self-explanation [63]. Knowledge of these effects can inform future design of instructional paradigms that support or require student self-explanations.

### 4.2.3 Conclusions

This study found that the results from the Math Input Study *did* generalize to the target population of algebra learners and simpler mathematical expressions, although the effect size decreased. In the Math Input Study, handwriting was three times as fast; here in the Lab Learning Study, handwriting was twice as fast. This difference is likely due to the simpler nature of the equations given in this study (fewer advanced characters used), the simpler nature of the typing interface used (to be more like the handwriting interface), and the addition of the learning task. The Lab Learning Study also found that, although students in the handwriting condition spent half as much time in the tutoring system, they learned just as much, implying that the extra time spent in the typing condition was wasted. With respect to the effect of worked examples, students experienced sizeable learning gains from pre-test to post-test in spite of being given only **answer-level feedback**, in the context of worked-examples-based instruction. This finding lends credence to the hypothesis that worked examples may be an appropriate method of instruction when using handwriting input

interfaces that may not be able to support step-targeted feedback. Finally, the most interesting finding here was that there was better transfer-to-paper in the handwriting conditions than in the typing condition: the tutor better predicted student performance on the post-test in the handwriting condition. Predicting paper-based performance is critical for accurate assessment of student knowledge within curricula that utilize intelligent tutoring systems.

### 4.3 Study 3: The Cognitive Tutor Study

Based on the findings of the Math Input Study and the Lab Learning Study, the motivation for including handwriting input in tutoring applications was very strong. The handwriting modality had been shown to have advantages both in terms of usability and in terms of impact on learning. The next step was whether or not the instructional paradigm could be modified to account for the fact that **step-targeted feedback** was difficult, if not impossible, to provide given current recognition levels.

The research questions addressed by this study include:

- Do students experience differences in learning due to the modality in which they generate their answers?
- Do the benefits due to the presence of worked examples sufficiently counteract the disadvantages of the lack of step-targeted feedback?
- Do the results from the Math Input Study and the Lab Learning Study for the benefits of handwriting, in terms of time and user satisfaction, generalize to a more complex tutoring system and classroom environment?
- Do the results from the Lab Learning Study, where students in a lab study experienced better transfer to paper when handwriting than when typing, generalize to a more complex tutoring system and classroom environment?
- Do students experience less cognitive load as measured by self-report when they use handwriting to solve problems than when they use typing?

Overall, the Cognitive Tutor Study found that worked examples provided added value to the normal Cognitive Tutor, even without handwriting, which is positive evidence that they can be instructionally helpful in this context. In addition, step-targeted feedback was important for student learning, and handwriting, while outperforming typing input *without* step-targeted feedback, did not outperform typing input *with* step-targeted feedback *and* examples. Therefore, the Cognitive Tutor Study determined that step-targeted feedback is very important instructionally for students in the math tutoring domain, and so the technology needs to be improved to be able to provide more than just answer-level feedback.

### 4.3.1 Experimental Design

The Cognitive Tutor Study was a between-subjects, *in vivo* classroom study, enabling the examination of the use of handwriting-based input in a real-world setting. This is important because classroom environments are often much less organized, noisier, and students are less apt to cooperate with the experimenter than they are in lab studies. The three factors this study explored were input modality (handwriting or typing); level of feedback (step-targeted or answer-level), and presence of annotated worked examples (yes or no).

### Participants

Eight Algebra I classrooms taught by four different teachers at two different schools (one urban public high school; one suburban vocation/technical high school) participated in the study, for a total of 106 participants. Students in the classrooms were in grades nine through 12 (ages 13 through 18, exact ages unknown). Of these, 31 students were removed due to absences for one or both of the pre-test or post-test, or due to spending less than 30 minutes in the computer tutor during the study. These exclusion criteria left 76 participants, of which at least 27 were female (demographic data such as grade, gender and ethnicity were unavailable for 14 students). Approximately 48 were Caucasian and 28 were African-American. At least 27 were in ninth grade, 31 were in tenth grade, and four were in eleventh grade. Exploratory data analysis revealed no effects of gender, age or school on any factors, so they are excluded from the analyses reported here.

### Factorial Design

The experiment design was a between-classroom study in which two classrooms were randomly assigned to each of four conditions, with the constraint that no two classrooms taught by the same teacher or at the same school were in the same condition.

The four conditions selected for the study were as follows:

1. typing, no examples, step-targeted feedback (control: Cognitive Tutor; CogTutorS in tables and graphs)
2. typing, examples, step-targeted feedback (CTExamplesS in tables and graphs)
3. typing, examples, answer-level feedback (CTExamplesA in tables and graphs)
4. handwriting, examples, answer-level feedback (HWExamplesA in tables and graphs)

The 2x2x2 matrix of possible conditions is shown in Table 4.8; the selected conditions are shaded. The design of the study satisfied three constraints: the control condition was the normal Cognitive Tutor; the handwriting condition made use of worked examples as a way of mitigating the lack of step-targeted feedback, which was not possible at that time; and each condition differed from the next and previous by only one factor. This study was also designed to determine whether

Modality	Examples Present		Type of Feedback
	No	Yes	
Typing	1	2	Step-targeted
		3	Answer-only
Handwriting		4	Step-targeted Answer-only

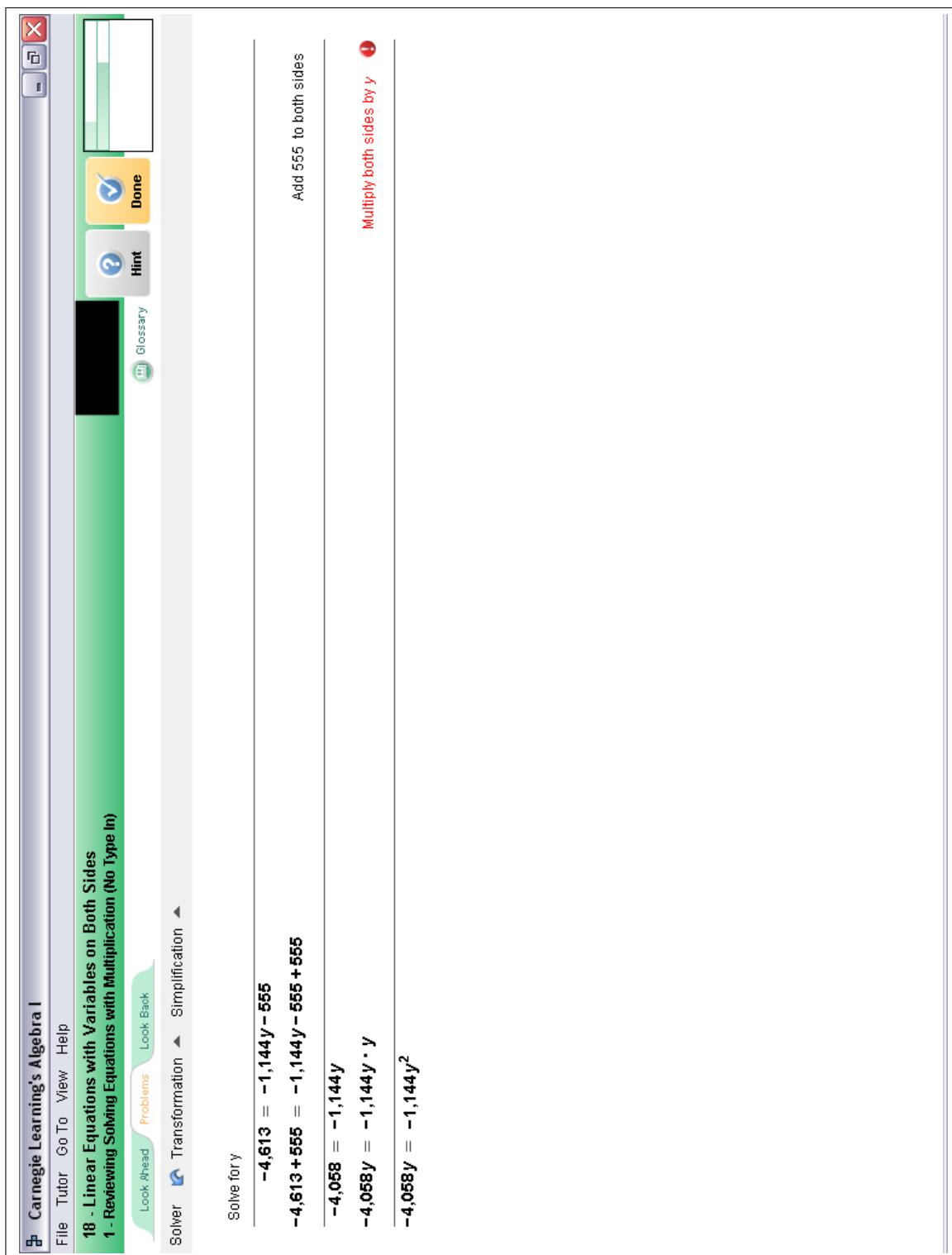
**Table 4.8:** Experimental design matrix showing the full cross of the three experimental factors; the shaded cells indicate conditions in the Cognitive Tutor Study.

or not the benefits of handwriting-based input, plus worked examples, would be sufficient to outweigh the disadvantage of not having step-targeted feedback. All four conditions as they were seen in the study are shown in Figures 4.10 (condition 1: CogTutor-NoExamples-StepFeedback), 4.11 (condition 2: CogTutor-Examples-StepFeedback), 4.12 (condition 3: CogTutor-Examples-AnswerFeedback), and 4.13 (condition 4: Handwriting-Examples-AnswerFeedback).

### Procedure

Students completed one unit in the tutoring system over two classroom periods of 75 minutes or three classroom periods of 49 minutes, depending on their school’s schedule. If students finished the unit early, they went back to their previous work. All students in each classroom took the post-test at the same time. There were three unique test forms (A, B, and C). Classrooms were randomly assigned as to which test form to use for each of the three tests. All materials for this study are given in Appendix D. The material covered in this study was Unit 18 in the 2006 Cognitive Tutor Algebra curriculum. This unit covered two- and three-step algebra equations, and contained problem types such as  $ax + b = c$ ,  $ax + b = cx + d$ , and  $a/x + b = c$ , with integers, decimals and large numbers (greater than 1,000). Students had to demonstrate mastery on each problem type before moving on to the next.

Students in the control condition (CogTutor-NoExamples-StepFeedback) worked with the standard Cognitive Tutor. The Cognitive Tutor equation solver widget requires them to solve a given equation by selecting equation-solving operations from a menu. In some sections of the unit, the result of the operation is computed for the student (*no-type-in*); in others, the student is required to type in the result (*type-in*). Students in two of the three treatment conditions (CogTutor-Examples-StepFeedback and CogTutor-Examples-AnswerFeedback) used the same equation solver interface, but were also given a worked example, annotated with step-level instructions about what operations were being performed, to copy each time they encountered a new problem type (in the *no-type-in* sections only). They copied the example by entering the solution steps into the interface, and the instructions encouraged students to study each step and consider why it was taken as



**Figure 4.10:** A screenshot of the CogTutor-NoExamples-StepFeedback condition (control condition) in the Cognitive Tutor Study.

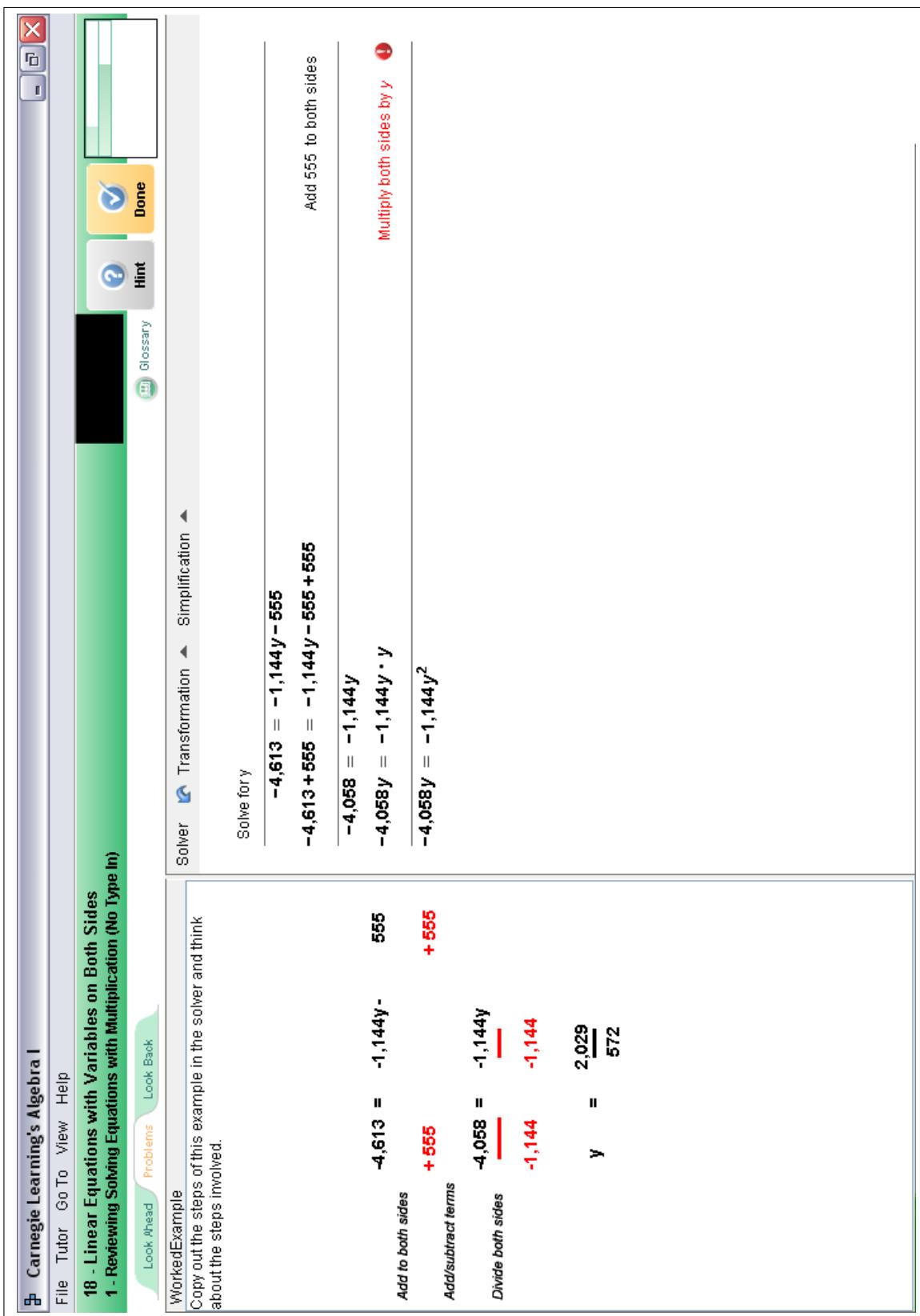


Figure 4.11: A screenshot of the CogTutor-Examples-StepFeedback condition in the Cognitive Tutor Study.

The screenshot shows the Carnegie Learning's Algebra I software interface. The top menu bar includes File, Tutor, Go To, View, Help, and a red X button. Below the menu is a toolbar with icons for zoom and a search bar. The main title is "18 - Linear Equations with Variables on Both Sides". A sub-section title is "1 - Reviewing Solving Equations with Multiplication (No Type In)". On the left, there are navigation buttons: "WorkedExample", "Look Ahead", "Problems", "Look Back", and "Glossary". A status bar at the bottom says "Add/subtract terms" and "Divide both sides". The central workspace displays the following steps of a worked example:

Solve for  $y$

$$\begin{aligned} -4.613 &= -1.144y - 555 \\ -4.613 + 555 &= -1.144y - 555 + 555 \end{aligned}$$

Add 555 to both sides

$$\begin{aligned} -4.613 + 555 &= -1.144y - 555 + 555 \\ -4.058 &= -1.144y \end{aligned}$$

Add/subtract terms in  $-4.613 + 555$

$$\begin{aligned} -4.058 &= -1.144y \\ -4.058y &= -1.144 \cdot y \end{aligned}$$

Multiply both sides by  $y$

$$\begin{aligned} -4.058y &= -1.144y \cdot y \\ y &= \frac{2.029}{572} \end{aligned}$$

Add/subtract terms in  $-1.144y - 555 + 555$

**Answer Checker**  **Check my Answer**

**Figure 4.12:** A screenshot of the CogTutor-Examples-AnswerFeedback condition in the Cognitive Tutor Study.

The screenshot shows a software interface for Carnegie Learning's Algebra I. The top menu bar includes File, Tutor, Go To, View, Help, and a search bar. Below the menu is a navigation bar with tabs: Look Ahead, Problems, Look Back, Worked Example, and a redacted tab. The main content area displays a handwritten equation  $-4613 = -1144y - 555$ . Above the equation are two buttons: a green 'Done' button and a blue 'HandwritingInput' button. To the right of the equation are 'Draw' and 'Erase' tools. The bottom section shows the steps for solving the equation:

$$\begin{aligned} -4613 &= -1144y - 555 \\ +555 &\quad \quad \quad +555 \\ -4058 &= -1144y \\ \underline{-1144} &\quad \quad \quad \underline{-1144} \\ y &= \frac{2,029}{572} \end{aligned}$$

On the right side, there is an 'Answer Checker' field containing 'y =' and a 'Check my Answer' button.

**Figure 4.13:** A screenshot of the Handwriting-Examples-AnswerFeedback condition in the Cognitive Tutor Study.

they copied it. On subsequent problems of the same problem type, an analogous worked example was displayed for reference as they solved a problem. In the *type-in* sections, no examples were given. In this way **scaffolding** of the examples was implemented, so that eventually students in all conditions had to solve problems on their own with no interface help.

In the CogTutor-Examples-StepFeedback condition, the detailed, immediate feedback of the standard Cognitive Tutor was used. In the CogTutor-Examples-AnswerFeedback condition, feedback was only given on the student's final answer. Students clicked a button in the tutor to check their final answer. If the answer given was incorrect, students were given error feedback and asked to try again. To prevent a student from becoming stuck, if he got the answer wrong three times, the solution to the problem was displayed for him to study and copy (like a worked example), which the same paradigm used in the Lab Learning Study.

Students in the Handwriting-Examples-AnswerFeedback condition (the only handwriting condition) entered their solution process in a blank space onscreen via an electronic stylus attached to their monitor. They also had examples during problem solving, and they were only given feedback on their final answer, which they typed into a text box to eliminate ambiguity.

## Measures

A 10-item pre-test and 10-item post-test were administered by the individual teachers. These tests contained two review problems (problems from prerequisite lessons), five “normal” problems (problems from the current lesson), and three transfer problems (problems requiring one additional step or skill than normal problems). The detailed log files of the students' tutor use during Unit 18, including total amount of time spent in the tutor, were also collected.

There was also a 10-item follow-up retention test three weeks after the post-test; and there was a short cognitive load survey given immediately before students took the post-test. Modeled after that used in [104], the survey asked students to rate, on a nine-point Likert scale, their perceived mental effort (*cognitive load*) during the study and how it compared to normal tutor use, and to indicate the primary source of this mental effort.

### 4.3.2 Results and Discussion

All means for quantitative measures reported for the Cognitive Tutor Study are given in Table 4.9. Individual tables are referenced in each relevant section that follows.

#### Gains from pre-test to post-test and retention test

Means of test scores on all three tests by condition, including gain scores from pre-test to post-test and pre-test to retention test, are shown in Tables 4.9(a) and 4.9(b), respectively. No difference was found in gains from pre-test to post-test, but there was a significant difference by condition in gains from pre-test to retention test.

**Table 4.9:** Means tables for all measures reported from the Cognitive Tutor Study.(a) Means table of test scores and gains from pre-test to post-test.  $N$  is the number of participants.

<b>Condition</b>	<b><math>N</math></b>	<b>Pre-Test</b>		<b>Post-Test</b>		<b>(Post - Pre) Gain</b>	
		<b>Mean</b>	<b>StdErr</b>	<b>Mean</b>	<b>StdErr</b>	<b>Mean</b>	<b>StdErr</b>
CogTutorS	13	16.92	5.04	13.85	4.82	-3.08	3.72
CTExamplesS	15	19.33	4.69	18.00	4.48	-1.33	3.46
CTExamplesA	23	24.35	3.79	18.26	3.62	-6.09	2.79
HWExamplesA	25	17.20	3.63	15.20	3.47	-2.00	2.68

(b) Means table of test scores and gains from pre-test to retention test.  $N$  is the number of participants.

<b>Condition</b>	<b><math>N</math></b>	<b>Pre-Test</b>		<b>Retention-Test</b>		<b>(Retention - Pre) Gain</b>	
		<b>Mean</b>	<b>StdErr</b>	<b>Mean</b>	<b>StdErr</b>	<b>Mean</b>	<b>StdErr</b>
CogTutorS	8	15.00	6.12	16.25	5.83	1.25	4.05
CTExamplesS	10	21.00	5.48	42.00	5.21	21.00	3.63
CTExamplesA	13	25.39	4.80	20.00	4.57	-5.39	3.18
HWExamplesA	14	17.14	4.63	30.00	4.41	12.86	3.06

(c) Means table of total time spent in tutor.  $N$  is the number of participants.

<b>Condition</b>	<b>Mean</b>	<b><math>N</math></b>	<b>StdErr</b>
CogTutorS	100.62	13	10.15
CTExamplesS	103.20	15	9.45
CTExamplesA	108.44	23	7.63
HWExamplesA	81.20	25	7.32

(d) Means table of the aggregate mental effort measure. Higher numbers mean students indicated they experienced more mental effort.  $N$  is the number of participants.

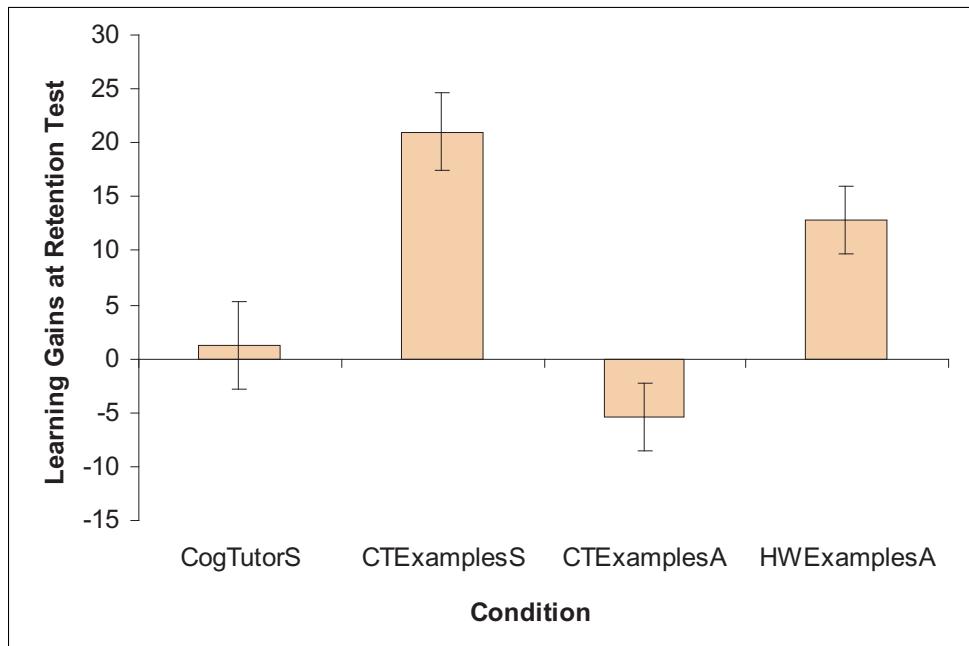
<b>Condition</b>	<b>Mean</b>	<b><math>N</math></b>	<b>StdErr</b>
CogTutorS	4.885	13	0.388
CTExamplesS	5.200	15	0.361
CTExamplesA	5.895	19	0.321
HWExamplesA	5.200	25	0.279

In a univariate ANOVA on learning gains computed as change in score (% correct) from pre-test to post-test, no significant difference by condition was found ( $F_{3,72} = 0.52$ , *n.s.*). However, a similar analysis on the 44 students who had taken retention tests did find a significant difference by condition in gain from pre-test to retention test ( $F_{3,41} = 11.91$ ,  $p < 0.05$ ). Tukey's post-hoc comparisons indicated that CogTutor-Examples-StepFeedback was better than CogTutor-NoExamples-StepFeedback ( $t(41) = 19.75$ ,  $p < 0.05$ ); CogTutor-Examples-StepFeedback was better than CogTutor-Examples-AnswerFeedback ( $t(41) = 26.39$ ,  $p < 0.05$ ); Handwriting-Examples-AnswerFeedback was better than CogTutor-Examples-AnswerFeedback ( $t(41) = 18.24$ ,  $p < 0.05$ ); and Handwriting-Examples-AnswerFeedback was marginally better than CogTutor-NoExamples-StepFeedback ( $t(41) = 11.61$ ,  $p = 0.118$ ). No other contrasts were significant. A graph of the estimated marginal retention gain means are shown in Figure 4.14.

While no significant difference in improvement from pre-test to *post-test* between conditions was found, a significant difference at *retention test* time was seen. Students may have been receiving subsequent instruction on these topics in the classroom, either through homework or class lectures. One explanation for the effect is that students in the better-performing conditions (the CogTutor-Examples-StepFeedback and Handwriting-Examples-AnswerFeedback conditions) might have been better prepared and experienced *accelerated future learning* [32]. Six out of the eight students in the CogTutor-NoExamples-StepFeedback condition who took the retention test did work in the control Cognitive Tutor on the unit covered by this study between the post-test and retention test times. It can be speculated that students in the other conditions also spent extra time on the material, in part because CogTutor-NoExamples-StepFeedback was not the best-performing condition on the retention test. However, this speculation could not be confirmed with the log data available at the time of writing; later additions to the DataShop of control Cognitive Tutor use (*i.e.*, students using the regular tutor as part of their normal classtime, outside of any study) might be able to shed light on this issue. Also, further evidence for this speculation comes from finding a significant correlation between post-test scores and retention scores (Pearson coefficient  $r = 0.351$ ,  $p < 0.05$ ). If there were no correlation, it would point to the strong possibility that students in different conditions were spending different amounts of extra time on this tutor lesson. Though the correlation is in the medium-strength range, this possibility cannot be completely ruled out.

An interesting finding of this study is that the CogTutor-Examples-StepFeedback condition was significantly better with respect to learning than the control condition, CogTutor-NoExamples-StepFeedback. The presence of worked examples was the sole difference between the two conditions. This study therefore provides good evidence in favor of using worked examples to supplement problem solving in the domain of algebra. Interspersing the annotated examples with the problems, leaving the examples for reference during problem solving, and fading the examples with time seemed to work well for these students. These findings contribute to the science of learning by providing instructional design recommendations for the use of worked examples and their effectiveness in algebra equation solving.

Although the handwriting condition performed well, there is reason to believe it did not perform as well as it could have. Bringing technology into a real-world classroom setting for the first time



**Figure 4.14:** Estimated marginal means of learning gains as measured from pre-test to retention test condition in the Cognitive Tutor Study. The key finding is that the handwriting condition (Handwriting-Examples-AnswerFeedback) is only marginally significantly better than the control (CogTutor-NoExamples-StepFeedback). Error bars indicate standard error.

is an error-prone process. Several technical glitches occurred during the study caused by the stylus devices used to allow the students to enter their solutions via handwriting. The devices interfered with each other in close proximity and caused the stylus to respond in unpredictable ways. As the technology improves and becomes more widely used (*e.g.*, TabletPCs in schools), these issues will no longer be present.

### Time spent in tutor

Means of total time spent in the tutor by condition are shown in Table 4.9(c). Students in the Handwriting-Examples-AnswerFeedback condition spent less time in the lesson than other students, but this difference was only marginally significant.

A univariate ANOVA with number of minutes spent in the tutor between pre-test and post-test as the dependent variable and condition as a between-subjects factor was conducted, and a marginally significant difference was found ( $F_{3,72} = 2.49, p = 0.067$ ). In the Math Input and Lab Learning studies, handwriting was much faster than typing interfaces for mathematics, whereas here the difference is only marginal. This diminished effect could be because students in the classroom are essentially controlled for time; knowing how much time they have remaining in

the class period may affect how quickly students work, whereas in a lab situation they are able to finish early and leave. In addition, the technical glitches involved in using the handwriting devices (mentioned in the discussion on learning) likely also interfered with students' efficiency. The difficulties with the technology slowed students down and forced them to re-do problems at times, which would affect their problem-solving speed.

### Cognitive load, or Mental effort

Cognitive load was measured via self-report—students answered two questions about the degree of mental effort they experienced during the study and how this mental effort differed from their normal use of the Cognitive Tutor. These two questions can be treated as two measures of the same quantity, that of cognitive load, along some *scale* whose exact parameters are unknown. Therefore, in analyses of the amount of cognitive load in this study, the two questions were aggregated by averaging each student's response to each question. Means of the aggregated mental effort rating by condition are shown in Table 4.9(d). A bivariate correlation of the two dependent measures represented by each question was run to verify that they can in fact be reliably assumed to be estimating the same quantity. The Pearson coefficient was statistically significant ( $r = 0.390$ ,  $p < 0.05$ ).

A univariate ANOVA with the aggregated mental effort rating as the dependent variable and condition as a between-subjects factor was conducted. No significant difference was found ( $F_{3,68} = 1.59$ , *n.s.*), although there was trend in favor of the Handwriting-Examples-AnswerFeedback condition. A planned contrast between the CogTutor-Examples-AnswerFeedback and the Handwriting-Examples-AnswerFeedback conditions, the pair of conditions that most directly show the impact of just the input modality, showed a marginally significant trend in favor of Handwriting-Examples-AnswerFeedback ( $t(68) = 0.695$ ,  $p = 0.11$ ).

This trend lends support to the hypothesis that handwriting interfaces lead to a reduction in cognitive load as compared to typing interfaces, although the difference is not significant. The technical difficulties of using the handwriting input devices during the study, as well as the familiarity students had with the other interfaces, are the likely contributing factors to the lack of significance. Students in these classrooms all used the normal Cognitive Tutor during their math courses, and were therefore already familiar with how the Cognitive Tutor interface worked. Students in the handwriting condition were therefore somewhat at a disadvantage; however, training time on the handwriting devices was not possible due to the strict constraints on teaching and lab time in the classroom environment. This issue deserves further exploration in future work on math tutoring systems and handwriting input.

In addition, a Pearson Chi-Square test of independence with condition and *source of mental effort* ("mostly the problems," "mostly the interface," or "about the same for both") was significant ( $\chi^2(6, N = 72) = 16.91$ ,  $p < 0.05$ ). The cross-tabulated responses are shown in Table 4.10. The responses in Table 4.10 indicate that the students in the CogTutor-Examples-AnswerFeedback condition were more likely to say that the problems were the source of their mental effort during

**Table 4.10:** Cross tabulated totals of responses of students to the question: “What was the source of the mental effort?”

Condition: Source:	CogTutorS	CTExamplesS	CTExamplesA	HWExamplesA
Mostly the problems	7 (54%)	8 (53%)	12 (63%)	6 (24%)
Mostly the interface	2 (15%)	0 (0%)	0 (0%)	9 (36%)
About the same for both	4 (31%)	7 (47%)	7 (37%)	10 (40%)
Total respondents	13	15	19	25

the study. Students’ difficulties in this condition suggest that the presence of worked examples may not completely compensate for the lack of step-targeted feedback. Students in this condition used the standard Cognitive Tutor equation solver interface, but without its step-targeted feedback. The standard Cognitive Tutor does not allow a student to go to the next step if their answer to the current step is incorrect; the prototype did allow this. Even though this was explained before the study began, students interpreted being allowed to move on as implicit feedback of being correct. Although the handwriting condition also did not have step-targeted feedback, its interface was completely new to them and so their mental model of how it should work was more flexible. In the future, effective interface designs that do not conflict with students’ mental models must be considered more closely.

The responses in Table 4.10 also indicate that students in the Handwriting-Examples-Answer-Feedback condition were more likely to say that working with the interface was the primary source of their mental effort experienced during the study. This is unfortunate, since the goal of introducing handwriting input was to *reduce* extraneous cognitive load caused by the interface, but not surprising given the events of the actual study. The technical difficulties of using the handwriting devices in the classroom (mentioned in the discussion on learning) frustrated students, which likely led directly to the results in Table 4.10.

### Transfer to Paper

The Lab Learning Study found that tutoring that utilized handwriting instead of typing tended to better predict student performance on the tests, implying that students experienced better transfer to paper and working on their own. In the Cognitive Tutor Study, the same analysis was conducted to determine if a similar result would hold true. In the cases in which there was a *modality switch* (*i.e.*, writing on the pre-test to typing in the interface to writing on the post-test or retention test), there should be a lower correlation of performance during training *vs* to performance on the tests.

A bivariate correlation was run on the proportion of problems solved correctly on the first try during training and the *pre-test* score, grouped by condition. Similar analyses were also done for the *post-test* and *retention-test* scores. The results of these correlations are shown in Table 4.11.

**Table 4.11:** Results of bivariate correlations between performance during training (full credit only) and on each of the three tests given during the study (pre-test, post-test, and retention test). The correlation results are grouped by condition. † indicates significance at the 0.05 level; ‡ indicates marginal significance at the 0.10 level.

<b>Training</b>	<b>N</b>	<b>Performance during:</b>		
		<b>Pre-Test</b>	<b>Post-Test</b>	<b>Retention Test</b>
CogTutorS	13	.501‡	.628†	.645†
CTExamplesS	15	.154	.342	.486‡
CTExamplesA	23	.620†	.575†	.414†
HWExamplesA	25	.310	.329‡	.055

The main results can be summarized as follows:

1. The CogTutor-NoExamples-StepFeedback condition shows significant or marginally significant correlations between training and all three tests, indicating that the anecdotal evidence about lack of transfer to paper does not hold up in this study.
2. The CogTutor-Examples-AnswerFeedback condition shows significant correlations between training and all three tests. This condition was the worst with respect to overall learning, however, so these correlation results simply show that students did just as badly during training with this interface as they did when later tested.
3. The Handwriting-Examples-AnswerFeedback condition does *not* show a significant correlation of the magnitudes seen in the Lab Learning Study.

It is likely that two factors were responsible for the lack of correlation in the handwriting condition. First, the technical difficulties with the handwriting devices, mentioned during the discussion on learning, interfered with students during training such that they were unable to effectively transfer their knowledge outside of the tutoring environment. Second, the familiarity that these students already had with the Cognitive Tutor interface most likely helped them in the other three conditions, especially the CogTutor-NoExamples-StepFeedback control condition.

### 4.3.3 Conclusions

In summary, the Cognitive Tutor Study found that the usability benefits of handwriting input continue to hold, in terms of total time spent during the lesson. Key findings with respect to pre-test to retention-test gains include the fact that Handwriting-Examples-AnswerFeedback was significantly better than CogTutor-Examples-AnswerFeedback, in which the direct comparison is the presence of handwriting input, implying that handwriting input is better than typing, all

else being equal; that CogTutor-Examples-StepFeedback was better than CogTutor-NoExamples-StepFeedback, in which the direct comparison is the presence of examples, implying that having interspersed examples is better than not having them, all else being equal; and that CogTutor-Examples-StepFeedback was better than CogTutor-Examples-AnswerFeedback, in which the direct comparison was step-targeted *vs* answer-level feedback, implying that step-targeted feedback is crucial to student success, all else being equal.

However, while a marginal effect was found in terms of reduction in cognitive load and improved transfer to paper of the handwriting condition, the benefits of handwriting input were not as strong in these areas as hypothesized or found in earlier studies. The technical difficulties that were encountered with using the handwriting devices in the classroom reduced the effectiveness of the handwriting condition in this study. In the Math Input Study and Lab Learning Study, strong evidence in favor of handwriting input was found in terms of speed, user preferences and transfer to paper. The Cognitive Tutor Study did find several important facts that support pursuing handwriting input further, but in order to do so in future studies, the technical difficulties must be addressed. Now that some of these difficulties are better known, especially in terms of the specific devices used here, future studies can be designed to avoid or account for them more directly.

Recall that Handwriting-Examples-AnswerFeedback was marginally better than CogTutor-NoExamples-StepFeedback in terms of learning gains from pre-test to retention test. From this, it can be concluded that, in spite of the difficulties with the handwriting modality, students were still able to perform marginally better in that condition than in the control condition. This is an important finding which indicates the value of further pursuit of this line of work.

## 4.4 General Conclusions from the Three Studies

The three studies presented in this chapter laid the foundation for proving that handwriting recognition can have benefits for math-oriented tasks, both in terms of usability and pedagogy. It was found that, in general, handwriting is faster and better-liked than typing; and in learning domains, that students sometimes experience better transfer to paper (and therefore better independent knowledge) when they transition from working in the tutor to working on their own in handwriting than they do in typing. All of these studies included at least one condition that allowed the participants to input mathematics on the computer via handwriting. None of the studies included recognition of said handwriting input. The focus was on discovering what could be achieved with the modality and how comfortable users would be with the modality. Therefore, the studies have helped establish the motivation to apply handwriting *input* in this domain. It was known before the studies were performed that handwriting recognition was not perfectly accurate. These studies also have helped establish that work is needed on the technical challenges of handwriting *recognition* in order to provide natural interfaces that support effective pedagogical strategy during tutoring, as step-targeted feedback was found to be critical in the Cognitive Tutor Study.

The next chapters focus on how handwriting recognition was adapted for this domain for the purposes of this dissertation, including training the recognition engine in advance on a training set of math writing from the target population, and using domain-specific context information during the recognition process to help refine the results.

# Chapter 5

## Improving Recognition: Baseline Writer-Independent Accuracy

Chapters 3 and 4 established some of the ways in which handwriting input can benefit users of math software and students learning math. In the three studies presented, the users only entered their input via handwriting; there was no recognition engine behind it. Interpretation of student input was either not performed, or was performed by a human, in the **Wizard-of-Oz** paradigm. The results of the Cognitive Tutor study showed that adding unrecognized handwriting input is enough to beat the traditional Cognitive Tutor without **worked examples**, but not enough to beat it with worked examples and **step-targeted feedback**. Therefore, the next two chapters discuss the ways in which handwriting recognition was incorporated into the interface of a state-of-the-art tutoring system, Cognitive Tutor Algebra, beginning with a search for the best recognition engine for this domain and then discussing the training, testing, implementing, iterating and evaluating that was conducted.<sup>1</sup>

### 5.1 Choosing a Recognition Engine: Case Studies

For the purposes of this dissertation, three different easily accessible and independently-developed recognition engines were compared: (1) the Freehand Formula Entry System (FFES) [131]; (2) JMathNotes [137, 138]; and (3) Microsoft's TabletPC recognizer<sup>2</sup>. The rationale for choosing these specific systems was to explore the most commonly used, or representative, recognition systems available for application developers wishing to incorporate handwriting input into new interfaces.

---

<sup>1</sup>The content in this section is partially based on content from these publications: [11, 14].

<sup>2</sup><http://msdn.microsoft.com/en-us/library/aa510941.aspx>

### 5.1.1 Freehand Formula Entry System

The Freehand Formula Entry System<sup>3</sup> [131] is based on the California Interface Tools (CIT) character recognizer<sup>4</sup>. It uses a nearest-neighbor classification based on a 48-dimensional feature space and is implemented in C++ under the Gnu Public License. In total there are almost 12,000 lines of code in 48 source files, not counting the included spatial math parser or user interface. Its minimal set of dependencies includes a Unix-like environment (or cygwin<sup>5</sup> on Windows) and ActiveTcl<sup>6</sup>. As it is typical of many research systems, its documentation is minimal and there is no discrete application programming interface (API).

FFES recognizes mathematical equations in a two-component process: character recognition as mentioned above [131], and mathematical expression parsing (DRACULAE) [149]. This recognizer is extremely flexible—it can be easily trained to any symbol set needed, for instance, the mathematical symbols and numbers that are needed in the beginning algebra domain. Stroke grouping (character segmentation) is performed via an algorithm that finds the highest confidence grouping of a set of  $m$  recently drawn strokes, where  $m$  is the maximum number of strokes in any symbol in the recognizer’s symbol set.

In papers published on FFES, accuracy measures on simple experiments involving five users were reported. *Symbol*-level accuracies of 77% were achieved for users to whom the system had not been specifically trained (*writer-independent*), and rates as high as 95% were achieved for users to whom the system *had* been specifically trained (*writer-dependent*) [131].

### 5.1.2 JMathNotes

The second recognition system tested was JMathNotes<sup>7</sup> [137, 138]. It uses a multi-class support-vector machine (DAG-SVM) classifier [110] using the radial basis function (RBF) kernel [26]. It is written in Java under the Gnu Public License. The entire system has several dependencies: Weka<sup>8</sup>, Jama<sup>9</sup>, and Lax<sup>10</sup>, but it can run on either Windows or Unix environments. Its documentation is nonexistent, and the tool primarily exists as a demonstration of the author’s SVM techniques for character recognition. The SVM implementation and recognition code consists of 42 source files and almost 9,000 lines of code.

JMathNotes recognizes characters using 12 different SVM models built for four subsets of characters: lowercase Latin alphabet, uppercase Latin alphabet, Greek letters, and digits/operators. These subsets are further divided by the number of strokes per character (one, two, or three). Some

---

<sup>3</sup><http://www.cs.rit.edu/~rlaz/ffes/>

<sup>4</sup>CIT was developed by Jim Arvo.

<sup>5</sup><http://www.cygwin.com>

<sup>6</sup><http://www.activestate.com/Products/activetcl/>

<sup>7</sup><http://ostatic.com/32644-software-opensource/jmathnotes>

<sup>8</sup>Open-source machine learning toolkit: <http://www.cs.waikato.ac.nz/ml/weka/>

<sup>9</sup>Java Matrix library: [math.nist.gov/javanumerics/jama/](http://math.nist.gov/javanumerics/jama/)

<sup>10</sup>“Launch Anywhere,” a retired platform-independent Java application launcher tool.

characters can be written in multiple styles with different numbers of strokes; they appear in more than one model.

SVM training can be quite slow due to the requirement of learning a boundary between all pairs of classes, whereas in the case of FFES and nearest-neighbor classification, training is extremely fast. Stroke grouping is performed based on the geometric distance from the current stroke to previously drawn strokes or symbols. See [137] for further details on the features, kernel functions, and type of multi-class SVM used in JMathNotes.

No accuracy was reported on JMathNotes in prior publications on the recognizer [137, 138].

### 5.1.3 Microsoft TabletPC Recognizer

The third system tested was the Microsoft TabletPC recognizer<sup>11</sup>. The handwriting recognition engine in Microsoft TabletPC Windows is widely regarded to be the state-of-the-art commercial handwriting recognition engine [146]. Microsoft's TabletPC writer-independent recognition software relies on a very large database of character samples from a variety of writers, as well as special-purpose heuristics to help distinguish easily confusable characters, such as 'O' (the letter oh) and '0' (the number zero).

The TabletPC recognizer is designed for general English text input rather than mathematics symbols and spatial notations. Unlike the feature-based single symbol classification used by FFES and JMathNotes, Microsoft's TabletPC recognizer uses a neural network to identify input on the *word* level, by using knowledge about common letter sequences in English words. While this may serve to improve the accuracy of the TabletPC's recognition on dictation, command input, or note-taking tasks, it may actually hurt recognition for single characters or mathematics.

The fact that its source code is not available to developers, coupled with its lack of support for training of its recognizer, were negatives as far as the goals of this dissertation were concerned.

It is especially difficult to obtain rigorous accuracy rates for commercial systems because companies's motives differ from academic research and these numbers are usually not reported. A third-party evaluation of Microsoft's TabletPC recognizer claimed it achieved very high accuracy rates (90-100% [146]) in spite of not being trainable or adaptable with time<sup>12</sup>.

## 5.2 Baseline Handwriting Recognition Accuracy

The main research questions addressed by the case studies of these three recognizers are as follows:

- What is the best possible writer-independent recognition accuracy achievable with each of the three recognizers tested on the corpus of student algebra handwriting?

---

<sup>11</sup><http://msdn.microsoft.com/en-us/library/aa510941.aspx>

<sup>12</sup>This feature is available in the latest version of Windows for the TabletPC, released some time after these experiments were run. In addition, Microsoft now offers an extension to the Equation Editor for the TabletPC that allows handwritten input called Equation Write. However, it is not customizable by the end-user or an application developer.

- What is the minimum number of handwriting samples needed to achieve this best baseline recognition engine accuracy?
- What are the limitations of each type of handwriting recognizer tested?

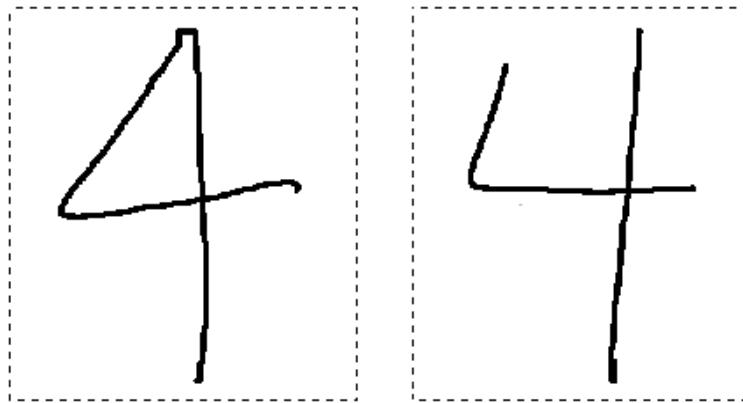
Evaluation of handwriting recognition systems has been limited by user-centered standards. Often there will be fewer than 10 users included, and they are usually required to train the engine to their own handwriting before beginning the evaluation, resulting in a **writer-dependent** test. However, in our application domain: classroom tutoring systems for math, students are not able to spend time training the system explicitly due to classroom time pressures. It is imperative in domains such as algebra learning to deliver a system with the best **writer-independent** accuracy possible, by training on a variety of users in advance. Writer-dependent accuracy tends to be higher than writer-independent accuracy, however. The system can perform on-the-fly adaptation to student writing over the course of a year-long curriculum in order to move toward a writer-dependent model and improve recognition further.

In addition, real-world handwriting recognition, in which users may write symbols just as they would on paper, involves two tasks: character segmentation or stroke grouping (*i.e.*, determining which strokes belong to which characters), and individual character recognition. In the mathematics domain, applications using handwriting recognition must also be able to parse the handwriting input into a structured mathematics representation so that the application can manipulate the input. This means that accuracy measured solely on individual *symbols* (*e.g.*, where the system knows that all strokes it receives are part of one character) may not be indicative of real-world performance, because the additional, challenging step of character segmentation is ignored. Therefore, it is important to consider accuracy measures on streams of characters (*equations*, in mathematics) to yield better accuracy predictions.

In summary, there are three main factors that must be considered when measuring recognition engine accuracy, especially in the domain of entering math expressions: type of usage (domain-specific *vs* domain-general), type of training set (*writer-dependent* *vs* *writer-independent*), and type of testing set (*symbol-level* *vs* *equation-level*). Each of these is described in more detail in the following sections.

### 5.2.1 Domain-Specific *vs* Domain-General Use

Domain-specific recognition is typically easier and more accurate than domain-general recognition, which attempts to recognize any type of symbol in any type of configuration and application. The recognition process can be made domain-specific either via hard-coding domain-specific rules such as vocabulary or grammar during training, or via on-the-fly use of context, such as semantic parsing. This dissertation only explores domain-specific recognition, by using recognizers whose vocabularies have been pruned, and by incorporating context information from the tutoring application as recognition is performed. The use context can provide detailed information to aid in



**Figure 5.1:** An example of two ways of writing the symbol '4', either with one continuous stroke or two separate strokes.

constraining the recognition process from the top down, or in refining the recognition hypothesis once it has already been formed from the bottom up.

### 5.2.2 Writer-Dependent vs Writer-Independent Training

To clarify, *writer-dependent* means the engine is trained solely on samples of handwriting from the same user that will be using the system or whose handwriting will be testing the system. *Writer-independent* means the engine is being trained and tested on distinct users: no users from the testing set appear in the training set, and vice versa.

Writer-dependent accuracy rates tend to be higher than writer-independent rates in most recognition technologies because differences in handwriting or speech vary more widely *across* users than *within* users. A particular user normally has a particular style of writing which does not vary much over time [39]. *Writer-dependent* recognition represents the “ideal” case for the recognizer: data consists of a set of similarly constructed symbols rather than differing styles, for instance, writing a ‘4’ with one continuous stroke *vs* with two separate strokes, as shown in Figure 5.1.

However, in certain domains, such as learning environments in classrooms, it is not feasible for the user (*i.e.*, student) to spend time training the system with no learning objectives. Training handwriting recognition engines to be *writer-dependent* usually involves a large upfront time commitment, called *enrollment*, during which the user inputs many (20 or more) examples of each character the recognizer is to understand, the recognizer’s *vocabulary*. In the vocabulary used in this dissertation, there are 22 characters: that would be over 400 examples for each student to enter! On the other hand, it is difficult to embed the handwriting training task into other, more learning-oriented tasks because the system cannot provide adequate feedback on the learning as-

**Table 5.1:** The set of all math symbols used throughout this dissertation.

0	1	2	3	4
5	6	7	8	9
x	y	a	b	c
+	-	=	(	)
_	/			

pects without good *a priori* recognition accuracy. Therefore, it is important to attempt to improve recognition without upfront training for each particular student. *Writer-independent* training allows a *walk-up-and-use* interaction style.

### 5.2.3 Symbol-Level vs Equation-Level Testing

The second crucial factor in calculating baseline recognition accuracy is the symbol *vs* equation difference, which refers to the way in which the system is tested. The problem of stroke grouping, that is, deciding which strokes belong to each character, is a challenge that real-time recognition applications must address. Testing an engine on single characters one at a time is not a realistic estimate of accuracy in a real-world context because, in a real-world interface, users will be writing their input with their own style and layout. Forcing users to input individual characters into bounding boxes removes the stroke grouping problem, but introduces a very unnatural and constrained interface style. Therefore, it is important to consider accuracy on full equations when judging how well a recognition engine will perform for real users. To perform stroke grouping, the engine is given a set of strokes that make up a full equation, and is iteratively asked to identify subsets of them that make up individual characters and then to identify each character.

### 5.2.4 The Algebra Learner Corpus

In order to conduct these baseline accuracy experiments and test the three recognizers, the hand-writing samples from the Lab Learning Study were extracted from the experiment logs. These samples yielded a corpus of data from over 40 high school and middle school algebra learners copying out equations. The data have been hand-segmented and hand-labeled. They are grouped by equation as originally written by the users in order to allow real-world equation testing; they can also be separated into individual symbols. The corpus contains 16,191 characters grouped into 1,738 equations. The symbol set includes 22 symbols, shown in Table 5.1. Each user wrote on average 404 samples (range: [227, 471]) and 17 samples per symbol (range: [0 ('\_'), 44 ('=')]). There are on average 770 samples per symbol from all users (range: [569 ('9'), 1581 ('=')]).

### 5.2.5 The Evaluation Method

Several ways of selecting the training data for the recognition engine were considered to determine the most effective method of choosing a minimal subset of data that yields the best writer-independent performance. These methods all involve randomly drawing from the entire available pool of training data with varying enforcements over the selection:

- No selection criteria, incrementing total number of *samples*, selected at random from the entire training set;
- Incrementing total number of samples per *user*, selected at random;
- Incrementing total number of samples per *symbol*, selected at random;
- Incrementing total number of samples per *symbol* per *user*, selected at random;
- Using all samples ( $n \approx 400$ ) for each user, but incrementing total *number of users* included, selected at random.

The two factors of samples per *symbol* (in the vocabulary) and samples per *user* are important because they are the two main sources of variance in the training set. Each *symbol* is a unique class to be identified by the recognizer. Ensuring an equal representation of symbols is important in order to provide the recognizer with a balanced model. In beginning algebra equation solving, the most common character may be the equals sign ('='), but otherwise most symbols are evenly represented. Each potential *user* may have a particular style of writing that differs significantly from any other user. In practice, there are not really an infinite number of writing styles, as some agreed-upon standards have been established in service of legibility. Our corpus is sufficiently large to expect to have covered the most common writing styles, which will aid a system trained on this corpus to recognize samples from new users whose writing is similar in style to those styles seen in the training set. Ensuring an equal representation of each possible user's handwriting (or each major style of writing) will prevent the system from having a bias toward a particular style of writing. A writer-independent system has been exposed to many styles and will use these to classify new users' writing. Work on writer-identification has found that writing styles are highly individual [133], but in terms of handwriting recognition algorithms, a degree of generality can be abstracted in order to cluster writing styles [39].

The experiments were conducted using five-fold cross-validation, in which the set of users was divided into five chunks, or *folds*. Each fold was used exactly once for the *testing set*, while the other four folds were used to build the *training set*. Because there were 40 total users, this meant that, in the *writer-independent* tests, the testing set always consisted of eight independent users, and the training set was built from the samples of the remaining 32 users. In the *writer-dependent* tests, a separate test was conducted for *each user*: the testing set consisted of one-fifth of the samples for *one* user and the training set was built from the remaining four-fifths. Each of the experiments is presented in the next section. Both *writer-dependent* and *writer-independent* baseline accuracies of each recognizer are reported, on the algebra learner corpus.

**Table 5.2:** Summary of baseline accuracy results for the three recognizers tested in this dissertation. Numbers in parentheses indicate the number of samples per symbol per user that yielded that accuracy value. Microsoft’s TabletPC recognizer could not be trained to a writer-dependent model at the time of these experiments. Symbol accuracy is the cumulative match score over all symbols tested. Equation accuracy is average score over all equations, computed via normalized Levenshtein string distance.

Recognizer	Writer-Dependent Symbol	Writer-Dependent Equation	Writer-Independent Symbol	Writer-Independent Equation
Freehand Formula Entry System	91%	78%	83% (2)	71% (2)
JMathNotes	65%	44%	66% (10)	47% (10)
Microsoft TabletPC	n/a	n/a	83% (n/a)	54% (n/a)

An important note is that the Microsoft TabletPC API recognizer used for these experiments was pre-Windows Vista and therefore was not trainable using user data. Therefore, only *writer-independent* results are reported for that recognizer; the recognizer was not trained to either the target vocabulary or the target population. As future work in this area, one could re-run the experiments with the latest version of the Microsoft TabletPC recognizer that can now be trained.

In each experiment, accuracy is reported at the point at which accuracy has leveled off and no significant benefit is obtained from continuing to add more data to the training set.

Accuracy was calculated via normalized Levenshtein string distance [80], which is equivalent to the “word error rate” metric used in speech recognition, given by:

$$W = \frac{S + I + D}{N},$$

where  $S$  is the number of substitutions,  $I$  is the number of insertions and  $D$  is the number of deletions; and  $N$  is the length of the string to recognize.

### 5.2.6 Accuracy Results for Each Recognizer

The engine-specific accuracy results, both writer-dependent and writer-independent, are reported here. In general, it was found that FFES was the most highly accurate recognizer tested on the algebra learner corpus, and that the Microsoft TabletPC recognizer was not well-suited for this domain. The summary table, Table 5.2, gives the best accuracy rates in each category of training/testing experiment for each recognizer, while Table 5.3 gives the detailed writer-independent accuracy means for both trainable recognizers, by size of the training set, and Table 5.4 gives the detailed writer-dependent accuracy distribution for both trainable recognizers.

**Table 5.3:** Means tables for baseline accuracy measures reported for the Freehand Formula Entry System and JMathNotes.

(a) Means table of writer-independent accuracy for FFES on both symbols and equations, showing what training iterations were performed.  $N$  is the total number of tests across all folds. (Note that the standard deviations are only for  $N = 5$ , and reflects the standard deviation of the number of folds, not individual tests.)

Samples per symbol per user	$N$	Symbol		Equation		
		Mean	StdDev	$N$	Mean	StdDev
1	16192	81.44	2.91	1738	64.81	5.00
2	16192	83.05	1.95	1738	70.58	3.00
3	16192	82.96	2.16	1738	71.20	3.41
5	16192	82.76	2.46	1738	71.77	3.44
7	16192	82.72	2.55	1738	71.93	3.15
10	16192	82.63	2.47	1738	71.85	3.06
15	16192	82.89	2.07	1738	71.80	3.33
20	16192	82.73	2.08	1738	71.95	3.29
25	16192	82.81	2.09	1738	71.95	3.28
30	16192	82.77	2.11	1738	71.98	3.30
35	16192	82.75	2.08	1738	72.04	3.28

(b) Means table of writer-independent accuracy for JMathNotes on both symbols and equations, showing what training iterations were performed.  $N$  is the total number of tests across all folds. (Note that the standard deviations are only for  $N = 5$ , and reflects the standard deviation of the number of folds, not individual tests. Note that JMathNotes tests were run with only 30 users instead of all 40.)

Samples per symbol per user	$N$	Symbol		Equation		
		Mean	StdDev	$N$	Mean	StdDev
1	12221	57.09	4.76	1311	38.04	4.21
2	12221	62.15	3.09	1311	40.75	4.27
3	12221	63.23	2.78	1311	41.41	3.52
5	12221	64.55	2.25	1311	43.27	3.43
7	12221	65.26	2.20	1311	44.85	3.76
10	12221	65.81	2.08	1311	46.97	4.38
15	12221	66.32	2.07	1311	48.75	4.12
20	12221	66.89	1.80	1311	49.24	3.96
25	12221	66.93	1.76	1311	49.66	3.90
30	12221	67.04	1.78	1311	50.30	4.02
35	12221	67.11	1.85	1311	50.70	4.03

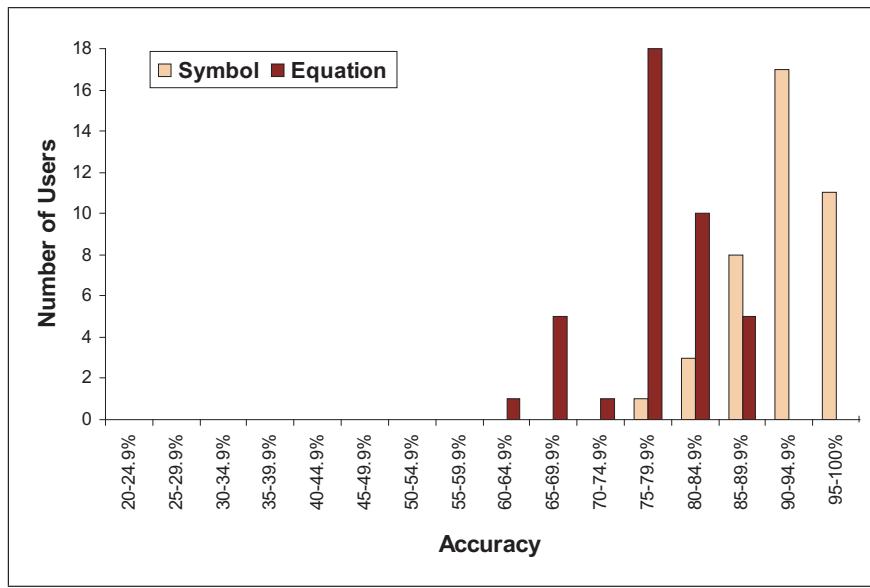
**Table 5.4:** Histogram table of all writer-dependent results for both the Freehand Formula Entry System and JMathNotes.

Accuracy Bucket	FFES		JMathNotes	
	Symbol	Equation	Symbol	Equation
20-24.9%				2
25-29.9%				1
30-34.9%				2
35-39.9%				5
40-44.9%				11
45-49.9%				9
50-54.9%			1	7
55-59.9%			5	3
60-64.9%		1	12	
65-69.9%		5	13	
70-74.9%		1	8	
75-79.9%	1	18	1	
80-84.9%	3	10		
85-89.9%	8	5		
90-94.9%	17			
95-100%	11			

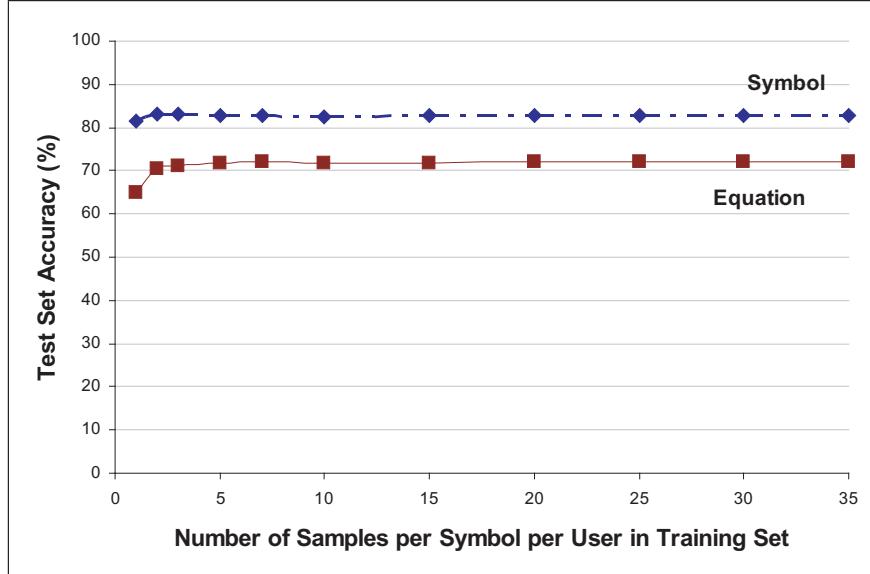
### Freehand Formula Entry System

The original FFES estimates of 95% (writer-dependent) and 77% (writer-independent) was not obtained from a large sample of users; this dissertation attempts to replicate these original FFES accuracy results via more systematic testing, using the algebra learner corpus. Two writer-dependent experiments were conducted (one for symbols and one for equations) for each of the 40 users in which the training set was four-fifths of that user's data and the remaining one-fifth was used as a test set. The five-fold cross-validated accuracy is above 90%, but not as good as the 95% quoted for FFES in [131] (*mean*: 91.476%, *stdev*: 4.63; *median*: 92.85%). Figure 5.2(a) shows a histogram of the distribution over all 40 users of accuracy on the test set for symbols and equations, which generally follows a normal distribution; actual values are given in Table 5.4. Accuracy on equations is much lower than on symbols, on the other hand. The average accuracy for equations is 78% (*mean*: 77.95%, *stdev*: 6.30; *median*: 78.76%) per user, an 14% decrease in accuracy from the symbol case, and an 18% decrease in accuracy from the previously reported statistics on FFES [131].

Writer-independent experiments were also conducted, in which the number of samples per *symbol* per *user* included in the training set was varied, testing on both single symbols and equations.



(a) Writer-dependent histogram of FFES' accuracy on symbols and equations. Overall symbol accuracy average is 91.48%; median is 92.85%. Equation accuracy average is 77.05%; median is 78.76%.



(b) Writer-independent FFES' accuracy on both symbols and equations when samples per symbol per user is varied in the training set from 1 to 35. Accuracy at the point at which adding further training samples does not significantly improve accuracy is 71.3% for equations, which is an approximate 9% decrease in accuracy from the writer-dependent equation tests.

**Figure 5.2:** Baseline accuracy results for the Freehand Formula Entry System.

Results of these tests are graphed in Figure 5.2(b). The recognition accuracy for single symbols levels off (*i.e.*, there is no significant benefit to accuracy of adding further training samples) around 83% after training on only two samples per symbol per user, a 9% decrease in accuracy from the writer-dependent symbol case. For equation-level testing, the performance of FFES levels off around 71% at the same point, a 14% decrease in accuracy from individual symbols and a 9% decrease in accuracy from the writer-dependent equation tests.

### JMathNotes

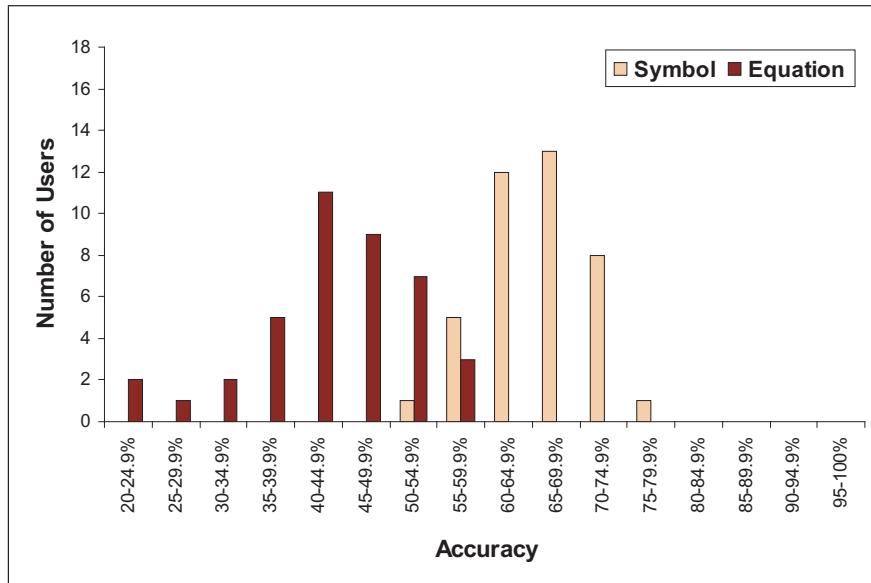
The same five-fold cross-validated data experiments on the algebra learner corpus were conducted with the JMathNotes recognizer, with one exception: only 30 users were included rather than all 40, due to the much greater time demands of training and testing this SVM-based recognizer. For each iteration, one SVM model was trained for all characters on varying subsets of the training data and tested on the test set for that fold.

In general, JMathNotes did not perform as well as FFES. The five-fold cross-validated *test* writer-dependent accuracy on symbols is 65% (*mean*: 65.46%, *stdev*: 5.71; *median*: 65.53%), about 29% lower than the performance of FFES. Figure 5.3(a) shows a histogram of the distribution, over all 40 users, of accuracy on the test set for symbols and equations; actual values are given in Table 5.4. It again follows a somewhat normal distribution, but is more spread out than that of FFES. The equation accuracy is much lower than for symbols. The average accuracy over all users for equations is 44% (*mean*: 44.11%, *stdev*: 8.60; *median*: 42.42%), which is a 32% decrease in accuracy from JMathNotes' own performance on symbols, and a 44% decrease in accuracy from FFES on equations.

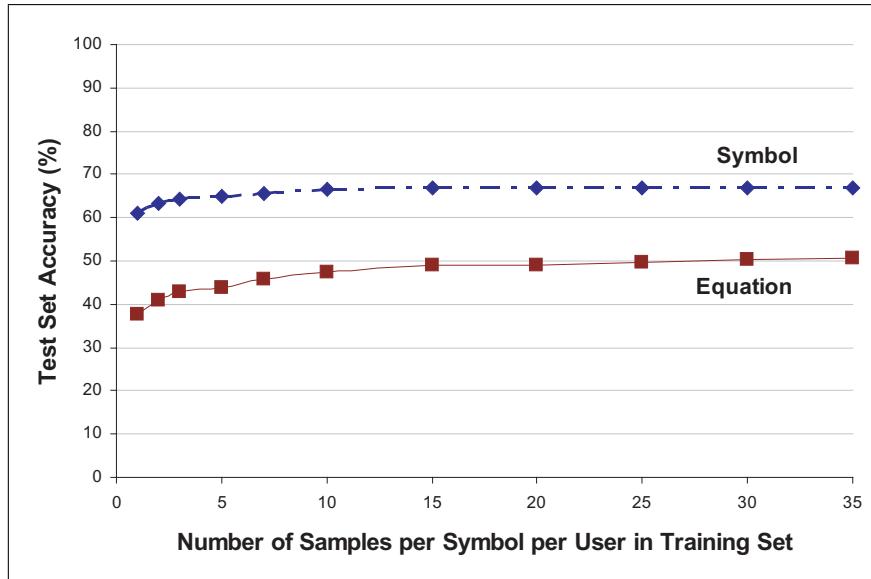
The same writer-independent experiments were conducted with the JMathNotes engine as with FFES, varying the number of samples per *symbol* per *user* included in the training set. The results for both equation and symbol are graphed in Figure 5.3(b). The recognition accuracy for single symbols levels off (*i.e.*, there is no significant benefit to accuracy of adding further training samples) at about 66% after training on ten samples per symbol per user, a 2% increase in accuracy from JMathNotes' own writer-dependent performance on symbols and a 15% decrease in accuracy from FFES' writer-independent accuracy on symbols (with only two samples per symbol per user). For equations, it is 47% at the same point, a 7% increase in accuracy from JMathNotes' writer-dependent equation performance and a 34% decrease in accuracy from FFES' writer-independent accuracy on equations.

### Microsoft TabletPC Recognizer

It was not possible to perform exactly the same set of experiments on Microsoft's TabletPC recognizer because this recognizer was not trainable. Therefore, only writer-independent accuracy estimates on the algebra learner corpus were computed. Two factors were varied: using or not using a "word list" to constrain the recognition results to contain only symbols within the symbol



(a) Writer-dependent histogram of JMathNotes' accuracy on both symbols and equations. Overall symbol accuracy average is 66.46%; median is 65.53%. Equation accuracy average is 44.11%; median is 42.42%. JMathNotes performed about 29% worse than FFES on symbols.



(b) Writer-independent JMathNotes' accuracy on both symbols and equations when samples per symbol per user is varied in the training set from 1 to 35. Accuracy at the point at which adding further training samples does not significantly improve accuracy is 47.0% for equations. JMathNotes performed about 34% worse than FFES on equations.

**Figure 5.3:** Baseline accuracy results for JMathNotes.

**Table 5.5:** Writer-independent recognition results for the Microsoft TabletPC recognizer.

Test Set Type	Accuracy	
	Symbol	Equation
No Word List	67.42%	54.20%
Word List	83.16%	36.42%

set; and testing symbols one at a time or full equations. The optimal results achieved through these experiments are shown in Table 5.5.

These results show that the real-world performance of an engine that has not been trained to the specific domain of use can be far lower than quoted rates of accuracy. This engine has achieved rates as high as 90-100% [146] on note-taking and memo-writing tasks, but the underlying word-level neural network structure of the recognizer apparently works quite poorly in the math domain. Later versions of the recognition engine in Windows for the TabletPC include the ability to be trained on user-defined data, which may help the rates improve for certain domain-specific tasks such as entering or solving math equations and expressions. Also, the use of a word list in these experiments did not precisely follow the intended use of such a list by the recognizer developers; it appears that using the word list makes the system’s stroke grouping algorithm far too greedy, clumping multiple symbols into one “word.” Because the “word list” used in these experiments was actually the vocabulary of symbols, it was ineffective on equations: strokes for multiple symbols would be grouped together greedily into one symbol. With the facilities available in the TabletPC recognizer at the time of these experiments, it was not possible to achieve highly accurate math recognition for full equations.

Writer-independent accuracy on symbols for Microsoft’s TabletPC recognizer was 83%, about the same as FFES performance on the same, and 26% better than JMathNotes. Performance on equations for Microsoft was 54%, a 24% decrease in accuracy from FFES, and a 15% increase over JMathNotes.

### 5.2.7 Summary of Results and Discussion

Table 5.6 shows a summary of the writer-independent results for all three recognizers discussed in this dissertation: the Freehand Formula Entry System, JMathNotes, and Microsoft’s TabletPC recognizer.

In the single symbol case, both FFES and Microsoft are competitive at 83% accuracy. It is in the equation case that the biggest difference emerges: FFES outperforms the others by almost 25%. The equation case allows users to input full equations without artificially placing bounding boxes in which users have to enter individual symbols. This is a more natural way of entering handwritten input because the users do not have to alter characteristics of their handwriting such as spacing, connectedness or order of strokes to use the system. Therefore, performance on this

**Table 5.6:** Summary of writer-independent recognition results for the three case studies.

Recognizer	Accuracy	
	Symbol	Equation
Freehand Formula Entry System	83.1%	71.3%
JMathNotes	65.8%	47.0%
Microsoft TabletPC	83.2%	54.2%

type of input should be emphasized in evaluations of handwriting recognizers.

Note that the equation-level accuracy is not what one would expect if processing of individual characters in the equation were independent. In fact, by virtue of the stroke-grouping process involved in recognizing full equations, the individual symbols are not independent. If the individual characters could be treated independently, there would be no difference between the character recognition rates in symbol and in equation modes. One could estimate the overall equation accuracy by taking the symbol accuracy to the  $n$ th power, where  $n$  is the length of the equation. For a three-character equation, it would be  $0.83 * 0.83 * 0.83$ , or 57%. Given that the average length of the equations in the baseline recognition tests was nine characters, the results indicate that equation level accuracy is better than independence would predict. Because the individual symbols in a full equation cannot be treated independently, the accuracy for equations is computed differently from the accuracy for symbols. Symbol accuracy is simply about substitution—given target  $a$ , did the recognizer return  $a$  or some other symbol? On the other hand, Levenshtein string distance was used for equations in order to account for issues in stroke grouping, which allows the recognizers to add extraneous characters or remove others. Given target  $a$ , the recognizer could return  $a$ ,  $b$ ,  $bc$ , etc. Therefore, the equation accuracy does not indicate that the system got 71% of equations perfectly correct (that is quite rare, in fact). Rather, the average normalized Levenshtein string distance over all equations is 71%. Hence, the equation accuracy does not represent what one would obtain by raising the symbol level accuracy to the  $n$ th power, and stroke grouping errors are the reason.

Some limitations of the experiments performed in this dissertation do exist. One is that newer versions of the Microsoft TabletPC recognizer are trainable, which might allow its accuracy to increase significantly for the equation case. Domain-specific training can be crucial to maximizing recognition accuracy within a specific application. Another limitation is that JMathNotes was trained using only one DAG-SVM for all characters in the training set. It might improve accuracy to train separate SVMs, for example, of characters with varying numbers of strokes, which would yield a more complete picture for comparing the different recognizers. Another possibility for future work would be to classify users' styles of writing more finely than just number of strokes per symbol and to ensure an equal representation of different writing styles.

The results of these experiments have implications for application developers who are intending to use handwriting input and adapt existing recognition engines. The recognizers were trained by ensuring an equal representation of samples per *symbol* per *user* in the training set in order to yield

the best baseline accuracy with the least amount of samples. If available, domain-specific data yields the best performance by pruning the symbol set and allowing the use of context to aid the recognition engine. Research-level and state-of-the-art recognition systems currently require some level of technical expertise to incorporate them into an application, but as more are adapted for real systems, APIs and toolkits should follow.

### 5.3 General Conclusions

Three handwriting recognition engines were reviewed for this dissertation, but ultimately only one was used as the base recognition system for the remainder of this work. FFES performed the best out of the three on the algebra learner corpus collected in the Lab Learning Study, particularly in the case of testing on full equations. The results of these experiments showed that with very little training data per person—just two samples per symbol per person—very good writer-independent accuracy can be achieved with FFES. In the future, other recognition engines may be found or built that perform even better in general or on this corpus in this domain, and as such, the methodology used in this chapter can help guide the training and adaptation of such an engine to this context.

# Chapter 6

## Improving Recognition: Domain-Specific Context Information

The second main approach taken by this dissertation to enhance handwriting recognition accuracy is to incorporate domain-specific context information into the recognition process, thereby constraining the recognition hypotheses to contextually relevant ones and yielding higher accuracies. Work on recognition accuracy tends to hold domain-general recognizers that minimize user training, are multilingual, and are completely unconstrained as the “Holy Grail” [79, 86]; however, domain-specific recognition can be just as valuable. Because it is typically more accurate in its particular context by virtue of its smaller vocabulary sizes and specific grammar rules [86, 108], it can be used in cases where domain-general recognition may not yet be suitably accurate.

In the tutoring domain, much context is available for free. For instance, the tutoring program assigns the problems for the student to solve, so it knows what the student *should* be writing (the correct answer). From the model of student knowledge, it knows the probability that this student will get the correct answer, *i.e.*, based on prior opportunities to practice the same skill(s) applicable to the current step. From years of learning science research into *difficulty factors assessment* [136], the system knows the most common misconceptions (known in learning science literature as “bugs”) that students might write. These pieces of context can be used to constrain the recognition hypothesis space before any recognition is performed (*i.e.*, reducing the number of symbols under consideration), or to further refine an initial recognition hypothesis after the recognition was conducted normally.

### 6.1 Adding Domain-Specific Information

After the baseline **writer-independent** training described in Chapter 5, the FFES recognizer was at a reasonably high level of accuracy even before adding any domain-specific context information (73% writer-independent accuracy on full equations). Therefore, the tutor context information was used to further refine the recognition hypothesis once the strokes had already been fed into

the stroke grouper and character recognizer, rather than as a top-down constraint on what the recognizer would consider as potential hypotheses.

During this process, it was much more complex to obtain the probabilities from the student model or the common errors from the tutoring system than originally supposed. This is a limitation of the tutor code rather than a challenge as a research topic, so for sake of demonstration of the use of contextual information, only the set of all possible *correct* next steps is used (obtained at each subsequent problem-solving step). This information is easily available from the tutor system, and could be the most helpful piece of information, depending on the typical error rate in the types of algebra equation-solving lessons used in this dissertation.

Several ways to incorporate this context information into the recognition process were considered, such as modeling the relationship between the recognizer confidences and the tutor information via Bayesian networks. The method actually implemented was inspired by the field of collaborative information retrieval and metasearch, which uses *ranking fusion* techniques to combine the results of multiple searches [122]. The recognition hypothesis is expressed in terms of an ***n*-best list** per symbol, which can also be treated as a **rank-list**: elements with higher confidence values are ranked more highly than elements with lower confidence. This simplifies the problem in the sense that it can be difficult to convert confidences to probabilities for use in a Bayesian network, depending on how these confidences are computed. In FFES, confidences are the inverse Euclidean distance from the candidate to the nearest training example. While transformations such as squaring the distance or taking the inverse log normal could be applied in order to convert these distances into probabilities, for simplicity's sake these transformations were not done here.

On the other hand, it is quite simple to convert the tutor probabilities into rank-lists. An example of the information given by the tutor for a sample problem-solving step is given in Figure 6.1, showing four possible correct next steps for the problem  $2x + 10 = 30$ . Each possible correct next step is treated as equally likely (*i.e.*, there is no bias toward any particular solution path). This is a useful simplifying assumption; however, student preferences could be learned over time in a comprehensive curriculum. The probability of student error, which could be estimated from the tutor's knowledge model, is not used here. Because rank-lists are used rather than the raw *n*-best lists, the probabilities and confidences are abstracted, and the error probability does not matter; only the order of elements in the lists matters. The elements that would constitute errors (*i.e.*, symbols not in any correct option) can simply be given rank  $n + 1$ , where  $n$  is the size of the set of possible correct steps.

### 6.1.1 Working with the Tutor Information

As mentioned, the tutor returns a list of the possible correct steps in equation (or expression) form that the student could be writing, indicating which operation(s) are possible and what the outcome would be. However, the recognizer is only recognizing one character at a time. The recognizer, upon receiving a set of strokes to recognize, iteratively groups them into stroke groups corresponding to the most likely characters using a confidence measure. The tutor information

$$\left\{ \begin{array}{l} -10 \text{ from both sides} \\ 2x = 20 \\ /2 \text{ on both sides} \\ x + 5 = 15 \end{array} \right.$$

**Figure 6.1:** The type of information received from the tutor: the set of possible correct options for the current step (the step following  $2x + 10 = 30$ ).

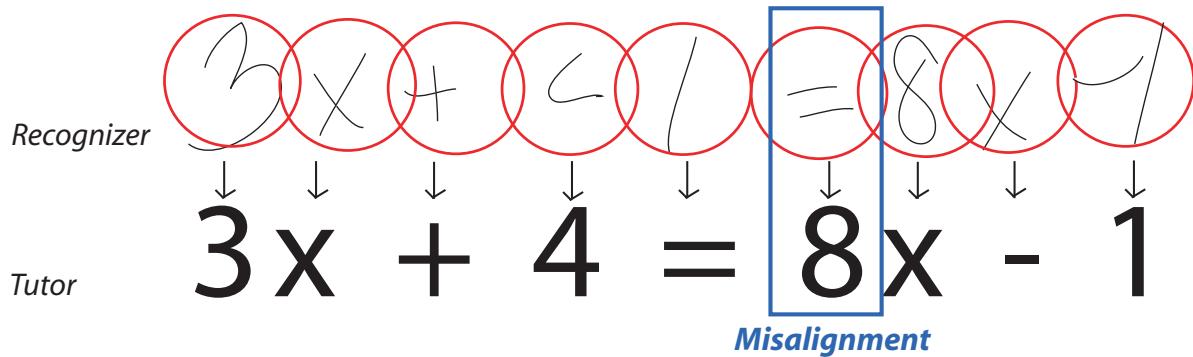
should not be applied in sequence one character at a time because the recognizer may have made errors in the stroke grouping process, for instance, breaking one multistroke symbol into two, or aggregating two symbols into one. This problem, known as the **alignment problem**, is also encountered in multimodal systems with two or more streams of co-occurring or synchronous input (e.g., [46]). The presence of stroke grouping errors means that it will not be useful for the tutor to simply provide a rank-list for the  $n$ th character in the target. The tutor could be looking at the wrong symbol if the stroke grouping algorithm has failed, causing the tutor information to be of no benefit, and potentially to be of significant harm. Figure 6.2 shows how aligning the two streams (the recognizer character-level recognition and the tutor equations) can fail given errors in stroke grouping. In this example, the '4' has been split into its component strokes by the recognition process because the recognizer's confidence levels indicated that a 'c' and '1' were more likely than a '4'. In addition, the '-1' strokes have been grouped into one symbol because the recognizer's confidence level indicated that a '7' was more likely. In trying to integrate the tutor information with this recognition stream, the streams are misaligned: the recognizer's sixth character is the '=', for example, while the tutor's is the '8'. This stroke grouping mistake makes the tutor unable to effectively benefit the recognition process by increasing the confidence that the sixth character is actually an '8'.

Therefore, another technique from information retrieval and natural language processing is used, called "bag of words" (e.g., [27]), in this case, "bag of symbols." Each set of correct steps is jumbled together as a set of symbols, unordered, and treated as equally likely. Symbols that appear more than once in the set of symbols are treated as more likely than symbols that appear once, by summing the probabilities normalized by length of the equation or expression in which each symbol appears. Figure 6.3 illustrates this process. The bag-of-words method abstracts away some information from the tutor-supplied information that could be useful, but the alignment problem is more potentially damaging to system performance because the tutor information cannot be of any use at all in the presence of the alignment issue. As future work, in § 8.3 ways are described to adapt the approach used here to alleviate the impact of this compromise, but which are out of scope for this dissertation.

*Input as written by student:*

$$3x + 4 = 8x - 1$$

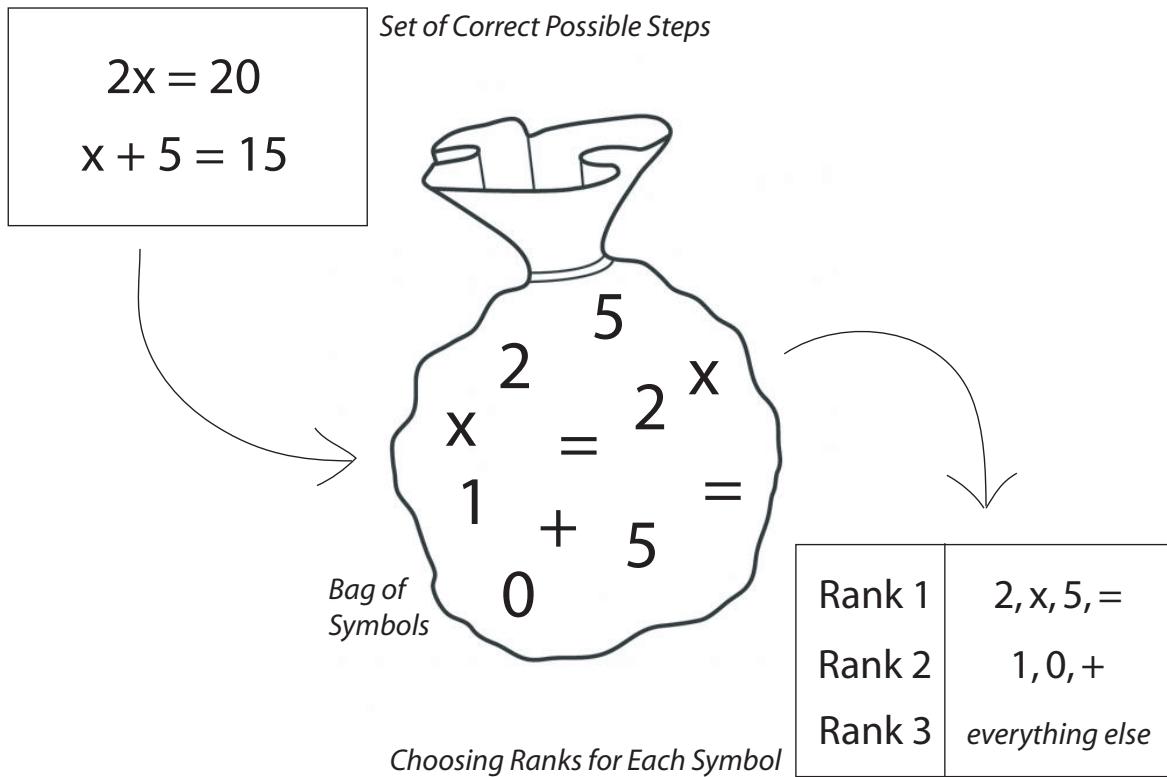
*Stroke grouping errors make aligning symbols from recognition result and tutor information difficult even with correct input:*



**Figure 6.2:** The alignment problem prevents comparing the tutor information to the recognizer information directly. Because the recognizer can make errors in stroke grouping, the tutor does not know which symbol (in order from right to left) to consider. In this example, stroke grouping errors have occurred on the '4', which has been split into two groups, and on the '-1' symbols, which have been combined into one group. When the recognizer asks the tutor for the sixth character, the tutor returns '8' but the recognizer is already looking at the '='. In this case, the tutor information would not help, and may in fact harm, recognition.

### 6.1.2 Average-Rank Sort

After putting the tutor-provided information into the bag-of-symbols format, both input streams are in equivalent and comparable forms: lists of symbols sorted by “likeliness”—in the recognizer case, this likeliness is represented as confidences; in the tutor case, this likeliness is represented as probabilities. The next step is to combine the two input streams. Confidences and probabilities are not directly comparable; therefore, the elements from both lists are converted into simple ranks. For example, the symbol with the highest confidence or probability is assigned rank 1, the next rank 2, and so forth. If two symbols are tied, they receive the same rank. Then, to combine the two rank-lists, a weighted *average-rank sort* algorithm is used, similar to a weighted Borda count [22], which effectively sums the rank position of documents from different results lists in search tasks.



**Figure 6.3:** Converting the tutor information, provided as a set of possible correct equations, into a “bag of words” rank-list involves the three steps shown above. The symbols in each of the equations are jumbled together and re-sorted by frequency. Symbols with the same frequency are assigned the same rank, and symbols in the vocabulary but not in any of the possible equations are assigned one more than the maximum rank.

Figure 6.4 outlines the steps involved in combining the two lists via average-rank sort. First, as described, convert the two lists into rank-lists, abstracting the raw probability or confidence information. Then, for each symbol in the vocabulary, take the rank of the symbol in the tutor list and average it with its rank in the recognition list (a straight or weighted average may be used). Then, sort the final list by this average rank; in case of ties, the recognizer rank is trusted more (*i.e.*, decide ties in the direction of the recognizer’s rank for that symbol), because the tutor information is so limited in scope. This does not require either list to output a hypothesis for every symbol: if the symbol does not appear in one or both lists, assign it rank  $n + 1$  for each list, where  $n$  is the size of the list.

As mentioned, the estimated student error probability (set here as 0.50) does not affect the recognition process because the values are abstracted away by the conversion to rank-lists. If an alternative means of combining the two lists were used, the error probability would have to be

$$\times (5+3)=24$$

The student is on the second step of the given problem ( $5x + 3x = 24$ ). This is what she writes.

### Recognition Process

$$1. \quad \textcircled{X}(5+3)=24$$

1. Stroke grouping selects first stroke as most probable character.

$$2. \quad \{ ('y': 0.997), ('+': 0.982), ('x': 0.976), ('4': 0.911), ('1': 0.790), ('7': 0.678), ('0': 0.608), ('2': 0.603), ('-': 0.597), ('=': 0.553), ('l': 0.542), ('a': 0.531), ('5': 0.519), ('9': 0.513), ('6': 0.508), ('b': 0.486), ('3': 0.484), ('c': 0.482), ('_': 0.453), ('0': 0.389), ('8': 0.378), ('/': 0.185) \}$$

2. The  $n$ -best list from the recognizer for the first stroke ('y' is the most likely candidate).

$$3. \quad \{ ('y': 1), ('+': 2), ('x': 3), ('4': 4), ('1': 5), ('7': 6), ('l': 7), ('2': 8), ('-': 9), ('=': 10), ('0': 11), ('a': 12), ('5': 13), ('9': 14), ('6': 15), ('b': 16), ('3': 17), ('c': 18), ('_': 19), ('0': 20), ('8': 21), ('/': 22) \}$$

3. The  $n$ -best list is transformed from using confidences to simple ranks for use in combination algorithm.

### Tutor Input Process

$$1. \quad 5x + 3x = 24$$

1. The tutor keeps the current problem for reference.

$$2. \quad \{ 'x(5+3)=24': 0.167, '8x = 24': 0.167, '1(5x+3x)=24': 0.167, \text{error}: 0.50 \}$$

2. The tutor knows the set of correct next steps given the current problem. The probabilities of the correct options are assigned as equal. Anything else the student inputs will be an error, so any symbol not in the correct set will be at end of the list.

$$3. \quad \{ ('x': 0.167*1/9+0.167*1/5+0.167*2/11), ('l': 0.167*1/9+0.167*1/11), ('5': 0.167*1/9+0.167*1/11), ('+': 0.167*1/9+0.167*1/11), ('3': 0.167*1/9+0.167*1/11), ('0': 0.167*1/9+0.167*1/11), ... \}$$

3. The equation list is transformed into a "bag of symbols" based on the distribution of symbols in the set of possibilities.

$$4. \quad \{ ('x': 0.08), ('-': 0.07), ('2': 0.07), ('4': 0.07), ('l': 0.03), ('5': 0.03), ('+': 0.03), ... \}$$

4. Now each symbol is associated with a probability.

$$5. \quad \{ ('x': 1), ('-': 2), ('2': 2), ('4': 2), ('l': 5), ('5': 5), ('+': 5), ('3': 5), ('0': 5), ('8': 10), ('1': 11), ('y': 12), ('7': 12), ('=': 12), ('a': 12), ('9': 12), ('6': 12), ('b': 12), ... \}$$

5. The sorted "bag of symbols" list is transformed into simple ranks. Any symbol not in the tutor list is added to the end with the highest rank.

### Combination Process

#### Recognizer

$$1. \quad \{ ('y': 1), ('+': 2), ('x': 3), ('4': 4), ('1': 5), ('7': 6), ... \}$$

#### Tutor

$$\{ ('x': 1), ('-': 2), ('2': 2), ('4': 2), ('l': 5), ('5': 5), ... \}$$

average

$$\{ ('y': 6.5), ('+': 3.5), ('x': 2), ('4': 3), ('1': 8), ('7': 9), ('0': 6), ('2': 5), ('-': 5.5), ('=': 11), ('l': 8), ('a': 12), ('5': 9), ('9': 13), ('6': 13.5), ... \}$$

sort

$$\{ ('x': 2), ('4': 3), ('+': 3.5), ('2': 5), ('-': 5.5), ('0': 6), ('y': 6.5), ('1': 8), ('l': 8), ('7': 9), ('5': 9), ('=': 11), ('3': 11), ('a': 12), ('9': 13), ('6': 13.5), ... \}$$

1. The two lists (from the tutor and from the recognizer) are combined via the average rank sort technique. A symbol's location in the final list is determined by the average of its rank in each list. The weights used on each rank-list were determined experimentally.

$$2. \quad \{ 'x', '4', '+', '2', '-', 'l', 'y', '1', '(', '7', '5', '=', '3', 'a', '9', '6', 'b', 'c', '_', '8', '0', '/' \}$$

2. The final list is output, showing 'x' as the most likely candidate.



The algorithm is repeated for every new stroke grouping in the input until all strokes have been recognized.

**Figure 6.4:** The step-by-step algorithm which takes the recognition results and the tutor information and combines them to better interpret students' written input.

empirically verified. A general error rate for all algebra equation-solving units, just one unit, or just one problem type could be obtained from the PSLC LearnLab's DataShop<sup>1</sup> estimates from learning logs from past courses in the curriculum, and hard-coded into the tutoring system where needed. In addition, detailed error probabilities for this particular student could be obtained from the tutor's model of student knowledge in future development of this concept.

In summary, the procedure for adding the domain-specific context information to the recognition process to yield a combined hypothesis of what was written by the student is as follows:

1. Identify the current step the student is working on.
2. Determine all possible correct options for this step, by computing the set of next possible operations and results of those operations based on the prior step.
3. Given the prior step, what step(s) do these operations yield?
4. Convert the list of possible steps into a “bag of symbols.”
5. Sort the “bag of symbols” by symbol frequency and convert to ranks.
6. Use a weighted average to sort the tutor rank-list and the recognizer rank-list (converted from the  $n$ -best list with confidences) into one list via average-rank sort.

## 6.2 Context-Enhanced Recognition Accuracy

In order to evaluate the improvement in recognition accuracy as a result of this combination technique, a test suite of six sample problems was constructed. Each problem was included in the test suite twice: once with a conceptual error leading to one or more incorrect steps, and once fully correct, for each user in the corpus (the handwriting corpus used is the algebra learner corpus, the same as that used in the baseline recognition accuracy experiments from Chapter 5 and collected during the Lab Learning Study). Ten iterations of each problem solution, correct and incorrect, were conducted per user to account for noise in recognition accuracy. With 40 users in the test corpus, this yields  $6 * 2 * 10 * 40$ , or 4800, total problems tested. Problems had between three to five steps, and each step had between three to nine characters. § 6.2.3 describes how the test problems were chosen and Appendix E lists both the correct and incorrect versions of the problems used.

All tests were writer-independent, meaning that the recognition component was trained on a different set of users than the users being tested. In addition, five-fold cross-validation was used, so that each student was part of the test set once and part of the training set four times. The entire set of students was broken up into five subsets (folds) by using a random number generator to select the eight students for each fold. All of the data for a particular student was assigned to the same fold. All training data was used for each fold.

---

<sup>1</sup><https://learnlab.web.cmu.edu/datashop/>

It was important for comparison's sake that the accuracies obtained in these experiments had to be comparable to the baseline accuracy experiments from Chapter 5. In addition, a segmented and labeled corpus of problem-solving data was not available. Therefore, these experiments used the algebra learner corpus from Chapter 5, which was not a problem-solving corpus. Each problem-solving step from the test suite of problems was artificially constructed for each user by choosing random samples of each particular symbol for that user from the corpus, and translating them on the virtual canvas to fit problem templates. Equations are lined up along a center line and fractions are centered vertically over and below the fraction bar. These translated strokes were then fed to the recognizer, which performed stroke grouping and character recognition as usual until all strokes on the canvas had been recognized. The frequency of stroke grouping errors was not significantly impacted by this template-based alignment, as compared to the experiments from Chapter 5.<sup>2</sup>

Each problem was tested one step at a time. After the recognition had been done for each stroke grouping (several times per step), the tutor was asked for all possible correct steps at this stage of the problem-solving process, to further refine the recognizer's  $n$ -best list for that character, as described in § 6.1.1. The step the tutor believes the student is on was determined on-the-fly, by matching the recognition result (of tutor-plus-recognizer) to the set of correct options for that step, and assuming the step that was written is the option with the least distance from the recognition result. Using the tutor information, in terms of the correct next steps, to track student progress through the problem artificially constrains the tutor to expect that the student is on the correct solution path. However, this constraint is reasonable because the error rate for these types of algebra units is low, making correct solution paths much more frequent (elaborated more in § 6.2.4). In addition, if the step written were assigned directly to the recognition result of the system, other problems could arise such as parsing errors where the recognition result is not legal math. The tutor uses this hypothesis about what step was written to determine the set of next correct steps, so it must be a legal, parseable expression or equation.

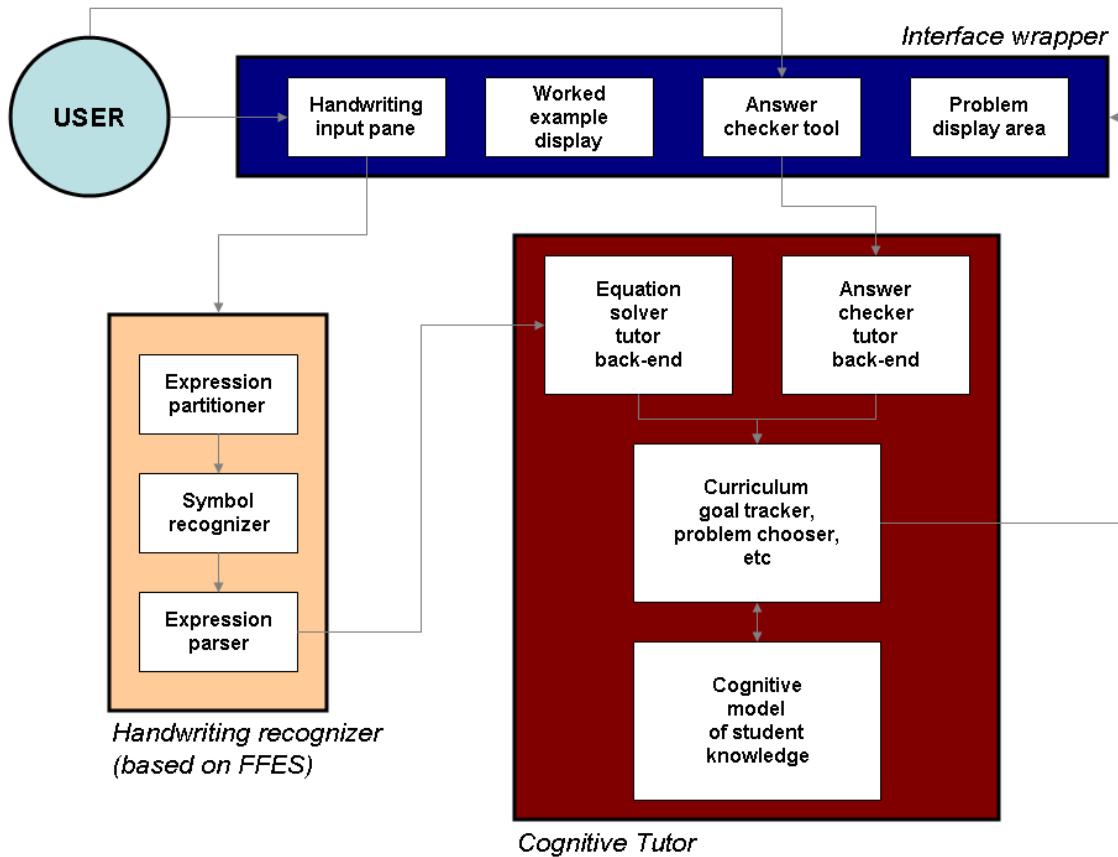
### 6.2.1 Tutor Testbed

A testbed platform was used throughout these experiments to obtain the appropriate information from the tutor side. In this proof-of-concept approach, the tutor was working in the background only (*i.e.*, a fully functional prototype was not built). It was trivially extended to allow user input for demonstration purposes, but it was considerably more involved to update the tutor model of student knowledge based on the system's interpretation of the student's input (necessary to allow the tutor to follow student progress and assign new problems). Therefore, the system cannot be used as it is now with students.

The testbed system integrates the tutoring system, implemented in Java, with the recognition system, implemented in C and C++. It uses a socket connection to the recognizer process in order to send stroke coordinates to the recognizer and receive the recognition results. The testbed

---

<sup>2</sup>This was roughly verified by comparing the average difference in length between the target and the recognition result from the baseline experiment log files for FFES to that of the current experiments.



**Figure 6.5:** System architecture diagram of the prototype tutoring system. For the experiments reported in this chapter, the “Interface wrapper” component is replaced with a batch testing program that reads handwritten strokes from corpus files and feeds them into the recognizer.

system can be run as a test suite, reading the strokes to recognize from files, or as a proof-of-concept for demonstration purposes that allows user input of the strokes and provides real-time recognition. The general system architecture is shown in Figure 6.5<sup>3</sup>. The tutor-provided context information is obtained in the testbed from a “Tutor” object running in the background, which can receive a problem or equation to solve and then be queried as to the set of next possible correct steps. In a real, deployed system, this Tutor object would not just be in the background, but the components of the testbed that handle the recognition calls and combination with the tutor context information would be incorporated into the back-end of the complete tutoring system, allowing real-time handwriting input in the context of problem-solving that is linked to the tutor’s

<sup>3</sup>This diagram has been updated from the one published in [10].

knowledge-tracing model.

For the purposes of running these recognition accuracy tests, the test suite was hard-coded into the program so that the recognizer would be run on the same problems for every user in the testing set. The testbed wrote detailed logfiles of recognition results of both the recognizer alone and the recognizer-plus-tutor-information system.

### 6.2.2 Iterative Algorithm Tuning

The average-rank sort allows different weights to be placed on both the recognition and the tutor information sides, *i.e.*, a weighted average as given by the following equation:

$$K_{avg} = W_T * K_T + W_R * K_R,$$

where  $K_R$  is the recognition rank,  $K_T$  is the tutor rank,  $W_R$  is the weight on the recognition side, and  $W_T$  is the weight on the tutor side. With a weighted average, the weights should always be normalized to sum to 1. For instance, a weight of 0.50 applied to each rank would correspond to weighting the two sources evenly.

Different parameter values were tested for these weights, in order to find the best possible accuracies obtainable with this method. The set of weights ( $W_T, W_R$ ) tested was (0.0, 1.0), (0.1, 0.9), (0.2, 0.8), (0.3, 0.7), (0.4, 0.6), (0.5, 0.5), (0.6, 0.4), (0.7, 0.3), (0.8, 0.2), (0.9, 0.1), (1.0, 0.0). The results of these tests to find the best weights are given in § 6.2.4. As future work, in § 8.3 other possible ways of combining the lists, such as using Bayesian networks, are discussed. More robust methods may yield further improvements upon those methods described here.

### 6.2.3 Choosing the Test Problems

The test set used in these experiments consists of six representative cases of problem-solving processes, with errors, from the learning phase of the Lab Learning Study. These cases were chosen by reviewing the screen captures of each problem-solving solution from the study (including both ones with errors and ones without). Out of 500 total problems for 28 students, 73 were not solved correctly on the first attempt (14% problem error rate). Each errorful problem example was roughly coded at a high level of abstraction with the type of error the student made. Table 6.1 lists the types of errors encountered in the corpus; the five most common were chosen for the errors to artificially introduce in the test suite.

A specific representative example of a student making each of the highest-frequency error types from Table 6.1 was chosen and included in the test cases. The types were “arithmetic error,” “negative sign dropped,” “example mirroring/using a number not in the problem,” “using a wrong number from the problem,” and “performing a different operation on each side.” The example-mirroring case and using-a-number-not-in-the-problem case can be combined for testing purposes because they amount to the same thing (the system sees numbers it does not expect); however, in a deployed system, the numbers from the example can be known to the tutor and can be used to

**Table 6.1:** Types of conceptual or other problem-solving errors represented in the corpus of problem-solving examples from the Lab Learning Study’s learning phase, grouped by similarity. The frequency column sums to more than 73 because some problems contained examples of multiple error types.

Error Type	Frequency	Description
Arithmetic error	24	The student makes a simple arithmetic error, such as indicating that $7 * 6$ equals 56.
Transcription error	5	The student copies the problem incorrectly or incorrectly copies a number from one step to another.
Example mirroring or copying	12	The student writes numbers in the problem that are mirrored from the example provided, rather than from the problem to be solved.
Negative sign dropped	12	The student divides by a negative number and does not include the negative sign in the result.
Performing different operation on each side	5	The student, for example, subtracts a number from one side but divides on the other side.
Using different numbers on each side	2	The student applies the same operation to both sides but uses different operands on each side.
Operated on terms rather than sides	2	The student applies an operation to two terms on the same side, ignoring the equals sign and the other side of the equation.
Using a wrong number from the problem	6	The student applies an operation using some number from the problem other than the correct one.
Using a number not in the problem or example	6	The student applies an operation using a number not in the problem at all (and not in the example, <i>i.e.</i> , not example mirroring).
Reciprocal confusion	2	The student multiplies by the numerator rather than the denominator to remove a fraction.
Not showing work	37	The student simply writes the final answer.
Incomplete solution	25	The student submits a partial solution without an $x = X$ line.
Expressed answer via “plug and chug”	12	The student writes the original equation with the value of $x$ in the place of the variable, for example, writing $5 * 3 + 3 * 3 = 24$ as the solution to $5x + 3x = 24$ .
Off-task writing	18	The student scribbles or writes messages unrelated to the problem-solving process in the input space.

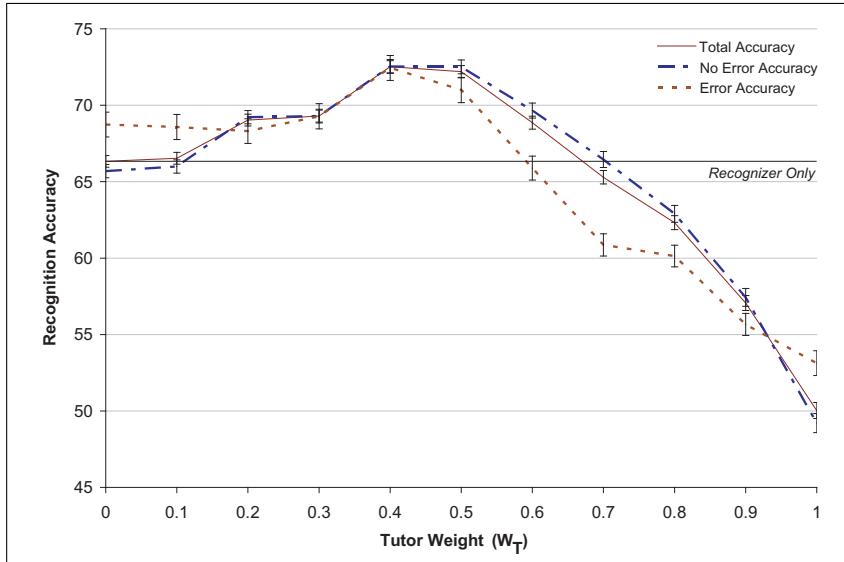
help identify this specific type of error. The test cases included one example problem with each of the five chosen errors, as well as the correct version of that problem. One more problem was also included to help the test suite problems' average deviation from correct match that of the corpus at large; this problem was an example of both the “negative sign dropped” and the “using a wrong number from the problem” error types. See § 6.2.4 for a further discussion of computing this deviation.

In a problem solution in which an error has been made, the error will propagate through the rest of the solution. Even if the student correctly applies operators to the remaining steps to achieve an answer, the steps he writes will not be part of the correct set of possibilities, since the tutor is only looking at the correct solution space. Therefore, a step is counted as an error step for analysis in any case where the input does not match any of the candidates in the set of correct options. In all, the test set had 11 steps that were error steps, out of 24 total steps. Half the problems had errors on them, for a 50% problem error rate. This error rate is much higher than the real-world corpus problem error rate of 14% mentioned above, in order to ensure that the proper cases were covered. That means the raw accuracy achieved could actually be higher in real-world datasets, since the combined system typically does better on steps without errors than steps with errors due to the tutor bias in favor of correct solutions.

#### 6.2.4 Evaluation Results

Two measures of success are reported here in order to evaluate or validate the approach taken by this work. One is the best raw accuracy achieved with this approach. The other is more application-specific: how good is this approach at identifying the step on which the error occurred?

Part of the original motivation to explore the use of handwriting recognition in the tutoring domain, in spite of imperfect recognition technology, was the hypothesis that aspects of the domain would allow interaction patterns that do not require the system to output the recognition result of each piece of input back to the student immediately, or possibly at all. Note that many recognition systems perform a type of “beautification,” in which the user’s writing is transformed into a machine font or otherwise standardized; this probably is initially surprising to the user, but the user eventually will become acclimated to it. It is more than beautification that is referred to here, but the additional burden of correcting the system and focusing on input and display rather than mathematics. In fact, this burden may impose its own type of **cognitive load**, whether or not recognition is highly accurate [116]. When the student becomes aware of the system working, this awareness may interrupt his cognitive flow and remove him from the context of problem-solving. For this reason, it may be enough to simply provide feedback of the type: “Your final answer is incorrect. It looks like you made an error on this step [highlight step in interface]. Why don’t you go back and check that step?” This feedback structures the interaction in the form of a tutoring intervention as a way to keep the student engaged in the problem-solving process, rather than exposing its true purpose, checking its interpretation of the student input. In order to identify on which step the problem occurred, the system must compare the distance from the set of correct possibilities with



**Figure 6.6:** The raw accuracy of the combined system at varying weights on the tutor and recognizer during the average-rank sorting process. The best improvement over the recognizer alone was seen at ( $W_T = 0.40$ ,  $W_R = 0.60$ ) pair during the average-ranking process, with an accuracy of 72.5%. Performance of the recognizer alone ( $W_T = 0.00$ ,  $W_R = 1.00$ ) is 66.3%.

the student’s input (after recognition) and determine which step has the highest deviation from the appropriate correct set.

Accuracy was determined in the same way as in the experiments presented in Chapter 5: using normalized Levenshtein string distance [80], counting the number of substitutions, insertions and deletions and then normalizing by string length. Means on all metrics are given in Tables 6.2 and 6.3, and are described further in the following sections.

### Raw Recognition Accuracy

The systematic tests revealed a relationship between the weight provided to the tutor and the accuracy obtainable. Figure 6.6 graphs the raw performance of the combined recognition system as a function of different ( $W_T$ ,  $W_R$ ) pairs; raw means are given in Table 6.2(a). The overall accuracy is shown, as well as the accuracy on steps with errors and steps without errors. Due to the context information’s bias toward correct answers, the combined system does better on steps without errors. The steps with errors do not bring the overall accuracy down as much as they might because of the distribution of steps with errors to steps without: there are many more steps *without* errors, which mirrors real-world datasets.

Weighting the tutor slightly less than the recognizer ( $W_T = 0.40$ ,  $W_R = 0.60$ ) appears to achieve the best recognition accuracy without as much penalty for the steps with errors. The mean

**Table 6.2:** Means tables for accuracy and other performance metrics reported in this chapter.(a) Means table of raw accuracy by  $(W_T, W_R)$  pairs.

$W_T$	$W_R$	$N$	Total Accuracy		Accuracy			Steps Without Errors		
			Mean	StdDev	$N$	Mean	StdDev	$N$	Mean	StdDev
0.0	1.0	18720	66.33	26.74	14820	65.69	26.94	3900	68.74	25.82
0.1	0.9	18720	66.53	26.79	14820	65.99	26.94	3900	68.58	26.10
0.2	0.8	18720	69.03	26.74	14820	69.22	26.99	3900	68.31	25.76
0.3	0.7	18720	69.29	27.44	14820	69.29	27.75	3900	69.28	26.24
0.4	0.6	18720	72.52	27.65	14820	72.54	28.06	3900	72.44	26.05
0.5	0.5	18720	71.20	27.83	14820	72.51	28.20	3900	71.00	26.33
0.6	0.4	18720	68.86	29.36	14820	69.65	30.35	3900	65.89	25.04
0.7	0.3	18720	65.29	30.72	14820	66.45	32.32	3900	60.86	23.17
0.8	0.2	18720	62.32	32.15	14820	62.90	34.21	3900	60.14	22.57
0.9	0.1	18720	57.06	34.19	14820	57.43	36.58	3900	55.66	22.92
1.0	0.0	18720	50.03	36.41	14820	49.21	38.69	3900	53.13	25.79

(b) Table of true-positive-rate and false-positive-rate for various threshold values by  $(W_T, W_R)$  pairs. The threshold indicates the level of deviation from correct used to determine whether a step is an error. Threshold values of 0.0 result in TPR and FPR of 100%; threshold values of 1.0 result in TPR and FPR of 0.0%. Bolded cells indicate “points of interest.”  $N$  for FPR rows is 5460; for TPR rows, 3900. Standard deviation is not included because this is the proportion across all folds/data.

$W_T$	$W_R$	<b>Rate</b>	Threshold Value								
			0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
0.0	1.0	FPR	77.9%	69.0%	57.7%	41.5%	<b>20.1%</b>	16.1%	8.7%	4.1%	3.5%
		TPR	99.8%	98.8%	90.3%	77.1%	<b>63.5%</b>	56.8%	50.9%	35.1%	34.3%
0.1	0.9	FPR	77.5%	67.8%	56.6%	41.4%	<b>20.5%</b>	16.3%	<b>8.6%</b>	4.2%	3.5%
		TPR	99.7%	98.7%	89.8%	76.6%	<b>64.1%</b>	56.9%	<b>51.9%</b>	35.9%	35.4%
0.2	0.8	FPR	71.5%	61.6%	50.3%	35.7%	17.0%	14.2%	7.2%	2.8%	2.4%
		TPR	99.5%	96.3%	87.8%	72.2%	59.1%	52.8%	47.7%	31.7%	31.4%
0.3	0.8	FPR	70.9%	60.9%	48.7%	37.1%	<b>14.0%</b>	10.6%	<b>6.5%</b>	2.2%	1.7%
		TPR	98.8%	94.6%	83.9%	66.1%	<b>57.9%</b>	52.3%	<b>48.6%</b>	34.1%	33.9%
0.4	0.7	FPR	65.1%	54.3%	46.2%	34.2%	12.5%	9.8%	6.4%	1.4%	1.0%
		TPR	96.9%	91.6%	80.2%	60.4%	43.5%	38.7%	35.9%	29.6%	29.1%
0.5	0.6	FPR	65.8%	55.4%	46.1%	34.2%	12.8%	10.1%	7.1%	1.6%	0.9%
		TPR	96.7%	91.0%	79.0%	59.1%	42.3%	38.2%	35.8%	29.6%	29.2%
0.6	0.5	FPR	65.4%	56.8%	50.8%	40.5%	17.4%	12.3%	7.9%	2.2%	0.6%
		TPR	93.0%	87.6%	73.8%	55.7%	43.9%	39.9%	37.8%	29.7%	29.3%
0.7	0.4	FPR	66.5%	58.9%	53.0%	45.3%	24.3%	20.8%	12.8%	2.8%	<b>0.3%</b>
		TPR	88.4%	82.8%	69.8%	58.0%	<b>53.6%</b>	50.0%	47.6%	37.9%	<b>37.3%</b>
0.8	0.3	FPR	70.7%	64.0%	57.1%	50.2%	29.4%	25.6%	17.4%	2.2%	0.3%
		TPR	86.2%	81.5%	70.8%	63.5%	60.8%	<b>57.4%</b>	53.7%	40.0%	37.3%
0.9	0.2	FPR	74.5%	69.4%	63.6%	59.0%	36.0%	31.0%	24.7%	3.8%	0.2%
		TPR	82.2%	79.4%	70.1%	62.8%	60.5%	58.0%	52.8%	34.1%	30.0%
1.0	0.1	FPR	80.5%	76.0%	73.1%	68.8%	49.0%	<b>45.7%</b>	43.8%	8.3%	0.0%
		TPR	82.2%	79.2%	69.6%	62.9%	<b>59.9%</b>	58.7%	57.3%	33.3%	29.8%

**Table 6.3:** Performance on the error identification task. Standard deviations are not included because these are raw proportions across all folds/data.

(a) Proportion of problems on which system correctly identifies error step by  $(W_T, W_R)$  pairs.

$W_T$	$W_R$	$N$	<b>ID First Error</b>	<b>ID Any Error</b>
0.0	1.0	390	0.426	0.585
0.1	0.9	390	0.426	0.580
0.2	0.8	390	0.432	0.619
0.3	0.7	390	0.371	0.558
0.4	0.6	390	0.424	0.597
0.5	0.5	390	0.401	0.580
0.6	0.4	390	0.390	0.529
0.7	0.3	390	0.396	0.505
0.8	0.2	390	0.456	0.567
0.9	0.1	390	0.465	0.598
1.0	0.0	390	0.320	0.403

(b) Proportion of problems on which system correctly identifies error step by problem type by  $(W_T, W_R)$  pairs.

$W_T$	$W_R$	$N$	$\frac{36}{x} = 4$	$5x + 3x = 24$	$8x = 16$	$\frac{x}{5} + 4 = 7$	$-x = 5$	$-x = 8$
0.0	1.0	390	0.103	0.664	0.436	0.292	0.221	0.841
0.1	0.9	390	0.077	0.692	0.459	0.297	0.203	0.826
0.2	0.8	390	0.069	0.774	0.510	0.192	0.215	0.833
0.3	0.7	390	0.056	0.744	0.441	0.182	0.041	0.762
0.4	0.6	390	0.159	0.531	0.849	0.172	0.044	0.790
0.5	0.5	390	0.126	0.492	0.831	0.167	0.026	0.767
0.6	0.4	390	0.287	0.210	0.885	0.182	0.056	0.721
0.7	0.3	390	0.377	0.172	0.844	0.228	0.056	0.697
0.8	0.2	390	0.467	0.336	0.867	0.195	0.018	0.856
0.9	0.1	390	0.518	0.244	0.879	0.274	0.000	0.874
1.0	0.0	390	0.454	0.000	0.856	0.613	0.000	0.000

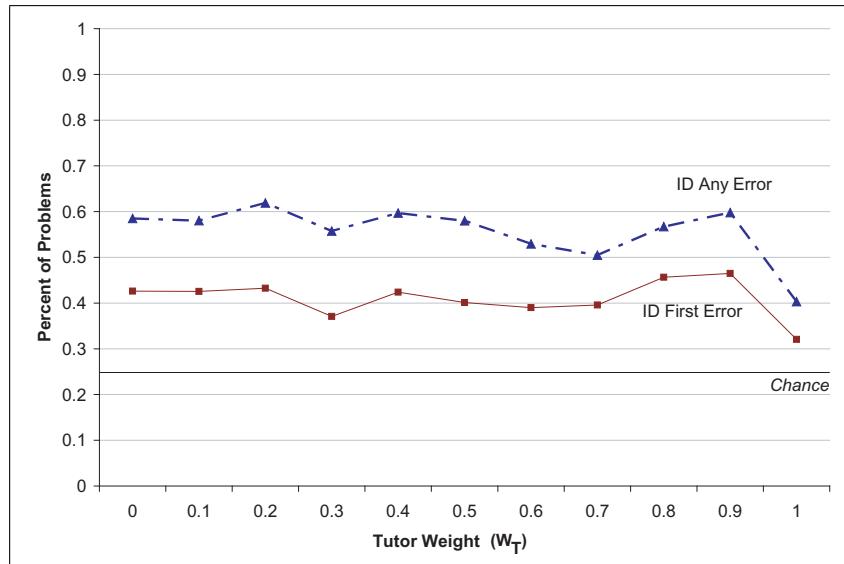
improvement at the peak weighting function is 72.5% ( $stdev: 27.7\%$ ), which is a 9% improvement over the performance of the recognizer alone on the same test set (66.3% recognizer-only accuracy on average). By a paired-samples t-test, the mean accuracy of the two samples (recognizer-only vs recognizer-plus-tutor) were significantly different ( $t(18719) = 28.0, p < 0.05$ ): the addition of context significantly improved the recognizer’s performance. These accuracy numbers are lower than those seen in the experiments presented in Chapter 5 (71.3% recognizer-alone equation accuracy), most likely due to the characteristics of this test set. The average length of the equations and expressions sent to the recognizer in these experiments was shorter, and although the accuracy is normalized by equation length, shorter expressions tend to have higher error rates. Furthermore, the recognizer in this set of experiments was trained on *all* the data from the proper fold, which may have added too much noise to the recognition process (rather than the minimal approach of two samples per symbol per user taken in Chapter 5).

Note that 72.5% accuracy is the normalized character recognition rate in the presence of the stroke grouping problem. It does not mean that the recognizer gets 72.5% of the equations perfectly correct. Rather, the proportion of equations that the combination recognizer gets perfectly correct at ( $W_T = 0.40, W_R = 0.60$ ) is 39.2%. This number can be used to model the amount of recognition noise interfering with the process of identifying whether a difference in the recognition result and the set of correct options is due to recognition error or student error.

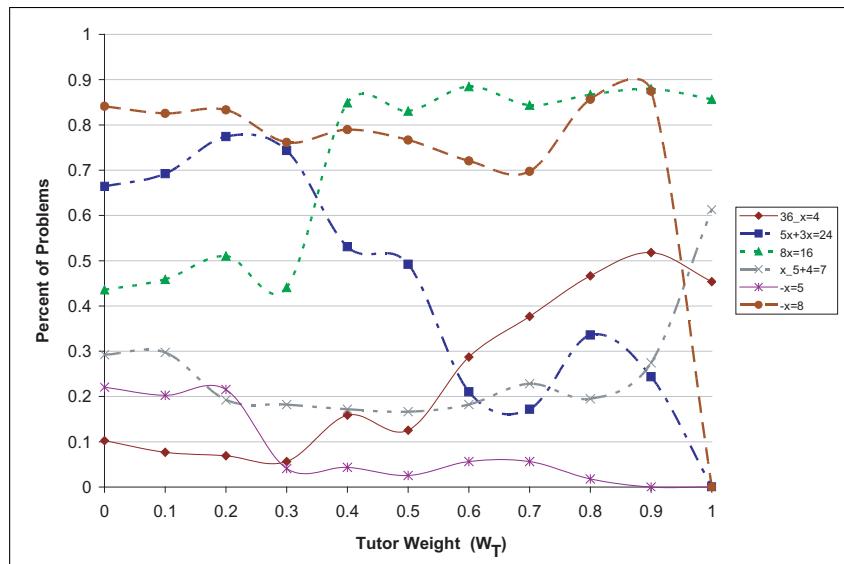
### Student Error Identification

Figure 6.7(a) graphs the performance of the system on a real-world validation metric of identifying where the error occurs in a problem-solving process; raw means are given in Table 6.3(a). For instance, this might be incorporated into a tutoring system as an improved modification of **answer-level feedback**. Once the student types in an incorrect final answer<sup>4</sup>, the system knows that the student made an error somewhere in the problem, and can go back to each step and measure the distance between the set of correct possibilities and what the recognizer interpreted. Figure 6.7(a) shows the performance of the algorithm at identifying the error step using the deviation of the recognizer results from the correct set. The two lines correspond to two possible “success” metrics of the error identification task. When a student makes an error in a problem, the subsequent steps she performs continue to deviate away from the set of correct options. So in essence, all subsequent steps are “error steps.” However, pedagogically, it is probably only effective to identify the *first* step on which an error was made. The graph shows that performance on this task is quite low: the best performance achieved on identifying the first error (“ID First Error”) was at ( $W_T = 0.90, W_R = 0.10$ ) and was about 46%. This is better than chance since the problems have between three to five steps: chance at identifying the error step correctly on the first try is 25%. The performance on the task of identifying any error (“ID Any Error”) was better, but not by much: the best was at ( $W_T = 0.20, W_R = 0.80$ ) and was about 60%. Identifying “any” error is easier because the

<sup>4</sup>Following the instructional paradigm used throughout this dissertation, typing in the final answer is done in order to remove ambiguity and because typing this step is simple.

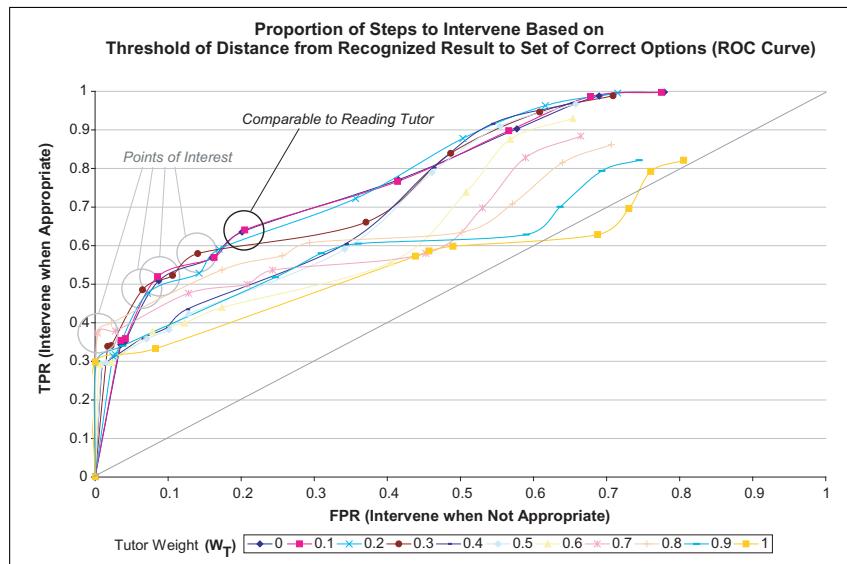


(a) The performance of the combined system at determining the erroneous step of a complete problem-solving solution. Chance is 25%.



(b) Detail by problem of the performance of the combined system at determining the erroneous step of a complete problem-solving solution.

**Figure 6.7:** Performance of the system on the error identification problem.



**Figure 6.8:** The ROC curve showing effect of applying different thresholds on identifying the error. Each line is a different ( $W_T$ ,  $W_R$ ) pair.

answers deviate more and more from correct as the student solution continues, so the system can eventually discover the student is on an error path, but not necessarily the point at which the student diverged from the correct path. This implies that the ability of the system to correctly identify the error the first time is not very good, and this algorithm needs improvement. Figure 6.7(a) does not show any real relationship between tutor weight and performance on this metric, but Figure 6.7(b) shows that performance differs dramatically when separated by problem; raw means are given in Table 6.3(b). Unfortunately, with only six test problems, it is not possible to draw any real conclusions about factors contributing to these differences. This would be a very interesting area for future work, however.

An alternative to choosing the step with the highest deviation from correct out of all the steps in a problem (and only provide delayed **step-targeted feedback**) is to treat each step individually and choose a threshold at which to intervene. If this strategy performed well, a tutoring system could provide immediate step-targeted feedback. By a one-way ANOVA, there is a significant difference in the deviation from correct on error steps *vs* non-error steps when the system knows the student made an error ( $F(1, 9358) = 631.8, p < 0.05$ ); error steps tend to deviate more from correct than non-error steps do. Means for the ( $W_T = 0.90, W_R = 0.10$ ) pair are shown in Table 6.4. This significance means it may be possible to rely on the difference to mark a threshold value.

Figure 6.8 is an **ROC** (receiver-operating-characteristics) curve of the error-detecting power of varying threshold values; an ROC curve graphs the ratio of false-positives to true-positives. When the recognition result for a step is above a certain threshold, indicating high deviation from the

**Table 6.4:** Means table for deviation from correct of the recognition result on all error steps *vs* on all non-error steps.  $N$  is the number of steps.

Error Step	Mean Deviation	N	StdErr
No	41.3%	5460	0.396
Yes	58.6%	3900	0.596

set of correct options, the system would identify this step as a student error and begin tutoring. A separate line is plotted for each  $(W_T, W_R)$  pair to determine how the behavior of the thresholds change.

Unfortunately, no extremely strong candidate threshold emerges as identifying a large proportion of the true-positives without also falsely identifying a large proportion of the false-positives. Four “points of interest” are highlighted on the graph. These are (from left to right) the 0.9 threshold value of the  $(W_T = 0.7, W_R = 0.3)$  pair ( $TPR: 37\%$ ,  $FPR: 0.3\%$ ), the 0.7 threshold value of the  $(W_T = 0.3, W_R = 0.7)$  pair ( $TPR: 49\%$ ,  $FPR: 7\%$ ), the 0.7 threshold value of the  $(W_T = 0.1, W_R = 0.9)$  pair ( $TPR: 52\%$ ,  $FPR: 9\%$ ), and the 0.5 threshold value of the  $(W_T = 0.3, W_R = 0.7)$  pair ( $TPR: 58\%$ ,  $FPR: 14\%$ ). The false-positive rate indicates the proportion of the time that the system would *incorrectly* identify a step as an error and intervene with the student when it is not needed;  $(1 - FPR)$  is the proportion of the time that the system would *not* identify a correct step as an error and therefore, properly not intervene. The true-positive rate indicates the proportion of the time that the system would identify an error step as an error and correctly intervene with the student;  $(1 - TPR)$  indicates the proportion of the time that the system would *fail* to identify the step as an error, and allow the student to move on with no intervention.

For instance, if we use the 0.7 threshold of the  $(W_T = 0.3, W_R = 0.7)$  pair with values  $TPR: 49\%$  and  $FPR: 7\%$ , then a little over 50% of the time a student error would go unidentified, but only 7% of the time would the tutor intervene unnecessarily. Compare this performance to that of the Reading Tutor from Project LISTEN<sup>5</sup>: overall accuracy in recent experiments was reported as 80% ( $TPR: 77\%$ ,  $FPR: 20\%$ ) [150], meaning that only 23% of the time would a student error go unidentified and 20% of the time the tutor would intervene unnecessarily. Intervening when it should *not* is arguably the more serious consequence in tutoring tasks [1], from both a usability perspective of user frustration and a pedagogical perspective of “undoing” good learning. To that end, the system presented here can achieve comparable performance with the 0.5 threshold of the  $(W_T = 0.3, W_R = 0.7)$  pair, with values  $TPR: 64\%$  and  $FPR: 20\%$  (missing a tutoring opportunity 36% of the time), also indicated on Figure 6.8; and with the 0.5 threshold of the  $(W_T = 0.0, W_R = 1.0)$  pair (recognizer only), with the same TPR and FPR values. The fact that this performance is comparable to the Reading Tutor, which has been shown to be a very effective learning aid [96], is a strong statement in favor of this threshold technique for error identification. However, due

---

<sup>5</sup><http://www.cs.cmu.edu/~listen/>

to the recognizer-only performance being competitive, it seems that the tutor information's bias toward correct prevents the system from being able to do as well as it might on this task. Further exploration is needed to enhance this system's performance on this metric.

### What is the Impact on the Student?

With either error identification algorithm, the student will have to perform extra, unnecessary steps as many times as it takes the system to find the step on which the error occurred. The system may not properly identify the error step the first time, so an important extension of the results of the previous section is to determine how often the system will have to unnecessarily intervene with the student.

The expected total number of unnecessary steps a student will have to do *on error problems* is given by the following equation:

$$\sum_{n=0}^{\infty} ne^{(n+1)},$$

where  $n$  is the number of unnecessary steps and  $e$  is the error rate in identifying the proper step where the student error is, given by one minus the success rate (*e.g.*, 0.50 or  $\frac{1}{2}$ ). For the value 0.50 of  $e$ , the sum converges to 1.0, meaning that with a 50% success rate at identifying which step the student error is on, the student will have to do an average of one extra step per problem, in other words, correcting the system once per problem *on which an error occurs*. Recall that the student will not have to do any extra steps on problems without errors, because he would have typed in the final answer correctly and been allowed to move on without the system needing to perform *any* recognition.

In the corpus of student problem-solving solutions from the Lab Learning Study, there were 73 problems out of 500 that the students did not solve correctly on the first try, which corresponds to a 14% problem error rate in the corpus. Therefore, the overall expected number of extra steps is the product of this rate and the expected number of extra steps on error problems. In this case it would be  $\frac{1}{7}$ , or, one out of every seven, problems that would require correcting, on average. In practice, some problems would require multiple unnecessary corrections by the student and some would require none, but the average would approach one out of seven problems.

For the maximum success rate of 46% seen in the results reported above for error identification, the sum (with  $e = 0.54$ ) converges to 1.378, meaning that *more than* once per problem *on which there was an error* students would have to enter an unnecessary step. Multiplying by the 14% problem error rate, this would yield approximately 0.193, or fewer than one out of five problems would require unnecessary extra steps overall.

Recall the time benefit shown for handwriting in the Lab Learning Study, in which students were twice as fast in handwriting as in typing for the same problems. Having to correct the system on one out of five problems (on average) would cut into that time benefit, at least by 20%. Even assuming conservatively that the added overhead of correcting costs the students twice as much

time, one can expect to cut into the time benefit only by 40%, meaning the students in handwriting would still be over 60% faster than students in typing for the same problems.

This formula is an important contribution in that it establishes the relationship between recognition accuracy and tutoring behavior, as manifested in terms of requiring extra steps. Given any goal tutoring behavior, this formula can be used to derive the required recognition accuracy. Furthermore, when better recognizers are developed for this domain, the formula can be used to predict performance on the error identification metric as has been shown here.

### 6.3 Limitations

The results reported in this chapter regarding performance of FFES when it is given domain-specific context information from the tutoring system are subject to some limitations. One such limitation is that these data are only as good as the original problems selected in the Lab Learning Study. They were representative of the types of algebra problems students at a specific level would encounter, and were also chosen to highlight some concepts which were known to present challenges to students, such as negative numbers. So the types of errors observed in the Lab Learning Study during the learning phase, while representative, are by no means exhaustive.

Both of the error identification algorithms described here (one with delayed feedback and one with immediate) did not perform as well as hoped on this task. Both rely on the deviation of the recognized result from the set of correct options as the flag to determine whether something is an error or not. However, given that the recognizer makes errors of its own, the deviation seen between the set of correct options and the recognition result may be the result of the recognition error rather than the student error. Because the recognition process is noisy, the recognition result cannot be trusted. Intuitively, the student's errors must deviate even *further* from the correct set than the amount of deviation caused by recognition noise in order to reliably identify them using this algorithm. The larger the average magnitude of the student's error from the set of correct options, the more confident the system can be; but in addition, the larger the standard deviation of either source of error, the less confident the system can be. The standard deviation of the raw recognition is almost 28% (73% accuracy, or 27% error), and the standard deviation of the (artificially introduced) student error is 14% (normalized by length, 36% on average; approximately 1.55 characters different). These numbers imply that the recognizer would have to be improved, either in terms of raw accuracy or reducing its variance, in order to be able to confidently determine on which step the error occurred on the first try.

One problem found with the current approach is the fact that the equals sign ('=') is *always* recognized as two minuses ('-'), given the way stroke grouping plus recognition is performed in the FFES recognizer (it disregards time information). This is easily correctable via a simple heuristic to replace all instances of '--' with '='. In fact, the DRACULAE parser [149] always corrects this error due to the added information of the spatial layout of the two '-' (one above the other). However, this correction cannot be made until both '-' have been recognized. Because

the tutor combination is done after each stroke is recognized, the tutor list must treat '=' as two '−', meaning that '−' becomes the most likely character to be seen in each equation. As the tutor weight increased, the output of the combined system became more and more heavily populated with '=' signs. Therefore, in these experiments, '=' was translated into *one* '−', to avoid this bias. Another possible way to reduce the impact that this limitation of FFES has on the combination algorithm might be to delay the point at which the tutor combination is performed, allowing the tutor list to work with one '=' rather than two '−'. It is possible that a recognizer that utilizes time information to perform stroke grouping may do better in general, and certainly on this particular problem.

## 6.4 General Conclusions

The experiments presented in this chapter focused on using domain-specific context information provided by the tutor to enhance recognition performance, both in terms of raw accuracy and in terms of identifying the location of student errors. The tutor information significantly reduced recognition error, by a factor of 9%, which is a significant improvement in terms of handwriting recognition research. However, on the task of identifying the location of student errors, the performance of the system using the error identification algorithm based on the deviation of the recognized result from the set of correct options did not perform as well as hoped. Nonetheless, the efficiency cost of the extra, unnecessary interventions the system might perform is not that high due to the low overall error rate students exhibit during equation-solving lessons. While further technical improvements are needed and should be explored, the results of these experiments indicate that there is great promise in the ability of a tutoring system that allows handwriting input to effectively provide feedback to students.

# Chapter 7

## Interaction Case Studies

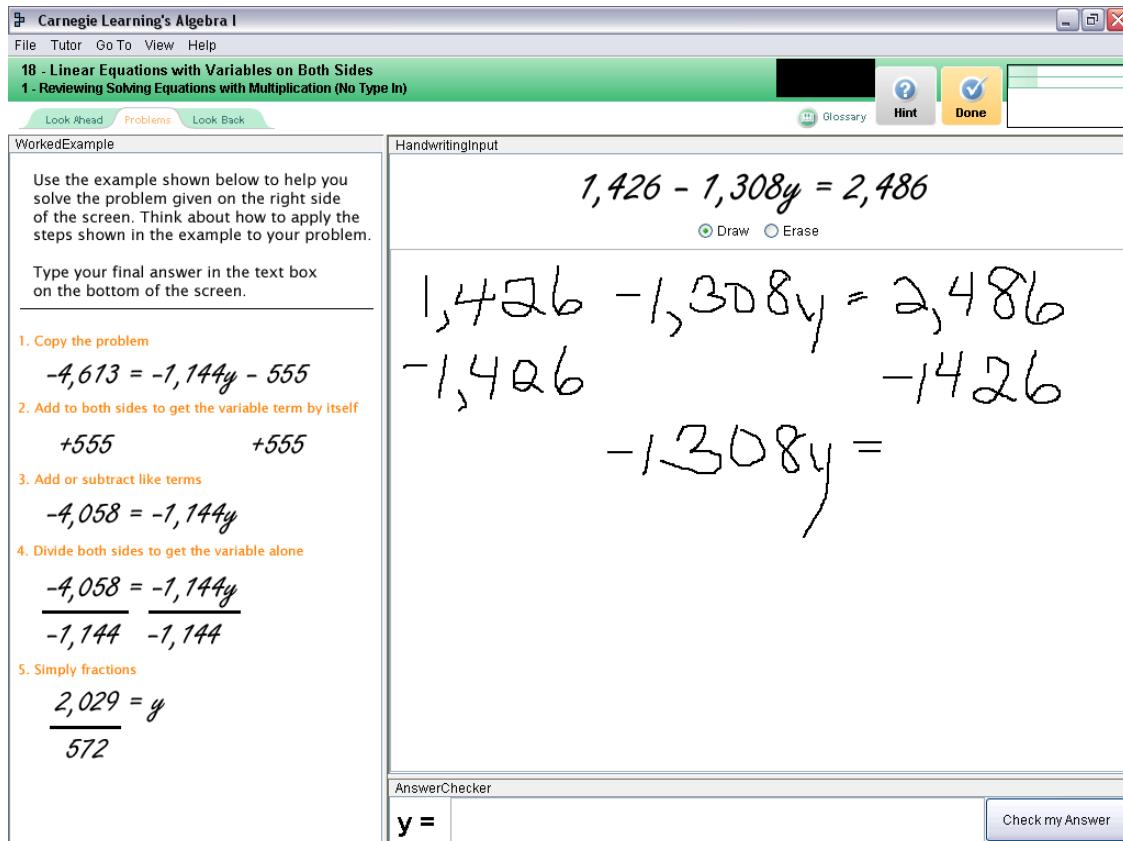
Based on all the results presented in this dissertation, it has been shown that handwriting input provides benefits for math learning, as well as that recognition can be improved through the use of even simple domain-specific context. These results point to several design recommendations for ways to structure a pedagogical interaction in a tutoring system using the recognition system with the enhancements outlined in Chapter 5 and 6. Intelligent tutoring systems developers can use the recommendations described in this chapter to build the next generation of math tutors that take advantage of the learning benefits allowed by handwriting input.

Figure 7.1 shows a proposed interface that is based on the Cognitive Tutor Algebra interface but allows the student to enter his solution via handwriting input. On the surface it looks largely like the prototype used in the Cognitive Tutor Study (Figure 4.13). It differs from that prototype in the way that the interaction flow would be structured; this chapter presents scenarios of use for such a tutor, describing how a student would use the system and how the system would handle tutoring opportunities.

The key points to note about this scenario include the following facts:

1. The tutoring paradigm uses **answer-level feedback**, as long as the student gets the correct final answer. Only if the student types the wrong answer does the recognizer attempt to interpret the student's solution, beginning a tutoring intervention.
2. **Worked examples** are used as part of the instructional paradigm, as they have been confirmed through this dissertation work to be effective learning aids in this task and context.
3. When the student makes an error on the final answer, the system launches recognition of the problem-solving process, combining the recognition hypothesis based on the written strokes with the tutor-provided context information. The system uses the degree of deviation of the result from the correct set of possibilities for each problem-solving step in order to determine which step is most likely to contain the error that led to the incorrect final answer.

The individual steps of the interaction scenario follow.



**Figure 7.1:** The proposed interface for a tutor using Cognitive Tutor Algebra as its base and allowing students to enter the problem-solving process via handwriting input. The worked example being used as reference by the student appears on the lefthand side of the screen. The handwriting input space is a blank, unconstrained input space. The text box for the student's final answer is on the bottom right of the screen, next to the “Check My Answer” button, which launches a tutoring intervention if the typed final answer is incorrect.

## 7.1 Interaction Scenario

Figures 7.2 to 7.12 illustrate the interaction paradigm envisioned for the tutoring system using handwriting input and recognition, made possible by the explorations of this dissertation.

The screenshot shows a software interface for Carnegie Learning's Algebra I. At the top, there is a menu bar with File, Tutor, Go To, View, Help, and a toolbar with icons for Undo, Redo, Hint, Done, and Glossary. Below the menu is a navigation bar with tabs: WorkedExample (selected), Practice, Look Back, and Look Ahead.

The main content area has a title "18 - Linear Equations with Variables on Both Sides" and a subtitle "1 - Reviewing Solving Equations with Multiplication (No Type In)". On the left, there is a "HandwritingInput" section with a text box containing the equation  $-4,613 = -1,144y - 555$ . Below the text box are buttons for Draw and Erase. A note says: "Copy the steps of the example shown below in the space to the right. Think hard about each step as you do so. What is the purpose of each step?" Another note says: "Type your final answer in the text box on the bottom of the screen."

On the right, there is a "Check my Answer" section with a text input field labeled "y =". Below it is an "AnswerChecker" section with a text input field labeled "y =".

The central part of the screen contains a numbered list of steps:

1. Copy the problem  
 $-4,613 = -1,144y - 555$
2. Add to both sides to get the variable term by itself  
 $+555$
3. Add or subtract like terms  
 $-4,058 = -1,144y$
4. Divide both sides to get the variable alone  
$$\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$$
5. Simplify fractions  
$$\frac{2,029}{572} = y$$

**Figure 7.2:** The scenario begins with the (fictional) student beginning a new lesson on solving linear equations with variables on both sides ( $ax + b = cx + d$ ). The lesson begins with a review of a simpler problem type,  $ax + b = c$ . The student is given an example of the first type of problem she will see, and is instructed to copy it out while thinking critically about the steps involved.

The screenshot shows a software interface for Carnegie Learning's Algebra I. At the top, there are standard menu options: File, Tutor, Go To, View, Help. Below that is a navigation bar with tabs: 18 - Linear Equations with Variables on Both Sides, 1 - Reviewing Solving Equations with Multiplication (No Type In), WorkedExample, Look Ahead, Problems, and Look Back. On the far right of the navigation bar are three buttons: Done, Hint, and Glossary.

The main workspace is titled "HandwritingInput". It contains handwritten mathematical steps for solving the equation  $-4,613 = -1,144y - 555$ . The steps shown are:

$$\begin{array}{r} -4,613 \\ + 555 \\ \hline -4,058 \end{array}$$

$$\begin{array}{r} -4,058 \\ -1,144 \\ \hline -2,914 \end{array}$$

$$\begin{array}{r} -2,914 \\ \hline 572 \end{array}$$

Below these handwritten steps, the final typed-in answer is  $y = 2029/572$ .

To the left of the handwritten steps, there is a text area with instructions:

Copy the steps of the example shown below in the space to the right. Think hard about each step as you do so. What is the purpose of each step?

Type your final answer in the text box on the bottom of the screen.

On the right side of the workspace, there is a "Check my Answer" button. At the very bottom of the screen, there is an "Answer Checker" button.

**Figure 7.3:** The student has copied out the example as instructed. When she clicks the “Check My Answer” button, the system will check that she has actually copied the example and done it successfully (by checking the final, typed-in answer). In this case, she has, so she will be allowed to move on.

The screenshot shows a software interface for Carnegie Learning's Algebra I. At the top, there is a menu bar with File, Tutor, Go To, View, Help, and tabs for HandwritingInput, WorkedExample, and Problems. The main area displays a problem titled "18 - Linear Equations with Variables on Both Sides" under the section "1 - Reviewing Solving Equations with Multiplication (No Type In)". The problem is:  $1,426 - 1,308y = 2,486$ . Below the equation, there are buttons for Hint, Done, Glossary, Draw, and Erase. A text box for handwriting input is present. To the right, there is a "Check my Answer" button and an "AnswerChecker" field where the student can type their answer. On the left, there is a sidebar with "WorkedExample" and "Problems" buttons.

**HandwritingInput**

$1,426 - 1,308y = 2,486$

Done Hint Glossary Draw Erase

WorkedExample

Use the example shown below to help you solve the problem given on the right side of the screen. Think about how to apply the steps shown in the example to your problem.

Type your final answer in the text box on the bottom of the screen.

1. Copy the problem  
 $-4,613 = -1,144y - 555$

2. Add to both sides to get the variable term by itself  
 $+555$

3. Add or subtract like terms  
 $-4,058 = -1,144y$

4. Divide both sides to get the variable alone  
 $\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$

5. Simplify fractions  
 $\frac{2,029}{572} = y$

Check my Answer

AnswerChecker

**y =**

**Figure 7.4:** After moving on, the student is given a new problem, analogous to the problem shown in the example that she has just copied, to solve on her own. The example remains onscreen to scaffold her problem-solving experience. The problem and the example are of the same type (*e.g.*,  $ax + b = c$ ), but may have different surface forms.

The screenshot shows a software interface for Carnegie Learning's Algebra I. At the top, there are standard menu options: File, Tutor, Go To, View, Help. Below the menu is a navigation bar with tabs: Look Ahead, Problems, Look Back, WorkedExample (which is selected), and HandwritingInput. The main content area has a green header bar with the title "18 - Linear Equations with Variables on Both Sides" and a sub-section "1 - Reviewing Solving Equations with Multiplication (No Type In)". On the right side of the header are three buttons: Hint (blue), Done (orange), and Glossary (green). Below the header, there is a text input field containing the equation  $1,426 - 1,308y = 2,486$ . To the right of the input field are three buttons: Draw (green), Erase (blue), and a small icon (green). The main workspace is divided into two sections. The left section contains a worked example with handwritten steps:

1. Copy the problem  
 $-4,613 = -1,144y - 555$
2. Add to both sides to get the variable term by itself  
 $+555 \quad +555$
3. Add or subtract like terms  
 $-4,058 = -1,144y$
4. Divide both sides to get the variable alone  
 $\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$
5. Simplify fractions  
 $\frac{2,029}{572} = y$

The right section of the workspace shows the student's handwritten work for the current problem:

$$\begin{array}{r}
 1,426 - 1,308y = 2,486 \\
 -1,426 \quad -1,426 \\
 \hline
 -1,308y = -1,486 \\
 \end{array}$$

Below the workspace, there is a text input field labeled "y =" and a button labeled "AnswerChecker". On the far right, there is a "Check my Answer" button.

**Figure 7.5:** The student is solving the problem given to her, by referring to the example onscreen.

The screenshot shows a software interface for solving linear equations. At the top, there are standard menu options: File, Tutor, Go To, View, Help, and tabs for HandwritingInput, WorkedExample, and Problems. Below the tabs, the title "18 - Linear Equations with Variables on Both Sides" and a sub-section "1 - Reviewing Solving Equations with Multiplication (No Type In)" are displayed. On the right side, there are buttons for Done, Hint, Glossary, Draw, Erase, and a "Check my Answer" button.

The main workspace contains handwritten work for solving the equation  $1,426 - 1,308y = 2,486$ . The student has written:

$$\begin{array}{r}
 1,426 - 1,308y = 2,486 \\
 -1,426 \quad -1,426 \\
 \hline
 -1,308y = 1,060 \\
 -1,308 \quad -1,308 \\
 \hline
 y = -265 \\
 \hline
 572
 \end{array}$$

Below the handwritten work, the typed-in answer  $y = -265/327$  is shown. To the left of the answer, the "Answer Checker" status is listed as "y = -265/327".

The bottom section of the workspace provides a step-by-step guide for solving the equation:

1. Copy the problem  
 $-4,613 = -1,144y - 555$
2. Add to both sides to get the variable term by itself  
 $+555$
3. Add or subtract like terms  
 $-4,058 = -1,144y$
4. Divide both sides to get the variable alone  
 $\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$
5. Simplify fractions  
 $\frac{2,029}{572} = y$

**Figure 7.6:** The student has completed solving the problem and types in her final answer. When she clicks the “Check My Answer” button, the system will check that she has actually solved the problem (by checking the handwriting space for input) and done it successfully (by checking the final, typed-in answer). In this case, she has, so she will be allowed to move on.

The screenshot shows a software interface for Carnegie Learning's Algebra I. At the top, there are standard window controls (Close, Minimize, Maximize) and a toolbar with icons for Help, Hint, Done, Draw, Erase, and Glossary.

The main area displays a math problem: "18 - Linear Equations with Variables on Both Sides" and "1 - Reviewing Solving Equations with Multiplication (No Type In)".

A "HandwritingInput" section contains the handwritten solution:

$$\begin{array}{r}
 1,426 - 1,308y = 2,486 \\
 -1,426 \\
 \hline
 -1308y = 1,060 \\
 \hline
 y = \frac{1,060}{-1308} \\
 y = -\frac{265}{327}
 \end{array}$$

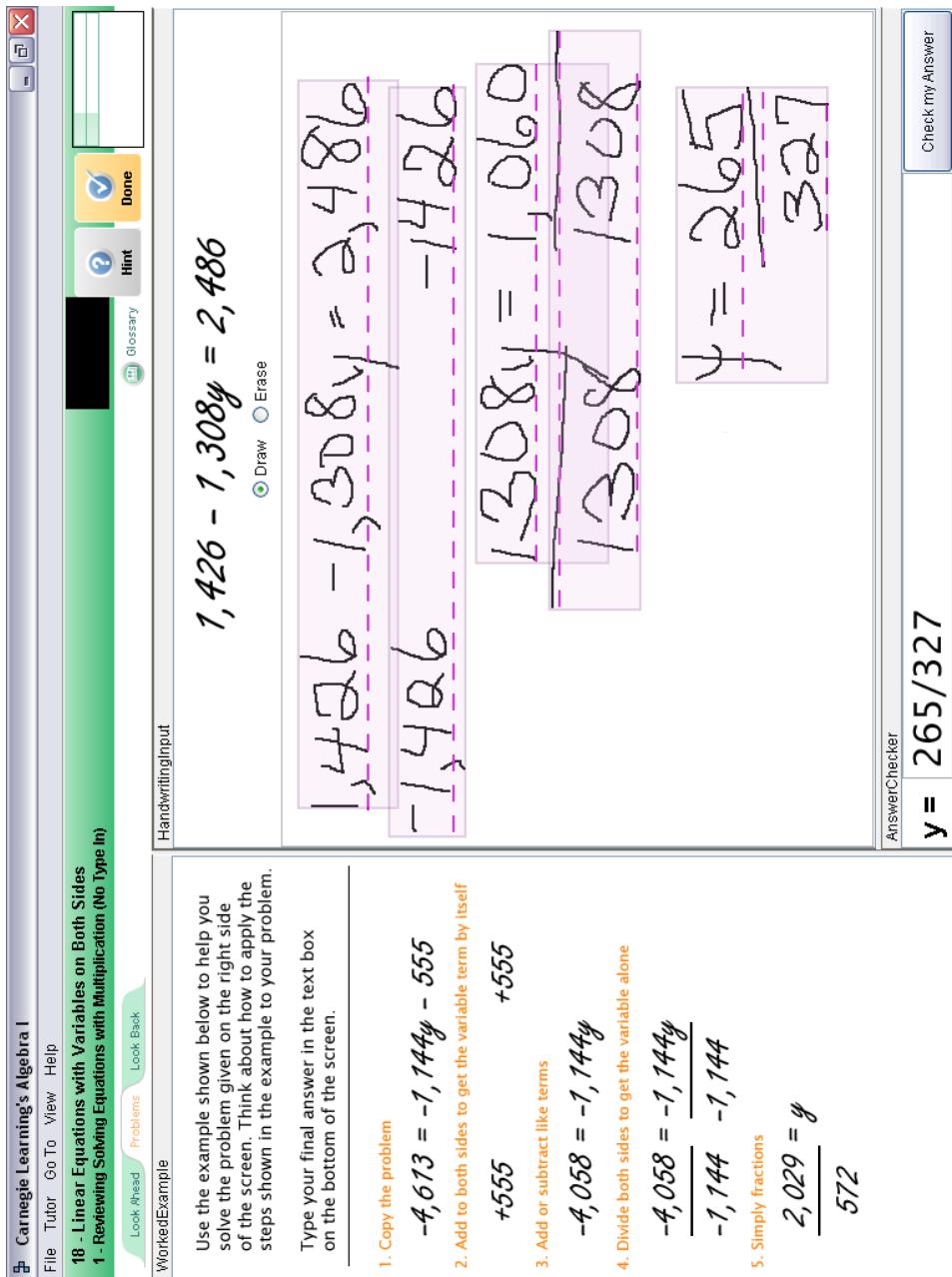
To the right of the handwriting, the student has typed "y = 265/327".

An "AnswerChecker" section on the right shows the correct answer: "y = 265/327". Below it is a "Check my Answer" button.

The bottom part of the screen shows a "WorkedExample" section with steps for solving the equation:

1. Copy the problem  
 $-4,613 = -1,144y - 555$
2. Add to both sides to get the variable term by itself  
 $+555$
3. Add or subtract like terms  
 $-4,058 = -1,144y$
4. Divide both sides to get the variable alone  
 $\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$
5. Simplify fractions  
 $\frac{2,029}{572} = y$

**Figure 7.7:** In an alternative scenario, the student has completed solving the problem and types in her final answer. When she clicks the “Check My Answer” button, the system will check that she has actually solved the problem (by checking the handwriting space for input) and done it successfully (by checking the final, typed-in answer). In this case, she has *not* solved the problem successfully: she has forgotten the negative sign when transcribing “-1308” in the third step, so the tutoring intervention begins.



**Figure 7.8:** The system launches recognition of the handwriting input space containing the student's complete solution. The system first extracts the strokes belonging to each step by finding baselines for each line or step of the problem and grouping strokes within steps. These strokes are then iteratively fed into the recognizer; as each character is recognized, the tutor context information about the set of correct options for each step is considered. Once the problem has been completely recognized, the system attempts to determine on which step the student's error occurred by calculating the deviation of the recognized steps from the correct options and choosing the maximum-deviation step as the most likely to contain the error. These are background processing steps and are *not* shown to the user, but are included here for illustrative purposes.

The screenshot shows a software interface for solving linear equations. At the top, there are standard menu options: File, Tutor, Go To, View, Help. Below that is a toolbar with icons for Hint, Done, Draw, Erase, and Glossary. The main area is titled "HandwritingInput". A problem is displayed:  $1,426 - 1,308y = 2,486$ . The student has handwritten the equation and attempted to solve it by subtracting 1,426 from both sides. The student's work is shown in red dashed lines:

$$\begin{array}{r} 1,426 \\ - 1,308y \\ \hline -1,426 \end{array}$$

A "Final Answer Incorrect" dialog box is overlaid on the screen, containing the following text:

Your answer is not correct. It looks like you have made an error on this step. Check your work and try to fix it.

Don't forget you can always ask for a hint!

To the right of the student's work, the correct solution is shown:  $y = 265/327$ .

**Figure 7.9:** Once the system has identified a step as the most likely to contain the error, it highlights the strokes associated with that step and prompts the student to revisit her solution beginning with that step. An alternative, shown in the next figure, is to ask the student to verify that the recognition result matches what she had written and commence tutoring once any ambiguity is resolved.

The screenshot shows a software interface for solving linear equations. At the top, there are standard menu options: File, Tutor, Go To, View, Help. Below the menu is a toolbar with icons for Hint, Done, Glossary, Erase, Draw, and a magnifying glass icon. The main workspace is divided into sections:

- HandwritingInput:** A large area where a student has handwritten the equation  $1,426 - 1,308y = 2,486$ . The system has interpreted this as  $1,426 - 1,308y = 2,486$ , with the original input crossed out.
- WorkedExample:** A section showing a worked example of solving the equation  $-4,613 = -1,144y - 555$ . It includes steps like "Copy the problem", "Add to both sides to get the variable term by itself", "Add or subtract like terms", "Divide both sides to get the variable alone", and "Simply fractions".
- Final Answer Incorrect:** A pop-up message states: "Your answer is not correct. It looks like you have made an error on this step. Did you mean to write  $1308y = 1060$  for this step?" It includes "YES" and "NO" buttons.
- AnswerChecker:** A section where the student can type their answer. The current input is  $y = 265/327$ .
- Check my Answer:** A button to check the correctness of the typed answer.

**Figure 7.10:** An alternative prompting style to the one presented in the previous figure, which is agnostic about what the student wrote, is the one presented here. The system asks the student to verify that the recognition result matches what she had written and commences tutoring once any ambiguity is resolved. An advantage of using this method is that the system exposes its interpretation to the user immediately, cutting off any error spirals which could occur in the previous prompting style. However, this very exposure could disrupt the student’s learning process.

Carnegie Learning's Algebra I

File Tutor Go To View Help

18 - Linear Equations with Variables on Both Sides  
1 - Reviewing Solving Equations with Multiplication (No Type In)

Worked Example Problems Look Back

Handwriting Input

1. Copy the problem  
 $-4,613 = -1,144y - 555$

2. Add to both sides to get the variable term by itself  
 $+555$

3. Add or subtract like terms  
 $+555$

4. Divide both sides to get the variable alone  
 $\frac{-4,058}{-1,144} = \frac{-1,144y}{-1,144}$

5. Simplify fractions  
 $\frac{2,029}{572} = y$

**Final Answer Incorrect**  
Your answer is not correct. It looks like you have made an error on this step.  
Did you mean to write " $1508y = 1100y$ " for this step?  
 YES  NO

Ok, sorry about that! Please type what you meant here:

**ANSWER CHECKER**

**y = 265/327**

**Check my Answer!**

Hint Done Glossary Draw Erase

**1308y = 1,060**

**Figure 7.11:** Following the prompting style from the previous figure, if the student indicates that the recognition result is not correct (*i.e.*, it does not match her intended input), the system brings up a text box in which the student can enter her input for that step unambiguously. If it is an error, tutoring commences as per Cognitive Tutor methods. If it is not an error, the recognizer can either iteratively attempt to identify another error step, or turn this problem into a worked example by providing the solution for the student to study, depending on its error identification confidence.

The screenshot shows a student working on a linear equation problem in Carnegie Learning's Algebra I software. The interface includes a toolbar at the top with 'File', 'Tutor', 'Go To', 'View', 'Help', 'Handwriting Input' (selected), 'Worked Example', 'Problems', 'Look Back', and 'Answer Checker'. The main area displays a math problem:  $1,426 - 1,308y = 2,486$ . The student has handwritten their work in the 'Handwriting Input' field:

$$\begin{array}{r}
 1,426 - 1,308y = 2,486 \\
 -1,426 \quad -1,426 \\
 \hline
 -1,308y = 1,060 \\
 -1,308 \quad -1,308 \\
 \hline
 y = -265
 \end{array}$$

The student has made a mistake in the first step, writing  $1,426 - 1,308y$  instead of  $1,426 - 1,308y =$ . The tutor provides feedback and steps to solve the equation:

1. Copy the problem  
 $-4,613 = -1,144y - 555$
2. Add to both sides to get the variable term by itself  
 $+555$
3. Add or subtract like terms  
 $-4,058 = -1,144y$
4. Divide both sides to get the variable alone  
 $-4,058 = -1,144y$   
 $-1,144 \quad -1,144$   
 $\hline$   
 $2,029 = y$   
 $572$
5. Simplify fractions

The student's final answer is  $y = -265/327$ , which is incorrect. A 'Check my Answer' button is visible on the right.

**Figure 7.12:** The student corrects her solution beginning with the step on which she made her first error. When she arrives at the correct final answer and types it into the text box, the tutor provides positive feedback and allows her to move on. The student continues working in this way until the tutor's knowledge-tracing model determines that a certain level of **mastery** is reached.

## 7.2 Interaction Flow

The scenario outlined in the preceding screenshot mockups (Figures 7.2 to 7.12) roughly embodies the following flow of interaction:

1. The student copies the worked example given when she is about to first see a specific problem type—this feeds forward into her problem-solving experience.
2. The student receives a problem to solve that is analogous to the example; the example stays onscreen for reference.
3. The student begins solving the problem.
4. The student types in her final answer in a text box; this input mode is used for minimal ambiguity—it is easy and desirable to allow the student to move on quickly if she is successfully solving problems.
5. The tutor checks the student's answer.
  - If it is correct, the student moves on—no tutoring is needed.
  - If it is incorrect, tutoring is needed, and the system begins a recognition cycle.
    - (a) The system detects the baselines of the strokes onscreen, separating them into distinct lines, or problem-solving steps.
    - (b) The system iterates over each step of the problem, recognizing the strokes and then applying the tutor information for that step as indicated in Chapter 6.
    - (c) The system uses an error detection algorithm once all steps have been recognized—choosing which step is furthest from any correct solution path based on deviation measured by Levenshtein string distance.
    - (d) The system highlights the strokes contained in the hypothesized error step and prompts the student to verify her answer for that step.
      - If the student indicates the recognition result *does* match what she had written, tutoring commences by providing a hint as to why the step is incorrect; the student re-writes her solution from that point on.
      - If the student indicates that the recognition result *does not* match what she had written, she is given a text box to enter her step again with no ambiguity.
        - \* If the non-ambiguous entry is correct, the system re-attempts to identify the error step.
        - \* If the non-ambiguous entry is incorrect, tutoring commences.
6. The student continues working in this way until the tutor model detects that mastery has been reached.

### 7.3 Errors and Error Repair Strategies

With the addition of the results laid out in this dissertation, there is plenty of HCI work to draw on in building an effective tutoring system for math that allows handwriting input, including work on interface recommendations regarding widget placement [115], user and usability requirements [118], recommendations for error repair and error models for children using handwriting input [114], and so on.

The interaction scenario presented in this chapter attempts to avoid exposing the system's recognition errors to the student as much as possible. In fact, that is a foundational approach taken by this dissertation work. However, it may not always be possible or desirable to do so, especially as an error spiral become more pronounced. For example, a student may be having technical difficulties with the pen input device that is causing his handwriting to be continually mis-recognized. For this reason, it is worth considering possible errors, both student and system, and strategies for error repair.

In systems with potential for error, such as the handwriting recognizer used in this dissertation, it often falls to the user to repair such errors. Mankoff and Abowd [90] identified five approaches to error handling in recognition-based interfaces: (1) error reduction, (2) error discovery, (3) error correction techniques, (4) validation of techniques, and (5) toolkit-level support. Error repair has been studied extensively in speech recognition interfaces, but less so in handwriting interfaces. The approaches to error repair taken in this dissertation are (1) error reduction—avoid making errors in the first place as much as possible, and (2) error discovery—attempt to find the system's errors before they are presented to the student. Error correction techniques and their validation can be taken into account if and when this system is developed and deployed. Many types of error correction techniques exist, and some work has been done in exploring their suitability for use with children. Read et al. [116] showed that students spend more time correcting errors when recognition is real-time (*i.e.*, displayed as characters are being written) than when it occurs at the end of a discrete unit, such as a sentence or equation; however, the total number of errors made does not differ. The extra time spent repairing errors is extraneous; delaying recognition feedback until later seems wise in this domain with this target population.

A system developer may choose to provide explicit error recovery techniques, and may find it useful to know what types of errors one can expect from this population and in this domain. Specific error types that are likely to occur in handwriting interfaces have been studied [127]. One can expect to find (1) discrete noises, (2) badly formed shapes, (3) input that is legible by the human but not by the recognizer, (4) misspelled words, (5) cancelled material and (6) device-generated errors. Types of repair strategies undertaken by users when these errors occur are deletion, completion, insertion and overwriting [65]. Error types for children using handwriting input include [121]: spelling errors, construction errors (*e.g.*, penmanship errors), execution errors in using the handwriting device, on top of recognition errors. In the learning domain, of course, the possibility of the student making math errors is very real and must be taken into account.

Additionally, observational studies of children using pen-based input and handwriting recog-

nizers have been undertaken, which have helped to identify specific types of device-generated errors, including position-based errors such as when the stylus and pointer onscreen are not properly calibrated, or when the student writing goes off the page [115]. It is true that these studies have used a paradigm in which the input device and display screen are separate, requiring the student to look at one *or* the other, whereas there are other possibilities, such as TabletPCs and the devices used in the Cognitive Tutor Study, which allow direct writing onto the display surface. This may eliminate the chances of some of the issues occurring, but others will still be present, such as the calibration issue.

The interaction scenario presented in this dissertation does not specifically deal with *recognition* error repair and recovery, and solely focuses on math errors, because it is the foundational assumption of this work that involving the student in error repair induces its own cognitive load and may reduce or eliminate the benefits for handwriting input that this work has found. However, it is a common approach in many recognition applications, and so it is worth considering the known research and what recommendations can be drawn from it. The relevant issues described in this section can help future application developers to design effective interactions that are robust to both student and system errors.

# Chapter 8

## Conclusion

### 8.1 Discussion and Summary

This dissertation has presented work and results in the area of applying handwriting recognition to intelligent tutoring systems for algebra equation-solving. The work began with a foundation of ground-laying studies that established that handwriting is beneficial for math input and math learning, and explored the factors responsible for handwriting's benefits. The Math Input Study found that, for college-level students performing math entry (not learning), handwriting was three times faster than typing, and this benefit increased as the equations got more complex. The Lab Learning Study found that, for middle and high school students performing a learning task, handwriting was two times faster, but learning did not suffer due to the decreased time spent in the lesson. In both studies, students indicated that handwriting was the most comfortable and enjoyable modality to enter math of all the ones they tried in the studies.

The Cognitive Tutor Study took this approach one step further and explored different pedagogical interventions to accompany handwriting input, such as **worked examples** and **answer-level feedback**, in order to account for the lack of highly accurate recognition. In this study, these interventions were compared to the traditional Cognitive Tutor Algebra interface that is based on typing. In this study, it was found that, in a classroom context with high school students using a state-of-the-art intelligent tutoring system, students using handwriting experienced higher learning gains than control students using the traditional Cognitive Tutor, in spite of the lack of **step-targeted feedback**. However, the “worked examples plus step targeted feedback” condition of that study performed the best of all the conditions, so it was clear that it was worth exploring ways to enhance handwriting recognition accuracy in order to be able to support step-targeted feedback while in handwriting.

This dissertation presented several techniques to improve recognition accuracy for use in the math tutoring domain, including training on a set of handwriting data from the target population in order to enhance **writer-independent** accuracy and to reduce or remove the need for students to take classroom time to train the system before they begin learning. In addition, the use of domain-

context information to refine recognition results was explored, by adding information from the tutoring system about the set of correct possible answers at each step of the problem-solving process. The domain-context information used, though very simple in scope, significantly improved recognition accuracy by approximately 9%. Furthermore, the impact on the student of this level of recognition and the system's ability to identify on what step a student error occurred was quantified and expressed in a general formula. This formula can be used to estimate the impact on the student when better recognizers become available as technology and algorithms continue to advance.

Finally, based on all these results, it is possible to structure a reasonable tutoring interaction paradigm, which this dissertation has outlined in the form of a scenario of use with screenshot mockups of the interaction. Designers of intelligent tutoring systems for mathematics can use this interaction scenario to build on the proof-of-concept prototype from this dissertation work and implement a tutoring system that can take advantage of the benefits of handwriting input, in spite of the technological limitations of recognition. Further work in this area to evaluate such a tutoring system would provide valuable commentary on the real-world validity of the results presented in this dissertation.

## 8.2 Contributions

This dissertation makes contributions in several areas:

- Learning science and intelligent tutoring systems:
  - Exploring and outlining the connections between learning gains and the input methods used to enter problem-solving processes in online tutors;
  - Designing and developing a handwriting interface for ITS in mathematics;
  - Providing a proof-of-concept of handwriting input being used with real students that can inform other domains such as geometry or physics;
- Handwriting recognition:
  - Conducting case studies exploring the success of several representative recognition engines when trained and tested on middle and high school students writing in the math domain;
  - Developing techniques and evaluation metrics for using handwriting recognition in the classroom, taking into account domain-specific context information to improve recognition results;
  - Discovering what levels of recognition accuracy are obtainable when the users are middle and high school students solving math problems;
  - Developing design guidelines for how to incorporate the handwriting input modality into new applications such as ITS for math;

- Applied machine learning and information retrieval:
  - Implementing co-recognition techniques for a new domain, that of math learning;
  - Applying ranking fusion methods in a new domain, that of handwriting recognition, especially for math;
- Human-computer interaction:
  - Establishing the usability advantages of handwriting interfaces over typing for math input on the computer, and especially in a learning task;
  - Exploring and understanding the factors causing these advantages;
  - Addressing the integration challenges of adapting handwriting recognition for use in interfaces for real users.

## 8.3 Future Work

There are many potentially fruitful avenues for future work and further exploration of this area. They can be roughly divided into two areas: (1) pedagogical explorations, and (2) technical explorations.

### 8.3.1 Further Pedagogical Explorations

An obvious avenue of further work is to extend the prototype developed in this dissertation into a complete system, and evaluate it with real students in the lab or a classroom. This dissertation outlines scenarios of use upon which the interaction flow of this prototype could be modeled in order to prove the conjecture that the levels of accuracy obtained by the methods presented here are usable by students in this domain. The results from the foundational studies, especially the Cognitive Tutor Study, indicate that this conjecture is probably true, but it has yet to be empirically established. The accuracy obtained via this method was not quite at the 91% standard set by Read et al. [117], but it was greatly improved over the original accuracy on the corpus. Such a study could use a **Wizard-of-Oz** style design with perfect recognition as the control in order to set the “gold standard” and determine how far behind, if at all, the methods presented in this dissertation leave the situation.

In addition, other interaction models besides those presented in this dissertation could be explored. For example, the tutoring system could engage the student in a *reciprocal teaching* [107] tutoring paradigm, in which the tutor plays the role of student and the student plays the role of tutor. Human-human tutoring has been shown to benefit tutors just as much as the tutees (*e.g.*, [61]). The system could then structure the interactions to verify its interpretation of student input as a tutoring dialogue: “It looks like you are adding two to both sides, is that true? Why are you doing that?” Such an interaction approach has the added benefit of engaging the student in self-explanation of

his solution process, which has been shown to engender deeper learning [31, 63]. Alternatively, the tutoring interaction could be structured more like an educational game, in which correcting the system is associated with rewards. For instance, in a spy game, the system could be construed as a devious agent attempting to interfere with and garble messages sent from the student to “command central,” requiring the student to find and correct places where the recognition system made errors. Educational games are a hot topic in educational technology and intelligent tutoring systems research [49], due to the expectation that such methods will increase engagement and therefore learning (*c.f.*, [47]).

This dissertation focuses on algebra equation-solving because the menu-based equation-solver tool is a crucial component of many of the Cognitive Tutor Algebra lessons, and because equation-solving is a problem-solving process that could benefit from an unconstrained input space allowing nonlinear layouts. However, the recommendations and methods used here can be adapted for other domains such as geometry, chemistry or physics tutoring, for which tutors already exist, and in which sketching and writing are crucial parts of the problem-solving process. Sketch recognition is essentially a similar problem to handwriting recognition, albeit more complex due to the even less constrained nature of sketching *vs* writing. Contextual cues and semantics have been used in approaches to sketch recognition, with good results (*e.g.*, [75, 4, 15]). The lessons from this dissertation that could be applied to those approaches are the benefits of training data from the target population, the application of ranking fusion techniques, and the concept of delaying feedback until enough information is available to be more confident about one’s recognition hypotheses.

### 8.3.2 Further Technical Explorations

Although the approach described in this dissertation achieved a 9% reduction in recognition error rates, it may be possible to improve this effect even further. For instance, while FFES performed the best out of the three recognition systems tested for this dissertation, recognition technology is continually improving. It may be that the new state-of-the-art can achieve higher baselines than FFES, and therefore, the techniques presented here could improve upon those recognizers to an even higher final accuracy rate, approaching or surpassing 91% (*c.f.*, [117])

Worth exploring are some other approaches to increase or magnify the recognition accuracy improvement provided by this technique presented in this dissertation. The tutor context information used in this dissertation included only the set of correct options at a particular step, biasing the tutor somewhat toward correct input by the students. Other information could help refine this behavior, such as including the probability, from the knowledge-tracing model, of this particular student’s **mastery** of a skill and likelihood of performing a step correctly.

The representation of the tutor context information is highly simplified (“bag of words”) in order to reduce the impact of the **alignment problem**. However, this abstracts some of the information provided by the tutor which could be useful, such as order of symbols and bi-grams which could be used to narrow down the set of correct options. One way to address this would be to remove from the bag any symbol which has been chosen by the combined system to fit a set of

strokes in that step already, or to reduce the probability that it would be written again to some low number.

Also, the combination of the recognition hypothesis and tutor-provided context information is currently very simple: average rank-sort. Other methods may be more robust and additionally remove some of the limitations of this approach, such as the alignment problem. Other ranking fusion techniques besides the modified Borda count used here, such as Markov chaining [45, 122], voting classifiers [72], or score-based rather than rank-based methods [64], might yield better ranking results. Using a Bayesian network [109] to model the relationship between the observed variable (the recognition result) and the sources of error (student error and recognition error) could improve the system’s ability to detect on which step the student’s error occurred. Using multiple classifiers is a technique already explored to some degree in the handwriting recognition community, and could be combined here with the tutor context information provided.

Finally, the parser included with FFES, called DRACULAE [149], was not incorporated into the recognition process used in this dissertation because of difficulties with altering the grammar to handle the special types of “operation steps” used in problem-solving, and the spatial arrangement of the templates in the test cases seemed to cause problems. In general, however, the parser is quite flexible, and could further enhance accuracy if it were used.

## 8.4 Final Remarks

This dissertation attempts to address the following thesis:

*The use of handwriting interfaces in intelligent mathematics tutoring software can yield higher learning gains in students through lower cognitive load than the use of standard typing interfaces. An important part of achieving this effect is increasing recognition accuracy to a level sufficient for adequate instructional feedback.*

While a prototype system that included handwriting *recognition* rather than simply handwriting *input* has not been built and evaluated during this dissertation, the results obtained regarding the benefits of handwriting interfaces for the math tutoring domain show that the above thesis holds.

The scenarios and mockups produced during the course of this dissertation can be used by, for instance, the Cognitive Tutor Algebra developers (Carnegie Learning, Inc.) to incorporate handwriting recognition into their tutoring system, especially for algebra equation-solving lessons, to capitalize on the benefits seen here for handwriting input in this domain and to push their state-of-the-art tutoring system to become even better. These tutors are used in over 2,600 schools nationwide at the time of this writing, allowing a great possibility for the work in this thesis to be widely disseminated to benefit many students. Developers of other tutoring systems can also learn from and use these methods in their own work, bringing unconstrained handwriting to the “desktop” once more.

# Bibliography

- [1] G.S. Aist. Challenges for a mixed initiative spoken dialog system for oral reading tutoring. Technical Report SS-97-04, AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction, 1997.
- [2] V.A.W.M.M. Aleven and K.R. Koedinger. An effective metacognitive strategy: Learning by doing and explaining with a computer-based cognitive tutor. *Cognitive Science*, 26:147–149, 2002.
- [3] V.A.W.M.M. Aleven, K.R. Koedinger, and O.A. Popescu. A tutorial dialog system to support self-explanation: Evaluation and open questions. In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 39–46. IOS Press, Amsterdam, the Netherlands, 2003.
- [4] C. Alvarado. Sketchread: a multi-domain sketch recognition engine. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 23–32. ACM Press, 2004.
- [5] J.R. Anderson, A.T. Corbett, K.R. Koedinger, and R. Pelletier. Cognitive tutors: Lessons learned. *Journal of the Learning Sciences*, 4:167–207, 1995.
- [6] R.H. Anderson. *Syntax-directed recognition of hand-printed two-dimensional mathematics*. PhD thesis, Department of Engineering and Applied Mathematics, Harvard University, 1968.
- [7] L. Anthony, A. Corbett, A.Z. Wagner, S.M. Stevens, and K.R. Koedinger. Student question-asking patterns in an intelligent algebra tutor. In J.C. Lester, R.M. Vicari, and F. Paraguacu, editors, *Proceedings of the International Conference on Intelligent Tutoring Systems; Lecture Notes in Computer Science 3220*, pages 455–467. Springer Berlin, Heidelberg, Germany, 2004.
- [8] L. Anthony, J. Yang, and K.R. Koedinger. Evaluation of multimodal input for entering mathematical equations on the computer. In *Proceedings of the ACM Conference on Human Factors in Computing (CHI)*, pages 1184–1187. ACM Press, New York, NY, USA, 2005.

- [9] L. Anthony, J. Yang, and K.R. Koedinger. Entering mathematical equations multimodally: Results on usability and interaction design. Technical Report CMU-HCII-06-101, Carnegie Mellon University, School of Computer Science, Human-Computer Interaction Institute, 2006.
- [10] L. Anthony, J. Yang, and K.R. Koedinger. Towards the application of a handwriting interface for mathematics learning. In *Proceedings of the IEEE Conference on Multimedia and Expo (ICME)*, pages 2077–2080. IOS Press, Amsterdam, the Netherlands, 2006.
- [11] L. Anthony, J. Yang, and K.R. Koedinger. Adapting handwriting recognition for applications in algebra learning. In *Proceedings of the ACM Workshop on Educational Multimedia and Multimedia Education*, pages 47–56. ACM Press, New York, NY, USA, 2007.
- [12] L. Anthony, J. Yang, and K.R. Koedinger. Benefits of handwritten input for students learning algebra equation solving. In *Proceedings of the International Conference on Artificial Intelligence in Education (AIED)*, pages 521–523. IOS Press, Amsterdamm, the Netherlands, 2007.
- [13] L. Anthony, J. Yang, and K.R. Koedinger. How handwritten input helps students learning algebra equation solving. Technical Report CMU-HCII-08-100, Carnegie Mellon University, School of Computer Science, Human-Computer Interaction Institute, 2008.
- [14] L. Anthony, J. Yang, and K.R. Koedinger. Toward next-generation, intelligent tutors: Adding natural handwriting input. *IEEE Multimedia*, 15(3):64–68, 2008.
- [15] J. Arvo and K. Novins. Appearance-preserving manipulation of hand-drawn graphs. In *International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia (GRAPHITE)*, pages 61–68. ACM Press, 2006.
- [16] C. Beal, W.L. Johnson, R. Rabrowski, and S. Wu. Individualized feedback and simulation-based practice in the tactical language training system: An experimental evaluation. In *Proceedings of the Artificial Intelligence in Education Conference (AIED)*, pages 749–749. IOS Press, Amsterdam, the Netherlands, 2005.
- [17] A.E. Beaton, I.V.S. Mullis, M.O. Martin, E.J. Gonzalez, D.L. Kelly, and T.A. Smith, editors. *The Mathematics Achievement in the Middle School Years*. International Association for the Evaluation of Educational Achievement, Boston, MA, USA, 1996.
- [18] A. Belaid and J.P. Haton. A syntactic approach for handwritten mathematical formula recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):105–111, 1984.
- [19] B.S. Bloom. The two sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13:4–16, 1984.

- [20] D. Blostein and A. Grbavec. Recognition of mathematical notation. In P.S.P Wang and H. Bunke, editors, *Handbook on Optical Character Recognition and Document Analysis*, pages 557–582. World Scientific Publishing Company, 1996.
- [21] C. Bonwell and J. Eison. Active learning: Creating excitement in the classroom. Technical Report Higher Education Report No.1, AEHE-ERIC, 1991.
- [22] J.C. Borda. Memoire sur les elections au scrutin. In *Histoire de l'Academie Royal des Sciences*. 1781.
- [23] J. Bransford, A.L. Brown, and R.R. Cocking (National Research Council), editors. *How People Learn: Brain, Mind, Experience, and School*. National Academies Press, Washington, D.C., USA, 2000.
- [24] C.M.L. Brown. Comparison of typing and handwriting in “two-finger typists”. In *Proceedings of the Human Factors Society*, pages 381–385. Human Factors Society, Santa Monica, CA, USA, 1988.
- [25] R.M. Brown. On-line computer recognition of handprinted characters. *IEEE Transactions on Electronic Computing*, 13:750–752, 1964.
- [26] M.D. Buhmann and M.J. Ablowitz. *Radial Basis Functions : Theory and Implementations*. Cambridge University Press, Cambridge, UK, 2003.
- [27] D. Buscaldi and P. Rosso. A bag-of-words based ranking method for the wikipedia question answering task. In C. Peters, P. Clough, F.C. Gey, J. Karlgren, B. Magnini, D.W. Oard, M. de Rijke, and M. Stempfhuber, editors, *Evaluation of Multilingual and Multi-modal Information Retrieval: 7th Workshop of the Cross-Language Evaluation Forum, Revised Selected Papers; Lecture Notes in Computer Science 4730*, pages 550–553. Springer Berlin, Heidelberg, 2007.
- [28] C.B. Cazden. Adult assistance to language development: Scaffolds, models, and direct instruction. In R.P. Parker and F.A. Davis, editors, *Developing literacy: Young children's use of language*, pages 3–17. International Reading Association, Newark, DE, USA, 1983.
- [29] K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: A survey. *International Journal of Document Analysis and Recognition*, 3:375–384, 2000.
- [30] M.-Y. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hidden markov model type stochastic network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.

- [31] M.T.H. Chi, M. Bassok, M.W. Lewis, P. Reimann, and R. Glaser. Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, 13:145–182, 1989.
- [32] M.T.H. Chi and K. VanLehn. Accelerated future learning via explicit instruction of a problem solving strategy. In R. Luckin, K.R. Koedinger, and J. Greer, editors, *13th International Conference on Artificial Intelligence in Education*, pages 409–416. IOS Press, Amsterdam, 2007.
- [33] R.C. Clark, F. Nguyen, and J. Sweller. *Efficiency in learning: evidence-based guidelines to manage cognitive load*. Pfeiffer, San Francisco, CA, USA, 2006.
- [34] G. Cooper and J. Sweller. Effects of schema acquisition and rule automation on mathematical problem-solving transfer. *Journal of Educational Psychology*, 79(4):347–362, 1987.
- [35] A.T. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In A. Kobsa, B.P. Woolf, D.N. Chin, and A.D. Esteban, editors, *Proceedings of User Modeling; Lecture Notes in Computer Science 2109*, pages 137–147. Springer Berlin, Heidelberg, 2001.
- [36] A.T. Corbett and J.R. Anderson. Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes. In *Proceedings of ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 245–252, 2001.
- [37] A.T. Corbett, K.R. Koedinger, and W.H. Hadley. Cognitive tutors: From the research classroom to all classrooms. In P. Goodman, editor, *Technology Enhanced Learning: Opportunities for Change*, pages 235–263. L. Erlbaum, Mahwah New Jersey, 2001.
- [38] A.T. Corbett and H. Trask. Instructional interventions in computer-based tutoring: Differential impact on learning time and accuracy. In *Proceedings of the ACM Conference on Human Factors in Computing (CHI)*, pages 97–104, 2000.
- [39] J.-P. Crettez. A set of handwriting families: style recognition. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 489–494. IEEE Computer Society, 1995.
- [40] Y.A. Dimitriadis and J.L. Coronado. Towards an art based mathematical editor that uses online handwritten symbol recognition. *Pattern Recognition*, 28(6):807–822, 1995.
- [41] T.L. Dimond. Devices for reading handwritten characters. In *Proceedings of the Eastern Joint Computing Conferences*, pages 232–237, 1957.
- [42] K. Mc Donald and A.F. Smeaton. A comparison of score, rank and probability-based fusion methods for video shot retrieval. In W.-K. Leow, M.S. Lew, T.-S. Chua, W.-Y. Ma, L. Chaisorn, and E.M. Bakker, editors, *Image and Video Retrieval; Lecture Notes in Computer Science 3568*, pages 61–70. Springer Berlin, Heidelberg, 2005.

- [43] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. In *Advances in Neural Information Processing Systems*, pages 155–161, 1996.
- [44] A. Druin and C. Solomon. *Designing multimedia environments for children*. John Wiley and Sons Ltd, New York, NY, USA, 1996.
- [45] C. Dwork, R. Kumar, M. Noar, and D. Sivakumar. Rank aggregation methods for the web. In *International Conference on the World Wide Web*, pages 613–622. ACM Press and Addison Wesley, 2001.
- [46] J. Eisenstein and C.M. Christoudias. A salience-based approach to gesture-speech alignment. In *In Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting*, pages 25–32. MIT Press, Cambridge, MA, USA, 2004.
- [47] E.S. Elliott and C.S. Dweck. Goals: An approach to motivation and achievement. *Journal of Personality and Social Psychology*, 54(1):5–12, 1988.
- [48] R. Fateman, T. Tokuyasu, B.P. Berman, and N. Mitchell. Optical character recognition and parsing of typeset mathematics. *Journal of Visual Communication and Image Representation*, 7(1):2–15, 1996.
- [49] Federation of American Scientists. *Summit on Educational Games*, 2006.
- [50] J.H. Flavell. Metacognitive aspects of problem solving. In L.B. Resnick, editor, *The nature of intelligence*, pages 231–236. Lawrence Erlbaum, Hillsdale, NJ, USA, 1976.
- [51] National Center for Education Statistics. International outcomes of learning in mathematics literacy and problem solving. <http://nces.ed.gov/pubs2005/2005003.pdf>, 2005.
- [52] C. Frankish, R. Hull, and P. Morgan. Recognition accuracy and user acceptance of pen interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 503–510, 1995.
- [53] R. Freedman. Atlas: A plan manager for mixed-initiative, multimodal dialogue. In *AAAI Workshop on Mixed-Initiative Intelligence*, 1999.
- [54] R.O. Frishkopf and L.D. Harmon. Machine reading of cursive script. In *Information Theory (4th London Symposium)*, pages 300–316, 1961.
- [55] S. Geisser. The predictive sample reuse method with applications. *Journal of the American Statistical Association*, 70(350):320–328, 1975.

- [56] R. Glaser. Components of a psychology of instruction: Toward a science of design. *Review of Educational Research*, 46:1–24, 1976.
- [57] D. Goldberg and A. Goodisman. Stylus user interfaces for manipulating text. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, pages 127–135, 1991.
- [58] D. Gopher and R. Braune. On the psychophysics of workload: Why bother with subjective measures? *Human Factors*, 26:519–532, 1984.
- [59] J.D. Gould, J. Conti, and T. Hovanyecz. Composing letters with a simulated listening type-writer. *Communications of the ACM*, 26(4):295–308, 1983.
- [60] A. Graesser, K.N. Moreno, J.C. Marineau, A.B. Adcock, A.M. Olney, and N.K. Person. Autotutor improves deep learning of computer literacy: Is it the dialog or the talking head? In *Proceedings of the International Conference on Artificial Intelligence in Education*, pages 47–54, 2003.
- [61] A.C. Graesser, N.K. Person, and J. Maglian. Collaborative dialog patterns in naturalistic one-on-one tutoring. *Applied Cognitive Psychologist*, 9:359–387, 1995.
- [62] I. Guyon and C. Warwick. Handwriting as computer interface. In R.A. Cole, J. Mariani, H. Uszkoreit, G.B. Varile, A. Zaenen, and A. Zampolli, editors, *Survey of the State of the Art in Human Language Technology*, pages 78–83. Cambridge University Press, Boston, MA, 1998.
- [63] R.G.M Hausmann and M.T.H. Chi. Can a computer interface support self-explaining? *Cognitive Technology*, 7:4–14, 2002.
- [64] D.F. Hsu and I. Taksa. Comparing rank and score combination methods for data fusion in information retrieval. *Informational Retrieval*, 8(3):449–480, 2005.
- [65] W. Hrst, J. Yang, and A. Waibel. Error repair in human handwriting: An intelligent user interface for automatic on-line handwriting recognition. In *Proceedings of the IEEE International Joint Symposia on Intelligence and Systems*, pages 389–395. IEEE, 1998.
- [66] R.M. Ingersoll. Teacher turnover and teacher shortages: An organizational analysis. *American Education Research Journal*, 37(3):499–534, 2001.
- [67] R.M. Ingersoll. Teacher shortages and education inequality. Technical Report Visiting Scholar Series, Spring 2005, Volume 1, National Education Association, 2005.
- [68] L. Jackson. Laptops, handhelds, or tablet pcs? Education World Technology: [http://www.education-world.com/a\\_tech/tech198.shtml](http://www.education-world.com/a_tech/tech198.shtml), 2004.

- [69] N. Kajler and N. Soiffer. A survey of user interfaces for computer algebra systems. *Journal of Symbolic Computation*, 25:127–159, 1998.
- [70] T. Kanahori, K. Tabata, W. Cong, F. Tamari, and M. Suzuki. On-line recognition of mathematical expressions using automatic rewriting method. In *Proceedings of the IEEE International Conference on Multimodal Interfaces*, pages 394–401, 2000.
- [71] J.F. Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems*, 2(1):26–41, 1984.
- [72] J. Kittler and F. Alkoot. Sum versus vote fusion in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):110–115, 2003.
- [73] A.L. Koerich, R. Sabourin, and C.Y. Suen. Large vocabulary off-line handwriting recognition: a survey. *Pattern Analysis Applications*, 6:97–121, 2003.
- [74] M.J. LaLomia. User acceptance of handwritten recognition accuracy. In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI)*, page 107, 1994.
- [75] J. Landay and B. Myers. Interactive sketching for the early stages of user interface design. In *Proceedings of the ACM Conference on Human Factors in Computing (CHI)*, pages 43–50. ACM Press, 1995.
- [76] J.J. LaViola. *Mathematical Sketching: A New Approach to Creating and Exploring Dynamic Illustrations*. PhD thesis, Brown University, 2005.
- [77] J.J. LaViola. An initial evaluation of a pen-based tool for creating dynamic mathematical illustrations. In *Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling, EG Workshop Series*, pages 157–164, 2006.
- [78] H.J. Lee and M.C. Lee. Understanding mathematical expressions using procedure-oriented transformation. *Pattern Recognition*, 27(3):447–457, 1994.
- [79] J.J. Lee and J.H. Kim. A unified network-based approach for on-line recognition of multi-lingual cursive handwritings. In *Proceedings of the Fifth International Workshop on Frontiers in Handwriting Recognition*, pages 393–397, 1996.
- [80] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [81] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 140:1–55, 1932.

- [82] D.J. Litman and S. Silliman. Itspoke: An intelligent tutoring spoken dialogue system. In *Proceedings of the Human Language Technology Conference: 4th Meeting of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL) (Companion Proceedings)*, pages 233–236, 2004.
- [83] R.H. Littin. Mathematical expression recognition: Parsing pen/tablet input in real-time using lr techniques. Master’s thesis, University of Waikato, Hamilton, New Zealand, 1995.
- [84] M. Liwicki and H. Bunke. Enhancing training data for handwriting recognition of whiteboard notes with samples from a different database. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 550–554, 2005.
- [85] J.L. Locke and F.S. Fehr. Subvocalization of heard or seen words prior to spoken or written recall. *American Journal of Psychology*, 85:63–68, 1972.
- [86] G. Lorette. Handwriting recognition or reading: What is the situation at the dawn of the 3rd millennium? *International journal on document analysis and recognition*, 2(1):2–12, 1999.
- [87] Liping Ma. *Knowing and Teaching Elementary Mathematics: Teachers’ Understanding of Fundamental Mathematics in China and the United States (Studies in Mathematical Thinking and Learning)*. Lawrence Erlbaum, Mahwah, NJ, USA, 1999.
- [88] I.S. MacKenzie and L. Chang. A performance comparison of two handwriting recognizers. *Interacting with Computers*, 11:283–297, 1999.
- [89] S. Madhvanath, D. Vijayasan, and T.M. Kadiresan. Lipitk: A generic toolkit for online handwriting recognition. Technical Report HPL-2006-115, HP Labs, 2006.
- [90] J. Mankoff and G. Abowd. Error correction techniques for handwriting, speech, and other ambiguous or error prone systems. Technical Report GIT-GVU-99-18, Georgia Institute of Technology, 1999.
- [91] N.E. Matsakis. Recognition of handwritten mathematical expressions. Master’s thesis, Massachusetts Institute of Technology, 1999.
- [92] R.E. Mayer. *Multimedia Learning*. Cambridge University Press, Boston, MA, 2001.
- [93] J. Van Merriënboer. *Training Complex Cognitive Skills: a Four-Component Instructional Design Model for Technical Training*. Educational Technology Publications, Englewood Cliffs, NJ, USA, 1997.
- [94] E.G. Miller and P.A. Viola. Ambiguity and constraint in mathematical expression recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 784–789. AAAI Press, 1998.

- [95] T. Mitchell. *Machine Learning*. WCB/McGraw-Hill, Boston, MA, USA, 1997.
- [96] J. Mostow, G. Aist, C. Huang, B. Junker, R. Kennedy, H. Lan, D. Latimer, R. O'Connor, R. Tassone, B. Tobin, and A. Wierman. 4-month evaluation of a learner-controlled reading tutor that listens. In V.M. Holland and F.P. Fisher, editors, *The Path of Speech Technologies in Computer Assisted Language Learning: From Research Toward Practice*, pages 201–219. Routledge, New York, NY, USA, 2008.
- [97] R. Narasimhan. Labeling schemata and syntactic descriptions of pictures. *Information Control*, 7(2):151–179, 1964.
- [98] R. O'Donnell and F.T. Eggemeier. Workload assessment methodology. In K. Boff, L. Kaufman, and J. Thomas, editors, *Handbook of perception and human performance*, pages 42.1–42.49. Wiley, NY, USA, 1986.
- [99] Pittsburgh Science of Learning Center (PSLC). Learnlab. <http://www.learnlab.org/>.
- [100] S. Oviatt. Mutual disambiguation of recognition errors in a multimodal architecture. In *Proceedings of the ACM Conference on Human Factors in Computing*, pages 576–583, 1999.
- [101] S. Oviatt and B. Adams. Designing and evaluating conversational interfaces with animated characters. In J. Cassell, J. Sullivan, S. Prevost, and E. Churchill, editors, *Embodied Conversational Agents*, pages 319–343. MIT Press, Cambridge, MA, USA, 2000.
- [102] S. Oviatt, A. Arthur, and J. Cohen. Quiet interfaces that help students think. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 191–200, 2006.
- [103] F. Paas. Training strategies for attaining transfer of problem-solving skill in statistics: A cognitive-load approach. *Journal of Educational Psychology*, 84:429–434, 1992.
- [104] F. Paas and J. Van Merriënboer. Variability of worked examples and transfer of geometry problem-solving skills: A cognitive-load approach. *Journal of Educational Psychology*, 86:122–133, 1994.
- [105] F. Paas, J.E. Tuovinen, H. Tabbers, and P.W.M. Van Gerven. Cognitive load measurement as a means to advance cognitive load theory. *Educational Psychology*, 38:63–71, 2003.
- [106] Microsoft: Education Pack. Equation write. <http://www.microsoft.com/windowsxp/downloads/tabletpc/educationpack/overview4.mspx>.
- [107] A.S. Palinscar and A.L. Brown. Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, 1:117–175, 1984.

- [108] R.D. Peacocke and D.H. Graf. An introduction to speech and speaker recognition. *IEEE Computer*, 23(8):26–33, 1990.
- [109] J. Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. In *Proceedings of the Conference of the Cognitive Science Society*, pages 329–334, 1985.
- [110] J. Platt, N. Cristianini, and J. Shawe-Taylor. Large margin DAGs for multiclass classification. *Advances in Neural Information Processing Systems*, 12:547–553, 2000.
- [111] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.
- [112] J.C. Read. Optimising text input for young children using computers for creative writing tasks. In *Proceedings of BCS British-HCI*. Springer Verlag, 2002.
- [113] J.C. Read. Using wizard of oz to evaluate mobile applications. In J. Lumsden, editor, *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, pages 802–813. IGI Global, Hershey, PA, USA, 2008.
- [114] J.C. Read, S. MacFarlane, and C. Casey. Oops! silly me! errors in a handwriting recognition-based text entry interface for children. In *Proceedings of the Nordic Conference on Human-Computer Interaction (NordiCHI)*, pages 35–40, New York, NY, USA, 2002. ACM Press.
- [115] J.C. Read, S. MacFarlane, and C. Casey. Pens behaving badly - usability of pens and graphics tablets for text entry with children. In *Adjunct Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 21–22. ACM Press, 2002.
- [116] J.C. Read, S. MacFarlane, and C. Casey. A comparison of two on-line handwriting recognition methods for unconstrained text entry by children. In *Proceedings of BCS British-HCI*, pages 29–32. Research Press International, Bristol, UK, 2003.
- [117] J.C. Read, S. MacFarlane, and C. Casey. Good enough for what?: acceptance of handwriting recognition errors by child users. In *Proceedings of the 2003 conference on Interaction design and children*, pages 155–155, New York, NY, USA, 2003. ACM Press.
- [118] J.C. Read, S. MacFarlane, and P. Gregory. Requirements for the design of a handwriting recognition based writing interface for children. In *Proceedings of the Conference on Interaction design and children*, pages 81–87, New York, NY, USA, 2004. ACM Press.
- [119] J.C. Read, S.J. MacFarlane, and C. Casey. Where's the 'm' on the keyboard, mummy? In *Womens' Engineering Society*, 2000.

- [120] J.C. Read, S.J. MacFarlane, and C. Casey. Can natural language recognition technologies be used to enhance the learning experience of young children? In *Computers and Learning*, 2001.
- [121] J.C. Read, S.J. MacFarlane, and C. Casey. Measuring the usability of text input methods for children. In *Proceedings of BCS British-HCI*, pages 559–572. Springer Verlag, 2001.
- [122] M.E. Renda and U. Straccia. Web metasearch: rank vs. score based rank aggregation methods. In *Proceedings of the ACM symposium on Applied computing*, pages 841–846, New York, NY, USA, 2003. ACM.
- [123] M. Ringenberg and K. VanLehn. Scaffolding problem solving with annotated worked-out examples to promote deep learning. In *Proceedings of the International Conference on Intelligent Tutoring Systems*, pages 625–634, 2006.
- [124] R.J.C.M. Salden, V. Aleven, A. Renkl, and R. Schwonke. Worked examples and tutored problem solving: Redundant or synergistic forms of support? In *Proceedings of the Annual Meeting of the Cognitive Science Society*, pages 589–594, 2008.
- [125] P.J. Santos, A.J. Baltzer, A.N. Badre, R.L. Henneman, and M.S. Miller. On handwriting recognition performance: Some experimental results. In *Proceedings of the Human Factors Society 36th Annual Meeting*, pages 283–287, 1992.
- [126] R.K. Sawyer, editor. *The Cambridge Handbook of the Learning Sciences*. Cambridge University Press, Cambridge, UK, 2005.
- [127] L. Schomaker. User-interface aspects in recognizing connected-cursive handwriting. In *Proceedings of the IEE Colloquium on Handwriting and Pen-based input*, pages 8/1–8/3, 1994.
- [128] R. Schwonke, A. Renkl, C. Krieg, J. Wittwer, V. Aleven, and R. Salden. The worked-example effect: is it just an artefact of lousy control conditions? *Computers in Human Behavior*, inpress.
- [129] M.M. Seastrom, K.J. Gruber, R. Henke, D.J. McGrath, and B.A. Cohen. Qualifications of the public school teacher workforce: Prevalence of out-of-field teaching 1987-88 to 1999-2000 (revised). Technical Report 2002603, National Center for Education Statistics, 2004.
- [130] K. Singley and J.R. Anderson. *The transfer of cognitive skill*. Harvard University Press, Cambridge, MA, USA, 1989.
- [131] S. Smithies, K. Novins, and J. Arvo. Equation entry and editing via handwriting and gesture recognition. *Behaviour and Information Technology*, 20:53–67, 2001.

- [132] J.C. Thomas Sr. A method for studying natural language dialog. Technical Report 5882, IBM Thomas J. Watson Research Center, 1976.
- [133] S.N. Srihari, C. Sung-Hyuk, H. Arora, and S. Lee. Individuality of handwriting: a validation study. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 106–109, 2001.
- [134] J. Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12:257–285, 1988.
- [135] J. Sweller and G.A. Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2:59–89, 1985.
- [136] H.J.M. Tabachneck, K.R. Koedinger, and M.J. Nathan. A cognitive analysis of the task demands of early algebra. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, pages 397–402, 1995.
- [137] E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical formulas in the e-chalk system. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*, pages 980–984. IEEE, 2003.
- [138] E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In J. Llads and Y.-B. Kwon, editors, *Graphics Recognition: Recent Advances and Perspectives; Lecture Notes in Computer Science 3088*, pages 329–340. Springer Berlin, Heidelberg, 2004.
- [139] C.C. Tappert, C.Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8):787–808, 1990.
- [140] J.G. Trafton and B.J. Reiser. The contributions of studying examples and solving problems to skill acquisition. In *Proceedings of the Cognitive Science Society*, pages 1017–1022, 1993.
- [141] K. VanLehn and R.M. Johns. Machine learning: Proceedings of the tenth annual conference. In P.E. Utgoff, editor, *Better learners use analogical problem solving sparingly*, pages 338–345. Morgan Kaufmann Publishers, 1993.
- [142] K. VanLehn, C. Lynch, L. Taylor, A. Weinstein, R. Shelby, K. Schulze, D. Treacy, and M. Wintersgill. Minimally invasive tutoring of complex physics problem solving. In *Proceedings of the Intelligent Tutoring Systems Conference*, pages 367–376, 2002.
- [143] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

- [144] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 1967.
- [145] E. Walker, B.M. McLaren, N. Rummel, and K.R. Koedinger. Who says three’s a crowd? using a cognitive tutor to support peer tutoring. In *Proceedings of the International Conference on Artificial Intelligence in Education (AIED)*, pages 399–408, 2007.
- [146] M. Witheiler. The tablet pc: An in-depth look with the fic slatevision. AnandTech.com: <http://www.anandtech.com/mobile/showdoc.aspx?i=1761>, 2002.
- [147] C. Wood. Technology and education. PC Magazine: <http://www.pc当地.com/article2/0,4149,15154,00.asp>, 2002.
- [148] L.S. Yaeger, B.J. Webb, and R.F. Lyon. Combining neural networks and context-drive search for online, printed handwriting recognition in the newton. *A.I. Magazine*, 19(1):73–89, 1998.
- [149] R. Zanibbi, D. Blostein, and J.R. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:1455–1467, 2002.
- [150] X. Zhang, J. Mostow, N.K. Duke, C. Trotochaud, J. Valeri, and A. Corbett. Mining free-form spoken responses to tutor prompts. In *Proceedings of the International Conference on Educational Data Mining (EDM)*, pages 234–241, 2008.

# **Appendices**

# **Appendix A**

## **Character Set Used**

This table gives all the math symbols and characters used throughout the studies and recognition tests in this dissertation.

0	1	2	3	4
5	6	7	8	9
x	y	a	b	c
+	-	=	(	)
-	/			

## Appendix B

### Study 1 Materials: The Math Input Study

#### B.1 Full List of Equations Used in the Math Input Study

$3x + 4 = 7$	$8x - 1 = 5$	$6x + 5 = 4$	$2x - 10 = 3$
$f(x) = x^4 - 31$	$6xy + \frac{4}{3}(x - y)$	$x\left(3 - \frac{2}{x}\right) = 0$	$y = 2^x 4x - 18 $
$\frac{y-4}{y^2-5y+4} = 9$	$\left \frac{2x}{3} - 17\right  \leq 3$	$\frac{y^2-8}{7(x+12)}$	$9(y + 4) = 4x_1 + x_2$
$y \leq  3x^{2/3} - 4x + 1 $	$y = (x - 7)^{-2}(x - 6)^3$	$f(x) = 5(y_2 - y_1)$	$\frac{2}{3}(x - 2) < \frac{4}{3}x + 8$
$\frac{4x_1+3x_2}{15(x-1)}$	$x^2 + y^2 - 5y - \frac{9}{4} = 0$	$\frac{1}{ x +1} - \frac{x^2}{2} \leq y$	$x^2y^2 - 4xy + 9x^2y + 18$
$f(x) = \frac{1}{18 - \left(\frac{5}{x^2}\right)}$	$\frac{(x+5)(x-5)}{3x^2+9} \geq 0$	$x(t_n) = x^2 + 8(t_{n-1} + 2)$	$\frac{2x^2}{x+1} <  y  < \frac{x+5}{2}$
$\sum [c_k^2 - 2c_k - 10]$	$\int \frac{8x^2}{(8x-2)^2} dx$	$\sqrt{\frac{4(1-3w^2)}{w^2+1}}$	$V = \frac{1}{3}\pi \left(\frac{y^2}{2}\right)^2$
$\int \left(\frac{u^7}{2} - \frac{\pi}{u^5}\right) du$	$s = \frac{\sqrt{t_1}-1}{\sqrt{t_2}+1}$	$\sum k^3 \approx \left(\frac{n(n+1)}{2}\right)^2$	$\int \frac{3\sqrt{7-3y^2}}{8y^2} dy$

## B.2 Math Symbols Test

### PRE-QUESTIONNAIRE

**Mathematics Symbols:** Please match the symbol on the left with its description on the right by writing the number of the description next to the symbol. If you haven't seen a symbol before, just leave it blank.

- |   |               |                            |
|---|---------------|----------------------------|
| — | $y^2$         | 1. integral                |
| — | $\leq$        | 2. variable                |
| — | $\approx$     | 3. fraction                |
| — | $\infty$      | 4. exponent                |
| — | $\sum$        | 5. pi                      |
| — | $ $           | 6. absolute value          |
| — | $\sqrt{x}$    | 7. expression              |
| — | $x$           | 8. infinity                |
| — | $\frac{1}{8}$ | 9. summation               |
| — | $\pi$         | 10. square root            |
| — | $x = 10y + 2$ | 11. approximately equal to |
| — | $4x - 9$      | 12. less than or equal to  |
| — | $ x $         | 13. equation               |

## B.3 Pre-Session Questionnaire

### PRE-QUESTIONNAIRE

**Input Methods:** What is the most frequently you've ever used (past or present) each of the following methods of computer input? Circle the appropriate number.

1. Keyboard and mouse

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

2. Speech dictation or speech commands

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

3. Graffiti® or other special-purpose pen-based alphabets

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

4. Pen-based handwriting recognition

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

5. Eye-tracking devices

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

6. Puff-and-sip devices

1-----	2-----	3-----	4-----	5-----	N/A
Use(d) it every day	Use(d) it fairly often	Use(d) it once in a while	Use(d) it rarely	Never used it, but have heard of it	Never heard of it

### PRE-QUESTIONNAIRE

**Input for Math:** Assuming the technology is near perfect, please rate how good you think the following input methods would be for mathematics equations, expressions, and/or symbols, by circling the appropriate number.

1. Keyboard and mouse

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

2. Speech dictation or speech commands

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

3. Graffiti® or other special-purpose pen-based alphabets

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

4. Pen-based handwriting recognition

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

5. Eye-tracking devices

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

6. Puff-and-sip devices

1.....	2.....	3.....	4.....	5.....	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

## B.4 Post-Session Questionnaire

### POST-QUESTIONNAIRE

**Input for Math:** Assuming the technology is near perfect, please rate how good you think the following input methods would be for mathematics equations, expressions, and/or symbols, by circling the appropriate number.

1. Keyboard and mouse

1-----	2-----	3-----	4-----	5-----	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

2. Speech dictation or speech commands

1-----	2-----	3-----	4-----	5-----	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

3. Pen-based handwriting recognition

1-----	2-----	3-----	4-----	5-----	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

4. Speech recognition PLUS pen-based handwriting recognition

1-----	2-----	3-----	4-----	5-----	N/A
Perfectly suited and natural	Somewhat suited and natural	Neither suited nor unsuited	Somewhat unsuited or unnatural	Not suited or natural at all	Never heard of it

## B.5 Demographics Questionnaire

**Demographic Data***Reminder: All information is anonymous and confidential!*

Native English Speaker	<input type="checkbox"/> Yes <input type="checkbox"/> No – Native Language: _____
Age	_____ (write age here) <input type="checkbox"/> I prefer not to say.
Gender	<input type="checkbox"/> Male <input type="checkbox"/> Female
Ethnicity	<input type="checkbox"/> African American <input type="checkbox"/> American Indian <input type="checkbox"/> Asian/Asian American <input type="checkbox"/> Caucasian <input type="checkbox"/> Other: _____ <input type="checkbox"/> I prefer not to say.
Highest Level of Education	<input type="checkbox"/> High School or equivalent <input type="checkbox"/> Some College <input type="checkbox"/> College Degree <input type="checkbox"/> Graduate Degree
IF IN SCHOOL: Current grade level	<input type="checkbox"/> College Freshman <input type="checkbox"/> College Sophomore <input type="checkbox"/> College Junior <input type="checkbox"/> College Senior <input type="checkbox"/> Graduate Student <input type="checkbox"/> Other: _____
Major/Area of Specialty	_____
IF EMPLOYED: Current Job	_____
Microsoft Equation Editor:	<input type="checkbox"/> Know it well <input type="checkbox"/> Know it a little <input type="checkbox"/> Don't really know it, never heard of it

# Appendix C

## Study 2 Materials: The Lab Learning Study

### C.1 Full List of Equations Used in the Lab Learning Study

#### C.1.1 Equations Used During Copying Phase

Typing			
1	$x + 19 = 60 + 2y$	11	$a + 21b - 8 = 3$
2	$5y = 72(3 + x)$	12	$\frac{(6c+9x)}{4y} = 7a$
3	$\frac{a}{38} = 4(8a - 1)$	13	$50 = 4(a + 7)$
4	$-b = \frac{29}{6c}$	14	$\frac{10b+2}{86} = 39$
5	$\frac{78}{c} = 53 + 5b$	15	$70 = \frac{(32c-98x)}{61}$
6	$4x - 10 = 6y + 23$	16	$4y - 5 = 67b - 2$
7	$7a - 8 = 15$	17	$-5c = \frac{4(90x+4)}{y}$
8	$\frac{b}{46} + 90 = 35$	18	$31 = a + 8b$
9	$12 = 6c - 45x$	19	$5c - 7 = x - y$
10	$\frac{y+70}{8} = 9$	20	$4a + b = 90c$

Handwriting			
1	$2x + 20 = 3$	24	$3a + b - c = 185$
2	$6y - 26 = 9$	25	$\frac{9}{4}a - \frac{4}{9}b = 100$
3	$12a + 14b = 165$	26	$9(4x - 2y + 1) = 7a + c$
4	$5y = 8x + 17$	27	$\frac{4-a}{7} = \frac{8c}{5}$
5	$\frac{4}{3}x - 12 = 85$	28	$6(5 + c) = 8(7 - a)$
6	$\frac{x+6}{62+8y} = 36$	29	$200 = 7x + 12$
7	$\frac{28-2y}{9-(4y+c)} = 6y + 4$	30	$x = 46(x - y)$
8	$4(a + b) = 9c$	31	$\frac{3}{7}(x - 10) = y$
9	$(4 + c)(2 - c) = b$	32	$-8x + 19 = -7$
10	$\frac{2}{3}y + \frac{5}{8}y = y - 3$	33	$b = \frac{-4+2c}{x}$
11	$\frac{5}{9}(a - b) = \frac{7}{5}c$	34	$8y = 20x - 4$
12	$\frac{6}{8}a + 5 = \frac{4}{3}a + 10$	35	$\frac{9(x-y)}{17} = 50$
13	$\frac{x+5}{9(6-68y)} = a - b$	36	$x - \frac{3}{7} = 10y$
14	$12x - x = 9$	37	$7b - 36 = 4c + 3$
15	$90b + 100c = 5000$	38	$280c + b - 35 = 55$
16	$y = ax + b$	39	$60a - 33b = 99$
17	$x = 10y - 6$	40	$3c - 7a = 6b$
18	$72x = 4(y - 8)$	41	$88 + c - 3b = 6a$
19	$5x + 1 = \frac{99}{x-4}$	42	$55 = 87a + c$
20	$\frac{(9x+1)(6+x)}{80} = y$	43	$\frac{(a+c-b)}{93} = 56$
21	$b + 4 = c - 8$	44	$63c = a(b + 7)$
22	$\frac{12}{7}(x - y) = \frac{4}{3}$	45	$33 + 9a - b + 67c = 98$
23	$(a - 7)(b - 5) = 2c$		

Speech			
1	$x + 32 = 89 + 7y$	11	$10a + 38b + 74 = 89$
2	$4y = 16(3 + x)$	12	$\frac{(50c - 62x)}{y} = 2a$
3	$\frac{a}{50} = 7(4a - 6)$	13	$30 = 6(a + 5)$
4	$-b = \frac{13}{8c}$	14	$\frac{4b+12}{86} = 90$
5	$\frac{46}{c} = 91 + 9b$	15	$11 = \frac{(35c+5x)}{7}$
6	$5x - 82 = 7y + 25$	16	$4y + 25 = 3b - 8$
7	$19a - 84 = 17$	17	$-6c = \frac{10(22x - 30)}{y}$
8	$\frac{b}{300} + 65 = 23$	18	$47 = a - 5b$
9	$78 = 9c - 4x$	19	$6c - 4 = x - 9y$
10	$\frac{y+6}{9} = 7$	20	$7a - 9b = 80c$

### C.1.2 Examples and Problems Used During Learning Phase

#### Copying (Examples)

$$\begin{array}{r} x + 4 = 9 \\ -4 \quad -4 \\ \hline x = 5 \end{array}$$

$$\begin{array}{r} 8x = 16 \\ \frac{8x}{8} = \frac{16}{8} \\ \cancel{\frac{8}{8}}x = 2 \\ x = 2 \end{array}$$

$$\begin{array}{r} \frac{x}{5} = 4 \\ 5 \cdot \frac{x}{5} = 4 \cdot 5 \\ \cancel{5} \cdot \frac{x}{\cancel{5}} = 20 \\ x = 20 \end{array}$$

$$\begin{array}{r} -x = 5 \\ -1 \cdot x = 5 \\ \frac{-1 \cdot x}{-1} = \frac{5}{-1} \\ \cancel{-1} \cdot \frac{x}{\cancel{-1}} = -5 \\ x = -5 \end{array}$$

#### Solving (Problems)

$$\begin{array}{r} x + 3 = 5 \\ -3 \quad -3 \\ \hline x = 2 \end{array}$$

$$\begin{array}{r} 3x = 15 \\ \frac{3x}{3} = \frac{15}{3} \\ \cancel{\frac{3}{3}}x = 5 \\ x = 5 \end{array}$$

$$\begin{array}{r} \frac{x}{7} = 6 \\ 7 \cdot \frac{x}{7} = 6 \cdot 7 \\ \cancel{7} \cdot \frac{x}{\cancel{7}} = 42 \\ x = 42 \end{array}$$

$$\begin{array}{r} -x = 8 \\ -1 \cdot x = 8 \\ \frac{-1 \cdot x}{-1} = \frac{8}{-1} \\ \cancel{-1} \cdot \frac{x}{\cancel{-1}} = -8 \\ x = -8 \end{array}$$

$$\begin{array}{ll}
 \frac{56}{x} = 7 & \frac{36}{x} = 4 \\
 x \cdot \frac{56}{x} = 7 \cdot x & x \cdot \frac{36}{x} = 4 \cdot x \\
 x \cdot \frac{56}{x} = 7x & x \cdot \frac{36}{x} = 4x \\
 56 = 7x & 36 = 4x \\
 \frac{56}{7} = \frac{7x}{7} & \frac{36}{4} = \frac{4x}{4} \\
 8 = \frac{7x}{7} & 9 = \frac{4x}{4} \\
 8 = x & 9 = x
 \end{array}$$

$$\begin{array}{ll}
 4x + 5 = 13 & 5x + 2 = 17 \\
 -5 \quad -5 & -2 \quad -2 \\
 4x = 8 & 5x = 15 \\
 \frac{4x}{4} = \frac{8}{4} & \frac{5x}{5} = \frac{15}{5} \\
 \frac{4x}{4} = 2 & \frac{5x}{5} = 3 \\
 x = 2 & x = 3
 \end{array}$$

$$\begin{array}{ll}
 \frac{x}{4} + 5 = 8 & \frac{x}{5} + 4 = 7 \\
 -5 \quad -5 & -4 \quad -4 \\
 \frac{x}{4} = 3 & \frac{x}{5} = 3 \\
 4 \cdot \frac{x}{4} = 3 \cdot 4 & 5 \cdot \frac{x}{5} = 3 \cdot 5 \\
 4 \cdot \frac{x}{4} = 12 & 5 \cdot \frac{x}{5} = 15 \\
 x = 12 & x = 15
 \end{array}$$

$$3x + 4x = 21$$

$$(3 + 4)x = 21$$

$$7x = 21$$

$$\frac{7x}{7} = \frac{21}{7}$$

$$\frac{7x}{7} = 3$$

$$x = 3$$

$$5x + 3x = 24$$

$$(5 + 3)x = 24$$

$$8x = 24$$

$$\frac{8x}{8} = \frac{24}{8}$$

$$\frac{8x}{8} = 3$$

$$x = 3$$

$$\frac{x + 5}{6} = 7$$

$$6 \cdot \frac{x + 5}{6} = 7 \cdot 6$$

$$6 \cdot \frac{x + 5}{6} = 42$$

$$x + 5 = 42$$

$$\begin{array}{r} -5 \\ -5 \end{array}$$

$$x = 37$$

$$\frac{x + 4}{3} = 8$$

$$3 \cdot \frac{x + 4}{3} = 8 \cdot 3$$

$$3 \cdot \frac{x + 4}{3} = 24$$

$$x + 4 = 24$$

$$\begin{array}{r} -4 \\ -4 \end{array}$$

$$x = 20$$

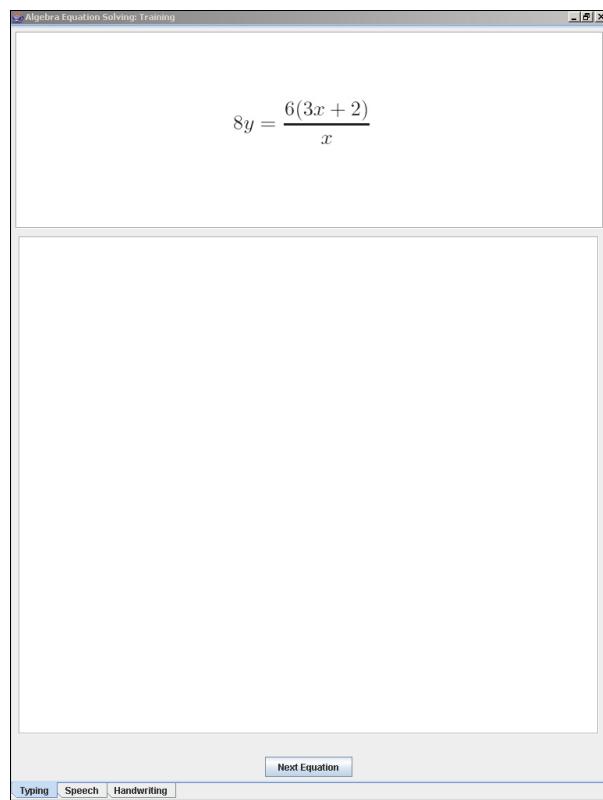
## C.2 Pre-Session Training Handout

### COPYING

There are 3 different ways we are going to do this. You will do all 3 ways but you might do them in a different order than shown here.

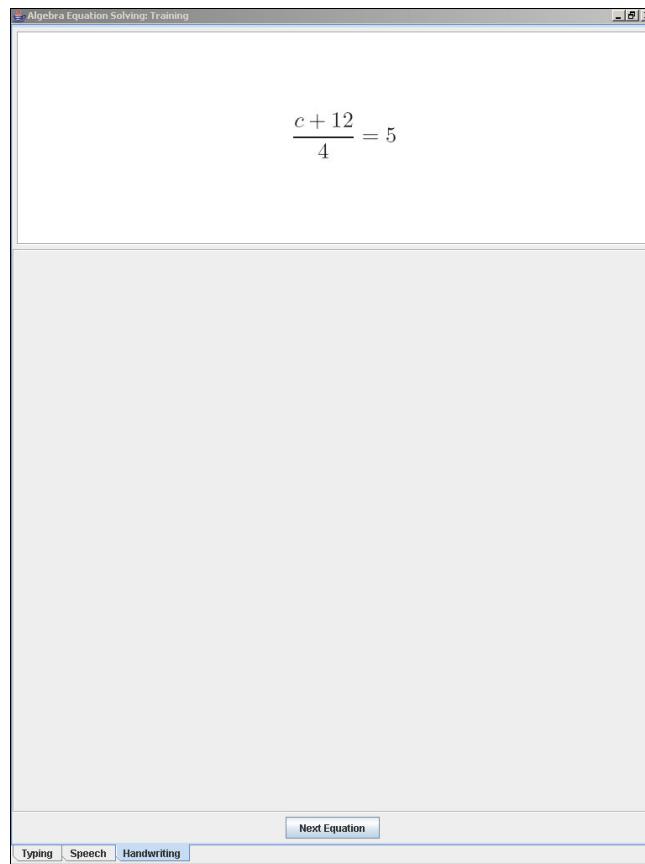
#### Keyboard

The top section shows the current equation. Type the equation into the text box on the bottom half of the screen. Use whatever keys are on the keyboard and skip lines and use spaces if you want to make it look like what is shown, or use special keys on the keyboard like ‘/’ to show things like division. Just type it out however feels most natural for you. When you are finished, click the “Next Equation” button at the bottom of the screen to continue.



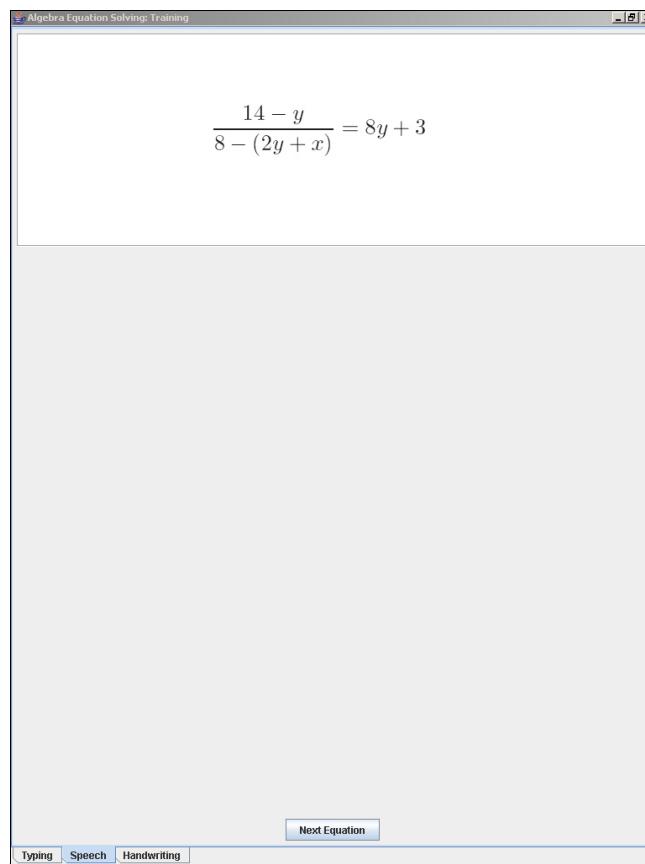
**Handwriting**

The top section shows the current equation. Write the equation into the white box at the bottom half of the screen using the pen. Just write it out like you would on paper. When you are finished, click the “Next Equation” button at the bottom of the screen to continue.



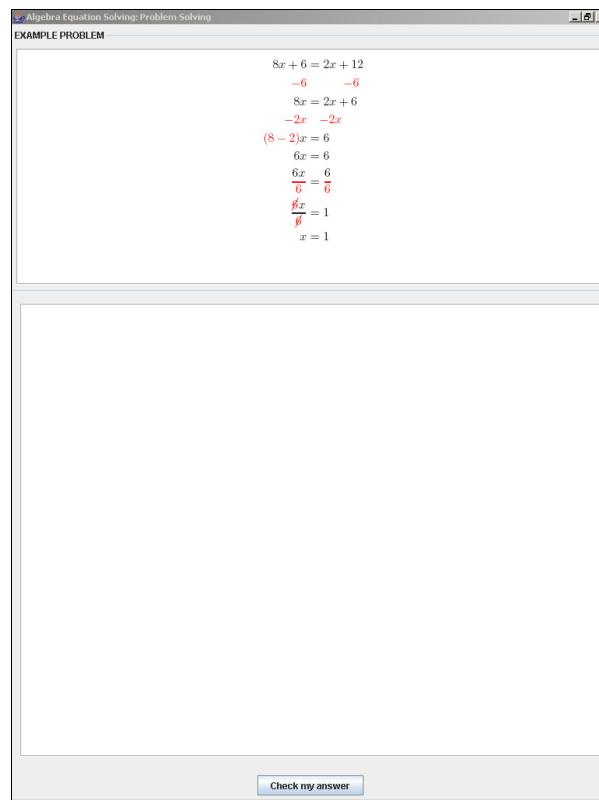
**Handwriting + Speech**

The top section shows the current equation. Write the equation into the white box at the bottom half of the screen using the pen. Just write it out like you would on paper. In addition, say the equation out loud into the microphone, similar to what a teacher would do when writing on the blackboard. Just write and say it however feels most comfortable and natural for you. When you are finished, click the “Next Equation” button at the bottom of the screen to continue.



**LEARNING****Examples**

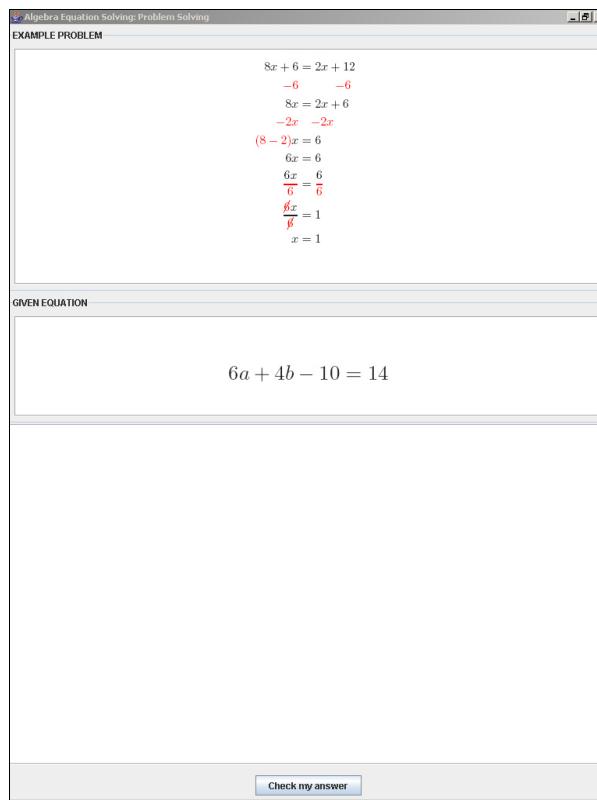
The screen you see is divided into two sections. First you will see an example problem on top which you will copy out step by step, just like it appears on the upper half of the screen. Copy it in the space on the bottom half of the screen in whatever way you have been assigned. You can skip any step that is just showing cancellations from the step before, like in line 8 below. Try to understand why you can take each step as you copy it down. When you have finished copying down the example, click the “Check my answer” button at the bottom of the screen and the computer will tell you if you copied it correctly or not.



**Problems**

After you finish the example, it will stay on the screen and an equation for you to solve on your own will appear below it. The new equation will be similar to the example you just copied.

Solve this problem step by step using the example as a guide. When you think you have the answer, click the "Check My Answer" button and the computer will tell you if you got the right answer or not. If you are not sure of what you are doing, you can ask the computer to check your last step by clicking the same button, if you are really stuck. The computer will take a few seconds to analyze your answer and may pause for a bit. If you didn't get the problem right, then you will get to try again. If you give an incorrect answer three times, then the computer will give you the answer for you to copy down and you will go to the next problem.



You will be doing one of the three methods below:

**Keyboard**

Use the keyboard to copy examples and solve problems just like the copying section before.

**Handwriting**

Use the pen to copy examples and solve problems just like the copying section before.

**Pen + Speech**

Use the pen, while speaking, to copy examples and solve problems just like the copying section before.

**C.3 Test A**

1.  $x + 5 = 8$
2.  $6x = 12$
3.  $\frac{x}{4} = 8$
4.  $-x = 3$
5.  $\frac{24}{x} = 8$
6.  $2x + 4 = 10$
7.  $\frac{x}{2} + 3 = 8$
8.  $2x + 3x = 15$
9.  $\frac{x+8}{5} = 4$
10.  $8x - 3x + 6 = 36$
11.  $\frac{3x+9x}{x} = 4x$
12.  $7x + 4 = 3x + 12$
13.  $3(x + 4) = 18$
14.  $\frac{2x+8}{3} = 4$
15.  $\frac{3x+5x}{4} = 4$

**C.4 Test B**

1.  $x + 2 = 10$
2.  $2x = 10$
3.  $\frac{x}{3} = 4$
4.  $-x = 6$
5.  $\frac{30}{x} = 6$
6.  $3x + 8 = 17$
7.  $\frac{x}{3} + 7 = 9$
8.  $3x + 2x = 20$
9.  $\frac{x+6}{8} = 4$
10.  $6x - 3x + 8 = 14$
11.  $\frac{7x+5x}{x} = 6x$
12.  $4x + 3 = 2x + 7$
13.  $3(x + 6) = 24$
14.  $\frac{3x+4}{5} = 4$
15.  $\frac{5x-2x}{5} = 3$

## C.5 Post-Session Questionnaire

Participant \_\_\_\_\_

### Questionnaire

### COMPUTER TOOLS

Other than today, have you ever used TYPING and the MOUSE to do stuff on the computer?

- Yes, all the time
- Yes, a few times
- Yes, once
- Not at all
- I don't know

Other than today, have you ever used HANDWRITING to do stuff on the computer, including on special handheld devices?

- Yes, all the time
- Yes, a few times
- Yes, once
- Not at all
- I don't know

Other than today, have you ever used SPEECH to do stuff on the computer?

- Yes, all the time
- Yes, a few times
- Yes, once
- Not at all
- I don't know

In school or at home, what method(s) do you use, if you use computers in math class?

- We don't use computers for math class.
- Typing, mouse
- Handwriting
- Speech
- Other: \_\_\_\_\_

<p>Participant _____</p> <p><b>Questionnaire</b></p> <p>What method did you use today during the problem-solving part (not the copying part)?</p> <p><input type="checkbox"/> Typing, mouse <input type="checkbox"/> Handwriting <input type="checkbox"/> Both handwriting and speech</p> <p>Did you like the method you used?</p> <p><input type="checkbox"/> Yes <input type="checkbox"/> No</p> <p>Why or why not? _____ _____ _____</p> <p>Would you want to use this method in the future if you had a choice?</p> <p><input type="checkbox"/> Yes <input type="checkbox"/> No</p> <p>Why or why not? _____ _____ _____</p> <p>During the copying part, which was your favorite method?</p> <p><input type="checkbox"/> Typing, mouse <input type="checkbox"/> Handwriting <input type="checkbox"/> Speech</p> <p>Can you say why you liked this method best?</p> <p>_____ _____ _____</p>	<p><b>LIKES and DISLIKES</b></p>
---	----------------------------------

<p><b>Questionnaire</b></p> <p>What was your last report card grade in math?</p> <p><input type="checkbox"/> A (100-90) <input type="checkbox"/> B (90-80) <input type="checkbox"/> C (70-80) <input type="checkbox"/> D (60-70) <input type="checkbox"/> F (below 60)</p> <p>What is your usual grade in math on assignments and tests?</p> <p><input type="checkbox"/> A (100-90) <input type="checkbox"/> B (90-80) <input type="checkbox"/> C (70-80) <input type="checkbox"/> D (60-70) <input type="checkbox"/> F (below 60)</p> <p>What was the last math class you took or subject you were studying?</p> <p><input type="checkbox"/> Pre-algebra <input type="checkbox"/> Algebra <input type="checkbox"/> Geometry <input type="checkbox"/> Other: _____</p> <p>What school do you go to? _____</p> <p><input type="checkbox"/> Private <input type="checkbox"/> Public</p> <p>What grade are you going into this fall?</p> <p><input type="checkbox"/> 6th <input type="checkbox"/> 7th <input type="checkbox"/> 8th <input type="checkbox"/> Other: _____</p> <p>How old are you? _____ years old</p> <p>Are you male or female?</p> <p><input type="checkbox"/> Male <input type="checkbox"/> Female</p> <p>What is your ethnicity?</p> <p><input type="checkbox"/> African American <input type="checkbox"/> Native American/American Indian <input type="checkbox"/> Asian/Asian American <input type="checkbox"/> Caucasian <input type="checkbox"/> Other: _____ <input type="checkbox"/> I prefer not to say.</p>	<p>Participant _____</p> <p><b>ABOUT YOU</b></p>
---	--

## **Appendix D**

### **Study 3 Materials: The Cognitive Tutor Study**

#### **D.1 Demographics**

This demographic information was filled in by the teachers at his/her option for each student:

1. School Name
2. Teacher Name
3. Class Period
4. Grade Level
5. Unit in Tutor Before Study
6. Average Grade in Math to Date
7. Attendance during Study

## D.2 Test A

1. Solve for  $y$ :

$$7 + 5y = 2$$

2. Solve for  $x$ :

$$\frac{x}{-5} + 8 = -2$$

3. Solve for  $x$ :

$$4x + 8x = 9$$

4. Solve for  $y$  and explain your steps:

$$3y + (-8) = 5 + 4y$$

5. Solve for  $x$ :

$$6 + (-7y) = -1 + 2y$$

6. Solve for  $x$  and explain your steps:

$$-3.2x + 4.1 = 12.8 + 6.4x$$

7. Solve for  $x$ :

$$-2.8 + 5.1x = 11.8x + (-3.4)$$

8. Solve for  $y$  and explain your steps:

$$y + 12.3 = -4.1y + (-9.2)$$

9. Solve for  $x$ :

$$\frac{4x+12}{3} = -9x$$

10. Solve for  $x$ :

$$14.1x + 19.1 + (-8.2x) = -6.2x + 12.8$$

### D.3 Test B

1. Solve for  $y$ :

$$9 + 2y = 4$$

2. Solve for  $x$ :

$$\frac{x}{-4} + 6 = -3$$

3. Solve for  $x$ :

$$2x + 5x = 6$$

4. Solve for  $y$  and explain your steps:

$$4y + (-7) = 5 + 6y$$

5. Solve for  $y$ :

$$7 + (-8y) = -3 + 2y$$

6. Solve for  $x$  and explain your steps:

$$-1.8x + 6.3 = 9.7 + 5.1x$$

7. Solve for  $x$ :

$$-13.8 + 6.3x = 21.1x + (-5.9)$$

8. Solve for  $y$  and explain your steps:

$$y + 8.2 = -3.4y + (-11.6)$$

9. Solve for  $x$ :

$$\frac{3x+8}{7} = -12x$$

10. Solve for  $x$ :

$$12.6x + 21.1 + (-7.4x) = -9.2x + 15.8$$

## D.4 Test C

1. Solve for  $y$ :

$$6 + 8y = 1$$

2. Solve for  $x$ :

$$\frac{x}{-3} + 4 = -7$$

3. Solve for  $x$ :

$$3x + 2x = 12$$

4. Solve for  $y$  and explain your steps:

$$6y + (-3) = 2 + 2y$$

5. Solve for  $y$ :

$$8 + (-5y) = -1 + 4y$$

6. Solve for  $x$  and explain your steps:

$$-4.8x + 3.2 = 8.9 + 2.5x$$

7. Solve for  $x$ :

$$-7.4 + 17.3x = 8.1x + (-24.6)$$

8. Solve for  $y$  and explain your steps:

$$y + 9.9 = -6.3y + (-4.2)$$

9. Solve for  $x$ :

$$\frac{7x+5}{2} = -3x$$

10. Solve for  $x$ :

$$16.3x + 18.4 + (-5.9x) = -7.5x + 17.3$$

## D.5 Cognitive Load Questionnaire

### Unit 18 Review of Linear Equations – Mental Effort Survey

Please circle the answer that best represents how you felt about the amount of mental effort you spent while working with in Unit 18.

How much mental effort during the study?

**In solving or studying Unit 18, I invested, spent, or felt**

1. very, very low mental effort
2. very low mental effort
3. low mental effort
4. rather low mental effort
5. neither low nor high mental effort
6. rather high mental effort
7. high mental effort
8. very high mental effort
9. very, very high mental effort

Was the mental effort the normal amount?

**The mental effort I spent or felt was**

1. very, very much less than normal tutor use
2. very much less than normal tutor use
3. much less than normal tutor use
4. a little less than normal tutor use
5. neither more nor less than normal tutor use
6. a little more than normal tutor use
7. much more than normal tutor use
8. very much more than normal tutor use
9. very, very much more than normal tutor use

Where did the mental effort come from?

**The mental effort I spent or felt came from**

1. mostly thinking about the problems
2. mostly trying to use the computer interface
3. about the same amount for both

## Appendix E

# Set of Test Problems Used in Recognition Experiments

### Problem 1

$\frac{36}{x} = 4$	$\frac{36}{x} = 4$	
*x * x	*x * x	
$36 = 4x$	$56 = 7x$	wrong
/4 /4	/7 /7	wrong
$9 = x$	$8 = x$	wrong

### Problem 2

$5x + 3x = 24$	$5x + 3x = 24$	
$x(5 + 3) = 24$	$x(5 + 3) = 24$	
$8x = 24$	$8x = 24$	
/8 /8	/3 /3	wrong
$x = 3$	$x = 8$	wrong

**Problem 3**

$$\begin{array}{r} 8x = 16 \\ /8 \quad /8 \\ x = 2 \end{array}$$

$$\begin{array}{lll} 8x = 16 & & \\ /8 \quad /8 & & \text{wrong} \\ x = 12 & & \text{wrong} \end{array}$$

**Problem 4**

$$\begin{array}{r} \frac{x}{5} + 4 = 7 \\ -4 \quad -4 \\ \frac{x}{5} = 3 \\ *5 \quad *5 \\ x = 15 \end{array}$$

$$\begin{array}{lll} \frac{x}{5} + 4 = 7 & & \\ -4 \quad -4 & & \text{wrong} \\ \frac{x}{5} = 7 & & \text{wrong} \\ *5 \quad *5 & & \text{wrong} \\ x = 35 & & \text{wrong} \end{array}$$

**Problem 5**

$$\begin{array}{r} -x = 5 \\ / -1 \quad / -1 \\ x = -5 \end{array}$$

$$\begin{array}{lll} -x = 5 & & \\ / -1 = / -1 & & \\ x = 5 & & \text{wrong} \end{array}$$

**Problem 6**

$$\begin{array}{r} -x = 8 \\ / -1 \quad / -1 \\ x = -8 \end{array}$$

$$\begin{array}{lll} -x = 8 & & \\ /8 \quad /8 & & \text{wrong} \\ x = 8 & & \text{wrong} \end{array}$$