Educational Systems for Maximizing Learning Online and in the Classroom

Eleanor O'Rourke

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

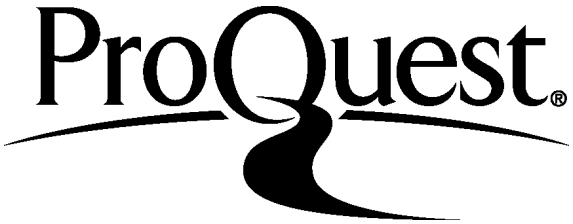
Reading Committee:

Zoran Popović, Chair

Carol S. Dweck

Daniel S. Weld

Program Authorized to Offer Degree:
Computer Science and Engineering



University of Washington

Abstract

Educational Systems for Maximizing Learning
Online and in the Classroom

Eleanor O'Rourke

Chair of the Supervisory Committee:
Professor Zoran Popović
Computer Science and Engineering

Over the past few decades, technology has evolved to a point where there is great potential for educational system to transform how we teach and learn. This has led to many exciting advances in educational technology. Platforms such as Khan Academy and Coursera provide millions of students with free access to online educational content. Intelligent tutoring systems personalize instruction and give intelligent feedback in response to any student mistake. Educational games such as Dragon Box and Refraction have a seemingly endless capacity to engage students in learning activities. However despite their potential, these systems have not fundamentally changed the state of education in the United States. Recent data shows that only one in four high school graduates are prepared for college, and student test scores have remained stagnant since the 1970s. While many factors influence the effectiveness of educational technology, one central problem is that computer scientists are trying to reinvent the wheel, rather than learning from prior research in education and the learning sciences.

In this dissertation, I combine techniques from computer science and the learning sciences to develop novel technical systems grounded in learning theory that tackle core challenges in education. My collaborators and I apply this approach to address challenges in three key areas. First, we rethink the design of motivational incentives for educational games by considering psychology research on the importance of student mindsets. Next, we develop new

technical approaches to facilitate the automatic generation of empirically validated instructional scaffolding. Finally, we study the behavioral impact of a novel formative assessment system to better understand the integration of technology into real-world classrooms.

Taken together, this research demonstrates the value of leveraging findings from the learning sciences literature in the design of new educational technologies. The systems presented in this dissertation can have an immediate impact on learning: they have been used by over 100,000 students online, adopted by companies, and used in classrooms throughout the Seattle area. Most importantly, this research takes an important step towards the development of personalized data-driven systems that meet the unique needs of each student, teacher, and classroom.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

This dissertation is the culmination of many years of work that would not have been possible without the guidance, support, and collaboration of many wonderful people.

First, I am incredibly grateful to my advisor, Zoran Popović, for his guidance, advice, constrictive criticism, and encouragement throughout my graduate career. He taught me how to select important and exciting problems, how to approach research systematically, and how to stand up for myself. I am also very grateful to my collaborator and committee member Carol Dweck for her positivity, enthusiasm, and interest in my career. She inspired me to look beyond computer science and become involved in the educational psychology. I would also like to thank my committee member Min Li for helping me communicate my work to education researchers, and committee member Dan Weld for his feedback and support. Finally, I would like to thank Hank Levy for all of his encouragement during the job search process, and Lindsay Michimoto for helping me find my place in graduate school.

I am hugely grateful to all of my collaborators on the various parts of this dissertation: Erik Andersen, Christy Ballweber, Eric Butler, Yvonne Chen, Armando Díaz Tolentino, Sumit Gulwani, Kyla Haimovitz, and Erin Peach. I would also like to thank past and current Center for Game Science (CGS) graduate students for their support, insights, and friendship: Rahul Banerjee, Aaron Bauer, Seth Cooper, Dun-Yu Hsiao, Alexander Jaffe, Seong Jae Lee, Yun-en Liu, Travis Mandel, Oleksandr Polozov, Zuoming Shi, Adam Smith, and Kathleen Tuite. I would also like to thank the CGS staff who contributed to this work: Ourania Abell, Nova Barlow, Matthew Burns, Craig Conner, Kate Fisher, Ric Gray, Mostafa Hedayat, Tim Pavlik, Rich Snider, Cullen Su, Roy Szeto, and Aaron Whiting. Finally, I am hugely indebted to the wonderful CGS artists Brian Britigan, Barbara Krug, and Marianne

Lee for their creativity and hard work. I am particularly grateful to Marianne and Barb for all of the hours they put in to brainstorming, designing, and user testing the brain points art. The brain points intervention would not have been possible without you.

Throughout graduate school, I have been incredibly lucky to be surrounded by an outstanding group of friends. First and foremost, I want to thank Nicki Dell and Franzi Roesner for their love and unwavering support over the last seven years. You saw me through all the ups and downs, shared invaluable advice, and helped me balance work with awesome adventures. I couldn't have done it without you. I am also so grateful to all of my Seattle friends for cheering me on and making this place my home: Greg Akselrod, Tony Fader, Pete Hornyack, Brenna Nowak, Chris Roby, Erin Roby and Todd Schiller.

I want to thank my incredible parents, Marylynn Salmon and Joe O'Rourke, for supporting me through all 23 years of my formal education. Thank you for believing in me and teaching me to love intellectual work. I am also grateful to my brother Russell for his sense of humor and unconditional support, and for making sure I got to Seattle in one piece. Thanks to my in-laws Sandy Ahlum, Lent Johnson, Emily Johnson and Vijay Ramachandran for their love and support through this process. And finally, thank you to my husband Cliff Johnson, for supporting my career and being there for me every step of the way. Your enthusiasm and excitement in my work meant more to me than you can imagine. Thank you for believing in me.

The work presented in this dissertation was supported by the Office of Naval Research grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the Hewlett Foundation grant 2012-8161, and gifts from Adobe and Microsoft.

DEDICATION

To my husband Cliff, for encouraging me to live a fulfilling life.

Chapter 1

INTRODUCTION

One of the great challenges facing society today is to teach students the skills required to succeed in our modern global economy. While technology has advanced massively over the past fifty years, our formal educational systems have barely changed. There is great potential for computational systems to transform how we teach and learn, and this has led to many exciting advances in the field of educational technology.

Over the past few decades, new systems have been designed to provide access to online courses, to personalize learning experiences, and to tackle the problem of student motivation. Platforms such as Khan Academy, Coursera, and EdX provide on-demand access to world-class educational videos and activities to millions of students online [55, 93]. Intelligent tutoring systems, designed to emulate one-on-one tutoring with a human, can provide intelligent feedback and hints in response to any student mistake [10, 29]. Finally, educational games such as DragonBox and Refraction have shown a seemly endless capacity to engage students in learning activities [75, 104].

While these technical advances appear to have great potential to improve student learning, they have not fundamentally changed the state of eduction in the United States. Recent data shows that 1.2 million students drop out of high school every year [135], and only 28% of high school graduates are prepared for college in all four subject areas assessed by the ACT test (English, reading, mathematics, and science) [2]. More importantly, average math and reading performance for 17-year olds has remained stagnant since the 1970s [1].

The limited impact of educational technology on real-world learning outcomes is caused by many factors. Online learning platforms typically attract students who are already motivated to learn, so they miss the students who need the most help [47, 147]. Intelligent tutoring

systems are difficult for teachers to use, and provide little room for customization to different classroom environments [29]. Educational games have mixed learning outcomes in formal studies [78, 84], in large part because it is challenging for students to transfer learning outside of the game context. In each of these cases, a central problem is that the computer scientists who are leading the system design process are trying to reinvent the wheel, rather than learning from prior research in education and the learning sciences. Researchers in these domains have been studying challenges such as student motivation, classroom integration, and learning transfer for decades.

In this dissertation, I apply a different approach. I take my inspiration directly from education research to develop novel technical systems grounded in learning theory that address core challenges in education. First, I identify successful approaches for improving student learning from the education literature, including learning theories, validated interventions, and evaluation methodologies. Then I use my expertise in computer science to expand and adapt these ideas to create new learning experiences that would not be possible without technology. Using this approach, I work towards the development of new data-driven systems designed to meet the unique needs of each student, teacher, and classroom.

My collaborators and I have applied this approach to address educational challenges in three key areas. First, we rethink the design of motivational incentives for educational games by considering psychology research on the importance of student mindsets [107, 108, 109]. Next, we develop new technical approaches to facilitate the automatic generation of empirically validated instructional scaffolding [102, 105]. Finally, we study the behavioral impact of a novel formative assessment system to better understand the integration of technology into real-world classrooms [106]. I provide a background for each of these educational challenges, and then summarize the contributions made in this dissertation.

1.1 Incentive Structures for Educational Games

Studies show that student motivation is an important predict of academic success [112], however formal education methods engage only a small portion of students [129]. As a result,

there is great interest in designing learning environments that motivate students and help them develop a positive relationship with learning. Video games are famous for their ability to motivate players to perform complex, time-consuming tasks [40], and learning has been cited as a central component of gameplay [41]. As a result, educational technology researchers have begun to explore the possibility of leveraging games to address the problem of student motivation in educational environments [40, 84, 101]. Game incentive structures, or the systems of rewards that are given to successful players, have elicited particular attention for their potential to engage students [59]. However, little is known about how in-game praise and rewards impact student motivation and learning.

Researchers in educational psychology have studied student motivation for decades, and their findings have great potential to inform the design of incentive structures in educational games. Studies show that praise can have varying, and sometimes negative, effects on motivation [92]. Praising a student's inherent ability promotes the fixed mindset, or the belief that intelligence is unchangeable, while praising a student's strategies and effort promotes the growth mindset, or the belief that intelligence is malleable [92]. These mindsets can have a strong impact on students' motivation, reaction to failure, and academic achievement [19]. Game incentives are similar to praise in many ways, because they communicate which behaviors are rewarded and valued in the learning context. As a result, this psychology research suggests that in-game incentives could impact student behavior, and even have negative consequences if rewards inadvertently support the fixed mindset.

Chapter 2 presents our research on developing a new incentive structure for educational games that is specifically designed to promote the growth mindset. We developed a new "brain points" incentive structure that detects productive struggle and strategy use in real-time as students play the educational game *Refraction*. To explore the impact of this intervention, we conducted three in-depth user studies with online participants. First, we compared the brain points incentive structure to a control that rewards students for their progress through the game, and found that students who receive brain points are more persistent and use more strategy [108]. Next we studied the impact of the brain points intervention

on students of different demographics [107]. Finally, we dug deeper into the design by comparing four modified interventions to our original, and found that the incentive structure has the strongest positive effect on student behavior [109]. This work demonstrates the impact of grounding educational technology design in learning theory, and provides valuable insights into the impact of in-game incentives on student behavior.

1.2 Interactive Instructional Scaffolding

One-on-one tutoring is one of the most effective instructional techniques [22]. Human tutors are powerful in part because they can personalize instruction for individual students, providing appropriately difficult activities and tailoring feedback to address specific misconceptions. This type of support is often referred to as scaffolding, an instructional technique that helps students solve problems that would otherwise be beyond their reach [43, 146]. While there are many different ways to scaffold student learning, some effective techniques include step-by-step demonstrations, tailored progressions of practice problems, and individual feedback. Curriculum materials provide some static scaffolding and teachers can personalize some content, but it is challenging for a single teacher to provide the same level of support to her class as a one-on-one tutor.

To address this challenge, educational technologists have explored approaches of automatically personalizing scaffolding through in interactive learning environments. A large body of research has explored the design of intelligent tutoring systems, interactive software that adapts problem progressions and provides personalized feedback in a variety of domains [10, 29, 51]. Human-computer interaction researchers have also studies the design of scaffolded tutorials to support users as they learn to use new software products [17, 28, 45]. While these systems have been shown to be effective, creating interactive scaffolding is very time-consuming because most content must be authored by hand [6, 45].

Chapters 3 and 4 present our research on developing new approaches for efficiently generating interactive scaffolding in two domains: procedural knowledge and algebraic problem solving. In Chapter 3, we describe the challenge of generating step-by-step explanations

and problem progressions for procedural knowledge. We present a new domain-independent method for representing problem-solving procedures, and describe how this model can be used to generate scaffolding automatically [102]. In Chapter 4, we describe the challenge of generating interactive scaffolding in the less constrained domain of algebraic problem solving. We present a new approach for producing both example problems and solution explanations from an domain model written using the declarative programming paradigm *answer set programming* [105]. This work demonstrates how computer science approaches can be used to generate scaffolding content that has been empirically validated by the education literature.

1.3 Personalization in the Classroom

While studies show that cognitive tutoring systems that automatically adapt scaffolding for each student are effective [68, 141], they have not been widely adopted in real-world classrooms. Researchers have noted that most of these systems are not designed to integrate with teacher workflows nor evaluated in terms of how they impact classroom processes [33], which could explain this lack of adoption. Most personalized learning systems are designed to adapt lessons and scaffolding completely automatically, without any input from teachers. However, teachers often personalize their own instruction using information collected through formative assessment, and adaptive systems may be more effective if they are designed to support this process.

Formative assessment is the practice of frequently assessing students to collect data about their progress [18]. Large learning gains have been measured when teachers modify instruction based on formative assessment data [18, 38]. Studies show that student data is most relevant to teachers when assessments are embedded in the curriculum [127] and when data can be collected and analyzed quickly [144]. However, administering and grading assessment requires significant teacher effort, making frequent formative assessment difficult to implement in the classroom. As a result, educational systems offer great potential for supporting real-time formative assessment while also providing personalized scaffolding for students.

Chapter 5 presents our research on designing and evaluating a tablet-based personal-

ization system that integrates into classroom workflows. Our system augments an existing mathematics curriculum by replacing paper-based problem-solving activities. It adapts problem content for each student automatically, and also communicates student data to teachers in real time to help them identify and assist struggling students. We report on an in-depth study conducted in four Seattle-area schools that measures behavior in classes with and without the technology, keeping both the curriculum and teacher constant. We found that students with the tablet software complete more problems with more accuracy, and that teachers assist students three times more often when they have access to real-time formative assessment data [106]. This work shows how educational systems designed with the classroom workflow in mind can effectively support personalization in the classroom.

1.4 Contributions

In this dissertation, I present the design and evaluation of novel technical systems grounded in learning theory that address core challenges in education. My collaborators and I tackle three distinct challenges: promoting the growth mindset in educational games, generating interactive scaffolding automatically, and studying the impact of personalized learning systems in the classroom. We make the following contributions:

Educational game incentive structures. We present a new approach for directly rewarding growth mindset behaviors through a novel incentive structure, and demonstrate the viability of this approach through the development of the "brain points" version of the educational game *Refraction*. Through three large-scale online studies, we demonstrate that our growth mindset intervention student persistence and use of strategy, and encourages struggling students to stick with the game. These findings broaden our understanding of the behavioral impact of incentive structures in educational games.

Interactive instructional scaffolding. We present new approaches for automatically generating instructional scaffolding for both procedural domains and algebra. We demonstrate the feasibility of our approaches through three proof-of-concept implementations: a grid

mathematics application, the educational game *Refraction*, and an algebraic problem solving application. We present preliminary results from user studies showing that participants can learn new problem-solving approaches from generated content. This work expands our ability to produce the example problems and scaffolding that will be needed to truly personalize student learning experiences in the future.

Personalization in the classroom. We present the design of a new personalization system that supports both adaptive problem progressions and real-time formative assessment. We evaluate the behavioral impact of introducing this system into real-world classrooms through a 10-week study in eleven sixth-grade math classes. We present results demonstrating that students complete more problems and teachers assist more individual students when the personalization system is present. These findings contribute to our understanding of how personalization technology can be designed to integrate with classroom workflows.

Taken together, these contributions demonstrate the value of leveraging approaches from the education literature in the design of new educational technologies. The systems presented in this dissertation have been used by over 100,000 students online, adopted by companies, and used in classrooms throughout the Seattle area. This research takes an important step towards the development of personalized data-driven systems that can meet the unique needs of each student, teacher, and classroom.

Chapter 2

BRAIN POINTS: A GROWTH MINDSET INCENTIVE STRUCTURE FOR AN EDUCATIONAL GAME

This chapter explores the design and implementation of an incentive structure for an educational game that encourages the growth mindset. Researchers have argued that video games could be leveraged to improve student engagement and motivation in a variety of learning environments. However, educational games are not uniformly effective, and little is known about how in-game rewards affect children's learning-related behavior. In this chapter, we show that educational games can be improved by fundamentally changing their incentive structures to promote the growth mindset, or the belief that intelligence is malleable. In particular, we present "brain points," a system that encourages the development of growth mindset behaviors by directly incentivizing effort, use of strategy, and incremental progress. This chapter describes the brain points system and presents results from three studies that explore the impact of this growth mindset incentive structure on student behavior in the educational game *Refraction*.

2.1 Introduction

Video games are famous for their ability to engage players and motivate them to perform complex, time-consuming tasks. Most children today are exposed to games on a daily basis; 92% of children ages 2 to 17 play video games, for an average of 20 to 33 minutes per day [57]. As a result, there is a growing interest in leveraging games to address the problem of student motivation in educational environments [40, 84, 101]. Game incentive structures, or the systems of rewards that are given to successful players, have elicited particular attention for their potential to motivate students [59]. However, games have produced mixed learning

outcomes in the classroom [50, 74, 84], and the effects of in-game praise and rewards on motivation and learning are not well understood.

A growing body of research in psychology suggests that feedback such as praise can have varying, and sometimes negative, effects [46, 92]. Praising a student's inherent ability has been shown to promote the fixed mindset, or the belief that intelligence is unchangeable, while praising a student's strategies or effort promotes the growth mindset, or the belief that intelligence is malleable [46, 92]. Studies have shown that children with a fixed mindset view mistakes, challenge, and effort as negative indicators of their intelligence, while children with growth mindset view effort as positive and challenges as opportunities to learn [36, 53]. More importantly, holding a fixed mindset predicts static or decreasing academic performance over time, while holding a growth mindset predicts academic improvement [19, 44].

Many laboratory and classroom studies have shown that children's mindsets can be changed through careful intervention [19, 58, 92]. Directly teaching students that intelligence is malleable was shown to improve classroom motivation and grades compared to a control group [19]. Even minimal interventions, such as praising children for their strategy or effort (e.g., "you must have worked really hard!") instead of their ability (e.g., "you must be smart at this!") as they solve problems produces a growth mindset, as well as higher motivation and task persistence [58, 92]. This research suggests that the praise and rewards given to students in educational games could impact their behavior, and even have negative consequences if rewards inadvertently support the fixed mindset.

In this chapter, we show that making a fundamental change to a game's incentive structure can positively impact children's behavior. We present "brain points," a system that rewards students for their effort, use of strategy, and incremental progress. Unlike previous mindset interventions, this incentive structure provides children with real-time feedback as they work to develop growth mindset behaviors. We present results from three studies that explore the behavioral impact of brain points. The first study shows that persistence and use of strategy are encouraged in the educational game *Refraction* through the introduction of this unorthodox incentive structure. The second study replicates these findings and

explores how brain points impact students from different demographic groups. Finally, the third study provides deeper insights into why the growth mindset intervention is effective by comparing five different versions of the intervention. We will first provide a background on the growth mindset and related research and then describe each study in turn.

2.2 Related Work

Psychologists have been studying motivation and academic achievement for decades, and many of their discoveries have important implications for the designers of educational technologies. Messages and rewards that support the growth mindset could be used to significantly improve the impact of such systems. In this section, we provide background on mindset research and discuss previous efforts to integrate growth mindset interventions into classrooms and educational tools. We also describe previous work on educational games, and suggest ways that the growth mindset could be used to improve their effectiveness.

2.2.1 Theories of Intelligence

Psychologist have shown that beliefs about the malleability of human attributes such as intelligence can have strong effects on motivation, reaction to challenge or failure, and academic achievement [19, 44, 53]. Individuals who hold a *fixed mindset* believe that they have a certain amount of intelligence, and that this is an unchangeable attribute. Studies have shown that people with a fixed mindset view challenging situations as "tests" of how much intelligence they have, and view effort and mistakes as indications of low ability [19, 53, 92]. On the other hand, individuals who hold a *growth mindset* believe that intelligence is malleable, and that people can increase their intelligence through hard work. They have been shown to value learning over performance, and view effort as a necessary part of the learning process [19, 53, 92]. These beliefs affect not only behavior but also academic achievement. In a longitudinal study of 373 seventh graders, Blackwell et al. showed that holding a growth mindset predicted improving grades over the two years of middle school, while holding a fixed mindset predicted static or decreasing grades [19].

A growing body of research shows that students' mindsets, and subsequently their behaviors and academic performance, can be changed through intervention [13, 19, 44, 58, 92]. One type of intervention involves changing the type of praise given to children when they are successful. In a now-famous 1998 study, Muller and Dweck showed that praise can have negative consequences when it supports the fixed mindset [92]. They gave fifth grade students a set of achievable problems and praised them for either their ability ("you must be smart at these problems") or their effort ("you must have worked hard at these problems"). Then they gave students very challenging problems on which they did poorly. Finally, they gave students another set of the original achievable problems. Students who received praise for their ability attributed their failure to lack of ability, and performed worse on the subsequent problem set, but students who received praise for their effort viewed the difficulty as a cue to try harder, and performed better on the subsequent problem set [92]. More recently, Gunderson et al. have confirmed the role of praise in mindsets, showing that the type of praise parents give to their young children at home predicts the child's mindset and desire for challenge five years later [46].

Another type of intervention involves teaching the growth mindset directly. Blackwell et al. taught a group of seventh grade students that intelligence is malleable during an eight-session workshop through readings and discussions about the neural connections that are formed in the brain when it works hard [19]. Before the intervention, students' math grades had been steadily decreasing (and this decline persisted for children in the control group), but after the intervention students grades improved significantly [19]. Aronson et al. also report the positive impact of a growth mindset intervention on the academic achievement of both African American and white college students [13]. The GPAs of both the African American and white students in the experimental condition were higher at the end of the academic quarter than those in the control condition [13].

The effects of these interventions on student motivation and academic achievement are impressive, and it would clearly be beneficial to integrate similar programs into educational technologies. Recently, efforts have been made to develop online materials that teach the

growth mindset. Brainology[1], a for-purchase online program based on the Blackwell intervention, teaches students the scientific basis of the growth mindset through readings and interactive exercises. Paunesku et al. studied the effectiveness of an online growth mindset intervention, delivered through two 45-minute sessions involving reading and writing exercises, and found student grades in core academic courses increased in comparison to a control [111]. However, we are not aware of any other work that integrates the growth mindset into the incentive structures of educational games.

2.2.2 Games as an Educational Platform

Video games are increasingly recognized as a compelling platform for instruction that could significantly improve student motivation in the classroom [40, 84, 101]. While empirical evidence supporting learning outcomes of educational games is mixed [50, 74, 84], there have been clear successes that highlight the potential of games as instructional tools. In a comparative review of STEM game studies, Mayo found that some games produce a 7 to 40% positive increase in learning outcomes [84]. Games have been shown to increase time-on-task, an important indicator for academic success [72, 74], and also increase student motivation [120]. Researchers have noted that successful educational games are those designed around effective pedagogical practices, perhaps explaining some of the mixed learning outcomes [84, 101]. These results indicate the importance of grounding educational games in valid pedagogical theory.

Many researchers have explored methods of effectively integrating learning theories into games, and of leveraging game features to maximize student motivation, persistence, and learning [27, 74]. For example, Chase studied how task framing in a genetics game affects student persistence and learning. She found that students who are told their performance is dependent on both chance and skill persist for a longer period of time after failure than those who are told their performance is dependent on skill alone [27]. However, the effects

[1]Brainology: http://www.mindsetworks.com/webnav/program.aspx

of game features on student motivation are still not well understood.

To our knowledge, no prior research has explored how to improve in-game motivation and persistence through growth mindset incentives and feedback. We believe that educational games provide a set of properties that make them particularly conducive to introducing and incentivizing growth mindset concepts. Game narratives provide a forum for directly teaching about brain growth, weaving messages that support the growth mindset throughout the game world. Constant interactive feedback provides a medium for showing students that their effort translates into progress [42, 84]. But most importantly, game incentive structures provide a way to support and reward behaviors consistent with the growth mindset, such as persistence and use of strategy.

2.3 Study 1: Brain Points Boost Player Persistence

The goal of our first study was to explore the impact of teaching the growth mindset and incentivizing effort in an educational game. To address this question, we designed an experiment that compared two versions of the game *Refraction*. The experimental version of the game rewards growth mindset behavior by leveraging the game's narrative and incentive structure, while the control version provides a neutral view of intelligence.

2.3.1 Experiment Design

In this section, we describe the game *Refraction* and then discuss the modifications we made to the game to create the experimental and control versions used in this study.

Refraction

This educational puzzle game was designed by game researchers and learning science experts at the Center for Game Science[2] to teach fraction concepts to elementary school students. The game was designed to support a conceptual understanding of fractions rooted in the

[2]Center for Game Science: http://centerforgamescience.org/

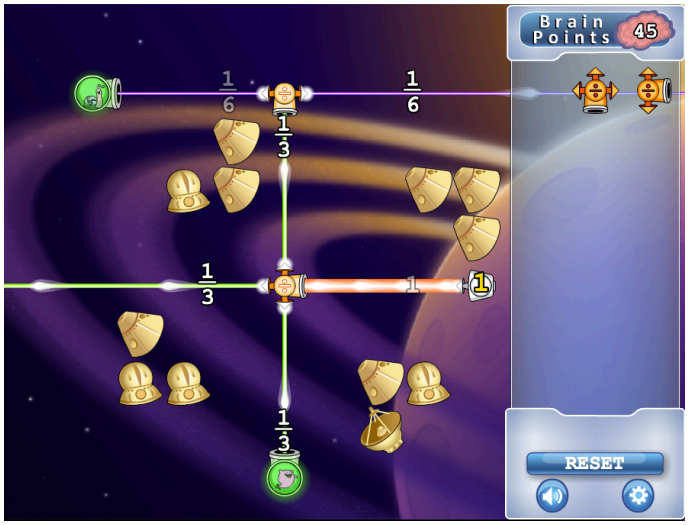
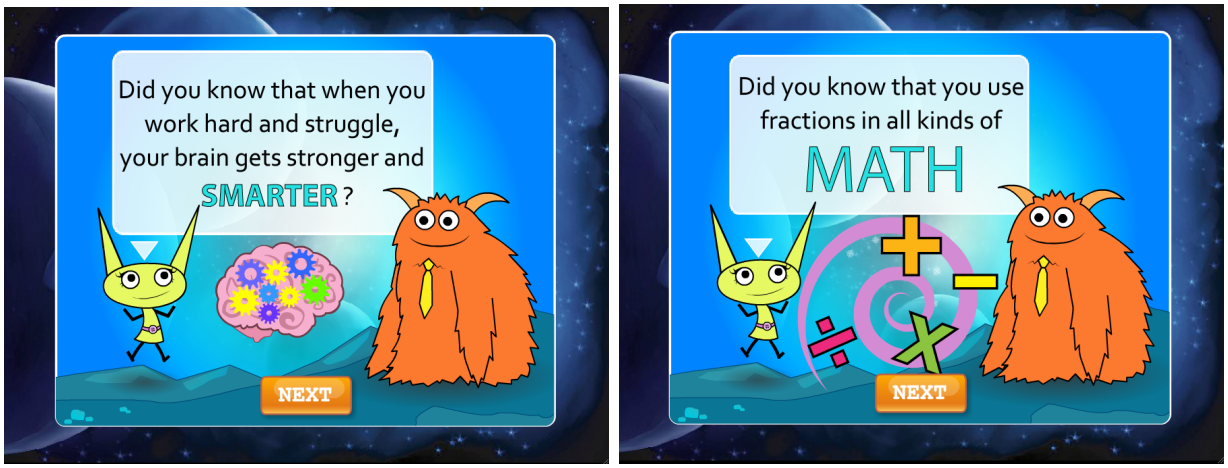


Figure 2.1: **A Refraction Level.** The pieces in the bin on the right are used to split lasers into fractional amounts and redirect them to satisfy the target spaceships. All ships must be satisfied at the same time to win.

concept of splitting [82]. To play, a child must interact with a grid that contains laser sources, target spaceships, and asteroids, as shown in Figure 2.1. The goal of the game is to satisfy target spaceships by splitting the laser into the correct fractional amounts and avoiding asteroids. The player uses pieces that either change the laser direction or split the laser into two or three equal parts to achieve this goal. To win, the player must correctly satisfy all the target spaceships at the same time. *Refraction* has been successful at attracting elementary school students, and has been played over one million times on the educational website BrainPOP since its release in April 2012.

Game Narrative

Both the experimental and control versions of *Refraction* are based on a central narrative. At the beginning of the game, players watch a 25 second introductory animation featuring Zuzu and Copper, characters who describe the game and its incentive structures. In the experimental version, we leverage this animation to teach players about the growth mindset



(a) (b)

Figure 2.2: **Narrative Animation Screenshots.** Figure (a) shows a screenshot of the animation that teaches the growth mindset in the experimental version of Refraction. Figure (b) shows a corresponding screenshot in the control, which presents a neutral mindset about the importance of fractions. The animations use identical art and text wherever possible.

Experimental
Z: Hey Copper, guess what!
Z: Did you know that when you work hard and struggle, your brain gets stronger and SMARTER?
C: Whoa cool!
Z: In this game, when you work out your brain and try new ideas, you'll get BRAIN POINTS!
C: I want to get lots of brain points! Let's play, Zuzu!
Control
Z: Hey Copper, guess what!
Z: Did you know that you use fractions in all kinds of MATH?
C: Whoa cool!
Z: In this game, when you make the fractions on the spaceships you'll get FRACTION POINTS!
C: I want to get lots of fraction points! Let's play, Zuzu!

Table 2.1: **Narrative Animation Scripts.** The experimental animation teaches the growth mindset, while the control animation presents a neutral message about the importance of fractions. Z indicates Zuzu's line, and C indicates Copper's line.

directly, using language based on that used in the Brainology curriculum and the Blackwell intervention [19]. In the control version, we present a neutral message about the importance of fractions. The text and art used in the animations are nearly identical, differing only in the messages they present to players. Screenshots are shown in Figure 2.2 and the full animation scripts are included in Table 2.1. We do not use audio in the animations because we cannot guarantee that players will have access to speakers or headphones. To accommodate different reading speeds, we added "Next" buttons that allow players to manually advance to the next part of the animation when they have finished reading.

Incentive Structures

Both versions of *Refraction* have point-based incentive structures designed to reward different types of behavior. For the experimental version, we designed a system of "brain points" that rewards children for their effort, use of strategy, and incremental progress. These types of behaviors support "learning" goals rather than "performance" goals, and rewarding similar behaviors with praise has been shown to promote the growth mindset [46, 92]. However, no existing growth mindset interventions have used point-based reward systems, so this is an entirely new method of teaching the growth mindset that is particularly well suited to the game context.

Children earn brain points while they are working to solve levels. We use a combination of four metrics that capture desirable strategic behaviors to determine when players should receive points. The *new hypothesis* metric captures each new idea the child tries. It triggers when the child makes two successive new moves with distinct pieces. The *board cleared* metric emphasizes stepping back to consider the puzzle from a fresh perspective. It triggers when there are at least two pieces on the laser, and the child returns all the pieces to the starting bin. The *math* metric captures incremental mathematical progress. It triggers when the child makes a target fraction for the first time. Finally, the *moves* metric captures effort. It triggers when the child makes ten distinct moves.

To earn brain points, the child must trigger two of these metrics. For example, a child

(a) (b)

Figure 2.3: **Incentive Screenshots.** Figure (a) shows the brain points interface in the experimental version, where players earn points for their effort, use of strategy, and incremental progress as they work to solve levels. Figure (b) shows the interface in the control version, where players earn points for each level they complete.

might try a new hypothesis and then clear the board, earning five points. We require two metrics to be triggered because this makes the system more difficult to game, since the child cannot repeat his last sequence of moves to earn another point. Gaming the system is a problem in many educational domains [15], so we wanted to ensure that our point system rewarded desirable behaviors without being overly transparent. All metric data, such as the number of moves made so far, are cleared when the child earns a point so that subsequent points are earned from scratch.

Brain points are given to players in increments of five to make them more exciting. Each time the child triggers two metrics, she earns five points. When this happens, a small animated brain icon with a short message appears, as shown in Figure 2.3(a). The icon moves to the upper righthand corner of the screen, adding five to the total number of points shown in the brain points bar. The short message reflects the last metric the child triggered. The messages associated with the four metrics are "New Idea" for the *new hypothesis* metric,

"Fresh Start" for the *board cleared* metric, "Math Effort" for the *math* metric, and "Working Hard" for the *moves* metric.

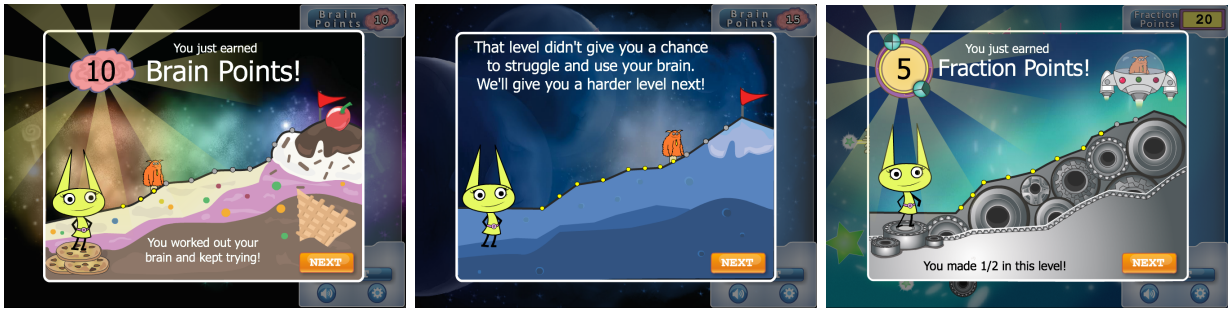
For the control version of *Refraction*, we designed a system of "fraction points" that reward children for completing levels. We chose to incentivize advancement through the level progression because this is a commonly used metric of success in video games, as Schell notes in his popular game design book [124]. Players earn five fraction points every time they win a level, displayed on the summary screen described in the next section. The total number of fraction points earned so far are shown in the upper righthand corner of the screen during the level, as shown in Figure 2.3(b).

Summary Screens

Each time a child completes a level of *Refraction*, a summary screen is displayed. This screen is designed to reinforce the game narrative and provide an opportunity for children to reflect on their success or progress. Identical art is used in the experimental and control versions, but different messages are given to support either the growth mindset or a neutral mindset. The summary screen has three components: a progress indicator, a points message, and a praise message. Screenshots of summary screens are shown in Figure 2.4.

The *Refraction* world has seven different planets, each with unique level art. In both conditions, children advance to a new planet each time they collect 50 points, an exciting reward that highlights their progress through the game. Since this advancement is based on points, children in the experimental condition move forward based on their effort and strategy use, while children in the control condition move forward based on their performance. Each summary screen shows a mountain with ten dots that indicate the player's progress through the planet. The player moves up one dot for every five points earned, and completed dots are shown in yellow. When the player reaches the top of the mountain, a reward animation shows the orange character Copper flying to the next planet.

In the experimental version, an animation shows Copper climbing the mountain every time the child earns brain points, as in Figure 2.4(a). This physical climbing metaphor was



(a) (b) (c)

Figure 2.4: **Summary Screen Screenshots.** Figure (a) shows the summary screen that is displayed when the child earns brain points in the experimental condition, and Figure (b) shows the screen that is displayed when the player does not earn brain points. Figure (c) shows the control condition summary screen that is displayed every time the player wins a level. In all cases, the mountain represents progress through the current *Refraction* planet.

designed to symbolize the hard work required to earn brain points and support the growth mindset. In the control version, Copper waits at the top of the mountain in his spaceship while newly acquired dots are highlighted in yellow, as in Figure 2.4(c). This presents a more neutral message, showing progress without symbolizing effort.

At the top of the summary screen, a message tells the player how many points were earned. Since players in the experimental condition earn points based on their behavior, this number changes dynamically. If a player does not earn any brain points on a given level, a message apologizing for not giving the player an opportunity to struggle is displayed, as shown in Figure 2.4(b). This message is based on Dweck's suggested growth mindset response to situations that do not challenge children or provide them with an opportunity to learn [36]. Players in the control condition always earn exactly five fraction points for each level completed.

At the bottom of the summary screen, a message praising the child's work in this level is shown. In the experimental condition, one of four messages is chosen, depending on which of the four brain point messages the player saw most frequently during the level. The messages,

Metric	Growth Mindset Praise Messages
New Hypothesis	You worked hard and tried new ideas!
Board Cleared	You worked hard and got a fresh start!
Math	You worked hard and used your math brain!
Moves	You worked out your brain and kept trying!

Table 2.2: **Growth Mindset Praise Messages.** The four messages displayed at the bottom of the summary screen in the experimental version of *Refraction*. A praise message is selected based on the brain points message the child saw most frequently during the level.

which are based on text from Muller and Dweck's praise-based intervention [92], are shown in Table 2.2. In the control condition, a message summarizing the fractions the child made in this level is shown. If the child won a level with spaceships that required 1/6 and 1/9 laser power respectively, the message would read "You made 1/6 and 1/9 on that level!"

Adaptive Level Progression

In order to effectively incentivize effort in the experimental version of *Refraction*, it was important to design a level progression that would provide every player with an opportunity to struggle, irrespective of their incoming skill level. To address this requirement, we created an adaptive level progression that was used in both versions of the game. Our design is based on the mastery learning model used in many knowledge tracing systems, in which students must display mastery of a concept before advancing to the subsequent concept [31]. The adaptive progression covers eleven mathematical concepts typically included in *Refraction*, which are listed in Table 2.3. We designed ten levels for each concept: one tutorial level that introduces the concept without providing additional pieces, and nine puzzles in which the player must correctly choose between splitter pieces to create the correctly valued fractions.

All children play the eleven concepts in the order listed in Table 2.3, but they move through the concepts at different speeds depending on their skill level. Every child plays the tutorial level for a given concept, and is then given levels from that concept until she either

Order	Concept	Order	Concept
1	Halves	6	Halves and Fourths
2	Thirds	7	Thirds and Ninths
3	Fourths	8	Halves and Sixths
4	Ninths	9	Thirds and Sixths
5	Sixths	10	Fourths and Sixths
		11	Sixths and Ninths

Table 2.3: **Refraction Concepts.** The fraction concepts covered in *Refraction*'s adaptive level progression. All children play the concepts in the same order, but they advance through concepts at different speeds based on their performance.

wins two levels under a performance threshold, or completes all nine levels for the concept. To win a level below the performance threshold, the player must complete the level in less than 1.5 times the minimum number of moves required to beat the level without making any mathematical mistakes. Mistakes are defined as moves that split the laser to make fractional values that are not needed to solve the level. For example, if a child is working on a level with a 1/9 target spaceship, she should never split the laser to make 1/2 or 1/4.

Every child will play between three and ten levels for each concept. The levels are given to all players in the same order, so every child plays the same first three levels for a concept, and some will continue on to play more levels. This means that many more children play the first three levels than play the ninth and tenth level.

Skipping Levels

For the growth mindset version of *Refraction*, we wanted players to think about their progress in terms of effort and strategy, symbolized by brain points, rather than in terms of the number of levels they have completed. As a result, we designed the summary screens and adaptive level progression in a way that minimizes the visibility of speed by making it difficult for children to compare the number of levels they have completed. One downside of this design is that it provides no way for players to back out of levels when they get stuck.

To address this issue and deter frustrated players from quitting the game, we added a "New Level" button that appears in each level after three minutes of play. Children are required to work on each level for at least three minutes, but then they are given an opportunity to move forward when they are stuck. To ensure that children notice the button, a tutorial message is displayed the first time it appears.

When a child clicks the "New Level" button, a summary screen is displayed. In the experimental version, if the child earns brain points before clicking the button, the standard summary screen shown in Figure 2.4(a) is displayed. If the child does not earn any brain points, the message "Work harder and use your brain on the next level to earn points!" is displayed in a screen similar to Figure 2.4(b). In the control version, children only earn points when they win levels, so the same summary screen with the message "Try again on a new level! To move up the mountain, save the spaceships!" is always shown. In all cases, the summary screen has a "Back" button that returns to the level and a "Next" button that advances to the next level in the current concept.

Challenge Levels

In addition to observing children's behavior as they interact with the standard game levels, we wanted to measure how they react to a particularly challenging level. In their 1998 study of praise, Muller and Dweck saw that failing to solve a very difficult problem affected children differently based on whether they received fixed or growth mindset praise [92]. We designed a similar test for *Refraction* players by giving them a "challenge level" 30 minutes after they started the game. This level had two 1/2 spaceships, and presented a tricky and unintuitive spatial problem. We used simple fractions because we could not guarantee that all players would have reached more complex concepts.

The challenge was framed as a special level separate from the standard game. We used an exciting animation to introduce the level, and designed a new background to accompany it. We wanted to measure whether any effects caused by the growth mindset intervention would transfer to levels where effort is not explicitly rewarded, so we did not give children points

Experimental	
Timing	*Message*
Tutorial	You won't earn points on this level, but it will let you challenge your brain and make it grow stronger!
Skip	You didn't master that challenge yet, but you can keep growing your brain by earning points on the next level!
Win	You beat the challenge! You really worked out your brain on that level.
Control	
Timing	*Message*
Tutorial	You won't earn points on this level, but it will let you show off your fraction skills!
Skip	You didn't beat that challenge, but you can use your fraction skills to earn points on the next level instead!
Win	You beat the challenge! You really used your fraction skills on that level.

Table 2.4: **Challenge Level Messages.** The three challenge level messages for each condition. The first message is shown during the challenge level tutorial, the second is shown when the level is skipped, and the third is shown when the level is won.

during the challenge level. We added a bright orange "New Level" button to the sidebar so that players could skip at any point. We also added a tutorial message highlighting the presence of the "New Level" button to make sure players knew it was available. The challenge level was identical in the two conditions except for the text displayed to children. There are three challenge messages, shown in Table 2.4. One message is shown at the beginning of the level, another when the child wins the level, and the last when the child skips the level.

2.3.2 Experimental Hypotheses

We had several hypotheses about how children would behave in response to our two versions of *Refraction*. First, we expected that children who played the experimental version of the game would exhibit behaviors consistent with the growth mindset, specifically persistence and strategic behavior.

Hypothesis 1: *Players in the experimental condition will be more persistent and more strategic than players in the control.*

While we expected all players to be affected by the growth mindset intervention, we thought that players who struggle with *Refraction* would be most strongly influenced by our incentive structure that rewards effort. We expected them to be more strongly motivated to persist in the game.

Hypothesis 2: *Struggling players will be most strongly motivated by the experimental intervention.*

In addition to observing general behavior, we also designed our experiment so that we could explicitly study how children reacted to challenge and failure by observing their behavior during the challenge levels. We expected children who played the experimental version of *Refraction* to react well to challenge, struggle, and failure.

Hypothesis 3: *Players in the experimental condition will react better to challenge than players in the control.*

We evaluate these three experimental hypotheses by capturing relevant behaviors during gameplay, as described in detail in the next section.

2.3.3 Method

To gain an understanding of the effects of incentivizing productive effort and teaching the growth mindset in educational games, we studied how children play the experimental and control versions of *Refraction* on the popular educational website BrainPOP[3]. BrainPOP is best known for its curriculum resources, but it also provides an educational game portal designed for use in the classroom. The BrainPOP Educators community has over 210,000 members, and the website is used as a resource in around 20% of elementary schools in the United States (Traci Kampel, personal communication).

One benefit of using the BrainPOP game portal to study the impact of our growth mindset intervention is that it provides us with access to a large, diverse population of students, and allows us to quickly learn whether our intervention has promise in the classroom. How-

[3]BrainPOP: http://www.brainpop.com/

ever, one downside of this resource is that we know very little about the children who visit BrainPOP or the contexts in which they play. We cannot collect any demographic information, and while we know that the website is primarily used in schools, we cannot tell whether children are playing in the classroom, in a computer lab, or at an after-school program. We mitigate the effects of these uncontrolled variables by evenly distributing them between conditions through the use of randomization and large sample sizes. We are also unable to directly measuring learning through formal pre and post tests in this environment. Instead, we analyze how our intervention impacts observable in-game behaviors such as persistence, use of strategy, and reaction to challenge. These are key components of learning that capture how students react to our growth mindset system.

Our study has a single between-subjects factor *intervention* with two levels: experiment or control. To collect our data, we set up *Refraction* to randomly assign new players to either the experimental or control version of the game, and logged all interactions players made with the game or its interface. We only included new players who were not familiar with *Refraction* in our analysis, and only used data from a player's first session to control for issues with shared computers in schools. To track players, we stored their progress in the Flash cache, which allowed us to selectively include new players and exclude return sessions. One drawback of this method is that a player who clears the cache or changes computers will be treated as a new player by our system. While we cannot assess the seriousness of this risk, its effects will be evenly distributed across conditions. Furthermore, it is unlikely that children will clear the cache because this option is inconvenient to access and it deletes all saved game progress.

Another challenge of studying student behavior online is that average play times are typically small. Previous research conducted on BrainPOP shows that children play *Refraction* for about three minutes on average [104]. We expected our brain points system to influence student behavior even during this short period of play because previous studies have shown that very minimal interventions, such as praising children for their effort or strategies, can improve motivation and persistence [58, 92]. However, we filtered our data to ensure that all

students included in our analysis had an opportunity to be influenced by our growth mindset intervention. The first growth mindset message presented to students is the introductory animation, so for both conditions we only included those who watched the entire introductory animation and made at least one move in the game. We also conducted our analysis with a stricter filtering criterion that only included students who completed the first four levels of the game. After four levels, 99% of players in the experimental condition have received brain points and seen the summary screen. Both analyses showed the same patterns of significance and led to the same conclusions, so we chose to present results for the looser filtering method that included more students in the analysis.

After filtering, our data set contained 15,491 players, with 7,807 players in the experimental condition and 7,684 in the control condition. These data were collected between September 3 and September 13, 2013. *Refraction* was featured on the front page of BrainPOP's game portal between September 3 and September 5, allowing us to attract large numbers of students. Since the data sets for the two conditions had different numbers of players, we randomly selected 7,500 players from each condition to include in our analysis.

2.3.4 Data Analysis and Results

We studied the effects of our growth mindset intervention on children's behavior by analyzing a number of outcome measures, each of which is described in detail below. Before performing this analysis, we evaluated the Kolmogorov-Smirnov test to assess the normality of our data, and found that it was statistically significant for all of our outcome measures. We therefore chose to use non-parametric statistical methods: a Wilcoxon rank sums test and an r measure of effect size for continuous variables, and a Chi-square statistic and a Cramer's V measure of effect size for nominal variables. We report effect sizes in addition to p-values to show the magnitude of the differences between our populations, since we are likely to find many significant differences due to our large sample sizes. For both of these statistical tests, effect sizes with values less than 0.1 are considered *very small*, 0.1 are *small*, 0.3 are *moderate*, and 0.5 or greater are *large*.

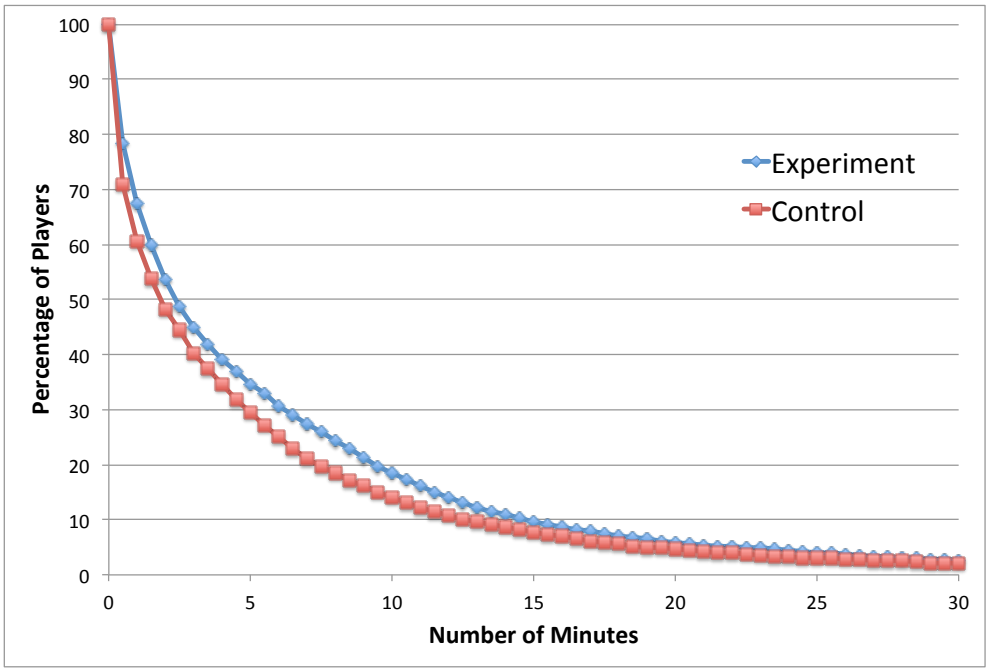


Figure 2.5: **Time Played Graph.** This graph shows the percentage of players in each condition who are still playing after a given number of active minutes. Children in the experimental brain points condition play significantly longer.

The Growth Mindset Intervention Increases Persistence

We expected children who played the experimental version of *Refraction* to be more persistent than those who played the control version. To evaluate this hypothesis, we analyzed two outcome measures: the amount of time children spent playing the game, and the number of unique levels they played. Since BrainPOP offers many other games that teach fraction concepts, we expected children to quit *Refraction* when they became bored or frustrated. As a result, these measures capture how long children are willing to persist before choosing to leave the game.

We calculate active time by counting the number of seconds each child played *Refraction*, excluding menu navigation and idle periods with more than thirty seconds between actions. Our analysis showed that *intervention* has a significant effect on active time played, with

children in the experimental condition playing a median of 118 seconds, compared to 89 for the control condition (Z=-8.61, p<0.0001, r=0.07). A graph of the active time for both conditions is shown in Figure 2.5.

We calculate the number of unique levels each child played by counting levels with at least one game action. Since *Refraction* has an adaptive level progression, each child plays a different set of levels based on their incoming skill. However, since children are randomly assigned to either the experimental or control condition, we expect skill to be evenly distributed across conditions. Our analysis showed that *intervention* has a significant effect on levels played (Z=-9.04, p<0.0001, r=0.07). Children in the experimental condition played more levels, a mean of 6.7 levels compared to 5.5 for those in the control. The median was 2 in both conditions.

These results suggest that children who play the growth mindset version of *Refraction* are more persistent than those who play the control version. They stay in the game significantly longer and play significantly more levels. However, both of these effects are very small. Given the BrainPOP environment, where the majority of children play for less than three minutes, this is not surprising. Therefore, the fact that we observe a 33% increase in median time played is encouraging, despite the small effect size.

The Intervention Promotes Growth Mindset Behavior

We expected children who played the experimental version of *Refraction* to display behaviors consistent with the growth mindset more frequently than those who played the control version. To evaluate this hypothesis, we analyzed an outcome measure that combines the four metrics used to award brain points to children in the experimental version. Recall from the Experiment Design section that we used a *new hypothesis* metric, an *empty board* metric, a *math* metric, and a *moves* metric to decide when to give brain points. In both versions of *Refraction*, we log an event every time one of the four metrics is triggered. We combine these four metrics to measure how frequently children exhibit the behaviors incentivized by the brain points system.

To combine the metrics, we summed the number of times each metric was triggered during play. Since children in the experimental condition play for significantly longer, they have more opportunities to trigger these metrics. To control for the amount of time played, we divided our combined metric by the number of minutes of active time played. This produced the average number of metric triggers per minute, our strategy outcome measure. Our analysis showed that *intervention* had a significant effect on this combined measure. Children in the experimental condition triggered an average of 2.49 metrics per minute, compared to 2.18 triggers per minute for the control (Z=-8.18, p<0.0001, r=0.07).

These results show that children who play the growth mindset version of *Refraction* learn to use the strategies incentivized by the brain points system. While the size of this effect is small, it is encouraging to see that such a short intervention can change the way children approach a problem, teaching them to exhibit growth mindset behaviors.

The Intervention Makes More Struggling Children Persist

We expected the experimental version of *Refraction* to encourage more struggling children to persist because effort was incentivized rather than speed or performance. To evaluate this hypothesis, we grouped players into three performance levels by measuring how quickly they mastered the first two concepts in the adaptive level progression. For each of the 483 players who mastered both concepts, we calculated the number of levels needed to complete the concepts. The minimum number of levels to master the concepts is six, and the maximum is twenty. To determine our grouping criteria, we ordered players by the number of levels they needed to master the concepts, and divided them into three equal groups. We consider children who complete the concepts in seven or fewer levels to be advanced, eight to ten levels to be average, and more than ten levels to be struggling.

To determine which types of players are encouraged to persist in each version of the game, we analyzed the performance level distribution of children who completed the first two concepts and reached the challenge puzzle that appears after thirty minutes of play. We found that *intervention* had a significant effect on performance level after thirty minutes

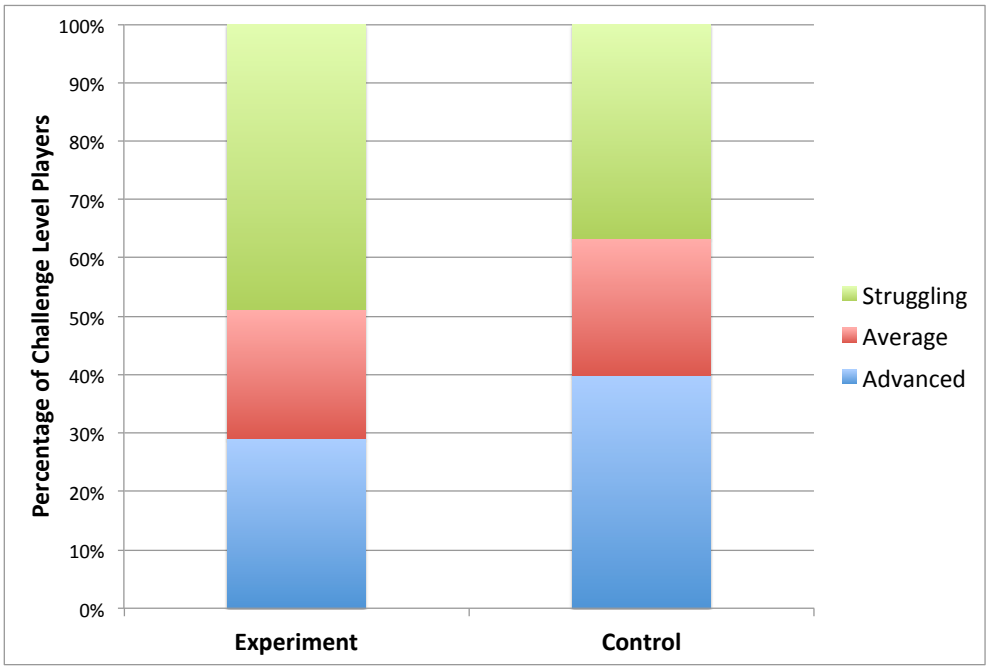


Figure 2.6: **Performance Distribution Graph.** A graph showing the performance level distribution of players who persist for thirty minutes and reach the challenge level. The brain points version of *Refraction* is better at retaining struggling players.

(χ^2=8.00, p<0.05, V=0.13). As shown in Figure 2.6, 49% of children in the experimental version were labeled as struggling, 29% were advanced, and 22% were average, compared to 37% struggling, 40% advanced, and 23% average for the control. This suggests that our growth mindset intervention encourages more low performing students to persist for extended periods of time in *Refraction*.

The Intervention May Improve Reaction to Challenge

We expected children who played the experimental version of the game to react more favorably to an especially challenging level than children who played the control version. To evaluate this hypothesis, we analyzed children's behavior in the "Challenge Level" given to players thirty minutes into the game. Very few children play *Refraction* for long enough to

reach the challenge level, so our analysis only includes 281 players from the experimental condition and 248 from the control condition.

First, we analyzed the time played and combined strategy outcome measures used previously. We found no significant effect of *intervention* on amount of active time played (Z=-0.81, $n.s$) or the number of strategy metrics fired per minute (Z=0.69, $n.s$). We also looked at whether children win the challenge level, skip by clicking the "New Level" button, or quit the game entirely. We did find a significant effect of *intervention* on win rate, most likely due to the uneven performance distribution at this point in the game. Children in the control condition won 18% of the time, compared to 11% for the experimental condition (χ^2=4.79, p<0.05, V=0.1).There were no significant differences in the skip rate (χ^2=2.0, $n.s.$) or the quit rate (χ^2=0.08, $n.s.$).

Finally, we studied how children were impacted by struggle by measuring how long children continued playing *Refraction* after the challenge level. We calculated the amount of active time children spent playing after this level, and while there was no significant effect, the results trend in the hypothesized direction of children in the experimental condition playing longer (Z=-1.71, p=0.088, r=0.07). They played for a median of 381 seconds after the challenge level, compared to 258 for children in the control condition, despite the fact that the remaining group of students in the experimental condition had a larger percentage of low performers.

We had hoped to see the increased persistence and use of strategy observed in the standard *Refraction* levels transfer to the challenge level, where effort was not explicitly rewarded through brain points. While it is possible that children in the experimental condition were less motivated to persist because they did not earn brain points on this level, it is also possible that we saw no effects due to the population of players included in this analysis. Children who play *Refraction* for more than thirty minutes on BrainPOP are unusual; they may be more persistent than average, and could already be inclined towards the growth mindset. Future research will need to explore this question in more depth in a environment where we can assess and control for children's incoming mindsets.

Despite the overall lack of difference in children's behavior during the challenge level, we measured a promising trend showing that children in the experimental condition may play longer after experiencing struggle and failure than children in the control condition. This trend suggests that even among the unusual group of children who play *Refraction* for over thirty minutes, our brain points intervention may affect how long they are willing to persist. This is especially encouraging because a larger percentage of the children who persisted until the challenge level in the experimental condition struggled with the game. However, this trend is not significant and will need to be confirmed through future research.

2.3.5 Summary

This results of this initial study show that making a fundamental change to an educational game's incentive structure can positively impact children's behavior. We designed a new "brain points" incentive structure that shows children how to practice and achieve growth mindset behaviors, in addition to teaching them the theory behind this mindset. The study results are encouraging. Children were only exposed to our intervention for a short period of time, three minutes on average, and yet we were able to capture the effects it had on their behavior. Children in the experimental condition played longer and completed more levels than those in the control condition. They also learned to use the strategies incentivized by the brain points system, exhibiting strategic behavior more often than children in the control condition. Finally, an analysis of the children who played for thirty minutes showed that those in the experimental condition may persist longer after struggling with a challenging level than those in the control condition.

2.4 Study 2: Demographic Differences in Brain Points Effectiveness

The goal of our second study was to expand on our original research by studying the brain points intervention in a new environment, the online schooling website K12[4]. Our collabora-

[4]K12: http://www.k12.com/

tion with K12 provides us with access to students who play for much longer than students on the casual website BrainPOP, where we conducted our initial study. Furthermore, K12 provides demographic data for their students. Growth mindset interventions have been shown to have a particularly strong effect on traditionally underperforming groups [13, 19], so we were interested in measuring whether our intervention has a similar effect.

2.4.1 Experiment Design

We conducted our study on the online schooling website K12.com. K12 is a for-profit company that provides curriculum content for online public schools as well as for home-schooled students. K12 was interested incorporating educational games into their curriculum, so we partnered with them to release the games developed by our research group, the Center for Game Science, through their website. Links to the game *Refraction*, designed to teach fraction concepts, were provided under an "additional activities" heading in 64 different locations within the K12 curriculum for third, fourth, and fifth grade math.

Our K12 study has an identical design to the BrainPOP study described in the previous section. It has a single between-subjects factor *intervention* with two levels: experiment or control. The experimental version of the game included an introductory animation that describes the growth mindset, and used the brain points incentive structure that rewards effort, use of strategy, and incremental progress. The control version included an introductory animation that presents a neutral message, and used a "level points" incentive structure that awards points for each completed level. We chose to change the name of our control incentive structure from "fraction points" to "level points" for this second study to remove any negative connotations associated with math terminology. Otherwise the design was unchanged. We used this incentive structure for the control because progress is commonly rewarded in games.

2.4.2 Results

We collected data from 7,940 students in between October 2013 and August 2014 (3,924 students in the experimental condition and 4,016 in the control). We report an analysis of two

Cond.	**Time** (minutes)	**Strategy** (per minute)
Experiment	14.7	2.9
Control	10.8	2.7

Table 2.5: **Intervention Results.** The median time played and strategies used for students in the experimental condition and the control condition.

Gender	**Time** (minutes)	**Strategy** (per minute)
Male	19.3	3.0
Female	12.5	2.8

Table 2.6: **Gender Results.** The median time played and strategies used for students of different genders.

outcome measures. The first captures student persistence by measuring the amount of time played. The second captures use of strategies that are associated with the growth mindset by combining the four metrics used to award brain points, as described in the previous section. We use non-parametric statistics because our data is non-normally distributed.

First, we analyzed our data to determine the impact of the *intervention* factor. We found that students in the experimental condition were significantly more persistent (Z=6.81, p<0.0001) and exhibited significantly more growth mindset behaviors (Z=4.93, p<0.0001) than students in the control condition. Median values are given in Table 2.5. These results confirm the findings from our original study, showing that the brain points intervention promotes students' persistence and use of strategy.

Next, we analyzed the impact of student demographics. For this analysis, we only included students who reported demographic information to K12, a total of 2,024 players in the experimental condition and 2,064 in the control. We studied three demographic factors: *gender*, *grade*, and *free/reduced lunch status*. We were particularly interested in measuring interactions between the *intervention* factor and each demographic factor, so we performed a factorial analysis. Since our data was non-normally distributed, we applied the Aligned Rank Transform [54] procedure, which aligns and ranks non-parametric data so that a standard ANOVA model can be used to perform a factorial analysis. We used the ARTool program developed by Wobbrock et al. to align and rank our data [145].

Grade	**Time** (minutes)	**Strategy** (per minute)
3rd	24.3	2.7
4th	19.1	2.8
5th	16.1	3.0
6th	9.5	3.1

Table 2.7: **Grade Results.** The median time played and strategies used for students of different grades.

Free Lunch	**Time** (minutes)	**Strategy** (per minute)
Yes	12.8	2.7
No	19.1	3.0

Table 2.8: **Free Lunch Results.** The median time played and strategies used for students of who were and were not eligible for free/reduced lunch.

The *gender* factor has two levels: male and female. We found that male students played for significantly longer than female students ($F(1,4084)=42.74$, $p<0.0001$), and that male students used significantly more growth mindset strategies than female students ($F(1,4084)=56.41$, $p<0.0001$). The median values for each metric by gender are given in Table 2.6. However, the *intervention*gender* interaction did not have a significant effect on either metric, showing that the brain points intervention was equally effective for students of both genders.

The *grade* factor has four levels: 3rd, 4th, 5th, and 6th grade. We found that younger students played for significantly longer ($F(1,4084)=63.61$, $p<0.0001$), but that older students used significantly more growth mindset strategies ($F(1,4084)=37.9$, $p<0.0001$). The median values are given in Table 2.7. While *intervention*grade* did not have a significant effect on either outcome measure, the results for the time played measure trend towards younger students being more positively impacted by the brain points intervention than older students ($F(1,4084)=2.44$, $p=0.06$). This suggests that the intervention may be more motivating for younger students.

The *free/reduced lunch status* factor has two levels: eligible for free and/or reduced lunch, and not eligible. Lower income students in the United States are eligible for free or reduced school lunch, and as a result lunch status is often used as a proxy measure for socioeconomic status. For this analysis, we grouped all students who were eligible for

free and/or reduced lunch and compared them to those who were not eligible. We excluded students with an unknown lunch status. We found that eligible students play for significantly less time ($F(1,3226)=43.70$, $p<0.0001$) and use significantly fewer growth mindset strategies ($F(1,3226)=59.47$, $p<0.0001$) than students who are not eligible. The median values are given in Table 2.8. More importantly, while the *intervention*lunch status* interaction did not have an effect on strategy use, it did have a statistically significant effect on time played ($F(1,3226)=4.67$, $p=0.03$). The brain points intervention was more effective at engaging students of higher income and encouraging them to play for long periods of time.

2.4.3 Summary

The results of this second study replicate the findings from our first study, showing that our growth mindset intervention has a positive impact on student persistence and strategy use. However, we also found that it does not benefit all students equally. The intervention may be more engaging for younger students, and is less engaging for lower income students. While we hoped that brain points would have a stronger effect on female students, as has been measured for other growth mindset interventions [19], we found that it has the same effect on students of both genders.

These results may say more about *Refraction* than about the brain points incentive structure. The game itself may be less appealing to girls and lower income students, and the cartoony style of the game characters may be more engaging for younger students. However, it is also possible that the effects are caused in part by the design of the intervention. The growth mindset messages used in the game are all presented through text in the user interface, which might be less engaging for students who are weak readers. We are currently exploring whether presenting growth mindset messages using an audio voiceover improves the effectiveness of the intervention for lower income students. We are also working on integrating brain points into other educational games to determine whether these effects are primarily driven by the design of *Refraction* or the design of the brain points intervention.

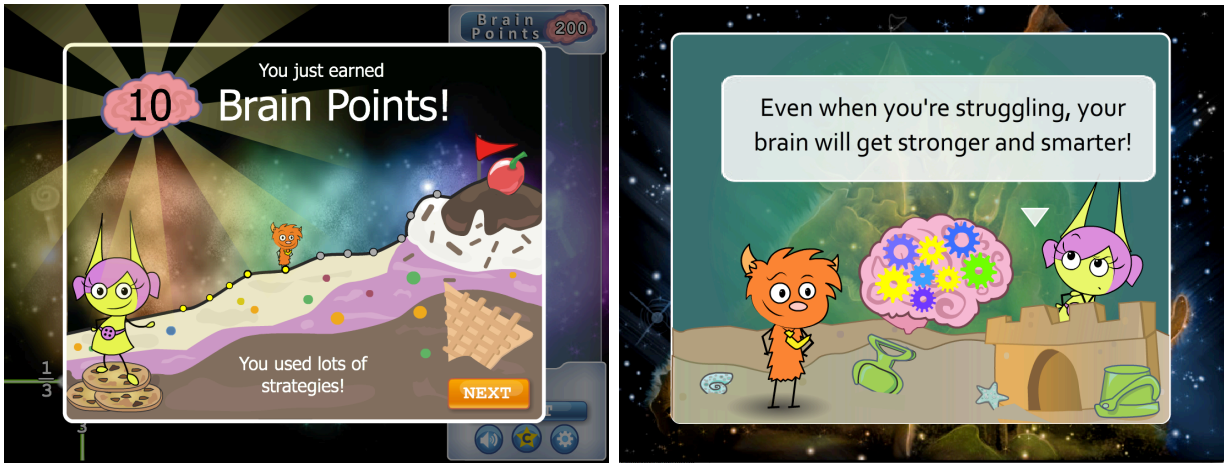
2.5 Study 3: A Deeper Look at the Brain Points Design

The goal of the third study was to expand on our previous work to gain a deeper understanding of the growth mindset intervention. While our initial studies showed encouraging results of the intervention's impact on persistence, strategy use, and reaction to challenge, they left a number of important questions unanswered. In particular, it was not clear whether the mindset narrative, brain points incentive structure, or progress visualization had the strongest effect on player behavior. Pevious work shows that the growth mindset can be taught directly [19], so it is possible that the narrative contributed the majority of the effect. The original study also compared brain points awarded during gameplay to a control where points were awarded after each completed level. It is possible that children simply enjoy earning points while playing, and that rewarding growth mindset behaviors is unimportant. Finally, it is unclear whether the progress visualization contributes to the observed effects.

We explore these unanswered questions through a large-scale experiment that compares five alternate versions of the growth mindset intervention. Our results provide new insights about the design of the brain points intervention that improve our understanding of how to promote growth mindset behaviors through educational games. We believe our findings will prove instrumental for future efforts to generalize this approach to other learning contexts.

2.5.1 Experiment Design

To tease apart the impact of each component of the brain points intervention, we designed an experiment that compares five different versions of *Refraction*. The first version is a control based on our original intervention from the first study. The remaining four versions either remove or modify components of the intervention to address specific research questions. In our analysis, we compare each modified version to to the control version, which allows us to measure the added impact of each component of the design. We leave pairwise comparisons of the modified versions for future work. In this section, we describe the control and then present our four research questions and the game versions used to address each question.



(a) (b)

Figure 2.7: **Intervention Modifications.** Figure (a) shows a screenshot of a summary screen with the new art for the Copper and Zuzu characters that we developed based on feedback from user studies with students. Figure (b) shows a screenshot from one of the three new animations we added to strengthen the growth mindset message.

Control

The control version of *Refraction* used for this study was nearly identical to the experimental version from the first study, described in detail in Section 2.3.1. This version incorporates three central intervention components: a growth mindset narrative presented through an introductory animation, the brain points incentive structure, and a progress visualization. We made a few minor modifications to the intervention design for this experiment. First, we included three additional growth mindset narrative animations, shown after the player completes the first three *Refraction* planets, which provide additional detail about the growth mindset. The animations are designed to directly teach the growth mindset and reinforce the value of strategy use and productive struggle. Second, we updated the art for the characters Copper and Zuzu based on feedback from students during formative user testing. Screenshots of these changes are shown in Figure 2.7.

RQ1: Does the growth mindset narrative matter?

The first question we wanted to explore was whether the growth mindset narrative presented through the sequence of four animations have a strong impact on player behavior. We hypothesized that the narrative would make an important contribution, increasing persistence and improving players' reaction to challenge. To answer this question, we created a version of *Refraction* that did not include the four animations. After loading this version of the game, players are immediately dropped into the first level, without any introduction to the growth mindset or the brain points incentive structure. Players receive points just as they would in the control version, and they see the progress visualization after each level.

RQ2: Do brain points matter?

Next we wanted to explore whether brain points impact the effectiveness of the intervention. We hypothesized that points would contribute to encouraging growth mindset behaviors. However, since many effective mindset interventions focus exclusively on textual narratives [19], it is possible that brain points have no impact on student persistence, strategy use, and reaction to challenge. To explore this question, we created a new version of the intervention that included the growth mindset animations but no points.

In this version, players begin by watching a modified version of the introductory animation that teaches the growth mindset but does not mention the brain points incentive structure. After the animation, children play the game exactly as they would in the control condition, except that that no points are earned during gameplay. Since the progress visualization displayed after each level is centered around earning points, we removed the summary screen in this version. Instead, we display a neutral dialog box, shown in Figure 2.8(a). While players do not earn points in this version, we silently calculate the number of brain points each player would have earned. We use this point count to determine when to advance players to the next *Refraction* planet, so that players advance at the same rate that they would have in the control condition.

(a) (b)

Figure 2.8: **Condition Designs.** Figure (a) shows a screenshot of the neutral dialog box displayed after each level in the condition with no brain points. Figure (b) shows a screenshot of the neutral summary screen used in the condition with no mountain visualization.

RQ3: Do brain points need to reward specific behaviors?

We were also interested in studying whether brain points are more effective when they reward specific behaviors. We hypothesized that rewarding growth mindset behaviors would contribute to the effect. However, it is possible that players simply enjoy earning points during gameplay, and that rewarding growth mindset behaviors is unimportant. To explore this question, we developed a version of the game that awarded points randomly, rather than in response to strategic behavior.

In this version, players view the same growth mindset narrative and progress visualization as in the control condition. Players still earn points during gameplay, but we do not use the combined strategy metric to determine when to award points. Instead, we set a timer to award points after a certain number of seconds. When the random points timer finishes, we wait until the player makes a move, and then award brain points. We chose this approach to make the points seem less random, since earning points when you have not made any move feels unnatural. When players earn random points, the "Working Hard" message is

displayed to insinuate that the points are rewarding effort. The "Working Hard" message is always displayed on the summary screen in this version.

To ensure that random points are awarded at the same rate as brain points, we randomly select each wait time from the distribution of times between brain points in the data set collected for the first study. However, as we discuss in detail in the analysis section, random points are awarded significantly more often than brain points. This is because we award random points using this time distribution during all levels the game. In the control condition, it is more difficult to earn brain points during the four tutorial levels at the beginning of the game because these levels do not include fractions.

RQ4: Does the progress visualization matter?

Finally, we wanted to study whether the progress visualizations displayed on the summary screens contribute to the effects measured in the original study. We hypothesized that the visualization would contribute to engagement, since it provides a clear indication of player progress through the game. However, it is possible that the visualization and mountain metaphor are too abstract to have a strong impact on children. To study this question, we created a version of the intervention that showed a summary screen without the mountain visualization. The summary still displays the points earned during the previous level and the growth mindset message. However, the mountain is replaced by an animated image of a spaceship, as shown in Figure 2.8(b).

2.5.2 Method

To address our research questions, we studied how children played the five versions of *Refraction* on the educational website BrainPOP. This is the same website used in our first study. As in the previous studies, we measure how our interventions impact observable in-game behaviors that are associated with the growth mindset. We describe our behavioral metrics and data collection process in detail below.

Behavioral Metrics

To measure student persistence and growth mindset behavior, we use the same metrics as in the previous to studies. To capture student persistence, we use two outcome measures: the amount of time children spend playing the game, and the number of levels they play. To measure whether students exhibit the growth mindset behaviors rewarded by brain points, we use an outcome measure that combines the four metrics used to award brain points in the control version of *Refraction*, described in detail in Section 2.3.1.

In addition to observing behavior during standard game levels, we wanted to measure how children are impacted by a particularly challenging level. We use a challenge level design similar to the one described in Section 2.3.1 with a few minor modifications. Rather than displaying the challenge level after 30 minutes of play, children are given access to the challenge when they earn sixty points. We also changed the design to give children more opportunities to choose whether or not to challenge themselves. The challenge is framed as a special reward given to players for their progress in the game. Screenshots of the challenge level and associated messages are shown in Figure 2.9.

When children earn a challenge level, they are given the option of either playing it immediately or skipping the level. Players can return to the challenge at any point during gameplay by clicking the "C" button on the game's sidebar. This allows persistent children to attempt the challenge multiple times. We wanted to measure whether effects produced by the growth mindset intervention would transfer to levels where effort is not explicitly rewarded, so no points were given during the challenge level. Players could exit the challenge at any point by clicking the "New Level" button on the game sidebar. The challenge-related text is identical in all versions of the game except the "no points" version, in which references to points are removed.

We capture player behavior in relation to the challenge level using four different metrics. For all of these analyses, we only include children who play the game long enough to earn the challenge level. First, we calculate the number of times each player attempts the challenge.

(a) (b) (c)

Figure 2.9: **Challenge Level.** We designed a "challenge level" to observe how children react in the face of a particularly difficult puzzle. Figure (a) shows the message displayed when the player earns the challenge level. Figure (b) shows the introductory message displayed when the player starts the challenge level. Figure (c) shows the challenge level interface. The player can skip the challenge at any point by clicking the "New Level" button. For the no points condition, all textual references to points are removed from these dialog boxes.

Next, we calculate whether each player wins the challenge. Since the level is very difficult, and players can skip the level at any time, we expect only a small percentage of children to win. We capture persistence by calculating the amount of active time each player spent on the challenge. Finally, we calculate the average number of growth mindset behaviors per minute using the combined measure described above. These measures provide us with a robust picture of how children in each condition react to a particularly challenging level.

Data Collection

To collect our data, we modified *Refraction* to randomly assign BrainPOP players to one of the five versions of the game. We only include new players in our analysis, and only use data from a player's first game session to control for issues with shared computers in school. We track players by storing information in the Flash cache, which allows us to exclude returning players. One disadvantage of this approach is that a player who clears the cache or

switches computers will be treated as a new player in our experiment. However, this effect is distributed evenly across conditions due to randomization.

We collected data on BrainPOP between September 3 and October 28, 2015. *Refraction* was featured on the front page of the BrainPOP game portal once per week during this period, allowing us to attract large numbers of students. We randomly selected 5,000 players per condition from the total data set of 26,662 players to include in our analysis.

2.5.3 Data Analysis and Results

We study the effects of our modified versions of the growth mindset intervention by comparing each to the control version of the intervention. We chose not to compare the four experimental versions to each other because this would result in a large number of pairwise comparisons. As a result, we are not able to formally reason about the relative impacts of the four versions. However, this analysis allows us to effectively explore our four research questions, which focus on understanding the added impact of each aspect of the intervention design. We leave comparative analyses of the four modified intervention designs for future work.

Before performing our analysis, we evaluated the Kolmogorov-Smirnov test to assess the normality of our data, and found that it was statistically significant for all of our outcome measures. As a result, we chose to use non-parametric statistical methods. We use a Wilcoxon rank sums test and an r measure of effect size for continuous variables, and a Chi-square statistic and a Cramer's V measure of effect size for nominal variables. We report effect sizes in addition to p-values to show the magnitude of the differences between our populations, since we are likely to find many significant differences due to our large sample sizes. For both tests, effect sizes with values less than 0.1 are considered *very small*, 0.1 are *small*, 0.3 are *moderate*, and 0.5 or greater are *large*. The results of our statistical analyses are reported in Tables 2.9, 2.10, 2.11, and 2.12. We describe these results and their implications in the following sections.

Condition	Active Time Played			# Levels Played		
Control	2.54 min	$N = 10,000$	$p < 0.0001$	4 levels	$N = 10,000$	$p < 0.0001$
No Narrative	3.01 min	$Z = -5.05$	$r = 0.05$	4 levels	$Z = 4.74$	$r = 0.05$
Control	2.54 min	$N = 10,000$	$p < 0.0001$	4 levels	$N = 10,000$	*n.s.*
No Points	2.21 min	$Z = -4.89$	$r = 0.05$	4 levels	$Z = -1.49$	
Control	2.54 min	$N = 10,000$	$p < 0.01$	4 levels	$N = 10,000$	$p < 0.0001$
Random Points	2.19 min	$Z = -2.66$	$r = 0.03$	3 levels	$Z = -5.62$	$r = 0.06$
Control	2.54 min	$N = 10,000$	$p < 0.0001$	4 levels	$N = 10,000$	$p < 0.005$
No Progress Viz	2.26 min	$Z = -3.80$	$r = 0.04$	3 levels	$Z = -2.89$	$r = 0.03$

Table 2.9: **Persistence Results.** Results from the statistical tests analyzing children's persistence while playing *Refraction*, specifically the amount of active time children spend playing the game and the number of levels attempted. In each column, the median values for each condition are reported on the lefthand side.

Condition	Growth Mindset Strategies Per Minute		
Control	2.40 strategies	$N = 10,000$	$p < 0.0001$
No Narrative	2.61 strategies	$Z = 7.08$	$r = 0.07$
Control	2.40 strategies	$N = 10,000$	$p < 0.0001$
No Points	2.71 strategies	$Z = 8.47$	$r = 0.09$
Control	2.40 strategies	$N = 10,000$	$p < 0.0001$
Random Points	2.02 strategies	$Z = -7.22$	$r = 0.07$
Control	2.40 strategies	$N = 10,000$	*n.s.*
No Progress Viz	2.46 strategies	$Z = 0.63$	

Table 2.10: **Growth Mindset Strategy Results.** Results from the statistical tests analyzing children's growth mindset strategy use while playing *Refraction*, specifically the average number of growth mindset behaviors they exhibit per minute. In each column, the median values for each condition are reported on the lefthand side.

Condition	# Challenge Attempts			Challenge Win Rate		
Control	1 attempt	$N = 1,580$	*n.s.*	7.69%	$N = 1,580$	*n.s.*
No Narrative	1 attempt	$Z = 0.05$		5.75%	$\chi^2 = 2.38$	
Control	1 attempt	$N = 1,309$	$p < 0.05$	7.69%	$N = 1,309$	$p < 0.05$
No Points	1 attempt	$Z = 2.07$	$r = 0.06$	11.19%	$\chi^2 = 4.67$	$v = 0.06$
Control	1 attempt	$N = 1,326$	$p < 0.01$	7.69%	$N = 1,326$	*n.s.*
Random Points	1 attempt	$Z = -2.78$	$r = 0.08$	7.69%	$\chi^2 = 0.00$	
Control	1 attempt	$N = 1,325$	*n.s.*	7.69%	$N = 1,325$	*n.s.*
No Progress Viz	1 attempt	$Z = 1.51$		9.55%	$\chi^2 = 1.14$	

Table 2.11: **Challenge Level Results.** Results from the statistical tests analyzing children's behavior while interacting with *Refraction*'s challenge level, specifically the number of times children attempt the challenge level and their win rate. In each column, the median values for each condition are reported on the lefthand side.

Condition	Active Time Played			Growth Mindset Behavior Per Minute		
Control	2.08 min	$N = 1,580$	*n.s.*	0.99	$N = 1,580$	*n.s.*
No Narrative	2.06 min	$Z = 1.12$		0.87	$Z = 1.52$	
Control	2.08 min	$N = 1,309$	$p < 0.05$	0.99	$N = 1,309$	$p < 0.05$
No Points	2.48 min	$Z = 1.99$	$r = 0.06$	1.10	$Z = 2.47$	$r = 0.07$
Control	2.08 min	$N = 1,326$	$p < 0.05$	0.99	$N = 1,326$	$p < 0.005$
Random Points	1.86 min	$Z = -2.50$	$r = 0.07$	0.73	$Z = -2.87$	$r = 0.08$
Control	2.08 min	$N = 1,325$	*n.s.*	0.99	$N = 1,325$	*n.s.*
No Progress Viz	2.33 min	$Z = 1.06$		1.06	$Z = 1.37$	

Table 2.12: **Challenge Level Behavior Results.** Results from the statistical tests analyzing children's behavior while interacting with *Refraction*'s challenge level, specifically the amount of active time they spend working on the challenge and the average number of growth mindset behaviors they exhibit per minute during the level. In each column, the median values for each condition are reported on the lefthand side.

Growth Mindset Animations Provide No Added Benefit

We expected the growth mindset narrative presented through animations to encourage persistence and strategic behavior. However, our analysis showed the opposite effect. Children in the no narrative condition played significantly longer, a median of 3.01 minutes compared to 2.54 minutes for the control. They also completed significantly more levels; the median was 4 levels for both conditions, but children in the no narrative condition completed a mean of 8.08 levels compared to a mean of 7.65 levels for the control. Finally, players who did not see the animations were more strategic, exhibiting a median of 2.61 growth mindset behaviors per minute compared to 2.40 per minute for those in the control.

These results surprised us. Previous research shows that interventions that teach the growth mindset directly can be effective [13, 19]. On the other hand, this is the first intervention to attempt to teach the growth mindset through in-game animations, and games research has shown that players dislike tutorials that interrupt gameplay [9]. This made us wonder if a large percentage of players quit during the introductory animation itself, leading to lower overall persistence. To explore this question, we performed a follow-up analysis that only included players in both conditions who completed the first level of the game.

This analysis tells a different story. A larger number of children complete the first level in the no narrative condition, 3,536 players compared to 3,278 players in the control. However, we found no significant difference in the amount of time played ($Z = 1.32$, $N = 6,814$, $n.s.$), with medians of 6.09 minutes for the no narrative condition and 6.38 minutes for the control. There was also no significant difference in the number of levels completed ($Z = -0.14$, $N = 6,814$, $n.s.$), with medians of 8 levels for both conditions. Finally, we found no significant difference in the number of growth mindset behaviors observed per minute ($Z = 0.57$, $N = 6,814$, $n.s.$), with medians of 2.95 for the no narrative condition and 2.93 for the control.

This analysis suggests that the growth mindset animation dissuades some children from playing the game, reducing overall engagement. However, the animations do not have a

negative impact on children's persistence or growth mindset behavior. This interpretation is further reinforced by our analysis of children's behavior during the challenge level. As reported in Tables 2.11 and 2.12, there were no significant differences between these two conditions for any of our challenge level measures.

There are a number of possible explanations for this result. It may be that children do not like watching animations during gameplay, and that those who persist through the animation do not pay attention to its content. We display growth mindset messages throughout the game, on the summary screens at the end of each level and on the brain points themselves. It is therefore possible that reinforcing the growth mindset narrative through animations is not necessary. We will need to confirm this hypothesis through further studies, however our findings indicate that growth mindset animations are not necessary to effectively promote desirable behaviors in *Refraction*.

Brain Points Increase Player Retention

We expected the control version of *Refraction* with the brain points incentive structure to be more effective at encouraging growth mindset behaviors than the version without points. Our analysis confirms that children in the no points condition play for less time, a median of 2.21 minutes compared to 2.54 minutes for the control. However, there is no significant difference in the number of levels completed. More surprisingly, players in the no points condition exhibit significantly more growth mindset behaviors during gameplay, a median of 2.71 per minute compared to 2.40 per minute for the control.

A smaller number of children persist until the challenge level in the no points condition, 581 compared to 728 in the control. However, our analysis of challenge level behavior shows that players in the no points condition are more persistent and strategic. They attempted the challenge significantly more times; the median was 1 for both conditions, but players in the no points condition tried the challenge slightly more, a mean of 1.06 times compared to 1 time for players in the control. Players in the no points condition spent more time on the challenge, used more growth mindset strategies, and won the challenge more often.

There are a number of possible interpretations of these results. It may be that the no points version of *Refraction* is more effective at encouraging use of strategy and persistence in the face of challenge. However, our analysis also shows that children in the no points condition completed the same number of levels as those in the control condition in less time. It is therefore more likely that the players who choose to persist in the no points version are more skilled at *Refraction*. Our second study of player demographics on the homeschooling website K12.com showed that older players exhibit more growth mindset behavior per minute than younger players. Since older players are typically more skilled at the game, this suggests that the combined growth mindset behavior metric may be correlated with incoming ability.

Further research is needed to understand the full implications of these results. It is clear from this analysis that brain points encourage a larger number of players to persist for long periods of time in *Refraction*. While it is likely that players who persist in the no points condition exhibit more strategic behaviors because they have a high level of incoming ability, it is not possible to confirm this hypothesis through our current study.

Rewarding Players Randomly is Not Effective

We expected that rewarding players with brain points randomly, instead of in response to growth mindset behaviors, would be less effective. Our analysis confirms this hypothesis. Players who received points randomly played for significantly less time, a median of 2.19 minutes compared to 2.54 for the control. They also played significantly fewer levels, a median of 3 compared to 4 for the control. Finally, they exhibited significantly less growth mindset behavior, a median of 2.02 per minute compared to 2.40 per minute for the control.

Fewer players persist until the challenge level in the random points condition, a total of 598 compared to 728 in the control. Those that did make it to the challenge level played fewer times; while the mean was 1 attempt for both conditions, those in the no points condition made a mean of 0.92 attempts, compared to 1 attempt for those in the control. Children in the random points condition spent less time working on the challenge level, used less strategy, and were less likely to win.

To confirm that brain points are not awarded more frequently than random points, we compared the number of points earned per minute in each condition. Recall that points are awarded in increments of five in both versions. We found that players in the random points condition received points significantly more often ($Z = 31.72$, $N = 10,000$, $p < 0.0001$, $r = 0.3$), a median of 5.31 points per minute compared to 2.39 per minute for the control. However, this difference is driven by a difference in earned points on the tutorial levels. Brain points are challenging to earn during the tutorial levels, but random points are awarded at the same rate on all levels.

These results suggest that randomly praising students for their effort is not effective. While it is possible that random points are less engaging because they are given out more frequently, this would not explain the differences in observed growth mindset behaviors and students' reaction to challenge during gameplay. As a result, it appears that the brain points incentive structure is successful specifically because it rewards students for their productive struggle and use of strategy. This finding has important implications. In order to effectively integrate this intervention into a new context, it will be necessary to develop new behavioral metrics that accurately capture growth mindset behaviors to determine when to award points.

The Progress Visualization Increases Player Retention

We hypothesized that the summary screen progress visualization would contribute to player engagement. Our analysis confirmed this hypothesis. We found that players with no visualization played for significantly less time, a median of 2.26 minutes compared to 2.54 minutes for the control. They also completed significantly fewer levels, a median of 3 levels compared to 4 levels for the control. However, the progress visualization had no impact on growth mindset behaviors. There was no significant difference in the growth mindset behavior per minute, nor in any of the challenge level measures.

This finding suggests that showing players their progress, where progress measures growth mindset behavior rather than number of levels completed, increases player retention. This suggests that children may be motivated by seeing their progress and anticipating the re-

ward of moving on to the next *Refraction* planet. The progress visualization is therefore a worthwhile addition to the growth mindset intervention.

2.5.4 Summary

The results of this third study provide new insights into the design of the brain points incentive structure, exploring questions left unanswered by our previous work. In particular, we explored the impact of the growth mindset narrative, brain points incentive structure, and summary screen progress visualizations on player persistence, use of strategy, and reaction to challenge. We found that narrative animations provide no added benefit to the intervention, and cause many players to quit before trying the first game level. However, we found that the progress visualizations and the brain points incentive structure both increase player persistence. Most importantly, we found that brain points are effective specifically because they reward desirable growth mindset behaviors. A similar incentive structure that rewards players at random intervals fails to encourage strategy use and persistence in the face of challenge.

While these results provide a deeper understanding of the design brain points intervention, a number of questions remain unanswered. We found that growth mindset animations had no positive impact on player behavior, however we cannot confirm why these animations were disengaging. It is possible that players never enjoy watching animations in casual games, however it's also possible that the cartoony style of the animations was unappealing to older students. We also found that players who did not receive brain points were more strategic and persistent in the face of challenge than players who did. This may be because more skillful players choose to persist in the no points version of the game, but we cannot confirm this hypothesis with the current data. Further research is needed to explore these questions.

Despite these limitations, our findings make an important contribution to our understanding of growth mindset interventions for educational games. We believe the brain points incentive structure could be generalized to other interactive learning environments, and we hope our research contributes to this goal in the future.

2.6 Limitations and Future Work

The research presented in this chapter has a number of limitations that should be addressed in future work. While we have measured how the brain points intervention impacts in-game behavior, we have not yet evaluated its impact on in-game learning and student mindsets. Furthermore, we have only implemented the intervention in a single learning environment, the educational game *Refraction*. More research is needed to understand how to generalize this approach to other learning environments and measure its effectiveness in new contexts. We discuss these limitations and areas for future work in more detail below.

2.6.1 Evaluating Learning and Mindsets

One of the biggest limitations of the work presented in this chapter is that we have not yet evaluated the impact of brain points *Refraction* on students' learning and mindsets. The majority of our experiments were conducted on BrainPOP, where students play for very short periods of time and we cannot formally assess changes in learning and mindsets through formal pre- and post-tests. In an ongoing experiment on K12, we using pre- and post-tests to study whether students who play the brain points version of *Refraction* learn more during play, and whether they develop a growth mindset. While the behavioral results presented in this chapter show that the intervention encourages student persistence and use of strategy, we have yet to confirm whether these behavioral changes translate into increased learning and reported growth mindset.

Once we confirm that brain points *Refraction* has a positive impact on student learning and mindsets, it will be important to measure whether students' growth mindset behaviors transfer to new learning environments. The ultimate goal of the brain points intervention is to change how students approach their own learning process by teaching them that their intelligence is malleable. Future work should evaluate mindset transfer and iterate on the intervention design if transfer is not effectively supported.

2.6.2 Generalizing Brain Points for New Contexts

Another important limitation of this work is that we have only implemented the brain points intervention in one learning environment. We believe that this intervention could be integrated into other educational games, and even used as a form of gamification in more traditional learning environments. However, there are many open questions about how to generalize our approach for new contexts.

The most challenging component of the intervention is the brain points incentive structure. In order to effectively implement this incentive structure in a new interactive learning system, the system must capture fine-grained information about student process during learning activities. Brain points cannot be added to a system that only collects a student's final answer to a problem. Once the new system is set up to log fine-grained process information, the designer must develop heuristics for the learning environment that detect strategy use and productive struggle. These heuristics will be unique to each learning environment, and it may require significant iteration to develop a set of heuristics that accurately captures growth mindset behaviors. Additional research will be required to study heuristic design in more depth, with the goal of developing general design recommendation to support the extension of this approach to a broad array of interactive learning environments.

Once the brain points intervention has been successfully integrated into other learning environments, an number of research questions can be explored to better understand the intervention's effectiveness. The research presented in this chapter studies the intervention in the context of a game that targets third to fifth grade students. We do not yet know whether the intervention will be effective in non-game-based learning environments, or whether it will appeal to older students. Furthermore, it is possible that some of the results presented in this chapter only occur in *Refraction*, and do not transfer to other contexts. In order to truly understand the effectiveness of this approach, we must study the generality of our findings across multiple learning environments and student populations.

2.7 Conclusion

In this chapter, we demonstrated that fundamentally changing a game's incentive structure to promote the growth mindset can have a positive impact on students' behavior. We presented "brain points," a system that rewards students for their effort, use of strategy, and incremental progress. Unlike previous mindset interventions, this incentive structure provides children with real-time feedback as they work to develop growth mindset behaviors. In this chapter, we described results from three studies that explore the behavioral impact of brain points. The first study showed that persistence and use of strategy are encouraged in the educational game *Refraction* through the introduction of this unorthodox incentive structure. The second study replicated these findings and explored how brain points impact students from different demographic groups. Finally, the third study provided deeper insights into why the growth mindset intervention is effective by comparing five different versions of the intervention. In combination, the results of these studies demonstrate the behavioral impact on incentivizing growth mindset behaviors in an educational game, and provide insights that can contribute to the future generalization of these ideas to new learning environments.

Chapter 3

A FRAMEWORK FOR AUTOMATICALLY GENERATING INTERACTIVE INSTRUCTIONAL SCAFFOLDING

This chapter considers the challenge of generating interactive instructional scaffolding to teach procedures across multiple learning domains. Interactive learning environments such as intelligent tutoring systems and software tutorials often teach procedures with step-by-step demonstrations. This instructional scaffolding is typically authored by hand, and little can be reused across problem domains. We present a new framework for generating interactive tutorials from an algorithmic representation of the problem-solving thought process. Given a set of mappings between programming language constructs and user interface elements, we step through this algorithm line-by-line to trigger visual explanations of each step. This approach allows us to automatically generate tutorials for *any* example problem that can be solved with this algorithm. In this chapter, we describe two prototype implementations in the domains of K-12 mathematics and educational games, and present results from two user studies showing that educational technologists can author thought-process procedures and that generated tutorials can effectively teach a new procedure to students.

3.1 Introduction

Procedural knowledge is required for a wide range of human activities, from cooking to mathematics to using software applications. Many of the concepts taught in interactive learning environments like cognitive tutors, educational games, and software tutorials are procedural in nature. Some procedures, such as opening a file in a text editor, are simple. Others, such as long division, involve complex control flow structures like loops and conditionals. Teaching complex procedures in interactive learning environments presents many challenges.

One effective method of teaching procedures is by providing scaffolding that helps students solve problems that would otherwise be beyond their reach [43, 146]. While there are many different approaches to scaffolding learning, interactive learning environments typically use step-by-step demonstrations, tailored progressions of practice problems, and individual feedback to support students. Intelligent tutoring systems provide student with adaptive sequences of problems and personalized hints [10, 29, 140], and software tutorials often use step-by-step demonstrations to teach procedural tasks [17, 62].

While interactive scaffolding is effective, this type of content is often time-consuming to create because it must be authored by hand [6, 45]. As a result, many researchers have explored methods of generating interactive explanations automatically [7, 17, 20, 28, 45, 49]. While these approaches have significantly reduced the amount of effort required to author scaffolding, they often apply to only one application domain or require the designer to write separate content for each example problem. In some domains, such as K-12 mathematics, we may want to demonstrate a procedure for a large number of practice problems and produce multiple different types of scaffolding. Ideally, we would like an application-independent method of explaining a given procedure for *any* example problem.

In this chapter, we introduce a new framework for automatically generating interactive tutorials to teach the entire space of problems that can be solved with a given procedure. In our framework, an educational technologist encodes the problem-solving thought process as an algorithm and provides a mapping between programming language constructs and visual objects in the user interface. By stepping through this algorithm line-by-line, we automatically generate visual explanations for each step. This approach has a number of advantages. Given a set of mappings and an encoded algorithm, our framework automatically produces tutorials for any example problem in the domain. Furthermore, our framework naturally supports the creation of many different types of interactive scaffolding, including problem progressions, just-in-time feedback, and fading worked examples.

We demonstrate the generality of our approach by showing how it applies to two distinct problem-solving domains: a K-12 mathematics application and an educational game.

Through a user study with fifth-grade students, we show that our generated tutorials can produce learning gains similar to those of a human teacher. We also show that that educational technologists can encode an algorithm to explain subtraction in just a few hours. We believe this approach can greatly improve the process of authoring interactive scaffolding, particularly in domains with large numbers of practice problems.

3.2 Related Work

Researchers have explored many methods of designing and authoring instructional materials to teach procedural knowledge through computer applications. In this section, we provide an overview of existing work in two domains: intelligent tutoring systems and software tutorials.

3.2.1 Intelligent Tutoring Systems

Intelligent tutoring systems (ITS) are applications that emulate one-on-one tutoring with a human teacher [10, 29, 140]. These systems teach procedural concepts using personalized progressions of practice problems and contextual feedback [140], and they have been shown to produce learning gains in classroom studies [68]. While ITSs have great potential, they have not been widely adopted because they are expensive to build [29]. It has been estimated that 200-300 hours of expert design is needed to produce one hour of ITS content [6].

Researchers have explored many methods of reducing tutor authoring effort. Brawner et al. provide design recommendations for authoring tools, highlighting the importance of domain-independent methods [24]. Sottilare describes efforts to develop a Generalized Intelligent Framework for Tutors based around a standard ontology for representing knowledge [132]. Stamper et al. developed the Hint Factory to make it easier to add ITS-style interaction to existing educational software by using Markov decision processes to automatically generate contextual hints from past student data [133].

Others have focused on reducing the expertise required to author content for an intelligent tutoring system. Blessing et al. developed a cognitive model SDK that allows authors to edit rule hierarchies through a user interface to reduce the need for programming exper-

tise [20]. Aleven et al. developed the Cognitive Tutor Authoring Tools (CTAT) to allow designers to create example-tracing tutors without programming [7]. With CTAT, designers demonstrate correct and incorrect procedures for an example problem to create a "behavior graph" that can be generalized to tutor multiple isomorphic problems. CTAT has been shown to reduce development costs by a factor of 4 to 8 [7]. Olsen et al. extended CTAT to support the authoring of collaborative ITSs built around collaborative scripts [100]. We take a different approach to improving the scaffolding authoring process. Rather than designing tools to reduce the expertise required to author tutors, we aim to maximize the content that programmers can create.

3.2.2 Software Tutorials

Since many tasks in software applications involve following procedures, there is a body of research in software usability that explores methods of teaching procedural knowledge. This work has shown that step-by-step tutorials are effective instructional tools, especially when they provide interactive elements such as stencils [62], videos that track the user's progress [114], and interface highlighting [17].

While step-by-step tutorials are effective, authoring this content can be time-consuming [17, 45, 115]. To address this issue, some researchers have focused on improving the quality of video tutorials, since these are comparatively low-cost to create. Pongnumkul et al. developed Pause-and-Play, a system that links events in a video to user events in the target application, pausing the tutorial when the user lags behind [114]. Others have focused on adapting existing tutorials for new domains. Ramesh et al. developed ShowMeHow, a system that automatically transfers step-by-step tutorials between similar applications such as Photoshop and GIMP [115].

Researchers have also explored methods for generating tutorials automatically [28, 45]. One approach is to generate tutorials directly from user demonstrations of a procedural task. Grabler et al. developed a system for automatically generating text and image tutorials from demonstrations in an instrumented version of GIMP [45]. Chi et al. developed MixT, which

automatically generates mixed media tutorials with both text and video from user demonstrations in Photoshop [28]. While these systems produce tutorials automatically, they only work for specific domains like image manipulation, and the tutorials cannot always be applied to new problems. Furthermore, these systems only support linear procedures. In contrast, our framework supports complex non-linear procedures, applies to multiple domains, and generates tutorials for any example problem that can be solved by the procedure.

Another approach is to generate tutorials that show users how to recreate an artifact in a user application. For example, Harms et al. augment a programming environment to automatically generate stencil-based instructions from code snippets [49], and Li et al. automatically generate tutorials to recreate AutoCAD drawings [73]. These systems can automatically produce tutorials for multiple example problems, but provide little control over how the tutorial teaches the procedure. In educational domains, it is important to teach students by describing the problem-solving thought process.

The work that relates most closely to ours is the DocWizards system by Bergman et al., which learns a procedure from multiple user demonstrations [17]. The DocWizards interface exposes a human-readable version of the learned procedure so a new user can step through it line-by-line. At each step, the current line is highlighted and the widget involved in the next action is circled in the user interface. While the DocWizards system uses an algorithmic representation of a procedure to automatically generate tutorials, it was not designed to explain the thought process. The learner is expected to read and understand $if-else$ statements, which significantly reduces the usability of this system. Furthermore, the procedure must be demonstrated on many distinct problems to create an accurate underlying algorithm, and the complexity of the procedures that can be represented is limited.

3.3 Framework Design

We present a new framework for generating step-by-step tutorials to teach procedural knowledge. In our framework, an educational technologist encodes the problem-solving thought process as an algorithm. Using a domain-independent interpreter, we step through this al-

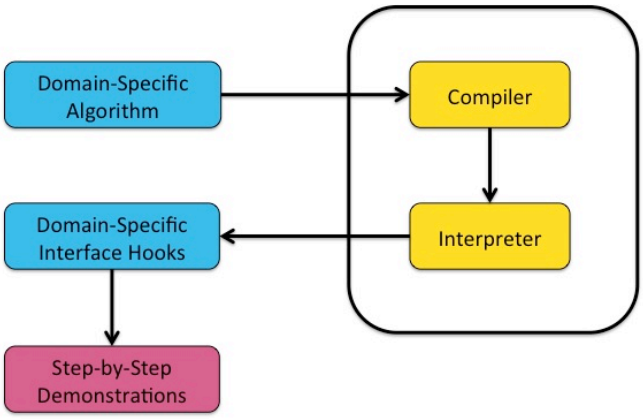


Figure 3.1: **Framework Workflow.** A domain-specific TPL algorithm representing the problem-solving thought process is given as input, compiled, and then passed to the interpreter. The interpreter steps through the algorithm line-by-line and throws events, which are passed to the domain-specific interface hooks, which generate visual explanations.

gorithm line-by-line. We explain each line using a set of mappings between programming language constructs and visual objects in the user interface, provided by the educational technologist. This approach allows us to automatically explain any problem that can be solved with the procedure. Figure 3.1 shows this workflow.

3.3.1 Thought Process Language

To complete a procedural task, a student must follow a certain thought process to decide when to perform each action. A simple procedure may be a list of actions to be executed in order, while a complex procedure will require the student to choose which steps to execute based on features of the current problem. To solve a subtraction problem, for example, the student needs to decide whether or not to borrow by comparing the top and bottom digits in a column.

We encode this problem-solving thought process in a language we call the Thought Process Language (TPL). The TPL algorithm for subtraction is given in Algorithm 1. This

Algorithm 1 Given as input a 2D array *table* containing a minuend p, represented as a set of digits $p_0, p_1, ..., p_m$ located in cells (0,0) to (0, m), and a subtrahend q, represented as a set of digits $q_0, q_1, ..., q_n$ in cells (1,$m-n$) to (1, m), subtract:

```
1:  procedure SUBTRACT(table)
2:      for each column in table.GetColsFromRight() do
3:          top := column.GetCell(0)
4:          bottom := column.GetCell(1)
5:          if bottom.NotEmpty() then
6:              if top.value < bottom.value then
7:                  borrow := table.LeftOf(top)
8:                  while borrow.value == 0 do
9:                      borrow := table.LeftOf(borrow)
10:                 borrow.value := borrow.value − 1
11:                 while borrow.NotNextTo(top) do
12:                     borrow := table.RightOf(borrow)
13:                     borrow.value := 9
14:                 top.value := top.value + 10
15:             result := column.GetCell(2)
16:             result.value := top.value − bottom.value
17:         else
18:             result := column.GetCell(2)
19:             result.value := top.value
```

algorithm solves all single-digit and multi-digit subtraction problems. TPL constructs are standard to many programming languages, and are not a contribution of this work. TPL includes basic primitives like variables and constants, as well as expressions that can include integer operators ($+$, $-$, $*$, $/$) or Boolean operators ($<$, $>$, $==$, $!=$). TPL has three types of statements: assignments, conditionals, and loops (for, while, do-while, for-each). TPL is based on object-oriented programming languages and also supports objects, object fields, and object methods. Additionally, TPL includes functions and a procedure can reference multiple sub-procedures.

3.3.2 Compiler and Interpreter

We generate explanations for a given problem by stepping through the TPL algorithm line-by-line like a debugger. To accomplish this, we use a compiler and an interpreter that are

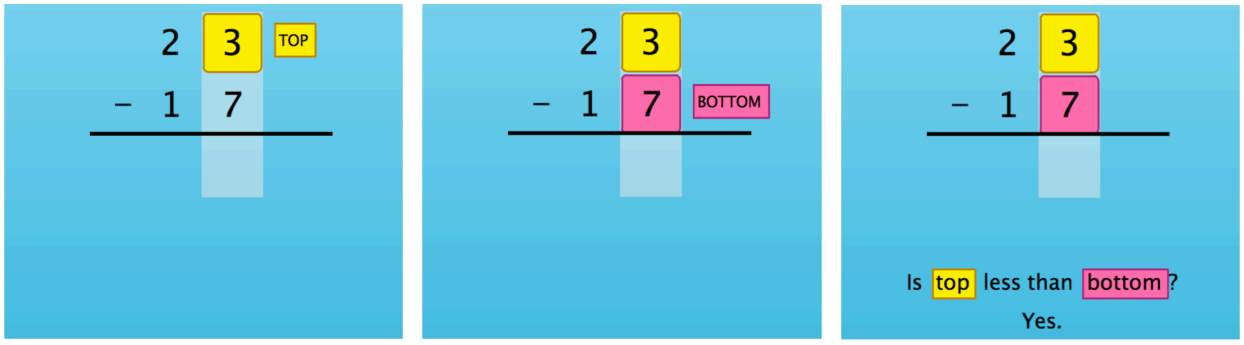
Event	Interface Hook
Put variable in context	Interface Highlighting
Remove variable from context	Interface Highlighting
Execute assignment statement	Textual Explanation
Execute conditional statement	Textual Explanation
Execute loop statement	Textual Explanation
Execute return statement	Textual Explanation

Table 3.1: **Interface Hooks.** The type of interface hook used to explain each type of event.

designed to be reused across applications. The TPL algorithm is compiled and passed into the interpreter, which maintains information about the set of variables that are currently stored in memory. As the interpreter executes each line, it broadcasts events about this state, such as when new variables are put into context, when statements are executed, and when sub-procedures return. A full list of the events thrown by the interpreter is given in Table 3.1. Each event contains relevant information like the name of the variable, the programming language object associated with the executed statement, or the name of the sub-procedure that was called. These events are passed to domain-specific interface hook functions, which generate and display visual explanations.

3.3.3 Interface Hooks

To explain the events that are broadcast by the interpreter, we use domain-specific functions called *interface hooks* that map events to visual explanations in the user interface. We use two types of visual explanations: textual descriptions and interface highlighting. The explanation used for each type of interpreter event is shown in Table 3.1. Authoring interface hook functions involves writing code that is similar to what a developer would write to create problem-specific tutorials for an application, but our framework allows this code to be re-used to explain any problem in the given domain.



(a) (b) (c)

Figure 3.2: **Subtraction Explanations.** Examples of the explanations generated for the subtraction procedure. Figure (a) shows the explanation of the assignment of a variable called *top* to the top cell in a column. Figure (b) shows the explanation of the assignment of a variable called *bottom* to the bottom cell. Since the variable *top* is still in scope it remains highlighted. Figure (c) shows the explanation of the condition statement that determines whether borrowing is needed.

Textual Explanations

Textual explanations are commonly used to teach procedural tasks in both software tutorials and intelligent tutoring systems [28, 140]. We use text to describe assignment statements, return statements, loops, and conditionals. To display a textual explanation in our framework, we first automatically generate the explanation and then display it in the user interface.

We generate explanation text in a completely application-independent manner using information that is encoded within the TPL algorithm itself. We use templates to convert lines of TPL code into textual explanations. A template is a single-sentence string that describes a certain type of statement, for example a conditional statement. To generate the explanation for a line of code, we fill the appropriate template with the names of the variables, properties, or functions that are used in that line. The template library includes 20 explanation templates, which are designed to be reused across applications.

Figure 3.2 shows explanations generated for the subtraction procedure in Algorithm 1. Assignment statements are explained using the name of the assigned variable, so the code $top := column.GetCell(0)$ in line 3 is explained with the text "Top". To explain the conditional statement *if top.value < bottom.value* in line 6, we fill in the conditional template "<explain boolean expression> <explain boolean result>". We populate the boolean expression template "Is <variable one> less than <variable two>?" to get "Is top less than bottom? <boolean result>." Finally, we fill the boolean result template with either "Yes" or "No" based on whether the expression evaluates to true or false. This produce the final explanation "Is top less than bottom? Yes."

To display the generated textual explanations in the user interface, our framework relies on application-specific interface hook functions. One interface hook function must be written for each location in the user interface where the designer wants to display text. As a result, the number of interface hooks required is highly dependent on the complexity of the domain. For the subtraction example, text is displayed in two locations. One interface hook function displays text next to a grid cell when a variable is assigned to a cell, and another displays text below the problem for all other types of programming language statements.

Interface Highlighting

We provide context to our textual explanations by highlighting visual objects in the user interface, an approach that is used in many effective software tutorials [17, 62]. Our goal is to highlight the set of interface objects that are relevant to the current steps in the procedure, without overwhelming the student by highlighting too many objects at once. To accomplish this, we determine highlighting based on the scope of variables within the procedure context.

Most imperative languages have a sense of *variable scope*, or the context in which a variable is defined and can be referenced. A variable that is defined within a branch of a conditional statement, like the *borrow* variable in the subtraction procedure, will only be in scope while that branch is executing. We draw a connection between variable scope and working memory, declaring that the objects a student needs to keep in working memory

while executing a procedure are equivalent to the variables that are currently in scope. We use the scope of variables to determine whether the visual representations of those objects should be highlighted.

To highlight objects in the user interface, our framework relies on application-specific interface hook functions. One interface hook function must be written for each type of object defined as a variable in the TPL algorithm. In the subtraction example, we define four variables: *top*, *bottom*, *borrow*, and *result*. All four of the variables are grid cell objects, so in this example we use one interface hook function that highlights arbitrary grid cells in the user interface. Using this interface hook function, our system highlights the objects referenced by the *top* and *bottom* variables while they are in context, as shown in Figure 3.2(b). Variables defined in the TPL algorithm must refer to visual objects in the user interface, however we have found that all variables do have visual representations in the procedures we implemented for our prototype systems.

3.3.4 Authoring Process

This framework utilizes domain-independent components where possible to reduce authoring effort. To apply this framework to a new application, an educational technologist first integrates the domain-independent compiler, interpreter, and explanation templates into their code base. Then, the developer writes interface hook functions that display textual explanations and highly visual objects in the user interface. Finally, the developer writes algorithms in TPL that encode the problem-solving thought process for each procedure to be taught. Note that the interface hook functions are only written once for a given application, and can then be used to explain multiple TPL algorithms. For example, in our grid mathematics prototype, one set of interface hooks can be used to explain a broad range of K-12 math procedures such as subtraction, long division, and fraction multiplication. The number of interface hooks required for a given application will depend highly on the complexity of the problem domain.

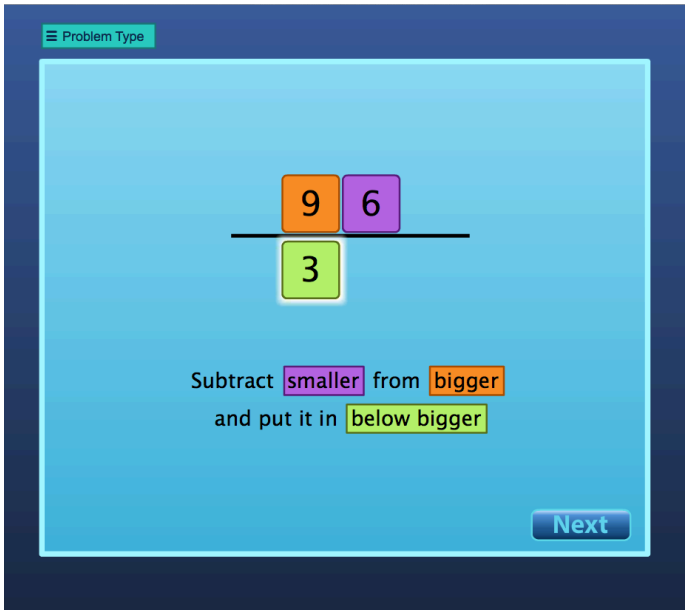


Figure 3.3: **Grid Mathematics Prototype.** Screenshot of the grid mathematics application, set up to explain a greatest common factor problem with Euclid's Algorithm. The user can switch to a new procedure using the "Problem Type" menu bar.

3.4 Prototype Implementations

Our framework for automatically generating step-by-step demonstrations can theoretically be used to explain any deterministic procedural process. However, it is more appropriate for some domains than others. This approach is most effective in rich domains where many problem types can be represented through a single interface. The framework is also best suited to complex procedures with loops and branching. This represents a wide variety of problem-solving domains in K-12 education. In this section, we demonstrate how the framework applies to two distinct domains: a K-12 mathematics application and an educational game. Both applications are written in ActionScript 3, so we implemented our Thought Process Language, Compiler, and Interpreter in AS3 as well. Through these prototypes, we show that our framework can produce instructional materials for a variety of procedures and tasks without any problem-specific design.

3.4.1 K-12 Grid Mathematics

The domain of grid mathematics includes all problems that can be solved on a two-dimensional grid. This covers a large portion of K-12 math, including addition, subtraction, long division, prime factorization, fraction arithmetic, and even algebra. For this prototype, we implemented a general-purpose grid math application that can display a wide variety of problem types. Figure 3.3 shows the application interface displaying a greatest common factor problem. Users can select the desired procedure using the "Problem Type" drop-down menu. Step-by-step demonstrations for each procedure are provided for a progression of increasingly complex problems. Users step through the explanations by clicking the "next" button.

We wrote three interface hooks for this application: one that highlights an arbitrary grid cell, one that displays an explanation next to an arbitrary grid cell, and one that displays an explanation under the problem. These three simple functions can explain a large variety of procedures. For this prototype, we implemented three TPL procedures: a subtraction procedure, a multiplication procedure, and a procedure for Euclid's algorithm to find the greatest common factor of two numbers.

For each TPL procedure, our prototype can automatically generate explanations for a massive number of problems. For example, there are 55 million unique four-digit subtraction problems. Some are simple, while others require borrowing or borrowing across zero. Each type of problem is solved with a different set of steps, which is represented by a trace through our TPL algorithm. Four-digit subtraction is covered with 73 unique traces. It would require a lot of effort to author tutorials for each type of problem individually. Our approach allows us to explain all problem types automatically.

3.4.2 Educational Puzzle Game

Our second prototype is an educational puzzle game that is solved with a highly complex procedure. We chose this application to show how our approach scales to domains that require

(a) (b) (c)

Figure 3.4: **Refraction Prototype.** Refraction PrototypeScreenshots of the *Refraction* prototype. To play, the student moves pieces from the bin on the right to the grid to bend and split the laser. Figure (a) shows an explanation of an assignment statement generated by our system. The text is generated from the variable name. Figure (b) shows an explanation of a conditional statement generated by our system. Figure (c) shows a summary explanation for the backtracking sub-procedure.

students to make complex decisions. *Refraction* was designed to teach fraction concepts to elementary school students. To play, the user interacts with a grid containing laser sources and target spaceships, as shown in Figure 3.4. The goal is to satisfy target spaceships by splitting the laser into the correct fractional amounts. The player uses pieces to change the laser direction and split the laser into fractional parts.

The most complex levels of *Refraction* cannot be solved with a deterministic procedure; heuristic search is sometimes required. For this prototype, we wrote a TPL procedure that covers the subset of *Refraction* levels that can be solved deterministically. This procedure is still much more complex than any of the grid math procedures implemented for our first prototype. It consists of 80 lines of TPL code and can solve thousands of distinct *Refraction* levels.

We wrote 27 interface hooks to explain the problem-solving process for *Refraction*. One set of functions highlights the following objects in the interface: pieces, arrays of pieces, spaceships, grid cells, grid columns or rows, spaceship fractions, laser fractions, and cardinal

directions. Other functions display textual explanations that point to those same visual objects. The code we wrote in these interface hooks is very similar to the code *Refraction* developers wrote to author the game's original introductory tutorials. As a result, the authoring effort was similar to the effort required to write level-specific tutorials. However, with this approach, our functions can be used to explain any level solved by our TPL algorithm.

The textual explanations are automatically generated using explanation templates, as described in the Framework Design section. To effectively describe the complex procedure used to solve *Refraction* levels, we encoded explanations within the variable names themselves. For example, the explanation shown in Figure 3.4(a) was generated from the assignment of a variable named *theShipNeedsThisMuchPower*. The explanation shown in Figure 3.4(b) was generated from a conditional statement that calls the function $areTheseTwoValuesEqual$. By encoding textual explanations directly in the TPL algorithm's variable and function names, we can automatically generate detailed walkthroughs for any game level that can be solved by our TPL algorithm. This could improve the process of authoring tutorials and hints for educational games; previous research shows that this process can be challenging and time-consuming [103].

3.5 Scaffolding Modalities

A central benefit of our authoring framework is that it naturally supports the creation of many different types of interactive instructional scaffolding, in addition to the step-by-step explanations described thus far. In this section, we show how our framework can be extended to produce tutorial progressions, just-in-time feedback, and fading worked examples.

3.5.1 Tutorial Progressions

The optimal amount of information to provide students as they work to master a new procedure varies throughout the learning process. Reigeluth and Stein argue that educators should introduce the simplest version of a new task first, and then give students progressively more complex tasks that build on the original task [116]. For novices, it is therefore desirable to

minimize cognitive load by starting with simple problems that can be solved with the most straightforward procedures. For example, when introducing a simple subtraction problem like $15 - 3$, we would not mention the procedure for borrowing because it is not needed in this problem.

To produce explanations with the ideal level of detail automatically, we trace the TPL algorithm for the given problem before generating the explanations to determine which statements should be explained and which should be skipped. For each statement that branches, like a loop or a conditional, we track whether the branching behavior of this statement varies during the execution of the algorithm for the current problem. If the statement always executes the same branch, for example if the "borrowing" conditional in subtraction never evaluates to true, we skip the explanation for that statement because it is not required for the current problem.

As a student masters portions of a complex algorithm, less explanation is required. For example, an algorithm for solving linear equations will include addition and subtraction as sub-procedures. However, since students studying algebra have already mastered these procedures, it is not necessary to step through them in detail. Our approach naturally supports multiple levels of explanation granularity. If the designer indicates that a sub-procedure should be summarized rather than explained in detail for a given problem, we automatically generate a summary explanation by fusing information stored in the procedure's comments. To author a sub-procedure summary, the designer writes a header comment with a high-level description of the procedure, and indicates which variables should be highlighted by adding the comment *//explain* to the end of the assignment statement for that variable. Figure 3.4(c) shows the summary explanation for the backtracking procedure in *Refraction*, which is used to determine where the next piece should be placed on the grid.

A key advantage of these techniques for producing appropriate explanations is that they build on recent work on automatically generating problem progressions for procedural tasks from an algorithmic representation of the procedure [8]. By combining our framework with these approaches, it would be possible use the TPL encoding of a procedure to automatically

produce a lesson that introduces practice problems in order of increasing complexity and explains each one with an appropriate level of detail. This would remove the need to author both practice problems and their explanations by hand.

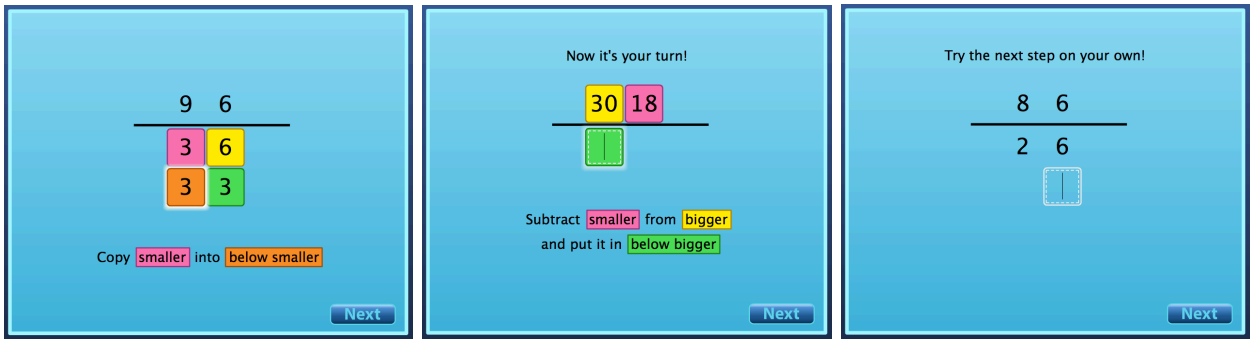
3.5.2 Just-in-Time Feedback

Another effective method of scaffolding learning is by providing students with feedback just as it is needed. Human tutors often give hints and suggestions when students make mistakes. Personalized hints are a central component of most cognitive tutors [140] and studies show that students perform better when these tutors provide hints [10]. Our framework naturally supports of interactive just-in-time feedback.

To show how we can generate interactive feedback directly from our step-by-step demonstrations, we created a "feedback mode" in each of our prototype applications. In this mode, the application waits for a user action and responds with a message if the action is incorrect. To author just-in-time feedback, the designer adds the comment *//action* to each line in the TPL algorithm that modifies the user interface. For example, for the subtraction procedure in Algorithm 1, this comment would be added to lines 11, 16, 19, and 22, all of which set the value of a grid cell to a new number.

While running in "feedback mode", the interpreter executes the TPL algorithm silently, without displaying any explanations, until it reaches a line with an *//action* comment. Then, the system waits until the user performs an action. If the user action does not match the expected action, then the user has made a mistake. The designer can define how many incorrect tries the students can make before the correct steps are explained. If the student exhausts all of their attempts, the interpreter reverts the incorrect action, backtracks to the previous *//action* comment, and displays a step-by-step demonstration of the steps the student should have made between their last correct action and their mistake.

This approach allows us to leverage the power of the interpreter to provide just-in-time feedback that directly targets student mistakes with minimal authoring effort.



(a) (b) (c)

Figure 3.5: **Fading Worked Examples.** Screenshots of the progression for Euclid's Algorithm that gradually fades between step-by-step explanations and independent problem solving. Figure (a) shows a full demonstration, Figure (b) shows the system asking the student to perform one step with help, and Figure (c) shows independent problem solving with just-in-time feedback.

3.5.3 Fading Worked Examples

The typical method for teaching procedures begins with a full demonstration of the procedure and ends with the student solving problems without help. One limitation of this approach is that the transition between demonstrations and independent problem solving is often abrupt; students observe a teacher demonstration on the board and then solve worksheet problems without help. Some researchers have suggested that scaffolding should be used to smooth this transition [14, 119]. Renkl et al. describe a method where some steps of a procedure are demonstrated while other steps are completed by the student. By gradually reducing the number of demonstrated steps, the student eventually learns to solve the problem without help. They found that this type of gradual fading improved student performance on near-transfer tasks [119].

Initial studies that explore the effectiveness of fading worked examples have all used explanations and fading that were authored by hand for each problem [88]. Melis et al. have explored methods of automatically generating and adapting fading examples within

the context of the ActiveMath intelligent tutoring system [88], but this approach does not generalize to other interactive learning environments or domains. Our approach naturally facilitates fading worked examples across multiple application domains.

We extended our framework to support fading in both of our prototype applications. To author fading for a given example, the designer indicates in the problem definition which actions in the TPL algorithm should be demonstrated and which should be performed by the student. Actions can also be fast-forwarded to focus more directly on a new part of the procedure. By defining whether each action in the TPL algorithm should be explained, performed by the user, or fast-forwarded, we can automatically create scaffolding that provides that level of fading. Figure 3.3 shows three different levels of fading that we created for Euclid's Algorithm in the grid math application. In Figure 3.5(a) the procedure is fully explained, in Figure 3.5(b) the student is given instructions but must complete the step on their own, and in Figure 3.5(c) the student solves the problem independently and receives just-in-time feedback in response to any mistakes.

The current design of our framework requires the designer to define which actions should be explained, fast-forwarded, and completed by the student for each problem. However, it could be extended further to produce faded progressions automatically. Gradual fading between worked examples and independent problem solving has great potential to support students as they learn new procedures, however this content is time-intensive to author and therefore difficult to study. We hope this new approach to authoring faded progressions can reduce the barrier to using and studying this technique.

3.6 User Evaluation

To gain a better understanding of the process of authoring scaffolding using our framework and the quality of the generated explanations, we conducted two user studies: one with educational technologists and one with fifth-grade students.

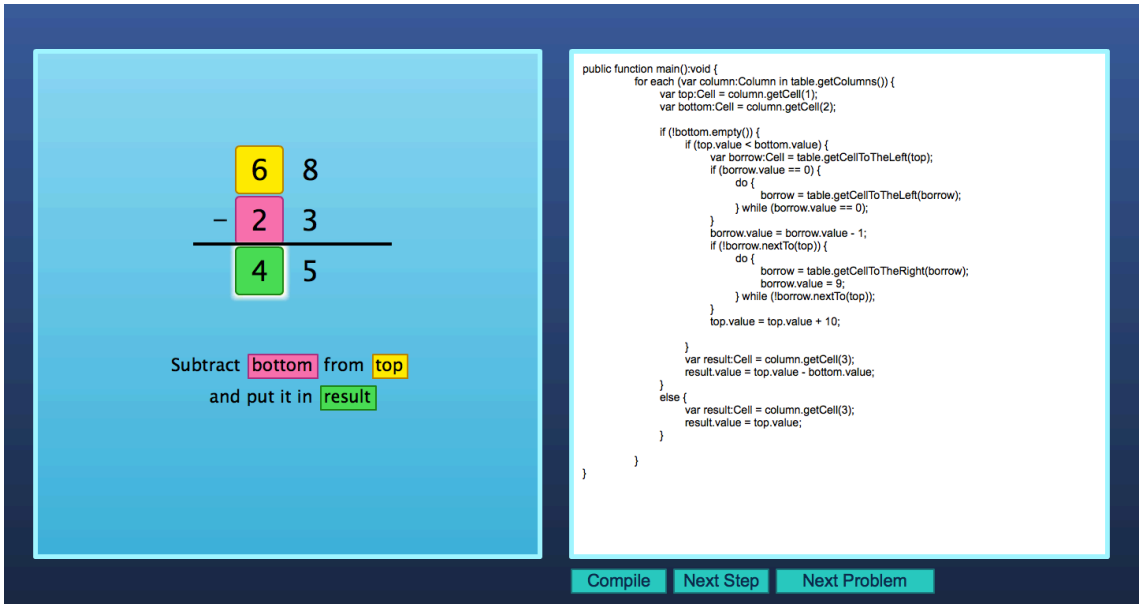


Figure 3.6: **Procedure Authoring Environment.** Screenshot of the authoring version of the grid mathematics application used in the authoring user study. Users can write Thought Process Language procedures in the panel to the right and step through the explanations they produce.

3.6.1 TPL Authoring User Study

The process of authoring step-by-step demonstrations with our framework consists of two parts: writing interface hook functions for the problem domain and writing a TPL algorithm for each procedure to be taught. In this study, we specifically examine the process of authoring TPL algorithms. To generate effective explanations, a TPL algorithm must abstract information into conditional blocks so that it is only explained when needed, and use variable names that describe the human thought process. This requires an unusual programming style that could be challenging for authors. In contrast, authoring interface hooks requires standard programming and design skills. We were therefore most interested in evaluating whether educational technologists can learn the unique skill set required to write effective TPL algorithms.

To evaluate the process of writing TPL algorithms, we conducted a small-scale user study with three educational technologists recruited through our institution. All three participants were professional developers of educational applications with no expertise in designing instructional content. We asked participants to write a TPL procedure for subtracting one to three digit numbers that supported borrowing. We developed an authoring version of the grid mathematics application for this study, which included a code-editing panel as shown in Figure 3.6. To author the step-by-step demonstrations, participants wrote their TPL procedure in the code panel, clicked "Compile", and stepped through the procedure for a set of provided example problems. We gave the participants a brief overview of the system before they began the task, and provided a few resources. This included documentation for the classes that their TPL procedures could reference, descriptions of the system's interface hooks, and an example procedure for Euclid's Algorithm.

All three participants produced TPL algorithms that solved the set of example problems we provided. They spent between 2.5 and 4 hours on the task in total, a reasonable amount of time considering that their procedures could explain all three-digit subtraction problems. The explanations generated by their algorithms provided clear descriptions of the subtraction procedure, although two participants included extra steps such as counting the number of columns rather than looping over them. Most interestingly, the authors each described a slightly different thought process for solving subtraction problems, displaying our framework's ability to support multiple types of explanations. All three participants iterated on their procedures extensively during the authoring process, and were able to learn effective ways of expressing the thought process using the feedback provided by the generated explanations themselves. This shows that educational technologists are able to learn to encode the problem-solving thought process to produce high-quality step-by-step tutorials without extensive training.

While we have not formally evaluated the process of authoring interface hook functions, we found it to be straightforward for our example implementations. This process requires standard programming and design skills, and a familiarity with the target application. The

effort required to author the interface hooks is highly dependent on the complexity of the application: we wrote three interface hooks for the grid math prototype and 27 for Refraction. Furthermore, interface hooks are only implemented once per problem domain, while TPL is written for each procedure to be taught.

3.6.2 Student User Study

The textual explanations generated by our framework are phrased slightly differently than hand-authored explanations due to the format of the underlying algorithm. We were therefore interested in evaluating whether the generated explanations can effectively teach students a new procedure. To explore this question, we collaborated with an elementary school math teacher to conduct a user study with two of his fifth-grade classes. We chose to teach Euclid's Algorithm for finding the greatest common factor of two integers because this is a procedure that is rarely taught in schools; the teacher we worked with was not familiar with the procedure.

We created two computer lessons to teach Euclid's Algorithm using our grid math prototype. One lesson displayed step-by-step demonstrations of the procedure for three example problems, and then provided a set of 18 practice problems with just-in time feedback. The second lesson presented the same 21 problems in the same order, but faded gradually between demonstrations and independent problem solving. We compared these two computer-based lessons to a traditional lesson given by the math teacher to see how the automatically-generated explanations compared to human explanations.

The Euclid's Algorithm lesson was conducted during a 50-minute math period. The first class received the traditional lesson, and the second class was split into two groups who received the two computer-based lessons. In both classes, the teacher began by giving an identical five-minute overview of factors and greatest common factors. Next, students had 10 minutes to work on a paper pre-test with three greatest common factor problems. One problem could be solved by listing the factors of each number and selecting the greatest common one, but the other two problems were designed to be difficult to solve without using

Euclid's Algorithm.

After finishing the pre-test, the first class was given a traditional 25-minute lesson by their teacher. We asked him to teach Euclid's Algorithm as he would teach any new procedure, but had him introduce the same set of 21 example problems in the same order as in the computer lessons. The teacher demonstrated the procedure for two example problems on the white board, and then asked students to work on the 19 practice problems on their own while he answered questions and worked with individual students. The second class moved to computers after completing the pre-test at their desks. Half of the students, randomly chosen, logged into a lesson with demonstrations and practice problems, while the other have logged into a lesson that faded between the demonstrations and practice problems. The students were given 15 minutes to interact with the system after logging in.

After the lessons, students went back to their desks to take the paper post-test. This test used the same three questions as the pre-test, given in a different order. We chose to use the same questions to be sure that the tests were of equal difficulty. Students were given 10 minutes to work on the post-test.

Results

We collected data from 28 students, and were given informed consent for participation by the students, their parents, their teacher, and their school principal. Of these students, 18 received the traditional lesson, 5 received the computer lesson with just-in-time feedback, and 5 received the computer lesson with fading worked examples. Due to a server issue we lost data from several students in the computer class, hence the smaller number of participants in these conditions.

We used non-parametric statistics in our analysis because the students' scores on the pre- and post-tests were non-normally distributed. We found that overall, scores improved between the pre- and post-test. We used a Wilcoxon rank sums test to measure the effect of *test* (either pre or post) on scores, showing that students performed significantly better on the post-test ($Z=-3.80$, $p<0.0001$), with a median of 2.5 questions correct out of three,

compared to 0 correct on the pre-test. We also calculated a learning gain for each student by taking the difference between their pre- and post-test scores, and found that the median learning gain was 1 point.

Next, we used a Kruskal-Wallis test to determine whether *condition* had any impact on students' scores. There was no significant difference in pre-test scores (χ^2=3.11, *n.s.*). Students in the traditional lesson condition got a median of 1 problem correct, compared to 0 for the two computer lesson conditions. There was also no significant difference in post-test scores (χ^2=3.11, *n.s.*); students in the traditional lesson condition got a median of 3 problems correct, compared to 2 problems correct for students in the fading computer lesson, and 1 problem correct for students in the just-in-time feedback computer lesson. Finally, *condition* had no significant impact on learning gains (χ^2=0.01 *n.s.*). Students in all three conditions had a median learning gain of 1 point.

Our results show that students who received the two computer lessons had learning gains roughly equivalent to those who received the traditional lesson. Only two students did not use Euclid's Algorithm to solve the post-test problems, one in the traditional lesson condition and one in the just-in-time feedback condition. This shows that the generated explanations were able to effectively convey a new procedure to students, even over a very short period of time. Students had less time-on-task and no help from their teacher with the computer lessons, but still performed equally well on the post-test.

While these results are promising, our sample size is small and we only looked at one TPL algorithm in this study. In future work, it would be valuable to conduct a more thorough experiment that focuses on the student experience. This would help us gain a better understanding of the strengths and weaknesses of our automatically generated explanations.

3.7 Limitations and Future Work

The research presented in this chapter has a number of limitations that will need to be addressed in future work. One of the biggest limitations of our approach is that the framework only supports one correct procedure. If the student deviates from the expected steps in any

way, even if she is following another correct procedure or is performing steps in a different order, the system will recognize the deviation as an error. The framework could be extended to support multiple procedures for a single type of problem to alleviate this issue. However, as currently written, the framework could provide a frustrating learning experience for students, particularly for more complex problems that have many valid approaches.

Our evaluation also leaves a few important questions unanswered. We evaluated the procedure authoring process, but we have not formally evaluated the effort required to write interface hook functions for a new application. To fully understand the efficiency gains of our approach, we would need to measure the time required to implement our framework for a new application and procedure, and compare this to the time required for other authoring approaches. This is a complex question to study, since authoring time will be highly dependent on the complexity of the target application and the procedure that we wish to teach. It is also difficult to gain access to other authoring tools, since many are not publicly available. However, understanding the complexity of the authoring process in more detail would help justify the value of our approach.

Finally, the scaffolding content generated using our framework provides many exciting avenues for future research. While fading worked examples have been proposed as a valuable scaffolding technique, most evaluations of this approach have been conducted on hand-authored progressions of problems that fade explanations away at a constant rate. Our system parametrizes the pace of fading, which could allow us to experimentally study the impact of fading at different rates. We could also personalize fading based on individual performance. This is a open area of future research that could have important implications for the design of fading worked examples.

3.8 Conclusion

In this chapter, we present a new framework for automatically generating interactive scaffolding to teach the entire space of problems that can be solved with a given procedure. We encode the problem-solving thought process as an algorithm and use this model to generate

explanations. This approach has a number of advantages. Given a small set of mappings between programming language constructs and visual objects in the user interface, we can produce tutorials for any Thought Process Language procedure that can be applied in the application. Furthermore, this framework naturally facilitates the authoring of problem progressions that introduce information gradually, just-in-time feedback, and scaffolding that fades between demonstrations and independent problem solving.

We demonstrate the generality of our approach by showing how the framework applies to two distinct problem-solving domains: a K-12 math application and an educational game. For each prototype, we authored interface hooks and TPL algorithms that produce in-depth explanations for problem-solving procedures in that domain. We also conduced an authoring user study to determine whether educational technologists can effectively encode a problem-solving procedure in our Thought Process Language. All three participants produced high-quality TPL procedures to explain subtraction in a reasonable amount of time. Finally, we conducted a user study with fifth grade students to determine whether the explanations generated by our framework can effectively teach a new procedure. This study showed that students who received computer lessons had similar learning gains to students who received a traditional lesson from their math teacher.

Our new approach for authoring interactive instructional scaffolding is designed to maximize the content that educational technologists with programming expertise can create. The authoring effort required to write interface hooks and TPL algorithms will not be not justified in all domains. However, our framework has incredible potential in domains where we would like to generate scaffolding for a wide variety of example problems. Perhaps most importantly, this approach naturally supports the creation of a variety of different types of instructional scaffolding, including problem progressions, just-in-time feedback, and fading worked examples. Since these types of scaffolding are costly to produce, many important questions about the relative effectiveness of different scaffolding methods remain unanswered. We hope that this framework will be used to improve tutorial authoring and also expand our understanding of effective scaffolding design.

Chapter 4

AUTOMATIC GENERATION OF PROBLEMS AND EXPLANATIONS FOR AN INTELLIGENT ALGEBRA TUTOR

This chapter considers the challenge of generating instructional content in learning domains that are less constrained than the procedural domains considered in the previous chapter. Building off our procedural approach, we present a new method of generating both example problems and interactive scaffolding for an intelligent algebra tutor. While many methods for implementing intelligent tutoring systems have been proposed, current approaches require authors to produce problems and explanations either by hand or with special-purpose tools. As a result, it is challenging for designers to tweak tutor parameters to study the implications of their design decisions. In this chapter, we present an approach for generating tutor content from a single representation of the learning domain. We use *answer-set programming* to model knowledge components, and show how this model can be used to automatically generate problems, problem solutions, structured problem ontologies, step-by-step explanations, mistake feedback, and fading worked examples. We demonstrate the feasibility of this approach through a proof-of-concept implementation in the domain of algebra. Through two preliminary user studies, we show that our automatically generated explanations are understandable, and can teach participants to solve equations more accurately and efficiently. Our novel approach for automatically generating content has great potential to facilitate future research into the effective design of intelligent tutoring systems.

4.1 Introduction

Few instructional techniques are as effective as one-on-one tutoring [22]. Human tutors personalize content for each student, providing appropriately difficult problems and tailoring

feedback to help them arrive at correct solutions. However, it is difficult for teachers to provide this level of assistance in traditional classrooms, and personalization becomes impossible in large lecture classes and online learning environments. To address this challenge, educational technologists have explored methods of emulating one-on-one tutoring through automated systems [10, 29, 51, 140]. These *intelligent tutoring systems* can provide adaptive problem progressions and personalized feedback in a variety of domains, and have been found to produce strong learning gains in classroom studies [68, 141].

While intelligent tutors are effective, they are difficult to build. A massive number of design decisions go into each system; designers must choose adaptive problem selection policies, determine how hints and feedback should be presented, and decide how much assistance to provide to students [140]. Furthermore, it is technically challenging to support robust personalization. A wide variety of implementation approaches have been explored, resulting in the creation of cognitive tutors [10, 29], constraint-based tutors [90, 98], example-tracing tutors [66], and ASSISTments [51]. However, all current implementation approaches depend on problems and explanations that are either authored by hand or generated with special-purpose tools [10, 69, 91].

The complex custom implementations of intelligent tutoring systems make it difficult for designers to tweak system parameters and study the impact of their design decisions. As a result, there are many open questions about how to effective intelligent tutor design. In this work, we propose a new approach for automatically generating a wide variety of tutoring content from a single representation of the problem domain. This approach supports the generation of parameterized content, allowing the designer to implement a wide variety of problem selection policies and feedback policies on top of that content by modifying parameters. We believe that this unified implementation design will support future research aimed at parametrizing and studying the myriad of features provided by intelligent tutors.

In this chapter, we show how we can automatically generate problems, problem ontologies, explanations, responses to misconceptions, and fading worked examples using a single underlying model in the domain of algebraic equations. We model algebra using if-then

production rules similar to those used by cognitive tutors. However, we implement our rules using the declarative programming paradigm *answer set programming*, which enables problem generation directly from the model. We show how we can generate all valid solutions to each generated problem, and organize these problems into an ontology based on their solution structure. We also present a new method for automatically generating step-by-step explanations directly from the same model of algebraic operations. We show how our explanation content can be used to create worked examples, tutored problem-solving, and a progression that gradually fades between the two.

The goal of this work is not to defend a particular instructional approach, but rather to show how a variety of tutoring features can be implemented using the modeling and generation techniques that we propose. However, to show that we can build a tutor on top of generated content, we developed a proof-of-concept implementation that generates problems and explanations for linear equations. We developed an application called the *Algebra Notepad* using our approach that allows users to manipulate equations through a gesture-based interface. We also conducted a preliminary evaluation of the quality of our generated explanations through two user studies: a formal study with Mechanical Turk workers and an informal study with eighth-grade students. We found that Turk workers solve problems more accurately and efficiently after using our tool, and that students can successfully solve problems in our application.

We make a number of contributions in this work. We present a novel method for generating tutoring content from a single unified domain model. We show how we can generate problems, solution traces, and explanations directly from this model. We also propose approaches for producing problem ontologies, responses to student mistakes, and fading worked examples. We present a proof-of-concept implementation of an algebra tutor that uses a gesture-based user interface for solving algebra problems. Finally, we present results from two user studies evaluating the quality of the explanation content that is automatically generated by our system. We believe this new approach for automatically generating tutor content can facilitate future research on the design of intelligent tutoring systems.

4.2 Related Work

A rich history of research in the domains of educational technology, artificial intelligence, and human-computer interaction has explored the design and implementation of intelligent tutoring systems. While researchers have begun to investigate the semi-automated exploration and optimization of design parameters in learning systems [76, 77], many open questions remain. The goal of our work is to facilitate further research into intelligent tutor design through the automated generation of parameterized problems and explanations. In this section, we provide a background on learning domain modeling, content generation, and scaffolding design.

4.2.1 Modeling Learning Domains

Many intelligent tutoring systems (ITS) use models of the target learning domain to enable functionality that personalizes feedback in response to student actions [10, 29, 90, 98, 140]. There are two primary classes of tutors that use domain models: cognitive tutors and constraint-based tutors.

Cognitive tutors, based on the ACT-R theory of cognition [10, 29], provide the most sophisticated form of intelligent feedback. Cognitive tutors represent knowledge using production rules that define if-then relationships. These rules model all knowledge required to solve problems in a particular learning domain and allow the tutor to solve each problem step-by-step along with the student [10, 29]. A typical tutor will involve as many as 500 production rules [11]. The rules are also used, with student data, to build a model that predicts a student's skill on the task, a process known as model tracing [10, 29]. Errors can be detected when the student action does not match any production rule in the model. Most tutors include explicitly programmed buggy production rules that match common mistakes and misconceptions [10]. We also represent our learning domain (algebraic equations) with if-then production rules. However, we implement our model in answer set programming to support problem generation, in addition to solving problems along with the student.

Constraint-Based Modeling (CBM) is an alternative approach stemming from Ohlsson's theory of learning from errors, which posits that students make mistakes when declarative knowledge has not yet been internalized in procedural knowledge [98]. According to this theory, students learn and improve their procedural knowledge by being informed of their mistakes [99], and an ITS can play the role of a mentor that helps students recognize mistakes [91]. Constraint-based tutors model the learning domain as a set of pedagogically important constraints [81, 90, 91]. In contrast to cognitive tutors, constraint-based tutors do not trace student actions but instead analyze the student's current state for violations of model constraints [91]. Constraint-based tutors typically require less authoring effort than traditional cognitive tutors, and can be applied to domains that are not heavily procedural such as English grammar, database design, and object-oriented software design [90]. However they can only provide feedback about constraint violations, not goal-oriented feedback [90].

Another class of ITS, sometimes referred to as *pseudo tutors*, exhibit many of the behaviors of ITS without requiring AI programming. Example-tracing tutors, one such example, are created by manually demonstrating multiple correct problem-solving solutions as well as common mistakes for particular problems. A behavior graph is created from the demonstrations and then used to trace student behavior and provide intelligent responses [66]. In the ASSISTments system, another type of ITS, authors write scaffolding questions for each problem that are used to respond to common student mistakes [51]. A downside of these approaches is that tutoring content must be demonstrated or authored for each problem type, while hand-authored production rules can generalize across many different problem types [66].

4.2.2 Problem Generation and Organization

Work studying problem generation for various educational domains exists as far back as the 1960s [143]. In a recent paper discussing new potential areas for intelligent tutoring system improvement, Koedinger et al. highlight automated problem generation as an interesting area for future development [69].

There has been some work on automatically generating problems for constraint-based tutors. Martin and Mitrovic developed an algorithm for automatically generating problems for their constraint-based SQL tutor, which first identifies a set of target constraints, then uses those constraints to generate random solution fragments, and then passes those fragments into the problem solving to generate a complete, novel solution. Finally, the solution is converted into a natural language problem that can be presented to students [80, 81].

More recently, work has looked at problem generation for various domains including word problems [113], geometry proof problems [128], natural deduction [4], procedural problems [8], and embedded systems [122]. Most of these approaches are template-based, that is, given a template generalizing a particular problem, they generate more problems fitting this template. Some generate these templates automatically [4], semi-automatically [8], or manually [122]. In our approach, rather than explicitly generalizing from templates, we generate all possible problems of a certain size and calculate the set of templates afterwards.

Several of these approaches are based on exhaustive search or logical reasoning, whereas we use a solver-based approach with *answer set programming* (ASP), a declarative programming paradigm. Andersen et al. use the code coverage toolkit Pex, built on the Z3 SMT solver, to generate problems in their domain of procedural mathematics (e.g., long division) [8]. Other approaches also use ASP directly for domains such as word problems [113] or educational math puzzles [25, 131]. Applying solvers to a new domain takes significant technical work. We focus on using this ASP model as a means for automatic explanation generation.

4.2.3 Explanation Generation

Most intelligent tutoring systems generate feedback and explanation content from hand-authored templates. In traditional cognitive tutors, both correct and buggy production rules are explained by writing textual templates for each rule [10, 11, 140]. This one-time cost occurs when the domain model is authored. In addition, to be able to fill in the template with the correct problem-specific content, each problem must be labeled to indicate which

phrases should be inserted into the template [11]. Next step hints and feedback to mistakes are produced with these templates [10, 140].

In constraint-based tutors, feedback messages are attached directly to the constraints [91]. Feedback messages, which are hand-authored for each constraint, can either be given after each student action or step, or at the end of the problem [90]. Feedback messages are completely problem-independent in this approach, but still must be hand authored for each constraint in the domain. The Cognitive Tutor Authoring Tools (CTAT) were developed to support efficient authoring of both model tracing tutors [70] and example-tracing tutors [66]. Example-tracing tutor authors can annotate hints and feedback messages for specific problems in the system [66]. Stamper et al. developed the Hint Factory, an approach for automatically selecting appropriate hints, based on student data [133]. The wording and ordering of hints was still authored by hand in collaboration with instructors [134].

We present an approach for automatically generating explanation content directly from the implementation of the underlying production rules, which come with advantages and disadvantages. Our approach requires authoring production rules in a way that will generate good explanations, possibly requiring additional up-front authoring effort. However, individual problems do not need to be labeled or annotated, because our system automatically uses problem-specific information produced by the answer set solver.

4.2.4 Scaffolding Design

A wide variety of different approaches have been explored for scaffolding student problem-solving during the learning process. Intelligent tutoring systems are typically designed to provide a specific form of scaffolding known as *tutored problem solving*. In this approach, students work through problems independently, and feedback and hints are given only when needed [10, 29, 140]. Worked examples, step-by-step instructions that shows students how to solve an example problem, are an alternate form of scaffolding. A large body of research has explored the use of worked examples in mathematics and science education [86, 117, 139]. They have been shown to improve learning in domains including algebra [136], debugging

electrical circuits [138], physics, and statistics [119].

Recent research has explored combining tutored problem solving and worked examples. Worked examples have been shown to be even more effective when the transition between viewing demonstrations and independent problem solving is gradual, a technique referred to as *fading worked examples* [118, 119]. Studies have suggested that tutored problem solving and fading worked examples can be synergistic forms of instruction [86, 123].

Our approach can be used to produce worked examples and tutored problem solving, gradually fading between the two types of content. The format of our explanation content makes it relatively straightforward to produce both full worked-examples and just-in-time feedback, which could facilitate future research that compares scaffolding approaches.

4.3 Modeling Algebra Problems

In this section, we describe modeling approach. First we define the sub-section of algebraic problem solving covered by our tutor. Then we describe how we model and generate example problems. Finally present our method for generating step-by-step explanations.

4.3.1 Problem Definition

In this work, we focus on an algebra tutor. A student solves algebraic equations by applying operators such as canceling common coefficients or subtracting a term from both sides. Each operator represents a valid algebraic transformation of an equation. These operators have both a precondition and an action, similar to the production rules in a cognitive tutor. The process of solving an algebra equation is selecting a valid sequence of operators (both the type of operator and operands) that result in a solved equation. Our goal is to both automatically solve problems and generate new problems that cover a space of possible solutions.

Operators include low-level axioms (e.g., canceling an added zero), and high-level operations (e.g., adding like terms). A more specific goal of our system is to teach students how to efficiently find *optimal* solutions, those with the shortest sequence of operators. In order to do that, we partitioned the various operators into categories, and the solving procedure in

Category	**Example**
Cancel	$x + 3y = 2 + 3y \mapsto x = 2$
Combine	$2x + 3x = 1 \mapsto 5x = 1$
Rearrange	$2x = 3 \mapsto (2x)/2 = 3/2$
Move	$x + 4 = 5 \mapsto x + 4 - 4 = 5 - 4$
Expand	$x^2 - 1 = x - 1 \mapsto (x + 1)(x - 1) = x - 1$

Table 4.1: **Operator Categories.** A prioritized list of our operator categories, with example equation modifications for each operator.

our tool puts precedence on each category of operation. For example, the student is required to consider whether an *cancel* operators can be used before *combine* operators. Table 4.1 shows these categories and some representative operations.

4.3.2 Modeling Algebra Problems

We model the problems using the declarative programming paradigm *answer set programming* (ASP) [39]. This choice is driven by our desire to both automatically solve and generate problems. While modeling problems is a technical challenge, ASP then enables both solving and generating in a relatively straightforward manner.

In ASP, programs declaratively describe a set of models called *answer sets* in a first-order-logic representation that syntactically resembles Prolog. Answer set solvers search the space of possible truth assignments of each predicate (logic statement representing facts that can be true or false). The returned solutions, called *answer sets*, are satisfying assignments consistent with all constraints specified in the program.

A conventional ASP program has 3 types of rules: first are *choice rules*, which allows the solver to guess whether predicates are true or not. *Deductive rules* are if-then rules that allow the solver to deduce new true predicates from established or guessed ones. Lastly, *integrity constraints* are rules that forbid certain solutions. By modeling the laws of algebra with deductive rules and integrity constraints but using choice rules for the solution and the problem, we can craft an ASP program that solves or generates algebra problems.

Algorithm 2 Psuedocode of the applicability rule for the *add like terms* operator. Each kind of precondition predicate is represented in this rule. The term T represents a time step, and the `holds` predicate describes which fluents hold at time T. For example, `holds`$(T,$`monomial`$(N))$ is true whenever N represents a monomial term at time T. Similarly, the fluents `are_addends`(L, R) means L and R are addends in the same sum, `are_on_same_side`(L, R) means L and R are on the same side of the equation, `same_degree`(L, R) means L and R have the same degree, and `zero`(N) means the node N is the constant 0.

```
applicable(T, add_like_terms(L, R)) ←
    time(T),
    % domain predicates
    node(L), node(R), L != R,
    % validity predicates
    holds(T,monomial(L)),
    holds(T,monomial(R)),
    holds(T,are_on_same_side(L, R)),
    holds(T,are_addends(L, R)),
    holds(T,same_degree(L, R)),
    % heuristic predicates
    not holds(T,zero(L)),
    not holds(T,zero(R)).
```

We model the algebra equation solving process using event calculus [126], a logical formulation in which the state at multiple time steps is represented with *fluents*. Fluents are facts which may or may not be true at a given point in time. In our model, fluents describe the state of the algebraic equation. The initial fluents (time step 0) are the configuration of the algebra problem. Each subsequent time step represents the application of a single operator.

Algebraic terms are represented as a tree. For example, the root is the equality symbol, and the left and right subtrees are the left and right sides of the equation, respectively. Deductive rules are used to determine the applicability of a given operator at a given time step. Further deductive rules describe how the fluents change on the next time step, when the operator is applied. Algorithm 2 shows a concrete example for the operation of adding like terms. Integrity constraints are used to ensure the final step is in solution form.

As a concrete example of a rule, consider the rule for adding two like terms. The rule for

the applicability has several preconditions: there must be two nodes that are being added, and the nodes must represent monomial terms of the same degree. The action deduces new fluents for the following time step in which the two terms are combined into one.

We include additional rules about applicability that are not about the validity of the rule, but a heuristic as to when it should be applied. For example, the adding like terms rule includes preconditions that neither coefficient is zero. In that case, we would prefer the system to give precedent to rules relating to canceling zero.

There are three different types of predicates in the precondition rules, illustrated in Algorithm 2. One type of precondition are *domain predicates*, which describe the potential space of all the arguments to an operator. Such predicates must be satisfied for an operator to be meaningfully applied to an equation at all, even in an incorrect manner. For example, `add_like_terms` takes 2 arguments, the terms to be combined. These arguments must be constrained to be distinct nodes of the equation tree.

The second type of precondition describes the validity of the rule, *validity predicates*. Such an example for adding like terms is that both terms must have the same degree. An operator is only a valid algebraic transformation if these preconditions all hold. An operator could, however, be incorrectly applied to any equation where all domain predicates hold but not the validity predicates.

Lastly, because we are interested in describing when it is a good idea to use an operator, we have *heuristic predicates* that describe whether a rule should be taken. For add like terms, such a predicate is that neither term should be zero, because in that case we would rather apply the operator for canceling a zero. However, if both the domain and validity predicates hold, this is still a *valid* algebraic transformation, just a potentially inefficient one.

4.3.3 Problem Generation and Solving

ASP allows us to both solve given problems and generate novel problems. In order to solve a given problem, we use choice rules for the predicates describing which operator is chosen at which time. Since the program constrains the final time step to be in solution form, ASP

will find the set of operators that produce a solution. We add further integrity constraints such that this solution must be the shortest possible. Additional integrity constraints force the program to select operators in the precedence order described in Table 4.1.

To generate a problem, we use choice rules for both the initial problem configuration and operators used in the solution. ASP requires us to choose a finite domain, so both the size of algebraic equations and number of steps in the solution must be constrained. However, in practice we are interested in small problems, so for our application we searched over problems of up to 4 steps with a maximum depth in the equation of 4.

With this process, we can generate a single problem and solution pair. Of course, many algebra problems have multiple possible solutions. Therefore, we must generate a problem and *all* of its possible (shortest) solutions. This requires some subtlety. Generating a single problem with *all* solutions is in a higher complexity class than generating only a single solution. ASP is capable of solving such problems, and previous work [130] has looked at the specific challenge of generating educational math puzzles with all solutions, but implementing this is technically challenging. However, we can take a much simpler approach. We do not wish to enforce constraints over all solutions, so we can produce all solutions in two steps. First we generate a set of problem-solution pairs, then, with a second ASP program, generate the set of all shortest solutions for each of these problems.

4.3.4 Explanation Generation

We designed our system to produce explanations that describe the priority of operator categories as well as why and how an operator can be applied to an equation. We generate these explanations directly from the ASP operators defined in our model. This allows us to produce explanations for each generated problem automatically, without hand-authoring any content. However, to produce good explanations, we must author the model slightly differently than we would if it was not used to generate explanations. We first describe how we generate operator explanations, then discuss the design and generation of priority explanations. Finally, we describe how these are combined to produce step-by-step explanations

automatically for any example problem generated using our approach.

Operation Explanations

There are two parts to explaining an operation that is applied to an equation. First, we want to convey to the student *why* the operation can be applied. Then, we want to convey *how* the operation is applied to modify the equation. We note that these two types of explanations correspond to the if-then structure of production rules. The set of predicates that define the preconditions that must hold in order for the operator to be applicable can be used to explain *why* the operation can be applied. The name of the operator can be used to explain *how* it transforms the equation. The operands passed into the operator references the terms in the equation to which the operation should be applied.

While the operator predicates include valuable information about *why* an operator can be applied to an equation, we may not want to explain all of the predicates to the user. For example, domain predicates that communicate to the solver that you cannot add a term to itself, as described in the *add like terms* example in Algorithm 2, should not be explained. To handle this case, we communicate to our system that some predicates should never be explained to the user by writing an underscore at the beginning of the predicate name. More importantly, we may want to have the option of producing a concise high-level explanation by combining multiple predicates into a single explanation. To handle this case, use a tiered approach that defines multiple explanation levels for each operator. The first level provides a high-level explanation for *why* the rule applies, while the second level provides a more nuanced explanation that includes all predicates that must hold.

Algorithm 3 shows the ASP code we used to model the *add like terms* operator in our system. The high-level *why* explanation generated for this operator is the name of the single pre-condition defined in the operator body at the first level, specifically "we are adding two terms with variables that have the same degree." To generate a more detailed *why* explanation, we can look a level deeper to see that the terms must be monomials on the same side of the equation with the same degree that are added together. The *how* explanation is

Algorithm 3 ASP code showing the definition of the *add two terms* operator. We separate the preconditions into two tiers. The first tier can be used to generate a high-level description of the preconditions, while the second tier can be used to provide a more detailed description. We use the second tier to generate explanations for rules that "almost fire" as well.

```
applicable(T, weCanSimplifyByAddingTheseTwoTermsTogether(L, R))
    :- weAreAddingTwoTermsWithVariablesThatHaveTheSameDegree(T, L, R).

weAreAddingTwoTermsWithVariablesThatHaveTheSameDegree(T, L, R)
    :- _areDistinctNodes(L, R), isMonomial(T, L), isMonomial(T, R),
       areOnTheSameSideOfTheEquation(L, R),
       areBeingAdded(T, L, R), haveEqualDegrees(T, L, R),
       _isNotZero(T, L), _isNotZero(T, R).
```

generated from the operator name "we can simplify by adding these two terms together."

This approach requires the author to abstract rule information into multiple levels of detail, which is not how rule would typically be written. Furthermore, explanation text must be incorporated into the names of operators and predicates. While authoring a model that generates high-quality explanations requires significant effort, once it is written explanations can be automatically generated for any example problem generated using the model.

Strategy Explanations

In addition to explaining *why* and *how* an operator can be applied to an equation, we also want to communicate whether it is a good strategic move to apply that operator. To communicate high-level strategy to students, we use explanations that highlight the priority order of our five classes of operator types: *cancel*, *combine*, *rearrange*, *move*, and *expand*.

To emphasize this priority, we designed a strategy dialog. For each equation, we want students to ask themselves whether they can cancel, and if not, then ask whether they can combine, etc. Therefore for each step, we generate a sequence of question-answer pairs for each strategy category until we find an operator that can be applied. For each strategy, we generate a question of the format "can we combine?" This question can be answered in three ways. If no operator in that category can be applied, we generate the response "no

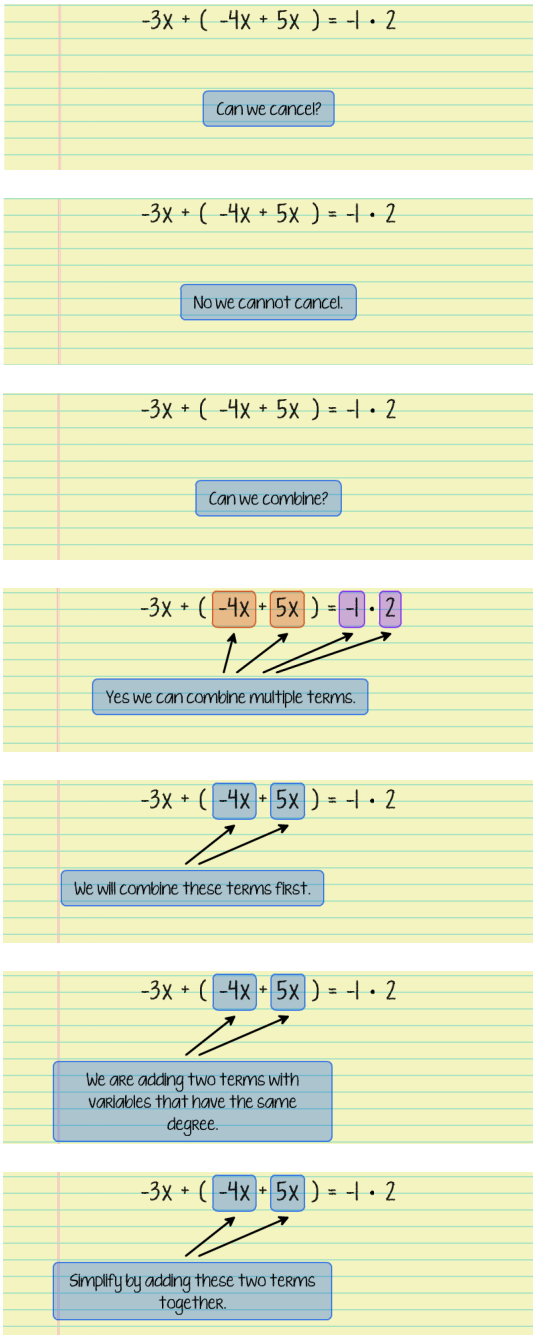


Figure 4.1: **Step-By-Step Explanation Example.** An explanation sequence automatically generated by our system for the *add like terms* operator.

we cannot combine." If a combine operator can be applied to exactly one pair of terms, we generate the response "yes we can combine these terms" along with the term ids. Finally, if there are multiple pairs of terms that can be combined, we generate the response "yes we can combine multiple terms" along with the pairs of term ids. In the case where multiple combine operators can be applied, the solver selects a pair of terms to combine first arbitrarily, which we explain with the response "we will combine these terms first."

The explanation text is generated from five templates, one for the strategy question, and four for the four types of strategy responses. These templates are populated with the name of the current strategy category. In combination with the identifiers of the terms to which each operator is applied, this information can be used to produce a strategy dialog.

Step-By-Step Explanations

We have described how we generate explanations that explain a high-level *strategy* for selecting an operator to apply, as well as explanations that describe *why* and *how* the operator is applied. To generate step-by-step explanations for a given problem, we combine these pieces as follows. For each step, we begin by displaying the strategy question-response pairs until we identify an operator that can be applied to the equation. Then, we present the *why-how* explanations that show how to use this operator. An example of an explanation sequence generated to explain the application of the *add like terms* rule is given in Figure 4.1.

4.4 Features Supported by the Model

There are several educationally interesting features and tools that the use of ASP as a modeling approach makes available to us. We describe some below.

4.4.1 Progression Generation

One benefit of this approach to modeling algebra problems is that we can straightforwardly cover the space of possible problems by asking the ASP solver to return all possible answer

sets. By then comparing the sequences of operators used in solutions, we can reason about the relationship between problems in the space. In particular, we can analyze or potentially generate progressions of problems, using the solutions (and possibly data gathered from students) to drive this reasoning.

Previous work has looked at automatic progression generation for learning domains based around solution traces for procedural knowledge [8] or features extracted from problem solutions [25]. Using the sequence (or set) of solution steps extracted from algebra problems, we can apply these techniques to any domains based on if-then operators as well.

One convenient way to think about and visualize relationships between problems is with a graph. Such a graph has categories of algebra problems as the nodes and edges that signify closely related problems. Let the nodes be sets of operators, where a problem belongs to a particular node if its solution sequence uses precisely that set of operators. For example, the problem $0 + (3x)/3 = 1$ uses the set of operators "canceling a zero" and "canceling a like numerator and denominator." Two nodes S, D have an edge from S to D if D contains all operators in S with precisely one addition. Figure 4.2 shows a graphical representation of a subgraph such a graph produced our set of generated algebra problems. The full graph, for all problems with up to 4 steps, has order 486 and size 1441. As evidenced by the figure, the graph is quite dense, and many combinations of possible operators are feasible.

As many tutors already formally model the steps required in the solution, and even already classify problems based on which operators are required, these techniques can be adapted to wide range of existing technology. Progression analysis and generation can be applied to hand-crafted problems and hand-crafted solutions, even if the solving/generation process of problems themselves is not available.

4.4.2 Misapplied Rules

One potential application of this framework is the automatic generation of buggy operators and misconceptions. Addressing student misconceptions are a common feature of computer tutors, of which many use hand-crafted buggy rules [10, 91]. Melis studied explicitly teaching

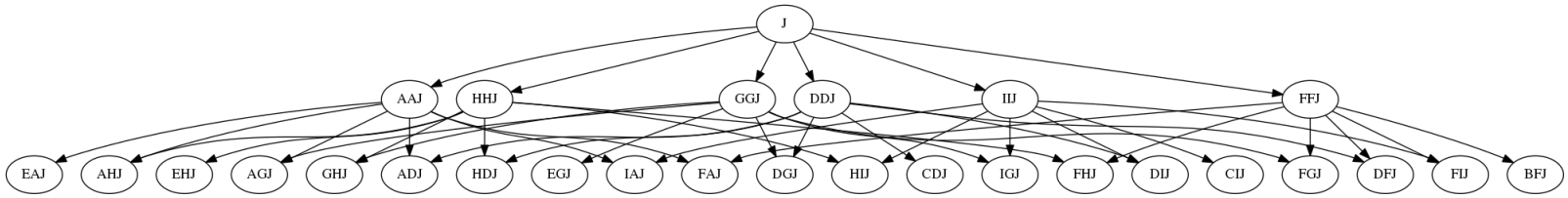


Figure 4.2: **Problem Graph.** A subgraph of our generated set of algebra problems. This portion of the graph shows all problems that include the operator for *canceling a coefficient on both side* (represented by the letter J), for problems of up to three steps. Each letter in a node label represents one of the operator types, for instance *canceling a zero* or *adding like terms*. Edges are drawn between nodes if the source node requires a subset of the operators required in the destination node. The full graph for all of our generated algebra problems contains 486 nodes.

students through erroneous examples that highlight common problem-solving errors in the intelligent tutoring system ActiveMath, presenting a number of methods of presenting examples with errors in them and asking students to find the mistake [87]. Subsequent research showed that erroneous examples can produce significant metacognitive learning gains in the classroom [137].

One category of misconceptions is misapplied rules. For example, given the equation $2x * 5x = 100$, a student could mistakenly think the add-like-terms rule applies to $2x$ and $5x$, since they are monomials of the same degree on the same side of the equation. However, they are being multiplied, not added, so the operation is not valid. Modeling algebraic operators in ASP allows us to automatically detect and reason about a subset of such misapplied rules. Each operator has several predicates which make up the precondition. If nearly all the predicates hold (e.g., there are two terms, both monomials of same degree) but one such predicate is missing (e.g., terms are not being added), then such a rule is almost, but not quite, valid to apply. We call such misapplications *almost-fires*.

Given the set of rules of when a rule is applicable (e.g., the predicate described by Algorithm 2), we can produce an exhaustive list of all the potential almost-fires for a given algebraic equation by looking at all rules that are applicable when a single predicate is omitted from the body of the rule. Recall that we partitioned predicates into 3 types: domain predicates which must hold for an operator to meaningfully applied, validity predicates which must hold for the operator to be a valid algebraic transformation, and heuristic predicates which must hold for the move to be optimal in our given solving process. Which type of predicate is omitted changes the type of almost-fire. It never makes sense to omit domain predicates, because the rule cannot meaningfully be applied to an existing equation, even in an incorrect way, and therefore cannot occur in practice. For example, a student cannot accidentally combine a like term with itself, so omitting $L \neq R$ from Algorithm 2 does not produce a reasonable almost-fire. Omitting a validity predicate produces a true buggy rule, where an operator can be applied to the equation incorrectly. Omitting `are_addends`(L, R) results in the misapplied in the $2x * 5x = 100$ example above. Finally, omitting a heuristic

predicate results in a algebraically sound application of an operator, but not one that the student should follow. For example, in $3 + 0$, we could correctly combine the like terms 3 and 0, but we would prefer the to use the operator that simply cancel the zero. This is a heuristic almost fire.

As with correct applications of operators, we can automatically extract explanations for almost-fire operators from the rules themselves. In this case we are only explaining why the rule does not apply. We take the name of excluded rule and negate it (e.g., "are being added" becomes "are not being added"). The structure and consistency of our rule names makes this negation straightforward, placing "not" after the first "is" or "are" that appears in the rule name. When explaining almost fires, the system generates the explanation text "it looks like we can ⟨*operator name*⟩, but we cannot because ⟨*negated rule that was excluded*⟩."

Often, misconceptions and buggy operators are hand-crafted into the tutors. In contrast, our model allows for the automatic detection of a wide set of potential misconceptions. While they only cover a subclass of potential misconceptions, they can be computed fully automatically. Because an almost-fire can still be applied, we can detect, whenever a student performs an operation incorrectly in the tool, which almost-fires are potential candidates for the mistaken operation. We hypothesize that, with data from students, one could automatically determine which of the automatically generated almost-fire rules are likely to occur in practice. Then such rules could be used for applications such as just-in-time detection and feedback of mistakes. If, through data analysis, we know particular operators or problems frequently cause certain mistakes, it may make sense preemptively explain the possible misconceptions.

4.4.3 Fading Worked Examples

There are a variety of different methods that education researchers have explored to teach students to solve problems in mathematical domains. On commonly used method is worked examples, in which step-by-step demonstrations showing students how to solve problems are displayed [86, 117, 139]. Another method is one-on-one tutoring with a student, in which an

expert provides students with hints and feedback to scaffold their problem solving [10, 29, 140]. Both of these types of content can be generated in our application.

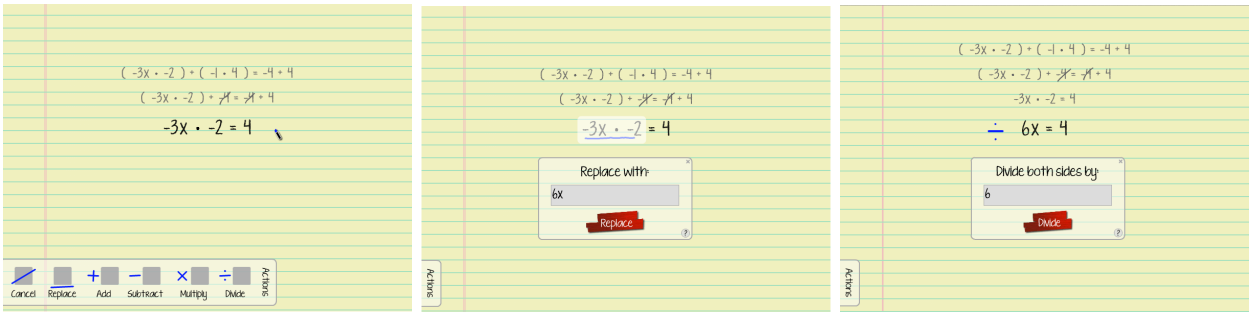
Worked examples have been shown to be even more effective when the transition between viewing demonstrations and independent problem solving is gradual, a technique referred to as fading worked examples [118, 119]. Salden et al. showed that gradually fading between worked examples and tutored problem solving increased learning over tutored problem solving alone, suggesting that the two methods can be synergistic forms of instruction [123]. We believe that faded worked examples are an important area for future research.

One benefit of the way in which we generate explanation content is that it can be easily adapted to provide instruction at multiple fading levels for any example problem. As we describe in the next section, we implemented fading worked examples into our proof-of-concept application.

While previous tutoring systems have also incorporated functionality to support fading worked examples, few tutoring systems use fading, and there is little understanding of how to automatically adapt fading to meet the needs of each student [123]. We believe this is an interesting area for future work, and we hope our approach to supporting fading directly from automatically generated problems and explanations will facilitate future research in this area.

4.5 Proof-of-Concept Implementation

The primary contribution of this work is the idea that a domain model can be used to generate both problems and explanations for intelligent tutors. There are many design decisions that go into the development of a tutoring system [140], and our goal is not to defend a particular instructional approach, but rather to show how a variety of approaches can be implemented using the modeling and generation techniques that we propose. However, to show that we can build a tutor on top of generated content, we developed a proof-of-concept implementation. We integrated our generated content into an interface for manipulating equations that we call the *Algebra Notepad*. The Algebra Notepad was developed in ActionScript 3, and provides



(a) (b) (c)

Figure 4.3: **The Algebra Notepad.** Screenshots of the Algebra Notepad interface. Each problem step is displayed on a separate line, and users manipulate equations using gestures. Figure (b) shows a *replace* gesture and Figure (c) shows a *divide* gesture.

a gesture-based interface designed to emulate solving equations on a piece of paper.

The Algebra Notepad interface is shown in 4.3(a). An "Actions" panel shows the gestures that can be performed to reduce the equation. The interface supports six gestures. To cancel, the user draws slashes through terms. To replace terms with another equivalent term, the user underlines the terms and enters the equivalent term in an input box as shown in Figure 4.3(b). To perform an operation to both sides of the equation, the user draws the appropriate symbol in the space to the left or right of the equation. An example of the division gesture is shown in Figure 4.3(c). When the user performs a gesture, the resulting modified equation is copied onto a new line. This allows the user to keep track of problem progress.

For this implementation, we generate a set of problems (along with their solutions and explanations) offline and input them into the application. We use the fading policy described in the Fading Worked Examples section above, which has five levels. Fully worked examples (Level 1) are shown with a step-by-step explanation showing how to solve each problem. The user steps through these explanations using a "Next" button. Strategy explanations are displayed first, followed by operator explanations, as shown in Figure 4.1. To visualize the terms to which the operation is applied, we highlight the terms in the equation and use arrows to point to the explanations.

The final level of the fading policy (Level 5) is a fully interactive mode that intervenes when the user makes a mistake. The application traces user actions and compares them to the set of correct solutions. If the user step matches a correct step, sparkly stars are displayed around the equation to communicate step correctness. If the step does not not match any of the correct solution, the user is shown a response. We use a three-tiered response system. First, we display the message *"Sorry, that step is incorrect. Try again!"* to give the user a chance to correct the mistake. Next, we show the message *"That step is incorrect"* followed by the *then* portion of the operator explanation. Finally, the application walks through the entire explanation for that step, and then performs the step to allow the user to move forward.

Levels 2, 3 and 4 in the fading policy blend these extremes. In Level 2, both strategy explanations and operator explanations are shown, but the user is asked to modify the equation. In Level 3, the strategy questions are shown, but the operator explanations are omitted. This level directs the user to the rule that should be applied, but does not explain how to perform that operation. In Level 4, the application asks the user to respond to the strategy question. The goal here is to encourage the user to ask the strategy questions internally. As with Level 3, the operator explanations are omitted.

4.6 User Evaluation

Thus far, we have have demonstrated proof-of-concept implementations that show how our ASP model can be used to generate problems, problem graphs, strategy and operator explanations, almost fires explanations, and fading worked examples. In addition to these implementations, we explicitly evaluate the quality of the explanation content generated by our system. Our textual explanations are phrased slightly differently than hand-authored explanations due to the format of the underlying model. We therefore wanted to determine whether these explanations are comprehensible, and the whether they can help improve algebra problem-solving skills. We conducted a formal user study of the Algebra Notepad on Mechanical Turk, and an informal study with eighth-grade students.

4.6.1 Mechanical Turk Study

The goal of the Mechanical Turk study was to recruit a large number of participants to try our application. While Turk workers are typically adults, many adults struggle with algebra and most have not solved an algebraic equation in many years. We designed our Mechanical Turk Human Intelligence Task (HIT) to market directly to adults who are not comfortable with math. We entitled our HIT *"Solve Simple Algebra Problems - No Algebra Experience Needed!"*. The description associated with the HIT was *"Want to learn algebra? Try out our new Algebra Notepad application by solving 15 simple algebra problems. No algebra experience necessary! We are studying how people solve algebra problems to improve educational technology."* Our intention was make the HIT sound approachable to people who do not like math.

We used a static progression of nine problems so that all users would have the same experience with the application, and we focused on a subset of algebra operations. During problem generation, we constrained our model to produce problems that use at least two of the following three operators: canceling a term from both sides, multiplying two terms, and dividing both sides by a constant. Six problems required a minimum of four steps to reach a solution, and the remaining three required at least three steps. We used the five-step fading policy described in the implementation section. Participants were first given one step-by-step example at Level 1, followed by two problems each at Levels 2, 3, 4, and 5.

Before and after using the Algebra Notepad, participants completed a three-problem test. The problems focused on our three target operations, and the post-test problems were identical to the pre-test ones except with different constants and coefficients. We designed a web form for our HIT that provided ten single-line input boxes for each problem. Participants were asked to show their work by writing one step on each line. After the post-test, participants completed a short survey with fix six-point forced-choice Likert questions, given in Table 4.2. We also asked participants about the explanations: *"What did you think about the explanations in the Algebra Notepad?"* to learn about explanation quality.

Likert Question	Result
The Algebra Notepad was hard to use	2.8
I liked drawing lines to change equations in the Algebra Notepad	3.9
The explanations in the Algebra Notepad were clear and understandable	4.8
The Algebra Notepad knew what I was struggling with	3.8
The explanations in the Algebra Notepad helped me solve problems	4.7

Table 4.2: **Survey Questions.** The five Likert scale questions for the Mechanical Turk study. We used a forced-choice six-point scale, where 1 was *Strongly Disagree* and 6 was *Strongly Agree*. Average results for each question are given in the right-hand column.

Measures

We use multiple measures in the study. We instrumented the Algebra Notepad application to log user interactions, including whether their steps were correct or incorrect on the first, second, and third try. We also graded the pre- and post-tests by hand. Solutions were only considered correct if users gave the correct final solution. For each correctly solved problem, we also counted the number of operations the participant performed to reach the correct solution. This allowed us to measure correct but inefficient solutions.

Results

We collected data from 57 Mechanical Turk workers. First, we analyzed pre- and post-test scores. We gave one point for each correct question, for a total of three possible points on each test. We ran a repeated measures ANOVA to analyze changes between the two tests, and found that participants performed significantly better on the post-test ($F(1, 56) = 8.02$, $p < 0.01$), with a mean score of 2.4 on the pre and 2.6 on the post. For correct solutions, we also counted the number of steps used. For participants who got at least one question correct, we calculated the average number of steps used on each test. All problems could be

solved in a minimum of four steps. We found that participants used significantly fewer steps to reach a solution on the post-test ($F(1, 50) = 80.06$, $p < 0.0001$), with a mean of 5.1 steps per problem on the pre and 4.3 steps per problem on the post. These findings suggest that participants learned from our generated explanations.

We also analyzed log data from the Algebra Notepad. Specifically, we calculated the percentage of steps participants got correct on their first, second, and third try for the two problems at fading Level 5. They got an average of 84.5% of steps correct on the first try, 7.3% correct on the secant try, 3.0% correct on the third try, and 5.1% incorrect on the third try. This shows that most participants were able to solve the steps on their own with the help of feedback, rather than having the system complete the step after three incorrect tries.

Finally, participants completed a survey about their experience. The results of the five Likert-scale questions are given in Table 4.2, showing that participants found the explanations to be clear and understandable, and that they helped participants complete problems. This finding is confirmed by the responses to the short answer question. One participant wrote *"I thought they were great. It has been years since I've done algebra and the explanations on the notepad refreshed by memory and improved my ability to solve problems correctly."* This indicates that some participants had not solved algebra equations in a long time. Another wrote *"They were great! Easy to understand and visually see what was meant."* These comments suggest that our explanations were effective even though they were generated from production rules.

4.6.2 Student Study

Since adults are not our target population, we also wanted to confirm that students can use our system. We conducted a second informal user study with seven eighth-grade students at a local Boys & Girls club. In this study, students used the Algebra Notepad application to complete a static progression of 20 problems. The first nine problems in the progression were identical to those used in the Mechanical Turk study, and the same fading policy was used. However, we added an additional 11 problems at Level 5, where students worked on

problems independently and were given feedback in response to mistakes as needed. All problems required the use of at least two of our three target operators, canceling from both sides, multiplying terms, and dividing both sides by a constant. The seven students all completed the progression of 20 problems. We analyzed their log data, and saw that for the 13 problems that asked students to solve problems independently, they got an average of 81.4% of steps correct on the first try, 11.3% correct on the secant try, 5.0% correct on the third try, and 2.3% incorrect on the third try. This suggests that most students were able to complete steps correctly in three tries with the help of our explanation content.

4.7 Limitations and Future Work

The research presented in this chapter has a number of limitations that will need to be addressed in future work. Our user evaluation is preliminary, and we were not able to measure learning through assessments in our student study. In future work, we would like to conduct a formal evaluation with students that compares our system to a more traditional paper-and-pencil lesson. This study could highlight areas for future iteration on the design of both our gesture-based interface and our explanations.

Another limitation of this work is our approach for teaching problem-solving strategy. The explanations generated by our system teach students to solve algebra problems strategically by organizing operations into prioritized categories. However, this approach assumes that the priority order is constant, rather than dependent on the problem state. Future research should explore more sophisticated methods of teaching strategic problem solving that focus on analyzing the current problem state and selecting rules based on properties of that state. We should also formally study whether strategy explanations are effective at teaching students to solve problems efficiently.

This chapter presets a broad range of tutoring features that could benefit from our approach, but each of these features needs to be investigated in more detail. For example, while we have suggested an approach for automatically generating buggy production rules that represent misapplied rules, we have not evaluated their value. Future research should

determine the subset of misconceptions that these misapplied rules cover, and the percentage of misapplied rules that correspond to common student mistakes.

In future research, we would like to explore how this approach generalizes to other problem-solving domains. It would be valuable to determine the class of learning domains that can be modeled using answer set programming, and the limitations of this modeling approach. One of the central benefits of our approach is that it supports the parameterization of both problem progressions and fading worked examples. As a result, this work provides a great opportunity to formally study the design of adaptive and personalize scaffolding.

4.8 Conclusion

In this chapter, we present a new approach for automatically generating tutoring content from a single representation of the learning domain. We use *answer-set programming* to model knowledge components, and show how this model can be used to automatically generate problems, problem solutions, structured problem ontologies, step-by-step explanations, mistake feedback, and fading worked examples. We evaluate our approach through a proof-of-concept implementation in the domain of algebra. Furthermore, we demonstrate that our automatically generated explanations can teach Turk workers to solve equations more accurately and effectively, demonstrating the potential value of our automated approach.

We believe this novel approach for generating a wide variety of tutor content automatically has great potential to augment personalized learning systems in the future. We believe our approaches could be integrated with existing tutoring techniques to introduce automation into developed systems. We also believe that this work opens up new possibilities for systematically exploring the relative effectiveness of different tutoring features. By producing content from a single unified model of the learning domain, we believe it will be more feasible to parametrize and formally evaluate the myriad features provided by intelligent tutoring systems.

Chapter 5

THE BEHAVIORAL IMPACT OF A PERSONALIZED LEARNING SYSTEM FOR THE CLASSROOM

This chapter considers the challenge of designing and evaluating personalized learning systems for the classroom context. Personalized instruction can improve student learning, but it is challenging for teachers to adjust content for each individual in real-time. As a result, technology-supported personalization has great potential in the classroom. Prior work shows that technical systems can provide teachers with real-time student data, adapt content for students automatically, and integrate with curriculum materials. While the ideal personalization system would combine these features, very few support all three. Furthermore, we have a limited understanding of how this type of system would impact classroom behavior. In this work, we present a new system that augments an existing math curriculum with real-time data communication and automated adaptation. We conduct formative research with eleven sixth-grade math classes in four urban schools to understand how this system impacts behavior. We find that students perform better and teachers assist more students with our system, but that student learning gains do not transfer onto paper assessments. We present design recommendations based on these findings with the goal of working towards the development of the ideal classroom personalization system.

5.1 Introduction

A central challenge faced by classroom teachers is to effectively support students with diverse backgrounds and needs. Great teachers find ways to personalize their assistance for each individual student, and research shows that personalized instruction can greatly improve learning outcomes [21, 22]. However, it can be difficult for teachers to adapt lesson content

in real-time, particularly in large classes. They rarely have access to the data needed to assess each student's current understanding [32], and curriculum materials are not typically designed to support personalization. As result, this is an area where educational technology has great potential to augment traditional classroom instruction.

Researchers have explored a number of features that could help support personalization in the classroom context. Some have shown that networked technologies can be used to communicate data to teachers and help them personalize instruction and assistance in real-time [3, 37, 51]. Others have developed intelligent systems that can personalize instruction automatically by adapting problem progressions [51, 140] and providing intelligent feedback in response to student mistakes [10, 29, 140]. Finally, researchers have highlighted the importance of designing systems that integrate with curriculum materials [127] and support classroom orchestration activities [33].

Given this research, it seems that the ideal personalization system would combine automated adaptation with real-time data communication, and would integrate smoothly with existing curricula and workflows. However, while each of these features has been studied extensively, few systems combine all three. Furthermore, while prior research demonstrates the pedagogical effectiveness of data communication systems [23, 37], intelligent tutors [68, 141], and classroom orchestration tools [34, 52], we still have a relatively limited understanding of how these technologies impact student and teacher behavior in the classroom. As a result, it is challenging to consider how to design a personalization system that would effectively support all three features.

In this chapter, we conduct formative research to advance towards the goal of creating this ideal personalization system for the classroom. We present the design of a new personalization system that augments an existing curriculum by providing digital teacher materials, an adaptive problem-solving application for students, and real-time visualizations of student progress for teachers. To understand how this system impacts both student and teacher behavior in the classroom, we conducted a ten-week study with eleven sixth-grade math classes in four urban schools. We compare classes that used our system to classes that did

not, keeping both teacher and curriculum constant. We use a wide variety of behavioral measures to capture student performance, problem-solving behavior, and affect, as well as teacher impressions, assistance, personalization, and use of class time.

Our findings highlight the potential benefits of classroom personalization systems, while also uncovering a number of areas for design iteration. In particular, we find that in classes that use our system, students complete more problems with more accuracy, and teachers assist individual students more frequently. However, student performance gains do not transfer onto a post-test, suggesting a need to better understand learning transfer between digital and paper interfaces. We also find that our system is more effective in classrooms with organized teachers who can manage the technology. We present a series of design recommendations given these findings, and suggest areas for future research towards the goal of creating effective classroom personalization technology.

5.2 Related Work

In this section, we review research that explores methods of communicating student progress to teachers, automatically adapting content for students, and integrating educational technologies into the workflow of in-person classrooms.

5.2.1 Communicating Student Data for Teachers

Teachers often personalize instruction in real-world classrooms by adjusting their assistance to meet individual student needs. Formative assessment, the practice of frequently testing students to collect data about their progress [18], can help teachers personalize more effectively. Researchers have measured large positive learning effects when teachers modify instruction based on formative assessment data [18, 38]. However, student data is most relevant to teacher decision-making when assessments are embedded directly in the curriculum [127] and when data is collected and interpreted quickly [144]. Since administering and grading assessments requires significant teacher effort, networked technologies offer great potential for supporting real-time formative assessment [32].

Some of the most successful technologies for collecting and communicating student data in real-time have focused on visualizing student data publicly to foster discussion. Student response systems such as ClassTalk™ can rapidly aggregate and display responses to multiple choice questions [3]. These systems can enable new levels of interaction in large lecture classes [35] and the collected data can help teachers adapt instruction [23]. Other systems focus on collecting richer student data. Classroom Presenter allows students to respond to in-class problems by annotating slides with stylus input [12], and SimCalc displays aggregated responses to collaborative activities completed on graphing calculators [52].

Another class of systems focuses on sharing student data with teachers, rather than to the class at large. TechPALS, a collaborative learning system for hand-held devices, provides teachers with a real-time visualization of group performance [121]. Look, a system that supports technology-based group exercises, allows teachers to view a snapshot of each group's screen to monitor progress [64]. In the learning analytics community, researchers have explored the design of learning dashboards [142] and open learner models [61] as a means of collecting and sharing data. While these systems are often designed to be used by students [60], teacher visualizations have been explored as well [56]. Some tutoring systems, such as ALEKS and ASSISTments, provide teacher reports with data about student assessment grades, learning pace, and time spent on activities [48, 51]. However, while an early study of the ASSISTments reports suggests that teachers find student data useful [37], few evaluations have explicitly studied how student data is used by teachers in the classroom.

In this work, we explicitly study how the availability of real-time formative student data impacts teacher behavior in the classroom. Our system is directly embedded into an existing curriculum and provides formative assessment data that is closely aligned with instructional goals on a daily basis. We are not aware of any other studies that provide such a detailed look at how real-time data is used in the classroom context.

5.2.2 Automated Personalization for Students

Automated personalization has been explored extensively by the educational technology community. There are two primary ways that systems personalize content for students: by selecting personalized progressions of problems, and by giving personalized feedback in response to steps and solutions.

The goal of personalizing problem progressions is to allow students to move at their own pace. Intelligent tutoring systems have explored many methods of adapting progressions, most of which are based on mastery learning, or the idea that each student should master the current concept before moving to the next one [22]. Some tutors, such as Sherlock [140] and ASSISTments [51], use mastery tests to determine when to advance students. Others, such as the PUMP Algebra Tutor [67] and ALEKS [48], determine mastery by consulting their underlying cognitive models. These mastery learning approaches can effectively support student learning; Corbett found that students who used a mastery learning version of the ACT Programming tutor performed significantly better than those who used a control [30].

While mastery learning supports the needs of individual students, it is rarely used in classroom. Van Lehn notes that personalized progressions work in self-paced courses, but can be challenging to integrate into traditional classrooms where students need to move through material together [140]. We address this issue by using a modified version of mastery learning that only allows students to explore content related to the current lesson. Students can advance to more challenging problems if they achieve mastery quickly, or move back to simpler problems if they need remediation, but all students work on the same type of problem. This is similar to the approach used in the PUMP Algebra Tutor [67].

In addition to personalizing problem progressions, many systems personalize the feedback given in response to student steps or solutions. The hallmark of cognitive tutors is their ability to respond intelligently to student mistakes using their underlying cognitive models, providing personalized hints and suggestions to help students arrive at a correct solution [10, 29, 140]. Simpler feedback systems have been explored as well. In the ASSISTments

system, authors write scaffolding questions for each problem that respond to common student mistakes [51]. Both types of tutors have produced significant learning gains in classroom studies, suggesting that personalized feedback can be effective [51, 68, 71, 89, 141].

Most intelligent tutors are designed with the expectation that the system will assist students during problem solving activities, rather than the teacher. In contrast, our goal is to help teachers provide targeted personal assistance during problem solving. As a result, we chose to focus on the design of teacher and classroom integration tools, rather than on providing personalized feedback. Our system gives students feedback on problem correctness, but provides no other assistance. However, studies have shown that correctness feedback alone can have positive feedback on student learning. Kelly et. al. found that students who receive immediate correctness feedback in the ASSISTments system learn more than those who receive feedback the following day [63].

5.2.3 Integration into the Classroom Workflow

While research shows that personalized learning systems have great potential to improve student learning outcomes, these systems have not been widely adopted. A number of researchers have suggested that this could be because most technologies are not designed to integrate with existing classroom workflows nor evaluated in terms of how they impact classroom processes [29, 32, 33].

Some researchers have addressed this challenge by designing systems to explicitly support classroom orchestration tasks. For example, TinkerLamp 2.0 is a tabletop system that explores the design space of classroom orchestration technologies [34]. It includes an interface of cards that teachers can use for orchestration tasks such as pausing groups or hiding information to pose questions to students [34]. Alavi et. al. developed two systems, Lantern and Shelf, designed to communicate information about student progress and their need for assistance from teaching staff in the classroom [5]. The SimCalc system, which supports collaborative group activities on TI-calculators, provides functionality for teachers to pause student devices to get the class' attention [52]. Our teacher application provides support for

orchestration activities such as enabling and pausing student problem-solving and communicating information about student progress.

While few studies of intelligent tutoring systems specifically focus on classroom workflows, a number of researchers have highlighted the importance of collaborating with educators during the design process. In a report on lessons learned in the development of ASSISTments, Heffernan and Heffernan note that giving teachers control over system content and functionality was crucial for adoption [51]. In a discussion of meta-design principals for cognitive tutor design, Koedinger and Corbett highlight the importance of co-designing content with experienced teachers and providing curriculum materials to support the tutor [65]. In one early study of intelligent tutoring systems, Schofield et. al. evaluated the impact of the tutoring system on student and teacher behavior in the classroom. They found that teachers spent more time with struggling students in tutored classrooms, and that student effort increased due to enjoyment and increased competition [125]. However, most studies evaluate tutoring systems according to their learning outcomes rather than their impacts on behaviors and workflows in the classroom.

In this work, we present a new personalization system designed to augment an existing curriculum, and therefore integrate with the curriculum's prescribed workflow. Through formative research, we explicitly study how our system impacts both learning outcomes and classroom behaviors.

5.3 System Design

The goal of this work is to explore the design of a personalized learning system that combines real-time formative assessment with automated personalization and that integrates smoothly with an existing curriculum. During the design process, we partnered with a non-profit company that specializes in tablet software for the classroom. We co-designed a system that augments a validated math curriculum. Our student application supports personalization by adaptively selecting problems and giving correctness feedback. Our teacher application supports classroom orchestration and provides a real-time visualization of student data.

5.3.1 Curriculum

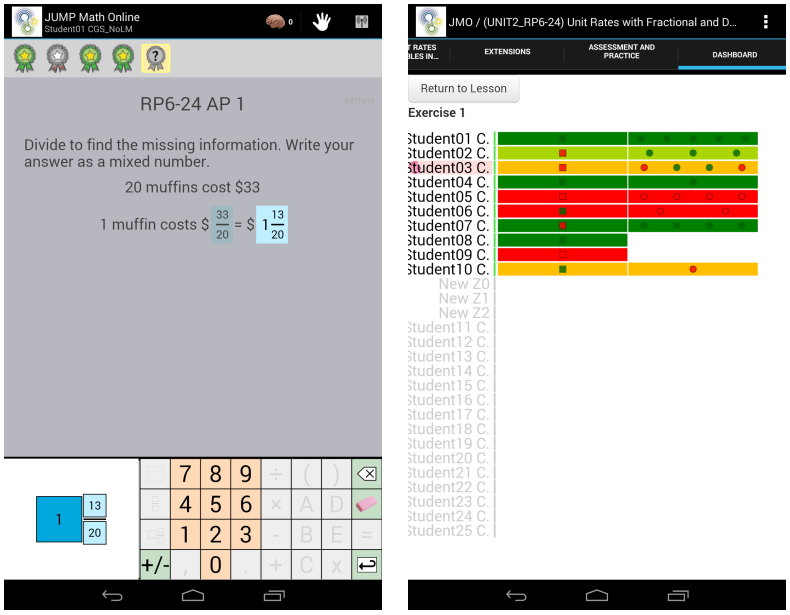
We chose to use the math curriculum developed by the Canadian company JUMP Math in this work [83]. JUMP Math's curriculum focuses on worked example problems, and has been shown to be highly effective in initial trials [94]. One benefit of this curriculum from our perspective is that its lesson plans are highly scaffolded. The teacher materials provide lesson plan scripts that indicate when new content should be introduced and when activities and problem-solving practice should occur. This well-defined structure made it simper for us to integrate our student and teacher applications into the prescribed lesson workflow.

We selected two units from the Common Core version of the sixth-grade JUMP curriculum to implement in our applications. These units focused on number systems concepts such as decimals, fractions, and proportions. The lessons follow a similar structure, beginning with the teacher *Teaching* a new concept. This is followed by individual practice on *Exercise* problems related to the introduced concept. Many lessons include an *Activity* or *Extension* that further explores the concept. At the end of the lesson, students work on *Assessment & Practice (AP)* problems, which are designed to prepare them for future assessments. The AP problems are not formal tests, so students may ask questions and work together. We designed our tablet applications to support this lesson structure.

5.3.2 Student Application

We designed the student application to personalize problem-solving practice sessions. From the student perspective, lessons operate as usual during most classroom activities. Students listen and answer questions during teaching sessions, and write notes in their paper notebooks. They participate in activities and extensions as usual. However, when it is time to practice either exercise or AP problems, students work on individual tablets using our application.

There is no difference in how exercise and AP problems are presented to students. When the teacher initiates a problem-solving activity, the first problem is displayed on the stu-



(a) (b)

Figure 5.1: **Student Application and Teacher Visualization.** Figure (a) shows the student problem-solving interface, with a calculator input interface at the bottom of the screen and badges at the top that indicate problem correctness. Figure (b) shows the teacher Dashboard that displays student problem-solving progress.

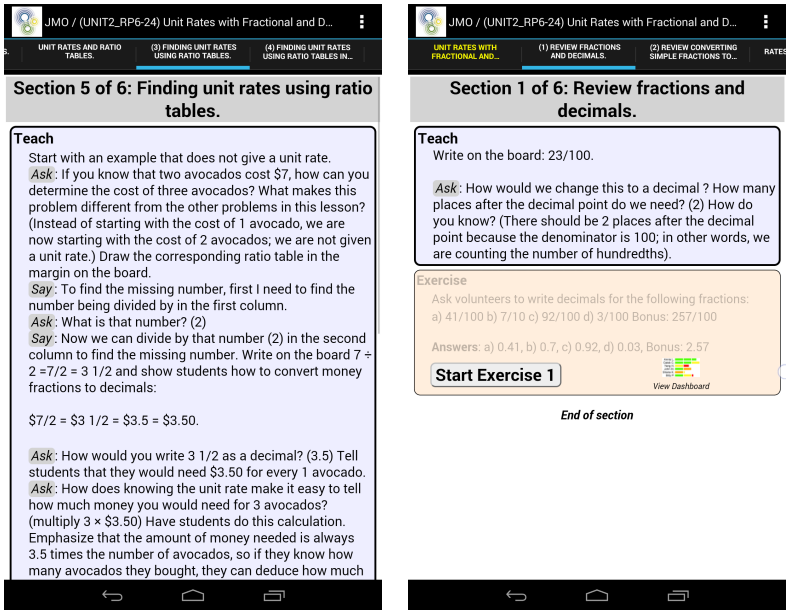
dent tablet as shown in Figure 5.1(a). Problems are displayed with blank spaces for student responses highlighted in blue. When the student taps on a blank space, the calculator interface for inputting a response opens at the bottom of the screen. After filling in the blanks, the student clicks a "Submit" button and the next problem is displayed. We communicate correctness information through the progression bar at the top of the screen. For each completed problem, a green badge is added if the problem is correct and a gray badge is added if it is incorrect. Students can revisit previous problems to fix mistakes at any time by clicking on the badge icons.

In addition to showing problem correctness, we adaptively select problems for each individual student. One of the challenges of integrating adaptivity into the classroom is that

students quickly become out-of-sync, making it difficult for the teacher to manage group activities and lessons [140]. To make adaptivity work in the classroom environment, we restrict our adaptive policy so that students are only given problems related to the concepts covered by the current lesson. For each lesson, we modeled the problem-solving thought process that students must apply to solve the example problems. We then generated a set of problems of varying levels of difficulty from this model, using an approach similar to [8]. By selecting problems from this set of related examples, we ensure that students stay in-sync with the rest of the class while allowing them to practice the concept at their current level.

We developed two separate adaptive policies, one for exercise problems and one for AP problems. Both are based on mastery learning, in that students must master each concept at a certain level before moving on to the next one [21]. Exercise problems in the JUMP curriculum are designed to help students learn newly introduced concepts, so we allow our policy to adapt freely for these problems. Students are given problems of a certain type until they pass our mastery measure, and only they are they advance to the next concept. In contrast, AP problems are designed to challenge students and prepare them for future assessment, so we chose to restrict adaptivity for these problems. For each problem type, students are given four problems from the original curriculum, and then assessed for mastery. If the student does not pass the mastery measure, up to two extra problems may be given, but then the student is advanced to the next problem type.

Our central design goal for this application was to provide some level of personalization while allowing students to move through the curriculum at the same pace. While this approach does not provide true mastery learning, this level of adaptivity is still much more personal than what students would experience with the standard JUMP curriculum. Furthermore, with access to immediate correctness information, students are able to identify and fix mistakes in real time.



(a) (b)

Figure 5.2: **Teacher Materials.** Figure (a) shows the teacher lesson view. Figure (b) shows a prompt for the teacher to "Start Exercise 1" on all student tablets.

5.3.3 Teacher Application

We designed the teacher application to directly reproduce the JUMP teacher materials. The teacher can reference the tablet application to view teaching and activity instructions, as shown in Figure 5.2(a). The navigation bar at the top of the screen can be used to move between lesson components. Most activities operate the same way as they would in paper-based JUMP classroom, except that lesson materials are accessed through the table application. However, when it is time to begin exercise or AP problems, the application displays a screen with a "Start Exercise" button, as shown in Figure 5.2(b). By pressing this button, the teacher can initiate the problem-solving activity on all student tablets.

As students work through practice problems, the teacher application displays real-time data through the table visualization shown in Figure 5.1(b). Each row in the table represents a student, each column represents a concept, and each dot represents a problem. Problem

dots are colored either red or green based on problem correctness, and the background of each row is colored according to average student performance across all problems attempted thus far. The goal of the dashboard is to allow teachers to monitor the pace and correctness of student work, and determine which students need individual assistance. Since the problem content is adaptive, this visualization does not show the specific problem each student is currently working on. However, the visualization provides high-level information showing which students are struggling most with the current lesson.

To support classroom orchestration, the application allows the teacher to pause all student tablets with the click of a button. This feature was designed to help teachers get students' attention to make announcements or re-teach concepts, and to ease the transition between problem-solving and then next lesson activity. Our central design goal for the teacher application was to provide real-time formative assessment data without disrupting the lesson workflow.

5.4 Study Method

The goal of our formative study was to discover how our new personalized learning system impacts student and teacher behavior in the classroom, and to identify areas for future iteration and improvement. To isolate the effects of our system, we designed an experiment with two conditions that kept both the curriculum and teacher constant, varying only the method of delivery.

Classes assigned to the experimental condition used our tablet-based version of the JUMP Math materials, described above. Classes assigned to the control condition used the standard paper version of the same materials. Teachers in the control classes accessed all lesson instructions in the paper teacher guide. During exercise activities, teachers wrote problems on the board, and students copied the problems into their paper notebooks to work out solutions. As a result, students worked through exercise problems at roughly the same pace. During AP activities, students worked on problems in their paper AP workbooks. In these workbooks, students could advance through problems at their own pace.

While we keep both the classroom teacher and lesson content constant, there are a number of differences between the two conditions. Students in the control condition used paper and pencil to work on a static progression of problems, while students in the experimental condition used tablets with touch inputs to work on a personalized progressions of problems. Our goal in this study is not to tease apart the relative impacts of each of these factors, but rather to gain a high-level understanding of how this particular system impacts student and teacher behavior in the classroom.

5.4.1 Participants

We recruited sixth-grade math teachers from four urban public schools to participate in our study. At each school, one teacher (or pair of co-teachers) participated. Each taught between two and four sixth-grade math classes, all of which were included in the trial. A total of 219 students participated in 11 classes. The schools represent a diverse set of communities; one serves a high-income neighborhood, and three serve different low-income neighborhoods. Students, teachers, parents, and principals gave informed consent for participation.

5.4.2 Procedure

At each school, one class was assigned to the control condition and the remaining classes were assigned to the experimental condition. We chose this approach because teachers wanted to use the technology in as many classes as possible. For a similar reason, we were not able to randomly assign classes to conditions. Teachers wanted to give their weaker classes the opportunity to benefit from the tablet-based system by assigning them to the experimental condition. One of the challenges of this type of fieldwork is that we must prioritize the needs of teachers and students.

We confirmed through an analysis of pre-test scores that students in the experimental classes at each school performed equally or significantly worse than those in the control class. This suggests that students' incoming ability does not bias our study in favor of the experimental condition. However, it is possible that teachers had other biases when

School	Unit 2 Start	Unit 2 End	Unit 4 Start	Unit 4 End
A	9/23/14	10/24/14	10/31/14	12/19/14
B	10/9/14	12/3/14	12/2/14	12/19/14
C	9/10/14	12/16/14	1/6/15	1/28/15
D	9/30/14	11/25/14	12/3/14	1/13/15

Table 5.1: **Study Dates.** The dates between which each school worked on JUMP Math Units 2 and 4. Due to interruptions and class pacing, teachers at each school spent different amounts of time on each unit.

assigning classes to conditions. As a result, we will need to confirm our findings in future studies. However, since five teachers with different biases participated, we expect any results that transcend these differences to be trustworthy. Furthermore, despite its flaws, we believe this study provides valuable insight into the impact of this personalized learning system on student and teacher behavior in the classroom.

We implemented two units from the JUMP Math curriculum in our system, Units 2 and 4. They were chosen because they cover closely related content on number systems, but they are not sequentially aligned in the curriculum. To stay true to the JUMP's design, teachers began by teaching Unit 1 on paper, then taught Unit 2 on either paper or tablets, then Unit 3 on paper, and finally Unit 4 on either paper or tablets. The dates that each school spent on Units 2 and 4 are given in Table 5.1.

The teachers who participated in the study use a variety of curriculums, and none were familiar with JUMP. While they may have reacted differently to our tablet-based system because they were learning a new curriculum, this design removes potential biases towards a familiar paper curriculum. To help familiarize teachers with the new materials, we conducted a curriculum training session at the beginning of the study. During the training, a JUMP Math representative described the curriculum philosophy and walked through example lessons. We also conducted a training to introduce teachers to the software, and taught students in the experimental classes to use the software during their first week of school.

5.4.3 Measures

The goal of this formative study is to gain a broad understanding of how classroom behavior is impacted by the introduction of our personalized learning system. Towards this end, we collected data on a wide variety of behavioral measures.

First, we gave students formal pre- and post-tests designed by JUMP before and after the two units to assess learning. The original curriculum only included one test for each unit, so we asked JUMP to create a second version of each test with the same items. This allowed us to give students different tests before and after the unit to control for learning effects. Half the students were given version A as their pre-test and version B as their post-test, while the other half were given the opposite. All tests were graded by the same researcher.

We also collected information about student problem-solving behavior throughout the study. We recorded two measures: the number of problems completed, and problem correctness. In the tablet classes, we logged student interactions with our system to capture this information. In the paper classes, we counted and graded AP problems, and counted exercise problems. The AP problems were clearly labeled in the JUMP AP workbook, but exercise problems were solved in students' personal notebooks, making them difficult to interpret and grade. To count exercise problems, graders used a well-organized notebook from each class as a template for counting the rest of the notebooks.

To learn how student engagement was impacted by our system, we observed students and logged information about their affect using the Baker-Rodrigo Ocumpaugh Monitoring Protocol (BROMP). BROMP is a protocol for capturing student affect through quantitative field observations [96] that has been used widely in educational technology research (e.g. [16, 110]). Under this protocol, a coder observes each student sequentially, assessing affect and logging it with a smartphone application. By looping through students repeatedly, the coder captures a large number of observations for each student. We recorded two variables: student *attention* (on-task, off-task, idle, or unknown) and student *mood* (positive, negative, neutral, or unknown). All coders were trained in advance and tested for inter-rater reliability.

We observed each class on three separate days, except at School B where we only observed twice due to scheduling constraints.

To measure how teachers behaved in tablet- and paper-based classes, we observed each class eight times during the study. Inspired by BROMP, we developed a tablet application for logging observations of teacher behavior which we call the "Teacher Taplog." This application provides buttons that the coder can press to log the current activity, such as "Teaching," "Demo," or "AP Problems." Actions that occur during an activity, such as "Assisting Student," can also be logged. All coders were staff members at the non-profit organization that we partnered with to develop the system, and all were trained in advance and tested for inter-rater reliability.

To learn about teachers' impressions of the tablet-based system, we asked each teacher to complete a survey about their experiences at the end of the study. The survey included both Likert-scale and free-form questions, which focused on the strengths and weaknesses of the system and the most positive and negative changes caused by introducing the system into classrooms.

Finally, we captured information about school demographics and teacher characteristics. The four schools that participated in our study serve different populations and present different challenges for teachers. Prior research shows that cultural, demographic, and organizational factors can influence the effectiveness of technology in the classroom [26, 97], so we wanted to consider these factors in our interpretation of our results. We looked at the publicly available demographic statistics for each school, and also conducted a survey of the four non-profit staff members who spent time in all four schools. These staff observed and supported teachers weekly throughout the trial. The survey included both Likert-scale and free-form questions, which focused on the quality of teacher instruction, the level of support principals provide for teachers, and the level of challenge presented by students (e.g. stress, issues at home, English language learners).

5.4.4 Analysis

In our analysis, we treat the four schools that participated in the study as four separate experiments. The standard method of analyzing a multi-school experiment in educational research is to use a nested analysis technique such as Hierarchical Linear Models [95]. However, in order to have the statistical power to measure differences between schools with a hierarchical model, a large number of schools is required [79]. We therefore chose to analyze the data from each school separately. Our design still allows us to informally compare findings across schools, providing more information about how our results generalize than we could have gained by studying a single teacher at a single school.

Within each school, our experimental design has a single factor *Condition* with two levels, experiment and control. All of our outcome measures were continuous. For each measure at each school, we first analyzed the Shapiro-Wilk test to assess the normality of our data, and we found that a number of our measures were non-normally distributed. We ran an independent-samples t-test for the measures that were normally distributed, and a Wilcoxon rank sums test for those that were non-normally distributed. We report either mean or median values for our measures, depending on the normality of the data, along with the results of the statistical test. Since we are treating each school as a separate experiment, we run a large number of statistical comparisons and risk inflating alpha. To address this issue, we sum p-values across all comparisons for each outcome measure to ensure that the combined alpha still falls below the 0.05 threshold.

We report effect sizes in addition to p-values to show the magnitude of differences. We use a Cohen's d measure of effect size for our normally distributed measures and an r measure of effect size for our non-normally distributed measures. For Cohen's d, effects with values 0.2 are *small*, 0.5 are *moderate*, and 0.8 or greater are *large*. For r, effects with values of 0.1 are *small*, 0.3 are *moderate*, and 0.5 or greater are *large*.

School	Quality	Org.	Support	Prep.	Rapport	Challenges
A	4	4	4.33	4.33	4.33	1.67
B	1.33	1	1.5	2	2.33	3.67
C	3	3	1.67	3.25	4.25	4.75
D	4.75	5	4.5	5	4.5	2.75

Table 5.2: **School Characteristics Survey.** Results for the Likert-scale questions of our survey of classroom observers. For each school, we collected data on the overall quality of the teacher, the level of class organization, the level of support provided by school administrators, the level of teacher preparedness, the rapport between teachers and students, and the level of challenge presented by students (issues at home, English language learners, etc).

5.5 School Characterization

Before presenting the results of our analysis, we provide a characterization of each school environment. Table 5.2 shows staff observers' average responses to our survey questions and Table 5.3 reports school demographic data collected from publicly available sources. By combining these two sources of data, we are able to gain some insight into the environment at each school. These school factors provide possible explanations for some of our results that vary across schools. Given our experiment design, we are not able to show causal relationships between classroom characteristics and the study results. Our interpretations of possible connections between classroom factors and results are simply hypotheses. However, we believe that this information provides a richer picture of the contexts within which our findings are situated.

School A. This school serves a high-income, predominantly white neighborhood. Two teachers co-taught, leading lessons in turn. Observers reported that both teachers were prepared, kept the classroom organized, and supported each other effectively. They had a strong grasp on lesson content, and changed teaching tactics throughout the day if a particular approach was not effective. One observer noted that "*the teachers have a strong*

School	Teaching	# Classes	# Students	Free Lunch	ELL	Math Score	Reading Score
A	Co-Taught	4	72	10.0%	1.9%	88.2%	90.4%
B	One Teacher	2	39	53.0%	7.0%	55.0%	69.5%
C	One Teacher	3	63	63.6%	16.4%	56.7%	70.3%
D	One Teacher	2	45	50.8%	5.7%	64.0%	64.0%

School	American Indian	Asian / Pacific Islander	Black	Hispanic	White	2+ Races
A	0.4%	10.8%	2.3%	7.1%	72.1%	7.3%
B	0.6%	16.1%	12.1%	18.9%	37.8%	14.4%
C	1.0%	27.1%	41.3%	10.5%	11.2%	8.9%
D	2.5%	14.9%	14.6%	23.4%	31.4%	13.4%

Table 5.3: **School Demographics.** Information about the four schools that participated in our study, including the number of teachers, classes, and students at each school. We also report publicly available data on the demographics of each school such as the percentage of students who are eligible for free or reduced lunch, and the percentage of students who are English language learners (ELL). We also report school performance on standardized math and reading tests.

rapport with the students, leading to greater connection and willingness to pursue help during free periods and even lunch time." Students were generally well behaved, with little need for disciplinary intervention. Although a few students had behavioral or learning challenges, they often had personal aides. Teachers were well-supported by the school's principal, who visited classes occasionally.

School B. This school serves a racially diverse low-income neighborhood. Observers reported that this teacher was ineffective and unable to restore order once students started to go off-task. The classroom was disorganized, and the teacher seemed inattentive. One observer noted that "*on every count, the instruction was weak.*" Although the teacher and students seemed to get along, his disciplinary approach did not work; consequently, both teacher and students would often raise their voices to draw attention, resulting in a chaotic learning environment. The students faced a moderate amount of home challenges, and a significant portion of students were English Language Learners (ELL), but external help was not present. The teacher did not effectively support students in figuring out how to be more engaged. Likewise, he did not receive support or feedback to improve his teaching.

School C. This school also serves a racially diverse low-income neighborhood. The teacher was typically prepared for class, although "*not ready to teach in any engaging way.*" Call-and-response or fill-in-the-blank teaching was common, which failed to retain some students' focus. However, the teacher was liked and respected by students. The students at this school presented many challenges; "*most of them have some sort of trauma that they are dealing with*" such as homelessness, unstable home environments, and language barriers. Despite these issues, the school only offered one part-time counselor. Observers reported that the principal was rarely on campus, and teachers saw her as "*having abandoned the children.*" Classroom activities were often disrupted by last minute schedule changes. Overall, the classroom environment was erratic, but the teacher managed her students well.

School D. This school serves a racially diverse low-income neighborhood, but is co-managed by the state and a non-profit organization with a STEM-education focus. The teacher was highly prepared, providing strong support and high-quality instruction. The

classroom was "*unbelievably organized,*" and students were obedient at all times while still having fun. One observer said that "*students would rush to do whatever she asked.*" Students showed respect, and the teacher reciprocated. The school principal dropped by occasionally, and the teacher kept in regular contact with both the principal and the school technology lead. Although many students came from immigrant families and faced challenges at home, there were no obvious signs of these issues based on the effective classroom management. One observer noted that the families who enrolled their children in this specially funded school were likely to be invested in their children's education.

5.6 Student Results

We first wanted to gain an understanding of how the introduction of our personalize learning system impacted student behavior. As discussed in the Measures section above, we assessed student learning, measured problem-solving behavior, and observed student engagement.

5.6.1 Student Learning

To determine how student learning outcomes differ in the two conditions, we examined the formal pre- and post-tests administered before and after each JUMP unit. We analyzed this data using a repeated measures ANOVA to measure the effect of *Condition* and *Test* (either pre-test or post-test). We were particularly interested in identifying any *Condition*Test* interactions, which would occur if the learning gains between pre- and post-test differed based on condition. We initialized analyzed the tests for Unit 2 and Unit 4 separately, but found that the results were the same for both units. For simplicity, we averaged the results of the pre-tests and post-tests for each unit and analyzed these averages. The results of the statistical tests are given in Table 5.4.

As mentioned in the Procedure section above, the mean pre-test scores in the tablet classes were lower than those in the paper classes at all four schools. Students in the tablet class at School B performed significantly worse than students in the paper class on both the pre- and post-tests. Our analysis shows that *Test* had a strong significant effect at all four schools,

School	Condition	Mean Pre	Mean Post	Condition		Test		Condition*Test	
A	Tablet	36.9	67.0	$DF = 70$	*n.s.*	$DF = 70$	$p < 0.0001$	$DF = 70$	*n.s.*
	Paper	38.8	70.4	$F = 0.6$		$F = 201.2$		$F = 0.1$	
B	Tablet	14.2	28.4	$DF = 37$	$p < 0.0001$	$DF = 37$	$p < 0.0001$	$DF = 37$	*n.s.*
	Paper	29.7	49.1	$F = 26.7$		$F = 57.8$		$F = 1.4$	
C	Tablet	39.1	60.0	$DF = 60$	*n.s.*	$DF = 60$	$p < 0.0001$	$DF = 60$	*n.s.*
	Paper	46.2	68.5	$F = 2.4$		$F = 155.8$		$F = 0.1$	
D	Tablet	37.9	68.2	$DF = 43$	*n.s.*	$DF = 43$	$p < 0.0001$	$DF = 43$	*n.s.*
	Paper	39.2	62.6	$F = 0.2$		$F = 140.3$		$F = 2.3$	

School	Condition	# Exercise Problems			# AP Problems			% AP Correct		
A	Tablet	506	$N = 72$	$p < 0.0001$	629	$N = 72$	$p < 0.0001$	98%	$N = 71$	$p < 0.0001$
	Paper	78	$Z = -5.9$	$r = 0.69$	400	$Z = -5.6$	$r = 0.66$	92%	$Z = -5.6$	$r = 0.65$
B	Tablet	898	$N = 39$	$p < 0.0001$	179	$DF = 37$	*n.s.*	88%	$N = 38$	*n.s.*
	Paper	284	$Z = 5.3$	$r = 0.84$	184	$t = -0.2$		80%	$Z = -5.6$	
C	Tablet	766	$N = 63$	$p < 0.0001$	466	$DF = 62$	$p < 0.001$	95%	$N = 63$	$p < 0.0001$
	Paper	135	$Z = -6.3$	$r = 0.80$	552	$t = -3.5$	$d = 0.90$	86%	$Z = -5.6$	$r = 0.54$
D	Tablet	1807	$N = 45$	$p < 0.0001$	364	$N = 45$	$p < 0.0005$	98%	$N = 45$	$p < 0.0001$
	Paper	390	$Z = 5.7$	$r = 0.86$	262	$Z = 3.6$	$r = 0.53$	89%	$Z = -5.6$	$r = 0.70$

Table 5.4: **Student Behavior Analysis.** Statistical results from our analysis of student behavior. The top table reports results from the repeated measures ANOVA analysis of pre-test and post-test scores, showing the effects of *Condition*, *Test*, and the *Condition*Test* interaction. The rightmost column reports the analysis of the learning gain between the two tests. The bottom table reports on student problem solving throughout the 10-week study, specifically the number of exercise and AP problems completed, along with AP correctness. The mean/median values are reported to the left of each column.

indicating that students learned between the two tests. However, we measured no significant *Condition*Test* interaction. We also analyzed the average learning gain between the pre- and post-tests at each school, and found no significant differences based on *Condition*. This analysis suggests that our personalization system had no positive effects on student learning. Students in the tablet classes did not exhibit larger learning gains between the pre- and post-tests than students in the paper classes.

5.6.2 Student Problem-Solving Behavior

While our system had no positive impact on learning gains, we were also interested in understanding how our system impacted problem-solving behavior during class. We analyzed data from all of the exercise and AP problem-solving sessions that took place during the trial, comparing both the number of problems completed and AP problem correctness. The results of this analysis are shown in Table 5.4.

In this analysis, we found that students in the tablet classes completed significantly more exercise problems than students in the paper classes at all four schools. The median number of problems completed varies by school, but the effect sizes between 0.69 and 0.86 indicate that this was a large effect. However, the story is not as clean for the AP problems. At schools A and D we see the same effect, with students in the tablet classes completing significantly more problems. However, we measured no significant effect at school B, and the *opposite* effect at school C, where students in the paper class complete significantly more problems. It is possible that the unrestricted adaptive policy we used for exercise problems allowed students to complete more than the policy used for AP problems. However, this does not explain why we see such different results for AP problems across the four schools.

Finally, we analyzed student performance on the AP problems. We found that students in the tablet classes at schools A, C, and D performed significantly better than those in the paper classes. There was no significant effect at school B, however students in the tablet class at this school performed significantly worse on the pre-tests. The fact that there was no significant difference in their performance on AP problems suggests that our system had the

School	Condition	% On-Task			% On-Task Excitement		
A	Tablet	78%	$N = 71$	*n.s.*	1.5%	$N = 71$	$p < 0.001$
	Paper	73%	$Z = -1.4$		0%	$Z = -3.9$	$r = 0.46$
B	Tablet	70%	$N = 38$	$p < 0.05$	0%	$N = 38$	*n.s.*
	Paper	83%	$Z = -2.4$	$r = 0.39$	0%	$Z = -0.4$	
C	Tablet	86%	$N = 62$	$p < 0.005$	1.6%	$N = 62$	$p < 0.005$
	Paper	74%	$Z = -3.4$	$r = 0.43$	0%	$Z = -3.2$	$r = 0.40$
D	Tablet	90%	$N = 45$	*n.s.*	1.9%	$N = 45$	*n.s.*
	Paper	90%	$Z = 0.2$		1.3%	$Z = 0.5$	

Table 5.5: **Student Engagement.** Statistical tests showing the differences in time observed time on-task and observed on-task excitement events. In each column, median values are given in the left of each cell, along with the test results.

same effect at school B. This result is surprising given that we saw no performance differences on the formal pre- and post-tests. It could be that students in the tablet classes reacted to correctness feedback and fixed mistakes, resulting in a better final performance. It is also possible that students in the tablet classes learned more, but that these learning gains did not transfer onto paper.

5.6.3 Student Engagement

We also examined student engagement by analyzing the BROMP data collected by our classroom observers. The results of this analysis are shown in Table 5.5. First, we analyzed how often students were observed to be on-task. We found no significant effect of *Condition* on time on-task at schools A and D, where teachers were organized and classroom management issues were minimal. However, at school B, students in the tablet class spent significantly *less* time on-task than students in the paper class. The teacher at this school was disorganized and struggled with classroom management, so it is possible that orchestrating tablets on top of other challenges was too much for this teacher. In contrast, at school C students in the tablet classes spent significantly *more* time on-task than students in the paper classes. The

teacher at school C was effective, but faced many challenges. It is possible that the tablets allowed students remain on-task while the teacher handled classroom management issues.

We also analyzed the amount of on-task excitement that students displayed. Overall, very few on-task excitement events were observed. However, at schools A and C there were significantly more on-task excitement events in the tablet classes than in the paper classes. There was no significant effect at school B, where there were very few excitement events observed in either class, or at school D, where relatively high levels of excitement were observed in both classes. Again, these variations are likely impacted by the differences in classroom environment at each school.

5.7 Teacher Results

Next, we were interested in understanding how the introduction of our personalized learning system impacted teachers. As discussed in the Measures section above, we captured detailed information about teacher activity in the classroom using our Teacher Taplog application. We also report on the results of our teacher survey.

5.7.1 Teacher Behavior

The results of our statistical analyses of teacher behavior are shown in Table 5.6. First, we examined how often teachers assist individual students during problem-solving activities. We define an assist as a one-on one interaction lasting more than 30 seconds. We hypothesized that teachers might assist students more frequently when they have access to real-time formative assessment data that highlights struggling students. We found that in all four schools, teachers assist individual students more often in the tablet classes than in the paper classes. While the mean number of assists varies across schools, the effect is significant at all four, suggesting that our system encourages teachers to provide more individual help.

We also analyzed the number of "re-teach" events in each class, in other words when teachers go back to teach a concept again after realizing that students are struggling. We hypothesized that teachers might re-teach more often with access to real-time data that

School	Condition	# Student Assists			# Re-Teaches		
A	Tablet	7.5	$N = 32$	$p < 0.05$	0	$N = 32$	$p = 0.07$
	Paper	2.5	$Z = -2.2$	$r = 0.38$	0	$Z = -1.8$	$r = 0.32$
B	Tablet	20.4	$DF = 12$	$p < 0.05$	0	$N = 14$	*n.s.*
	Paper	11.9	$t = 2.9$	$d = 1.66$	0	$Z = 0.9$	
C	Tablet	10	$N = 24$	$p < 0.01$	0	$N = 24$	*n.s.*
	Paper	1.5	$Z = -2.6$	$r = 0.58$	0	$Z = -0.6$	
D	Tablet	17.7	$DF = 15$	$p < 0.001$	0	$N = 17$	*n.s.*
	Paper	5.8	$t = 4.3$	$d = 2.2$	0	$Z = -1.0$	

School	Condition	Teaching Time			EX Time			AP Time		
A	Tablet	12.0	$N = 32$	*n.s.*	0.0	$N = 32$	*n.s.*	8.0	$N = 32$	*n.s.*
	Paper	19.5	$Z = 1.1$		0.0	$Z = 0.0$		7.0	$Z = 0.1$	
B	Tablet	18.1	$DF = 12$	*n.s.*	17.4	$DF = 12$	$p < 0.05$	1.0	$N = 14$	*n.s.*
	Paper	18.9	$t = -0.2$		12.5	$t = 2.5$	$d = 1.47$	0.0	$Z = -0.3$	
C	Tablet	11.5	$3N = 24$	*n.s.*	14.5	$N = 24$	*n.s.*	0.0	$N = 24$	*n.s.*
	Paper	11.5	$Z = -0.3$		13.0	$Z = -0.1$		0.0	$Z = 0.5$	
D	Tablet	9.0	$DF = 15$	*n.s.*	19.0	$N = 17$	$p < 0.05$	8.0	$N = 17$	*n.s.*
	Paper	8.0	$t = 0.0$		13.5	$Z = -2.3$	$r = 0.56$	0.0	$Z = -1.3$	

Table 5.6: **Teacher Behavior Analysis.** Statistical results from our analysis of teacher behavior. The top table reports the number of times teachers assisted individual students and the number of times teacher re-taught lesson content. The bottom table repots how class time was spent, specifically the amount of time (in minutes) spent on teaching, exercise problems, and AP problems. The mean/median values are reported to the left of each column.

shows when students are performing poorly. However, we found no significant differences in the number of re-teach events at any of the schools. There was a trend towards more re-teaches in the tablet classes at school A, but this effect was not significant at the 0.05 level. Overall, re-teach events were rare. This could be a result of the highly scripted curriculum, which did not encourage deviating from the lesson plan. It is also possible that teachers chose to assist individual students rather than re-teaching concepts to the entire class.

Finally, we looked at how teachers spent class time in the two conditions. We found no significant differences in the amount of time spent teaching. However, at schools B and D, teachers devoted significantly more time to exercise problems in their tablet classes. Looking back at the exercise problem results from the previous section, we see that the largest effects occurred at these two schools. There was no significant difference in the amount of time spent on AP problems at any of the schools, but there are large differences in the median time spend on AP problems. At schools A and D, where teachers spent the most time on AP problems, the tablet classes also completed significantly more AP problems than the paper classes. This suggests that the restricted adaptive policy may eventually allow students to complete more AP problems, but that it takes time for these effect to become visible.

5.7.2 Teacher Impressions

Overall, teachers viewed the system positively. When asked what was the single biggest benefit to their teaching practice, four out of five cited the real-time data communication features. One teacher noted that "*it was very easy to see what kids were struggling with the problems.*" Another said "*It is a great way to see how EVERY student is doing.*" When asked if using the tablet enhanced their ability to monitor student progress during class, teachers responded affirmatively, with an average score of 3.8 on a 4 point Likert scale.

Teachers also mentioned that students were more engaged while using tablets. One noted that "*students were catching their misunderstandings quickly and asking for help*". Another highlighted a benefit to classroom management, stating that "*students were engaged with learning while I was helping individuals with the work.*" Another mentioned that the system

increased accountability, saying that "*students couldn't just sit there pretending to know what they were doing.*"

Despite this positive feedback, teachers highlighted some issues with the system. One noted that "*the fill-in-the-blanks aspect seemed to prevent students from making important connections.*" Another cited gaming behavior: "*a few students would just mash random answers in an attempt to see how many problems they could do.*" When asked what was the biggest detractor from their teaching practice, two teachers said that they wanted more control over the adaptivity. One said it was frustrating "*not being able to adapt the lessons to students who needed accommodations*" and the other complained that "*students who finished quickly or understood the concept already could not work ahead in the next lesson.*"

5.8 Design Recommendations

Our formative research provides new insights into the behavioral impact of a personalized learning system designed to augment and existing mathematics curriculum by providing both real-time data communication and automated personalization of problems and feedback. In this section, provide a set of design recommendations based on our findings and highlight some important areas for future research.

Adaptive Policy Design. Our study shows that adaptive learning can be integrated into a traditional classroom to allow students to practice concepts at their own pace while ensuring that they stay in-sync with the rest of the class. Students solved significantly more problems with our system in some cases, but we also saw that the adaptive policy design had a strong impact on this effect. We observed a much stronger effect when we used an unrestricted policy for exercise problems, compared to the more constrained policy we used for AP problems. Furthermore, multiple teachers mentioned that they wished they could manually adapt content for students, suggesting that neither of our adaptive policies was providing optimal lesson pacing automatically. This highlights a need to conduct more research on adaptive policy design to more effectively support personalization within the constraints of the traditional classroom.

Learning Transfer Design. We found that students performed significantly better on AP problems in classes that used our tablet software, but these learning gains did not transfer onto the JUMP post-tests administered at the end of each unit. Transferring knowledge from one context to another is a known challenge in education [78, 85], and our tablet-based problem-solving environment provides a very different experience than paper. One teacher said that the fill-in-the-blanks interface could be hindering student understanding, and its possible that the tablet input modalities were too far removed for learning to transfer onto paper tests. The software also provides scaffolding in the form of correctness feedback, and it may be that students were not yet ready to have that support removed. In any case, further research is needed to understand the design factors that inhibit transfer, and how personalization systems can be designed to support learning more effectively.

Data Communication Design. Our results demonstrate that real-time data communication features are useful and can have a positive impact on teacher behavior. Teachers in all four schools assisted students significantly more often when they had access to real-time formative assessment data. They also cited this feature as the most valuable part of our system in the teacher surveys. However, while it is clear that data communication has great potential in the classroom, many open questions remain. While teachers helped more students in tablet classes, this did not have a significant impact on post-test scores. More research is needed to understand when personal assistance is effective, and how technologies can be designed to support effective assistance in the classroom.

Classroom Environment Design. Our results show that the impact of a personalized learning system can vary based on the classroom environment and teacher characteristics. In particular, we found that our system increased time-on-task at one school, but decreased it at another. This finding supports prior research on the importance of cultural, demographic, and organizational factors [26, 97], and suggests that interventions that are successful in low-challenge schools with highly competent teachers may not transfer to other environments. Further research is needed to understand how personalization technology can be designed to support struggling teachers, whose students could most benefit from additional support.

5.9 Limitations and Future Work

While this work makes many contributions in the domain of personalized learning systems, it also has a number of limitations that will need to be addressed in future work. Due to the constraints of working in real-world classrooms, we were not able to randomize the assignment of classes to experimental conditions. Furthermore, our study was not designed to tease apart the relative impacts of the different features of our system. For example, it is not possible to determine if students solve more exercise problems because of our adaptive policy, or the availability of real-time feedback, or additional teacher assistance. Finally, we did not interview students to learn about their impressions of the tablet-based software, which could provide valuable insights and suggest areas for iteration and improvement. These are questions that must be addressed in future research.

The student application also has a number of limitations. Research in cognitive tutors, and our own work on instructional scaffolding presented in Chapters 3 and 4, has highlighted the benefit of providing students with explanations and feedback through personalized learning applications. The student application used in this study only provides correctness feedback. Furthermore, the application interface was only designed to collect problem solutions, rather than the steps that students took to arrive at that solution. This makes it impossible to detect student misconceptions automatically. It is possible that our intervention did not produce significantly learning gains in part because it did not provide more sophisticated scaffolding. This will be an important question to study in future experiments measuring the effectiveness of personalized learning systems for the classroom.

Finally, this formative research highlights a number of research questions that we would like to explore in the future. As described in detail in the Design Recommendations section above, many open questions remain around the design of adaptive policies, supporting transfer between digital and paper environments, supporting effective teacher assistance, and designing technologies that support diverse classroom environments. This work takes a small step into a much larger research area that we will be exploring for decades to come.

5.10 Conclusion

In this chapter, we present formative research that works towards the development of an ideal personalization system for the classroom context. We introduce a new tablet-based system that augments an existing curriculum by providing digital teacher materials, an adaptive problem-solving application for students, and real-time visualizations of student progress for teachers. To study how this system impacts classroom behavior, we conducted a ten-week study with eleven sixth-grade math classes in four urban schools. We used a mixed-methods approach, and examined a wide variety of outcome measures to understand student behavior and engagement, teacher behavior and impressions, and classroom factors.

Our analysis shows that students complete more problems with more accuracy in classes that use our tablet-based system. Teachers also assist individual students significantly more often when they have access to real-time formative assessment data. However, student performance gains do not transfer onto a paper post-test. We also find that our system is more effective at keeping students on-task in classrooms with organized teachers who can manage the technology. We present a set of design recommendations given these findings, and suggest areas for future research in the domains of adaptive policy design, learning transfer, and data communication. We believe this formative research provides valuable insights into the behavioral impact of personalized learning systems that makes an important contribution to research in this field.

Chapter 6

CONCLUSION

Educational technology has great potential to help teach students the skills they will need to succeed in our modern economy. However, despite decades of research and many exciting advances in the field, educational technology has not fundamentally changed the state of education. One of the central problems with the current approach to educational technology design is that computer scientists are trying to reinvent the wheel, rather than learning from prior research in education and the learning sciences. Researchers in these domains have been studying challenges such as student motivation, classroom integration, and learning transfer for decades, and their findings have great potential to inform the design of new educational technologies.

This dissertation combines techniques from computer science and the learning sciences to develop novel technical systems grounded in learning theory that tackle core challenges in education. My collaborators and I applied this approach to address challenges in three key areas. First, we present a new method of directly rewarding growth mindset behaviors through a novel "brain points" incentive structure, and demonstrate that this intervention increases student persistence and use of strategy. Next, we present new approaches for automatically generating instructional scaffolding for procedures and algebraic problem solving, and demonstrate their viability through proof-of-concept implementations and user studies. Finally, we present a personalization system designed to support both adaptive problem progressions and real-time formative assessment in the classroom, and show that it encourages students to complete more problems and teachers to assist more individual students. This research demonstrates the depth of the interdisciplinary research space at the intersection of computer science and the learning sciences. Furthermore, it highlights the value of leveraging

findings from the learning sciences literature in the design of new educational technologies. The systems presented in this dissertation can have an immediate impact on learning.

While this dissertation makes many important contributions to the field of educational technology, we have only begun to scratch the surface of what is possible when technical innovations are grounded in learning theory and designed to transform how we teach and learn. This research tackles three separate educational challenges: teaching the growth mindset through incentives, supporting student learning through interactive scaffolding, and promoting personalization in the classroom. However, these approaches could be integrated into a single learning system in the future. Growth mindset incentives could be incorporated into an interactive learning system that provides personalized interactive scaffolding, and that system could be designed to support student and teacher interactions in the classroom. Ultimately, these ideas could be combined to work towards to the goal of leveraging personalized data-driven systems to meet the unique needs of each student, teacher, and classroom. Many research questions need to be explored before this vision becomes a reality, and I plan to continue exploring this interdisciplinary space throughout my career.

BIBLIOGRAPHY

[1] Trends in Academic Progress. (NCES 2013-456). U.S. Department of Education. White paper, Washington, DC: National Center for Education Statistics, 2013.

[2] The Condition of College and Career Readiness. White paper, ACT, Inc., 2015.

[3] A. L. Abrahamson. An overview of teaching and learning research with classroom communication systems. Paper presented at the Samos International Conference on the Teaching of Mathematics, Village of Pythagorion, Samos, Greece, July, 1998, 1998.

[4] Umair Z. Ahmed, Sumit Gulwani, and Amey Karkare. Automatically generating problems and solutions for natural deduction. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, pages 1968–1975. AAAI Press, 2013.

[5] HamedS. Alavi, Pierre Dillenbourg, and Frederic Kaplan. Distributed awareness for class orchestration. In Ulrike Cress, Vania Dimitrova, and Marcus Specht, editors, *Learning in the Synergy of Multiple Disciplines*, volume 5794 of *Lecture Notes in Computer Science*, pages 211–225. Springer Berlin Heidelberg, 2009.

[6] Vincent Aleven, Bruce M. McLaren, Jonathan Sewall, and Kenneth R. Koedinger. The cognitive tutor authoring tools (CTAT): preliminary evaluation of efficiency gains. In *International Conference on Intelligent Tutoring Systems*, ITS'06, pages 61–70, 2006.

[7] Vincent Aleven, Bruce M. Mclaren, Jonathan Sewall, and Kenneth R. Koedinger. A new paradigm for intelligent tutoring systems: Example-tracing tutors. *Int. J. Artif. Intell. Ed.*, 19(2):105–154, April 2009.

[8] Erik Andersen, Sumit Gulwani, and Zoran Popović. A trace-based framework for analyzing and synthesizing educational progressions. In *CHI*, pages 773–782, 2013.

[9] Erik Andersen, Eleanor O'Rourke, Yun-En Liu, Richard Snider, Jeff Lowdermilk, David Truong, Seth Cooper, and Zoran Popović. The impact of tutorials on games of varying complexity. In *CHI '12: Proceedings of the SIGCHI conference on Human factors in computing systems*, New York, NY, USA, 2012. ACM.

[10] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray Pelletier. Cognitive tutors: Lessons learned. *The Journal of the Learning Sciences*, 4(2):167–207, 1995.

[11] John R. Anderson and Ray Pelletier. A development system for model-tracing tutors. In *Proceedings of the International Conference of the Learning Sciences*, pages 1–8, 1991.

[12] Richard Anderson, Ruth Anderson, K. M. Davis, Natalie Linnell, Craig Prince, and Valentin Razmov. Supporting active learning and example based instruction with classroom technology. *SIGCSE Bull.*, 39(1):69–73, March 2007.

[13] Joshua Aronson, Carrie B. Fried, and Catherine Good. Reducing the Effects of Stereotype Threat on African American College Students by Shaping Theories of Intelligence. *Journal of Experimental Social Psychology*, 38, 2002.

[14] Robert K. Atkinson, Alexander Renkl, and Mary M. Merrill. Transitioning from Studying Examples to Solving Problems: Effects of Self-Explanation Prompts and Fading Worked-Out Steps. *Journal of Educational Psychology*, 95(4):774–83, 2003.

[15] R. Baker, J. Walonoski, N. Heffernan, I. Roll, A. Corbett, and K. R.. Koedinger. Why students engage in "gaming the system" behavior in interactive learning environments. *Journal of Interactive Learning Research*, 19(2):185–224, 2008.

[16] Ryan Shaun Baker, Albert T. Corbett, Kenneth R. Koedinger, and Angela Z. Wagner. Off-task behavior in the cognitive tutor classroom: When students "game the system". In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 383–390, 2004.

[17] Lawrence Bergman, Vittorio Castelli, Tessa Lau, and Daniel Oblinger. Docwizards: A system for authoring follow-me documentation wizards. In *UIST*, pages 191–200, 2005.

[18] P. Black and D Wiliam. Assessment and classroom learning. *Assessment in Education*, 5(1):7–74, 1998.

[19] Lisa S. Blackwell, Kali H. Trzesniewski, and Carol Sorich Dweck. Implicit Theories of Intelligence Predict Achievement Across an Adolescent Transition: A Longitudinal Study and an Intervention. *Child Development*, 78(1):246–263, 2007.

[20] Stephen B. Blessing, Stephen B. Gilbert, Stephen Ourada, and Steven Ritter. Authoring model-tracing cognitive tutors. *International Journal of Artificial Intelligence in Education*, 19(2):189 – 210, 2009.

[21] B. S. Bloom. Learning for mastery. *Evaluation Comment*, 1(2):1–12, 1968.

[22] Benjamin S. Bloom. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational Researcher*, 13(6):4–16, 1984.

[23] J. Bransford, S. Brophy, and S. Williams. When computer technologies meet the learning sciences: Issues and opportunities. *Journal of Applied Developmental Psychology*, 21(1):59–84, 2000.

[24] Keith Brawner, Heather Holden, Benjamin Goldberg, and Robert Sottilare. Recommendations for modern tools to author tutoring systems. In *The Interservice/Industry Training, Simulation & Education Conference (I/ITSEC)*, 2012.

[25] Eric Butler, Erik Andersen, Adam M Smith, Sumit Gulwani, and Zoran Popovic. Automatic game progression design through analysis of solution features. 2015.

[26] Rachelle Campigotto, Rhonda Mcewen, and Carrie Demmans Epp. Especially social: Exploring the use of an ios application in special needs classrooms. *Comput. Educ.*, 60(1):74–86, January 2013.

[27] C. Chase. The interplay of chance and skill: Exploiting a common game mechanic to enhance learning and persistence. In *Proceedings of the 2012 International Conference of the Learning Sciences*, 2012.

[28] Pei-Yu Chi, Sally Ahn, Amanda Ren, Mira Dontcheva, Wilmot Li, and Björn Hartmann. MixT: automatic generation of step-by-step mixed media tutorials. In *UIST*, pages 93–102, 2012.

[29] A. Corbett, K. R. Koedinger, and J. R Anderson. Intelligent tutoring systems. In M. Helander, T. K. Landauer, and P. Prahu, editors, *Handbook of Human-Computer Interaction, Second Edition*, pages 849–874. Elsevier Science, Amsterdam, 1997.

[30] Albert T. Corbett. Cognitive computer tutors: Solving the two-sigma problem. In *Proceedings of the 8th International Conference on User Modeling 2001*, UM '01, pages 137–147, London, UK, UK, 2001. Springer-Verlag.

[31] Albert T. Corbett and John R. Anderson. Knowledge tracing: Modeling the acquisition of procedural knowledge. *User Modeling and User-Adapted Interaction*, 4(4):253–278, 1995.

[32] V. M. Crawford, M. Schlager, W. R. Penuel, and Y. Toyama. Supporting the art of teaching in a data-rich, high performance learning environment. In E. B. Mandinach and M. Honey, editors, *Linking data and learning*, pages 109–129. Teachers College Press, New York, 2008.

[33] Pierre Dillenbourg and Patrick Jermann. Technology for classroom orchestration. In Myint Swe Khine and Issa M. Saleh, editors, *New Science of Learning*, pages 525–552. Springer New York, 2010.

[34] Son Do-Lenh, Patrick Jermann, Amanda Legge, Guillaume Zufferey, and Pierre Dillenbourg. Tinkerlamp 2.0: Designing and evaluating orchestration technologies for the classroom. In Andrew Ravenscroft, Stefanie Lindstaedt, CarlosDelgado Kloos, and Davinia Hernandez-Leo, editors, *21st Century Learning for 21st Century Skills*, volume 7563 of *Lecture Notes in Computer Science*, pages 65–78. Springer Berlin Heidelberg, 2012.

[35] Robert J. Dufresne, William J. Gerace, William J. Leonard, Jose P. Mestre, and Laura Wenk. Classtalk: A classroom communication system for active learning. *Journal of Computing in Higher Education*, 7(2):3–47, 1996.

[36] Carol S. Dweck. *Mindset: The new psychology of success*. New York: Random House, 2006.

[37] Mingyu Feng and Neil T. Heffernan. Towards live informing and automatic analyzing of student learning: Reporting in assistment system. *Journal of Interactive Learning Research*, 18(2):207–230, April 2007.

[38] L. S. Fuchs and D. Fuchs. Effects of systematic formative evaluation: a meta-analysis. *Exceptional Children*, 53(5):199–208, 1986.

[39] Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Schneider. Potassco: The potsdam answer set solving collection. *AI Communications*, 24(2):107–124, 2011.

[40] James P. Gee. *What Video Games Have to Teach Us About Learning and Literacy*. St. Martin's Press, revised and updated edition. edition, March 2008.

[41] James Paul Gee. Learning by design: Good video games as learning machines. *E-Learning and Digital Media*, 1(1):5–16, 2005.

[42] J.P. Gee and D.W. Shaffer. Looking where the light is bad: Video games and the future of assessment. (Epistemic Games Group Working Paper No. 2010-02), Madison: University of Wisconsin-Madison, 2010.

[43] Krista D. Glazewski and Peggy A. Ertmer. Scaffolding disciplined inquiry in problem-based learning environments. *International Journal of Learning*, 12(6):297–306, 2005.

[44] Catherine Good, Joshua Aronson, and Michael Inzlicht. Improving adolescents' standardized test performance: An intervention to reduce the effects of stereotype threat. *Journal of Applied Developmental Psychology*, 24:645–662, 2003.

[45] Floraine Grabler, Maneesh Agrawala, Wilmot Li, Mira Dontcheva, and Takeo Igarashi. Generating photo manipulation tutorials by demonstration. In *ACM SIGGRAPH*, pages 66:1–66:9, 2009.

[46] Elizabeth A. Gunderson, Sarah J. Gripshover, Carissa Romero, and Carol S. Dweck. Parent Praise to 1- to 3-Year-Olds Predicts Children's Motivational Frameworks 5 Years Later. *Child Development*, 84(5):1526–1541, 2013.

[47] John D. Hansen and Justin Reich. Socioeconomic status and mooc enrollment: Enriching demographic information with external datasets. In *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge*, LAK '15, pages 59–63, New York, NY, USA, 2015. ACM.

[48] Michael E. Hardy. Use and evaluation of the aleks interactive tutoring system. *J. Comput. Sci. Coll.*, 19(4):342–347, April 2004.

[49] Kyle J. Harms, Dennis Cosgrove, Shannon Gray, and Caitlin Kelleher. Automatically generating tutorials to enable middle school children to learn programming indepen-

dently. In *International Conference on Interaction Design and Children*, IDC '13, pages 11–19, 2013.

[50] Erik Harpstead, Brad A. Myers, and Vincent Aleven. In search of learning: facilitating data analysis in educational games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 79–88, New York, NY, USA, 2013. ACM.

[51] Neil T. Heffernan and Cristina Lindquist Heffernan. The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching. *International Journal of Artificial Intelligence in Education*, 24(4):470–497, 2014.

[52] Stephen J. Hegedus and William R. Penuel. Studying new forms of participation and identity in mathematics classrooms with integrated communication and representational infrastructures. *Educational Studies in Mathematics*, 68(2):pp. 171–183, 2008.

[53] Gail D. Heyman and Carol S. Dweck. Children's Thinking about Traits: Implications for Judgments of the Self and Others. *Child Development*, 64(2):391–403, 1998.

[54] J. J. Higgins and S. Tashtoush. An aligned rank transform test for interaction. *Nonlinear World*, 1(2):201–211, 1994.

[55] Andrew Dean Ho, Justin Reich, Sergiy O. Nesterko, Daniel Thomas Seaton, Tommy Mullaney, Jim Waldo, and Isaac Chuang. HarvardX and MITx: The First Year of Open Online Courses, Fall 2012-Summer 2013. White paper, Harvard and MIT, 2014.

[56] MD Johnson, G Cierniak, C Hansen, S Bull, B Wasson, C Biel, and K Debus. Teacher approaches to adopting a competency based open learner model. In *International Conference on Computers in Education. APSCE*, 2013.

[57] Kaiser Family Foundation. Key Facts: Children and Video Games, 2002. http://kaiserfamilyfoundation. files.wordpress.com/2013/04/5959.pdf.

[58] Melissa L. Kamins and Carol S. Dweck. Person versus process praise and criticism: Implications for contingent self-worth and coping. *Developmental Psychology*, 35(3):835–847, 1999.

[59] Karl M. Kapp. *The gamification of learning and instruction: game-based methods and strategies for training and education*. San Francisco, CA: Pfeiffer, 2012.

[60] Judy Kay and Susan Bull. *Artificial Intelligence in Education: 17th International Conference, AIED 2015, Madrid, Spain, June 22-26, 2015. Proceedings*, chapter New Opportunities with Open Learner Models and Visual Learning Analytics, pages 666–669. Springer International Publishing, Cham, 2015.

[61] Judy Kay and Bob Kummerfeld. Lifelong learner modeling. *Adaptive Technologies for Training and Education*, pages 140–164, 2012.

[62] Caitlin Kelleher and Randy Pausch. Stencils-based tutorials: Design and evaluation. In *CHI*, pages 541–550, 2005.

[63] Kim Kelly, Neil Heffernan, Cristina Heffernan, Susan Goldman, James Pellegrino, and Deena Soffer Goldstein. Estimating the effect of web-based homework. In H.Chad Lane, Kalina Yacef, Jack Mostow, and Philip Pavlik, editors, *Artificial Intelligence in Education*, volume 7926 of *Lecture Notes in Computer Science*, pages 824–827. Springer Berlin Heidelberg, 2013.

[64] Kibum Kim, Deborah Tatar, and Steve Harrison. Sharing visual context to facilitate late overhearer's understanding of the handheld-based learning activity. In *Proceedings of the 8th Iternational Conference on Computer Supported Collaborative Learning*, CSCL'07, pages 367–369. International Society of the Learning Sciences, 2007.

[65] K. R. Koedinger and A. T. Corbett. Cognitive tutors: Technology bringing learning science to the classroom. In K. Sawyer, editor, *The Cambridge Handbook of the Learning Sciences*, pages 61–78. Cambridge University Press, 2005.

[66] Kenneth R. Koedinger, Vincent Aleven, Neil Heffernan, Bruce McLaren, and Matthew Hockenberry. Opening the door to non-programmers: Authoring intelligent tutor behavior by demonstration. In JamesC. Lester, RosaMaria Vicari, and Fbio Paraguau, editors, *Intelligent Tutoring Systems*, volume 3220 of *Lecture Notes in Computer Science*, pages 162–174. Springer Berlin Heidelberg, 2004.

[67] Kenneth R. Koedinger and John R. Anderson. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

[68] Kenneth R. Koedinger, John R. Anderson, William H. Hadley, and Mary A. Mark. Intelligent tutoring goes to school in the big city. *International Journal of Artificial Intelligence in Education*, 8:30–43, 1997.

[69] Kenneth R. Koedinger, Emma Brunskill, Ryan Shaun Joazeiro de Baker, Elizabeth A. McLaughlin, and John C. Stamper. New potentials for data-driven intelligent tutoring system development and optimization. *AI Magazine*, 34(3):27–41, 2013.

[70] Kenneth R. Koedinger and Neil Heffernan. Toward a rapid development environment for cognitive tutors. In *in Proceedigns of the International Conference on Artificial Intelligence in Education*, pages 455–457. IOS Press, 2003.

[71] Kenneth R. Koedinger, Elizabeth A. Mclaughlin, and Neil T. Heffernan. A quasi-experimental evaluation of an on-line formative assessment and tutoring system. *Journal of Educational Computing Research*, 43(4):489–510, 2010.

[72] Jeremy Lee, Kathleen Luchini, Benjamin Michael, Cathie Norris, and Elliot Soloway. More than just fun and games: assessing the value of educational video games in the classroom. In *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, pages 1375–1378, New York, NY, USA, 2004. ACM.

[73] Wei Li, Yuanlin Zhang, and George Fitzmaurice. Tutorialplan: Automated tutorial

generation from cad drawings. In Francesca Rossi, editor, *IJCAI*, pages 2020–2027. IJCAI/AAAI, 2013.

[74] Conor Linehan, Ben Kirman, Shaun Lawson, and Gail Chan. Practical, appropriate, empirically-validated guidelines for designing educational games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1979–1988, New York, NY, USA, 2011. ACM.

[75] Yun-En Liu, Christy Ballweber, Eleanor O'Rourke, Eric Butler, Phonraphee Thummaphan, and Zoran Popović. Large-scale educational campaigns. *ACM Trans. Comput.-Hum. Interact.*, 22(2):8:1–8:24, March 2015.

[76] Yun-En Liu, Travis Mandel, Emma Brunskill, and Zoran Popović. Towards automatic experimentation of educational knowledge. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3349–3358, New York, NY, USA, 2014. ACM.

[77] Derek Lomas, Kishan Patel, Jodi L. Forlizzi, and Kenneth R. Koedinger. Optimizing challenge in an educational game using large-scale design experiments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 89–98, New York, NY, USA, 2013. ACM.

[78] Yanjin Long and Vincent Aleven. *Intelligent Tutoring Systems: 12th International Conference, ITS 2014, Honolulu, HI, USA, June 5-9, 2014. Proceedings*, chapter Gamification of Joint Student/System Control over Problem Selection in a Linear Equation Tutor, pages 378–387. Springer International Publishing, Cham, 2014.

[79] Cora J. M. Maas and Joop J. Hox. Sufficient sample sizes for multilevel modeling. *Methodology: European Journal of Research Methods for the Behavioral and Social Sciences*, 1(3):86 – 92, 2005.

[80] Brent Martin and Antonija Mitrovic. Automatic problem generation in constraint-based tutors. In *Intelligent Tutoring Systems*, volume 2363 of *Lecture Notes in Computer Science*, pages 388–398. Springer Berlin Heidelberg, 2002.

[81] Brent I. Martin. *Intelligent Tutoring Systems: The Practical Implementation of Constraint-Based Modelling*. PhD thesis, University of Canterbury, 2001.

[82] Taylor Martin, Carmen Petrick Smith, Erik Andersen, Yun-En Liu, and Zoran Popović. Refraction time: Making split decisions in an online fraction game. In *American Educational Research Association Annual Meeting*, AERA, 2012.

[83] JUMP Math. Jump math, 2015.

[84] Merrilea J. Mayo. Video Games: A Route to Large-Scale STEM Education? *Science*, 323:79–82, 2009.

[85] Anne McKeough, Judy Lee Lupart, and Anthony Marini. *Teaching for Transfer: Fostering Generalization in Learning*. Routledge, New York, New York, 1995.

[86] Bruce M. McLaren, Sung-Joo Lim, and Kenneth R. Koedinger. When and how often should worked examples be given to students? new results and a summary of the current state of research. In B. C. Love, K. McRae, and V. M. Sloutsky, editors, *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pages 2176–2181, Austin, TX, 2008. Cognitive Science Society.

[87] Erica Melis. Design of erroneous examples for activemath. In *Proceedings of the 2005 Conference on Artificial Intelligence in Education: Supporting Learning Through Intelligent and Socially Informed Technology*, pages 451–458, Amsterdam, The Netherlands, The Netherlands, 2005. IOS Press.

[88] Erica Melis, Georgi Goguadze, and Universitt Des Saarl. Towards adaptive generation of faded examples. In *International Conference on Intelligent Tutoring Systems , LNCS*, pages 762–771. Springer- Verlag, 2004.

[89] Michael Mendicino, Leena Razzaq, and Neil T. Heffernan. A quasi-experimental evaluation of an on-line formative assessment and tutoring system. *Journal of Research on Technology in Education*, 41(3):331–359, 2009.

[90] Antonija Mitrovic. Fifteen years of constraint-based tutors: What we have achieved and where we are going. *User Modeling and User-Adapted Interaction*, 22(1-2):39–72, April 2012.

[91] Antonija Mitrovic, Kenneth R. Koedinger, and Brent Martin. A comparative analysis of cognitive tutoring and constraint-based modeling. In Peter Brusilovsky, Albert Corbett, and Fiorella de Rosis, editors, *User Modeling 2003*, volume 2702 of *Lecture Notes in Computer Science*, pages 313–322. Springer Berlin Heidelberg, 2003.

[92] Claudia M. Mueller and Carol S. Dweck. Praise for Intelligence Can Undermine Children's Motivation and Performance. *Journal of Personality and Social Psychology*, 75(1):33–52, 1998.

[93] Robert Murphy, Lawrence Gallagher, Andrew Krumm, Jessica Mislevy, and Amy Hafter. Research on the Use of Khan Academy in Schools. White paper, SRI International, 2014.

[94] Beverley Murray. Increased Math Achievement in Elementary Students Participating in JUMP Maths 2013-14 National Book Fund Program. Technical report, 2015.

[95] Ann A. O'Connell and D. Betsy McCoach. *Multilevel Modeling of Educational Data*. Information Age Publishing Inc., Charlotte, North Carolina, 2008.

[96] J. Ocumpaugh, R.S. Baker, and M.M.T. Rodrigo. Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) 2.0 Technical and Training Manual. Technical report, New York, NY: Teachers College, Columbia University. Manila, Philippines: Ateneo Laboratory for the Learning Sciences, 2015.

[97] Amy Ogan, Erin Walker, Ryan S.J.D. Baker, Genaro Rebolledo Mendez, Maynor Jimenez Castro, Tania Laurentino, and Adriana de Carvalho. Collaboration in cognitive tutor use in latin america: Field study and design recommendations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 1381–1390, New York, NY, USA, 2012. ACM.

[98] Stellan Ohlsson. Constraint-based student modeling. In JimE. Greer and GordonI. McCalla, editors, *Student Modelling: The Key to Individualized Knowledge-Based Instruction*, volume 125 of *NATO ASI Series*, pages 167–189. Springer Berlin Heidelberg, 1994.

[99] Stellan Ohlsson. Learning from performance errors. *Psychological Review*, 103(2):214–262, 1996.

[100] JenniferK. Olsen, DanielM. Belenky, Vincent Aleven, Nikol Rummel, Jonathan Sewall, and Michael Ringenberg. Authoring tools for collaborative intelligent tutoring system environments. In *Intelligent Tutoring Systems*, pages 523–528, 2014.

[101] Harold F. O'Neil, Richard Wainess, and Eva L. Baker. Classification of learning outcomes: evidence from the computer games literature. *The Curriculum Journal*, 16(4):455–474, 2005.

[102] Eleanor O'Rourke, Erik Andersen, Sumit Gulwani, and Zoran Popović. A framework for automatically generating interactive instructional scaffolding. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1545–1554, New York, NY, USA, 2015. ACM.

[103] Eleanor O'Rourke, Christy Ballweber, and Zoran Popović. Hint systems may negatively impact performance in educational games. In *Proceedings of Learning @ Scale*, L@S '14, pages 51–60, 2014.

[104] Eleanor O'Rourke, Eric Butler, Yun-En Liu, Christy Ballweber, and Zoran Popović. The effects of age on player behavior in educational games. In *Proceedings of the International Conference on the Foundations of Digital Games*, FDG '13, New York, NY, USA, 2013. ACM.

[105] Eleanor O'Rourke, Eric Butler, Armando Diaz Tolentino, and Zoran Popović. Automatic generation of problems and explanations for an intelligent algebra tutor. In *submission to a human-computer interaction conference*, 2016.

[106] Eleanor O'Rourke, Yvonne Chen, Christy Ballweber, and Zoran Popović. Personalized learning and its behavioral impact on the classroom ecosystem. In *submission to a human-computer interaction conference*, 2016.

[107] Eleanor O'Rourke, Yvonne Chen, Kyla Haimovitz, Carol S. Dweck, and Zoran Popović. Demographic differences in a growth mindset incentive structure for educational games. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 331–334, New York, NY, USA, 2015. ACM.

[108] Eleanor O'Rourke, Kyla Haimovitz, Christy Ballweber, Carol Dweck, and Zoran Popović. Brain points: A growth mindset incentive structure boosts persistence in an educational game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3339–3348, New York, NY, USA, 2014. ACM.

[109] Eleanor O'Rourke, Erin Peach, Carol S. Dweck, and Zoran Popović. Brain points: A deeper look at a growth mindset incentive structure for an educational game. In *Proceedings of the Third (2016) ACM Conference on Learning @ Scale*, L@S '16, pages 41–50, New York, NY, USA, 2016. ACM.

[110] Zachary A. Pardos, Ryan S. J. D. Baker, Maria O. C. Z. San Pedro, Sujith M. Gowda, and Supreeth M. Gowda. Affective states and state tests: Investigating how affect throughout the school year predicts end of year learning outcomes. In *Proceedings of*

the Third International Conference on Learning Analytics and Knowledge, LAK '13, pages 117–124, 2013.

[111] David Paunesku, Gregory M. Walton, Carissa Romero, Eric N. Smith, David S. Yeager, and Carol S. Dweck. Mind-set interventions are a scalable treatment for academic underachievement. *Psychological Science*, 26(6):784–793, 2015.

[112] Paul R. Pintrich and Elisabeth V. De Groot. Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology*, 82(1):33– –40, 1990.

[113] Oleksandr Polozov, Eleanor O'Rourke, Adam Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. Personalized mathematical word problem generation. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, IJCAI '15, 2015.

[114] Suporn Pongnumkul, Mira Dontcheva, Wilmot Li, Jue Wang, Lubomir Bourdev, Shai Avidan, and Michael F. Cohen. Pause-and-play: automatically linking screencast video tutorials with applications. In *UIST*, pages 135–144, 2011.

[115] Vidya Ramesh, Charlie Hsu, Maneesh Agrawala, and Björn Hartmann. Showmehow: Translating user interface instructions between applications. In *UIST*, pages 127–134, 2011.

[116] C. M. Reigeluth and F. S. Stein. The elaboration theory of instruction. In *Instructional Design Theories and Models: An Overview of their Current States*, Hillsdale, NJ, 1983. Lawrence Erlbaum.

[117] Alexander Renkl. Learning from worked-out examples: A study on individual differences. *Cognitive Science*, 21(1):1 – 29, 1997.

[118] Alexander Renkl, Robert K. Atkinson, and Cornelia S. Große. How fading worked solution steps works a cognitive load perspective. *Instructional Science*, 32(1-2):59–82, 2004.

[119] Alexander Renkl, Robert K. Atkinson, Uwe H. Maier, and Richard Staley. From example study to problem solving: Smooth transitions help learning. *Journal of Experimental Education*, 70(4):293–315, 2002.

[120] Katrina E. Ricci, Eduardo Salas, and Janis A. Cannon-Bowers. Do computer-based games facilitate knowledge acquisition and retention? *Military Psychology*, 8(4):295–307, 1996.

[121] Jeremy Roschelle, Ken Rafanan, Gucci Estrella, Miguel Nussbaum, and Susana Claro. From handheld collaborative tool to effective classroom module: Embedding {CSCL} in a broader design framework. *Computers & Education*, 55(3):1018 – 1026, 2010.

[122] Dorsa Sadigh, Sanjit A. Seshia, and Mona Gupta. Automating exercise generation: A step towards meeting the mooc challenge for embedded systems. In *Proceedings of the Workshop on Embedded and Cyber-Physical Systems Education*, WESE '12, pages 2:1–2:8, New York, NY, USA, 2013. ACM.

[123] Ron J. C. M. Salden, Vincent A. W. M. M. Aleven, Alexander Renkl, and Rolf Schwonke. Worked examples and tutored problem solving: Redundant or synergistic forms of support? *Topics in Cognitive Science*, 1(1):203–213, 2008.

[124] Jesse Schell. *The Art of Game Design: A Book of Lenses*. Morgan Kaufmann Publishers, Burlington, MA, 2008.

[125] J.W. Schofield, D. Evans-Rhodes, and B.R. Huber. Artificial intelligence in the classroom: The impact of a computer-based tutor on teachers and students. *Social Science Computer Review*, 8:24–41, 1990.

[126] Murray Shanahan. The event calculus explained. In *Artificial intelligence today*, pages 409–430. Springer, 1999.

[127] L. Shepard. The role of assessment in a learning culture. *Educational Researcher*, 29(7):4–14, 2000.

[128] Rohit Singh, Sumit Gulwani, and Sriram Rajamani. Automatically generating algebra problems. In *AAAI*, 2012.

[129] E. A. Skinner and M. J. Belmont. Motivation in the classroom: Reciprocal effects of teacher behavior and student engagement across the school year. *Journal of Educational Psychology*, 85(4):571– –581, 1993.

[130] Adam Smith, Eric Butler, and Zoran Popović. Quantifying over play: Constraining undesirable solutions in puzzle design. In *Procedings of the 8th International Conference on the Foundations of Digital Games*, 2013.

[131] Adam M. Smith, Erik Andersen, Michael Mateas, and Zoran Popović. A case study of expressively constrainable level design automation tools for a puzzle game. In *FDG '12: Proceedings of the Seventh International Conference on the Foundations of Digital Games*, New York, NY, USA, 2012. ACM.

[132] Robert Anthony Sottilare. Considerations in the development of an ontology for a generalized intelligent framework for tutoring. In *International Defense & Homeland Security Simulation Workshop in Proceedings of the I3M Conference*, pages 19–25, 2012.

[133] J. Stamper, T. Barnes, L. Lehmann, and M. Croy. The hint factory: Automatic generation of contextualized help for existing computer aided instruction. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems Young Researchers Track*, pages 71–78, 2008.

[134] John Stamper, Tiffany Barnes, and Marvin Croy. Enhancing the automatic generation of hints with expert seeding. *Int. J. Artif. Intell. Ed.*, 21(1-2):153–167, January 2011.

[135] Patrick Stark and Amber M. Noel. Trends in High School Dropout and Completion Rates in the United States: 19722012. (NCES 2015-015). U.S. Department of Education. White paper, Washington, DC: National Center for Education Statistics, 2015.

[136] John Sweller and Graham A. Cooper. The use of worked examples as a substitute for problem solving in learning algebra. *Cognition and Instruction*, 2(1):59–89, 1985.

[137] Dimitra Tsovaltzi, Bruce M. McLaren, Erica Melis, and Ann-Kristin Meyer. Erroneous examples: Effects on learning fractions in a web-based setting. *Int. J. Technol. Enhanc. Learn.*, 4(3/4):191–230, January 2012.

[138] Tamara van Gog, Liesbeth Kester, and Fred Paas. Effects of worked examples, example-problem, and problem-example pairs on novices' learning. *Contemporary Educational Psychology*, 36(3):212–218, 2011.

[139] Tamara van Gog, Fred Paas, and Jeroen J.G. van Merriënboer. Process-oriented worked examples: Improving transfer performance through enhanced understanding. *Instructional Science*, 32(1-2):83–98, 2004.

[140] Kurt VanLehn. The behavior of tutoring systems. *International Journal of Artificial Intelligence in Education*, 16:227–265, 2006.

[141] Kurt VanLehn, Collin Lynch, Kay Schulze, Joel A. Shapiro, Robert Shelby, Linwood Taylor, Don Treacy, Anders Weinstein, and Mary Wintersgill. The Andes physics tutoring system: five years of evaluations. In *International Conference on Artificial Intelligence in Education*, pages 678–685, 2005.

[142] Katrien Verbert, Sten Govaerts, Erik Duval, Jose Luis Santos, Frans Assche, Gonzalo Parra, and Joris Klerkx. Learning dashboards: An overview and future research opportunities. *Personal Ubiquitous Comput.*, 18(6):1499–1514, August 2014.

[143] Jonathan D Wexler. A self-directing teaching program that generates simple arithmetic problems. *Computer Sciences Technical Report*, 19, 1968.

[144] D. Wiliam. Keeping learning on track: Formative assessment and the regulation of learning. In Jr. F. K. Lester, editor, *Second handbook of mathematics teaching and learning*, pages 109–129. Information Age Publishing, Greenwich, CT, 2007.

[145] Jacob O. Wobbrock, Leah Findlater, Darren Gergle, and James J. Higgins. The aligned rank transform for nonparametric factorial analyses using only anova procedures. In *CHI*, pages 143–146, 2011.

[146] D. Wood, J. Bruner, and G. Ross. The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry*, 17:89–100, 1976.

[147] Chen Zhenghao, Brandon Alcorn, Gayle Christensen, Nicholas Eriksson, Daphne Koller, and Ezekiel J. Emanuel. Whos Benefiting from MOOCs, and Why. White paper, Harvard Business Review, 2015.