Evaluating Proof Blocks Problems as Exam Questions

Seth Poulsen, Mahesh Viswanathan, Geoffrey L. Herman, and Matthew West
University of Illinois at Urbana-Champaign

Proof Blocks is a novel software tool which enables students to write mathematical proofs by dragging and dropping prewritten lines into the correct order, rather than writing a proof completely from scratch. We used Proof Blocks problems as exam questions for a discrete mathematics course with hundreds of students, allowing us to collect thousands of student responses to Proof Blocks problems. Using this data, we provide statistical evidence that Proof Blocks are easier than written proofs, which are typically very difficult. We also show that Proof Blocks problems provide about as much information about student knowledge as written proofs. Survey results show that students believe that the Proof Blocks user interface is easy to use, and that the questions accurately represent their ability to write proofs.

1.0 INTRODUCTION

Understanding and writing mathematical proofs is one of the critical yet difficult skills that students must learn as a part of the discrete mathematics curriculum. Proofs and proof techniques are included by the ACM curricular guidelines as a core knowledge area that should be understood by any student obtaining a degree in computer engineering, computer science, or software engineering [12, 16, 30]. A panel of 21 experts using a Delphi process agreed that 6 of the 11 most difficult topics in a typical discrete mathematics course are related to proofs and logic [9].

There are many aspects of writing mathematical proofs that are difficult. Many students fail to produce the basic building blocks that proofs have, such as properly declaring variables or referencing theorems [23]. Students get stuck working through the details of algebraic manipulations. They have a tendency to commit certain logical fallacies such as confusing a proposition with its converse [23, 27]. Studies have shown that even when students have all the prerequisite content knowledge to write a mathematical proof, they still struggle to construct one [32]. Thus, there is a gap that needs to be filled between students having the content knowledge to write a proof and the aptitude to actually construct one.

Vygotsky's theory of psychological development posits that between the tasks which a person can and cannot do, there is a so-called zone of proximal development: a set of tasks which a person cannot perform unaided, but which they can perform when given help and support, called scaffolding [31, 36]. Computer science instructors and researchers have used various approaches to scaffolding students learning to write code for the first time. Block based programming languages such as Scratch and Blockly [8, 15] scaffold students by providing them with building blocks from which to assemble their programs and guarding against the struggles of syntax errors. Research has shown that using block based languages can accelerate the student learning process when first learning to program [34]. Parson's problems are a kind of homework and exam question where students are asked to assemble prewritten

The original version of this paper was published in the *Proceedings of the 17th ACM Conference on International Computing Education Research* (Aug. 2021), 157-168.
Reprinted with permission.
This work was recognized with an Honorable Mention at ICER '21.

lines of code into a correct program [17]. Researchers have shown Parson's problems to be useful both as test questions [4] and as a learning tool for helping to accelerate the learning process for beginners learning to write code [7].

Following from the success of Parson's problem and similar approaches to teach programming, we propose Proof Blocks. Proof Blocks allows students to construct mathematical proofs by dragging and dropping prewritten proof lines into the correct order, rather than having to write the entire proof from scratch. Figure 1 shows an example of a Proof Blocks problem. Proof Blocks provides a scaffolded environment, enabling students to construct mathematical proofs without needing to worry about coming up with all of the details on their own. A Proof Blocks problem may also contain distractor lines which are not a part of any correct solution. The design of the Proof Blocks grader [20] is flexible in allowing any correct arrangement of the lines of the proof. This is enabled by the instructor specifying which lines of the proof depend on which other lines (the full dependence graph of the lines of the proof in Figure 1 can be seen in Figure 2). Students who fail to construct a correct proof on their first try can then receive automated feedback from the computer, as shown in Figure 3, before being given additional attempts at the discretion of the instructor.

Proof Blocks problems are also very promising for saving time for both students and course staff. Many computer science departments are experiencing a huge increase in enrollments. This increase in enrollments means course staff lose more time to grading, making it more difficult for them to spend the time they need helping students individually. Proof Blocks helps to alleviate this strain by providing a way to test some of students' proof skills in a way that can be automated, saving grading time and allowing course staff more time for other activities that help students such as office hours and review sessions.

The ability to receive automated feedback is also a boon to students. Due to staff time constraints, students in a discrete mathematics course may not be able to receive feedback on the correctness of proofs they write until long after they have completed them. Proof Blocks also helps with this, as it allows students to receive feedback instantly, just as they receive instant feedback from the compiler and from automated testing suites as they write code.

In using a new kind of test question with our students, we wanted to ensure that we were testing students on the correct set of skills and that we were providing them with a fair and equitable learning experience.

In this paper, we seek to answer the following three research questions:

RQ1: What statistical information about student knowledge do Proof Blocks problems provide relative to other course content?

RQ2: What is the relationship between the knowledge required to complete Proof Blocks problems and other types of problems in a discrete mathematics course?

RQ3: What are students' perceptions about the fairness, usability, and authenticity of being assessed by using Proof Blocks problems?

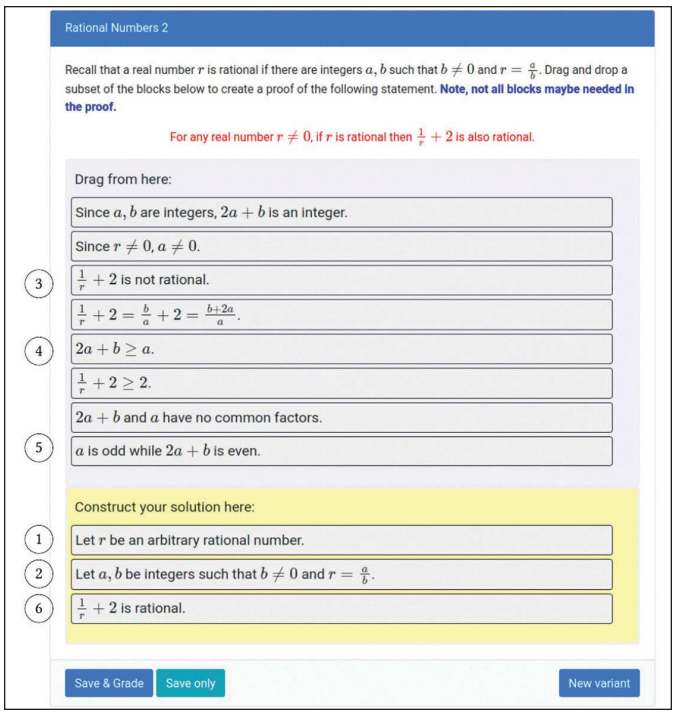

Figure 1: An example of the Proof Blocks user interface used by students. Individual lines of the proof start out shuffled in the light-blue starting zone, and students attempt to drag and drop them into the correct order in the yellow target zone. The instructor wrote the problem with 1, 2, 3, 4, 5, 6 as the intended solution, but the Proof Blocks autograder will also accept any other correct solution as determined by the dependency graph shown Figure 2. For example, both 1, 2, 5, 4, 3, 6 and 1, 2, 4, 5, 3, 6 would also be accepted as correct solutions.

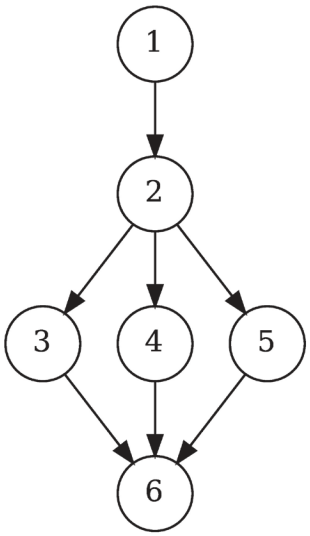

Figure 2: The dependency graph of the statements in the proof shown in Figure 1. The Proof Blocks grader will accept any topological sort of this directed acyclic graph as a correct solution. For more details of the implementation of the Proof Blocks grader, see [20].

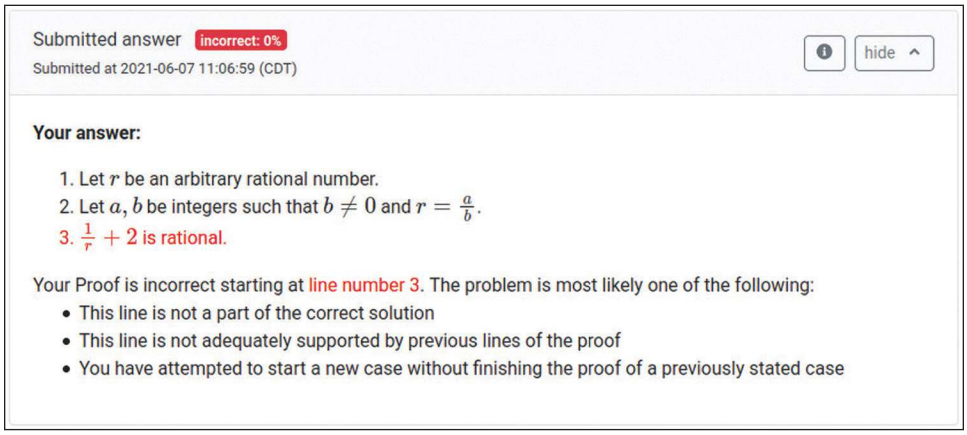

Figure 3: Example of feedback given to students working on Proof Blocks problems. To avoid giving students so much information that we are not actually testing their knowledge, they are only told at which line their proof fails, not the reason why or what the solution is. One area of future research is to investigate what kind of feedback is best for students to receive when using Proof Blocks as a tool for learning to write proofs.

2.0 RELATED WORK

Anecdotally, we have heard of instructors using scrambled proofs to assess student knowledge both in Euclidean geometry and in higher-level mathematics. In theory, instructors may have offered such questions on paper even before the advent of computers, though we can find no explicit record of this. Additionally, to our knowledge there has been no research into the merits of these questions either for learning or for assessment.

We will give a brief overview of related work including Parson's problems, research on teaching and learning proofs, and software tools for constructing mathematical proofs in an educational context.

2.1 PARSON'S PROBLEMS

The use of scrambled code problems was first documented by Parsons [17]. They have since been studied for their desirable properties both in assessment and learning [4, 5, 7]. The desirable properties of Parson's problems were a major inspiration for the creation of Proof Blocks.

Denny et al. [4] showed that Parson's problems are easier to grade than free-form code writing questions, and yet still offer rich information about student knowledge. We will show the same to be true with Proof Blocks problems in relation to free-form proof writing questions. Ericson et al. [7] showed that students learning to write code using Parson's problems learn at an accelerated rate in the early stages of learning compared to students being taught to fix code or write code from scratch.

2.2 RESEARCH ON TEACHING AND LEARNING PROOFS

There are many threads of research in seeking to illuminate students' understandings and misunderstandings about proofs [24, 26, 27]. One thread establishes that, as they learn, students go through different phases in the complexity of ways they are able to think about solving proof problems [33]. Another study demonstrated that even when students had all of the knowledge required to write a proof and were able to apply that knowledge in other types of questions, they were still unable to write a proof [32], thus highlighting the need to scaffold students through the proof-writing process.

On the other hand, there is little research on concrete educational interventions for improving the proof learning process [11, 27]. Indeed, a recent review of the literature on teaching and learning proofs concluded: "more intervention-oriented studies in the area of proof are sorely needed" [27]. Hodds et al. [11] showed that training students to engage more with proofs through self-explanation increased student comprehension of proofs in a lasting way. Proof Blocks problems similarly force deliberate engagement with proof content, as close reading is necessary to determine the correct arrangement of lines. Proof Blocks also shows promise as a tool that can provide scaffolding that students are so in need of when learning to write proofs.

2.3 EDUCATIONAL THEOREM PROVING SOFTWARE

A few other software tools have been created to enable students to create proofs in the computer in such a way that they can receive automated feedback. Some use text-based representations, while others use visual representations of proofs.

Polymorphic Blocks [13] is a novel user interface which presents propositions as colorful blocks with uniquely shaped connectors as a signifier of which types of propositions can be connected in a proof. While the user interface has been shown to engage students in learning proofs, it supports only propositional logic. The Incredible Proof Machine [3] guides students through constructing proofs as graphs. As with Polymorphic Blocks, the user interface is engaging, but the formality of the system limits the topics which can be effectively covered.

Jape [2] is a "Proof calculator," which guides students through the process of constructing formal proofs in mathematical notation with the help of the computer. While Jape can allow students to construct proofs in arbitrary logics, it requires the instructor to implement these logics in its own custom programming language before students can use them to construct proofs.

MathsTiles [1] is a block-based programming interface for constructing proofs for the Isabelle/HOL proof assistant. In theory, having an open-ended environment where students could construct arbitrarily complex proofs seems like it could be a huge advantage. However, in user studies, the authors found that students only had a chance at being successful while using MathsTiles if they were provided a small instructor-procured subset of blocks, namely, those needed to complete the problem at hand.

Ensley and Winston offer some scrambled proofs in a JavaScript applet as supplementary material to their discrete mathematics textbook [6]. However, their tools are restricted in only supporting grading by simple ordering, greatly restricting the types of proofs that students can construct using the tool. The directed acyclic graph-based grading that the Proof Blocks autograder uses enables assessing proofs which are more complex and use a greater variety of writing styles.

Most of these tools cover only small subset of the material typically covered in a discrete mathematics course, and those that are more flexible require learning complex theorem prover languages. In contrast, Proof Blocks enables instructors to easily provide students with proof questions on any topic. To our knowledge, no research has been published on using any of the above tools as part of student assessments.

3.0 COURSE CONTEXT

We evaluated Proof Blocks problems by using them for exams in a discrete mathematics course at the University of Illinois at Urbana-Champaign. At the University of Illinois, the discrete mathematics course in the computer science department is taught every semester (including during the summer) and is taken by hundreds of students each semester, across multiple sections. Most students are freshmen and take the course as part of their computer science major, computer science minor, or computer engineering major. The listed prerequisites for the course are introductory programming and introductory calculus. The course is designed to prepare students for the theory track in the computer science department and usually covers logic, proofs, functions, cardinality, graphs and trees, induction, recursion, number theory, probability, basic algorithm analysis, and sometimes additional topics as time permits. Though taught in the computer science department, it is solely a theory class, with no programming assignments.

In Fall 2020, the course was taught completely online due to the COVID-19 pandemic. The course was split into 3 sections, each with a unique instructor, with a total of 404 students. Each week's content consisted of a video lecture and small group assignments completed over video conferencing with teaching assistant guidance and support. Students were then assigned homework to provide additional practice with the material. At the beginning of each week, students took a short exam on the material covered the previous week. Some weeks, the students were also given a practice exam to assist in studying. If a student had to miss an exam for some reason, they were allowed to make up the exam the following week.

In lieu of a final exam, students were given the opportunity to retake any three of the exams. A full listing of the topics on each exam, as well as the number of each type of question on each exam, can be seen in Table 1. While the distribution of question types among tests may not be ideal for measuring the qualities of types of questions, it gives the study a large degree of *ecological* validity. That is, in the discrete mathematics class examined in this study, Proof Blocks problems were not used at an artificially inflated rate, but rather were used as one would want to use any type of test question—intermixed with other types of test questions, at times when they were appropriate.

Table 1: The breakdown of question types on each exam. Proof Blocks problems were used on exams throughout the semester as and when the instructors felt that they would be useful. They were not used at an artificially inflated rate for the purposes of this study.

Exam Number	Topics	Proof	Proof Blocks	Other
1	Logic and Proofs	1	2	3
2	Sets, functions, and Relations	2		7
3	Cardinality	1	2	3
4	Directed Graphs	1	1	4
5	Undirected Graphs and Trees	1	1	4
6	Induction	2		3
7	Recursive sets and Structural Induction	1		5
8	Number Theory	1	1	4
9	Probability and Counting			6
10	Series Sums and Solving Recurrences			5
11	Algorithm Analysis and Big O		2	4

Students took their exams using PrairieLearn, an open-source online homework and exam platform [35]. Especially for a course of this size, Proof Blocks' fully automated grading was a big advantage in saving course staff time which could be reallocated in other ways. In total, students were given 9 Proof Blocks problems on exams and 3 on practice exams. Students received immediate correctness feedback on each Proof Blocks problem on their exams and were given up to 4 or 5 attempts at each question, with a decreasing number of points awarded depending on the number of attempts used. Students were typically given 3 attempts for multiple choice questions, and 4 or 5 attempts for fill-in-the-blank computation questions, also with a decreasing number of points awarded depending on which attempt they successfully answered correctly. In all cases, the students were only awarded full points if they completed the question correctly on the first attempt. Students wrote free response proof questions in text entry box which supported markdown and LaTeX, but were told that using plain text (for example, spelling out "and" instead of using $\wedge$ and spelling out "intersection" instead of using $\cap$) was acceptable as we did not expect them to learn LaTeX for the course.

In order to combat student cheating efforts, almost all questions had multiple variants. Many questions had three or four static variants, one for each of the three course sections on the primary test day, and the fourth being used the following week for the make up exams. Other questions had variants generated uniquely for each student based on a random number generator. Questions randomized values such as elements of a set, edges in a graph, and other question properties which could be easily randomized and then computer graded.

All multiple choice, fill in the blank computation, and Proof Blocks problems were automatically graded by PrairieLearn as soon as the student completed them, and they were immediately shown their grade on these questions. An overview of the workings of the Proof Blocks autograder can be seen in [20]. Written proof questions were then hand graded by one of the course's 8 teaching assistants, based on rubrics created collaboratively between the instructors and teaching assistants. The rubrics were different for each exam, but generally students were awarded points for following the correct proof structure, properly declaring variables, knowing and correctly applying definitions, and logical flow from one step to another. Points were not awarded for style. The first author of this paper was a teaching assistant for the course, and the second was one of three faculty instructors for the course, with the other authors having no affiliation with the course.

4.0 DATA HANDLING

All submissions to exam questions were automatically saved to a database by PrairieLearn. With approval from our university's Institutional Review Board, the course data was accessed by an instructional technology specialist employed by the engineering college, and then fully anonymized before being delivered to the research team for analysis. All research team members handling the data were trained in proper student privacy and human subjects research protocols.

4.1 DATA PREPARATION

For our analysis, we treat all variants of a question as the same question. Though there are small differences in difficulty between question variants, we concluded that these small differences were not relevant to the research questions we are addressing with this study.

Because the final exam involved retaking exams which may have contained questions overlapping with questions that students had already seen, and we are focusing only on students' first interactions with a given question, we exclude the final exam from our analysis, focusing only on the 11 exams given to the students throughout the semester. Two questions on Exam 2 had user interface bugs in them, causing the course staff to award everyone in the course full points on those questions for fairness, so they were also excluded from the analysis. The Proof Blocks question given on Exam 8 was nearly identical to a question given on a practice exam, so we omit it from the analysis to avoid the analysis being skewed due to students knowing the answer in advance.

With the anonymity restrictions placed by our Institutional Review Board, we are unable to know which students in our data set formally dropped the course before the end, so for our analysis we only kept students who attempted at least 10 of the 11 exams (325 of 404 students). In some cases, different questions were used between primary exams and make up exams, so we do not have a response from every student for every question, even for students who took every exam. Our final data set consisted of 325 students over 62 questions. A complete data set would have been 20,150 answers, but we had 569 missing data points, giving us a total of 19,581 student answers.

To keep all questions on the same scale, remove effects of un-validated grading rubrics, and to remove the effects of guessing on additional submissions after feedback, we re-graded all questions on a dichotomous scale (1 for fully correct, 0 for not fully correct) and graded only the first submission. This decision aligns our data more closely with the two-parameter logistic model of item response theory (See Section 5.1).

5.0 METHODS

We use the two-parameter logistic model (2PL) from item response theory [14] to answer **RQ1** and correlation analysis to answer **RQ2**. To answer **RQ3**, we administered a survey.

5.1 PSYCHOMETRICS

To answer **RQ1**, we want to understand what level of student knowledge Proof Blocks questions assess, and how accurately it assesses that knowledge. 2PL has been used widely in psychometrics and has been used within computer science education mostly for validation of concept inventories [10, 18, 19, 37]. 2PL is a good fit for our needs because it provides a way to model the probability of each student answering each question correctly as a function of a question's difficulty and discrimination.

The *difficulty* is how hard it is to answer a question correctly, and the *discrimination* is how well a question differentiates between students of lower and higher ability levels. In the case of difficulty, we want to explore whether Proof Blocks problems have lower difficulty than written proofs, giving evidence that they may provide scaffolding. For discrimination, higher is always better in the sense that if a question's discrimination is higher, it will provide more information about student knowledge. We would like to explore if Proof Blocks problems have comparable discrimination to written proof problems.

We used the R programming language and the package ltm to clean the data and fit item response theory models [21, 22]. In 2PL, we assume that the probability of student n correctly responding to item i can be modeled as a function of the student's ability, θ_n, the discrimination of the item, a_i, and the difficulty of the item, b_i, as follows:

$$p_i(\theta_n) = \frac{1}{1 + e^{-a_i(\theta_n - b_i)}}.$$

The distribution of student ability parameters θ_n is normalized to a mean of 0 and standard deviation of 1. In this case, because the students were learning across the course of the semester in between these test questions, the difficulty measurement of the questions is relative to the student knowledge at the time they took that particular exam, rather than absolute.

After fitting the 2PL model, we test two null hypotheses:

1. The distribution of difficulties of Proof Blocks problems is the same as for written proof problems. (We desire for Proof Blocks to be easier.)
2. The distribution of discriminations of Proof Blocks problems is the same as for written proof problems. (We desire for Proof Blocks to be comparably discriminatory.)

After using a Shapiro-Wilk test to confirm the normality of these distributions, we use a t-test to test the hypotheses.

In order to fit the dichotomous response requirement of 2PL, we converted all problems to binary responses: 1 for full points and 0 for anything less than full points (See Section 4.1). To ensure robustness of our results, we also fit our data to a graded response model, a type of polytomous item response theory model that accounts for assignment of partial credit. This model supported our conclusions just as well as the 2PL model, so we present the simpler model for ease of presentation. As a further robustness check, we also used a standard classical test theory model, which again supported the same conclusions.

5.1.1 Item Response Functions. Inserting the difficulty and discrimination parameters for each test item into Equation 1 gives the *item response functions*, which help us visualize the difficulty and discrimination of test items, and the probability that a student with a given ability level will answer the question correctly. The difficulty of the item determines the ability level at which a student will have a 50% probability of getting the question correct. For example, in Figure 4 the solid line describes an item with difficulty 0, meaning that if we choose a random student with mean ability level, there is a 50% chance that student would have answered that item correctly. The dotted item is an easier item with a difficulty of -0.5, meaning that students who are half a standard deviation below the mean in ability level answer that item correctly at a rate of 50%, and students with mean ability level answer that item correctly at a rate greater than 50%. A good assessment will have questions with a variety of difficulty levels to assess student knowledge at all relevant levels of ability.

The discrimination of an item manifests in the item response function as the slope, with a higher positive discrimination leading to a more strongly positive slope. For example, the dashed line in Figure 4 denotes an item that has the same difficulty as the solid-line item, but with a higher discrimination, so that the probability that a student gets the questions correct rises more quickly for students above mean-ability level, and decreases more quickly for students below mean-ability level. Questions with high discrimination allow assessments to measure student knowledge with high accuracy and less error, so it is always desirable for items to have high discrimination.

5.1.2 Item Information Functions. The *item information function* for an item is the derivative of the item response function for that item. It shows how much information that item gives about students taking the test at each level of ability. Figure 4 shows example item information curves for the same items as it shows item response functions. The solid item collects more information about higher performing students than the dotted item, due to having higher difficulty. The high discrimination of the dashed item allows it to provide much more information across a range of ability levels than either of the other two items.

Summing together multiple item information functions gives the combined information that can be gained about a given student from a set of items. To better understand the quality of information that Proof Blocks problems provide about students in a discrete mathematics course, we calculate the average item information curve for each category by summing the information curves for all the items in each category (i.e., Proofs, Proof Blocks, Other), and then dividing by the total number

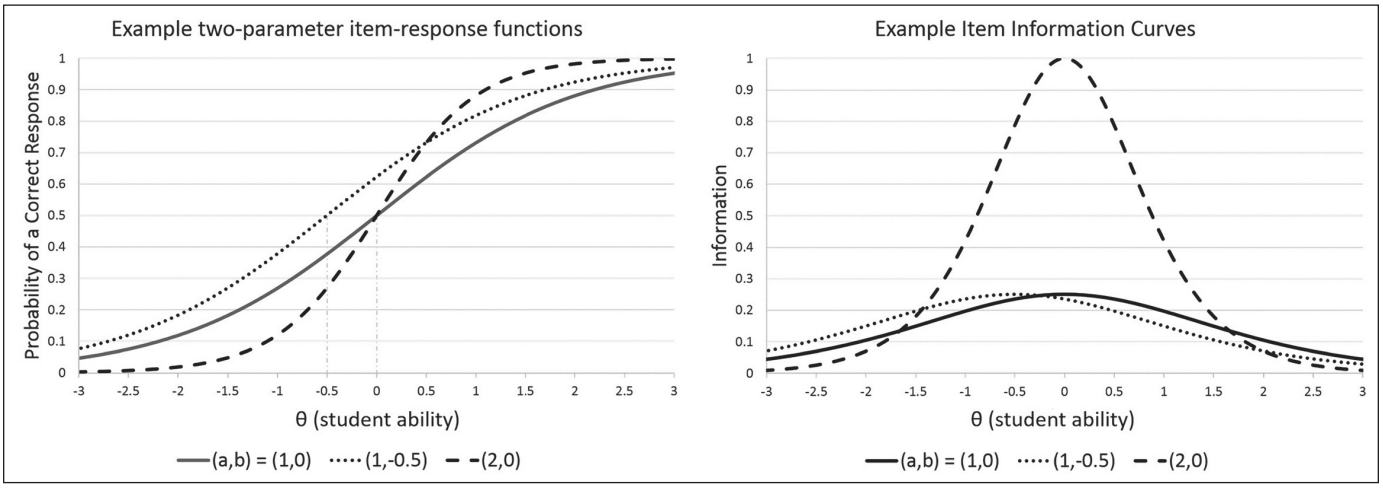

Figure 4: Left: Three example item response functions with varying discrimination (ai) and difficulty (bi). Right: Item information curves for the same example items.

of items in that category. We compare the average amount of information each problem category provides to further explore the relative utility of Proof Blocks problems.

5.2 CORRELATION

In order to examine the overlap between the skills needed for different question types (**RQ2**), we calculated the correlation between students' average scores in each question category. By design, Proof Blocks problems are scaffolded proof problems, and so we expect that only some of the skills required to solve proof problems are also required to solve Proof Blocks problems. Thus, we expect to find a correlation between students' scores across these question types, but not a correlation so strong that it would imply the questions are assessing the exact same knowledge. This shows one of the limitations of our study: based on our current data, we can take a broad look at the closeness of the association between Proof Blocks problems and proof problems, but without further data we are not yet able to comment on which exact skills are required to answer one type of question but are not for the other.

After using a Shapiro-Wilk test and finding that the data were non-normal, we used the Spearman correlation to calculate the correlation between students' scores in the different question categories.

5.3 SURVEY

We used an anonymous survey to help us answer **RQ3**: What are students' perceptions about the fairness, usability, and authenticity of being assessed using Proof Blocks problems? We asked these questions because we wanted to create a scaffolded learning tool that students would readily engage with during their learning process. We asked about fairness and usability, because a negative response to these issues would reveal student affect which may cause students to disengage from Proof Blocks problems. Likewise, when students feel that scaffolded learning environments are inauthentic, as some students feel about block-based programming languages [34], they may disengage. We asked the students Likert scale questions with 5 possible responses: strongly disagree, somewhat disagree, neutral, somewhat agree, and strongly agree. Out of the 325 students included in the psychometric analysis, only 51 responded to the survey (15.7%).

To evaluate student's perceptions of authenticity, we had students rate their agreement to the following:

1. Proof Blocks accurately represent my understanding of how to write proofs.
2. Written proofs accurately represent my understanding of how to write proofs.

We converted these items to numeric scales of 1-5 so that we could use statistical tests to help us answer **RQ3**. We used a Mann-Whitney U test to determine if students' responses to these two questions were significantly different, with the null hypothesis that students have the same perception of how well Proof Blocks problems and written proofs represent their understanding of how to write proofs. To evaluate students' perceptions of fairness, we had them rate their agreement to:

1. The assignment of partial credit for Proof Blocks was fair.
2. The assignment of partial credit for written proofs was fair.

Again, we used a Mann-Whitney U test to determine if students' responses to these two questions were significantly different, this time with the null hypothesis that students believed that the assignment of partial credit was equally fair for Proof Blocks problems and written proofs. To understand students' perceptions of the usability of Proof Blocks, we had them rate their agreement to:

1. The Proof Blocks user interface was easy to use.

We do not apply any statistical tests for this construct because the user interface for Proof Blocks is incommensurate with the interface for writing proofs.

Finally, to see if student's perceptions of Proof Blocks' difficulty aligned with the empirical evidence about question difficulty, we had students to rate their agreement to:

1. Proof Blocks problems are easier than written proofs.

Again, there were no statistical test for this item, but we felt it would be desirable to know if the students' perception of the difficulty of Proof Blocks questions aligned with the empirical evidence.

We also asked three optional open ended questions, mainly with the goal of giving students the opportunity to voice any major concerns they may have had with Proof Blocks:

1. How do you think we could improve Proof Blocks Questions?
2. Given more practice problems, what do you think Proof Blocks would help you learn?
3. Do you have any other feedback about Proof Blocks?

No major concerns were raised. While we did not have enough responses to the open ended questions to do a qualitative analysis, we will use some of them to help us interpret the results of the quantitative survey questions.

6.0 RESULTS AND DISCUSSION

6.1 PSYCHOMETRICS

6.1.1 Results. We will now examine the fit of the 2PL model to answer **RQ1**. The full model fit of the 2PL is shown in Table 2, with the test questions divided by category. It is important to recall that in this case, because the students were learning across the course of the semester in between these test questions, the difficulty measurement of the questions is relative to the student knowledge at the time they took that particular exam, rather than absolute.

Figure 5 is a box and whisker plot that compares the difficulty of the different types of questions. We first used a Shapiro-Wilk normality test to show that the distributions of difficulty of proof problems ($W = 0.92$, $p = 0.39$) and Proof Blocks problems ($W = 0.99$, $p = 0.99$) are both close enough to normal

Table 2: Difficulty (Diff.) and Discrimination (Disc.) parameters for all items in the 2PL model fit. Topic names have been shortened to save space. For the full names, refer to Table 1.

Type	Question	Topic	Diff.	Disc.	Question	Topic	Diff.	Disc.
Proof	**1**	Logic and Proofs	-0.54	1.04	**2**	Sets, functions	-0.28	1.68
	3	Sets, functions	0.55	0.84	**4**	Cardinality	1.66	0.67
	5	Directed graphs	0.24	0.93	**6**	Undirected Graphs	2.41	0.64
	7	Induction	0.75	1.19	**8**	Induction	0.87	1.18
	9	Recursive sets	0.54	1.45	**10**	Number Theory	0.27	1.20
ProofBlocks	**11**	Logic and Proofs	-1.18	0.80	**12**	Logic and Proofs	-0.99	1.16
	13	Cardinality	-0.52	1.38	**14**	Cardinality	-0.28	1.26
	15	Directed graphs	-0.34	0.49	**16**	Undirected Graphs	0.38	0.90
	17	Algorithm analysis	-1.76	0.57	**18**	Algorithm analysis	-0.71	1.04
Other	**19**	Logic and Proofs	-2.54	0.91	**20**	Logic and Proofs	-4.97	0.39
	21	Logic and Proofs	-3.32	0.73	**22**	Sets, functions	0.02	0.64
	23	Sets, functions	-6.75	0.37	**24**	Sets, functions	-2.06	0.83
	25	Cardinality	-2.85	1.06	**26**	Cardinality	-3.77	0.61
	27	Cardinality	-1.79	0.66	**28**	Directed graphs	-0.01	0.58
	29	Directed graphs	-2.33	0.45	**30**	Directed graphs	0.54	0.78
	31	Directed graphs	-1.97	0.99	**32**	Undirected Graphs	-0.45	0.64
	33	Undirected Graphs	-0.35	0.44	**34**	Undirected Graphs	-1.72	0.88
	35	Undirected Graphs	0.25	0.27	**36**	Induction	-1.44	1.37
	37	Induction	-0.29	1.69	**38**	Induction	-1.06	1.29
	39	Recursive sets	-0.12	0.87	**40**	Recursive sets	-1.42	0.91
	41	Recursive sets	-2.48	1.14	**42**	Recursive sets	-2.33	1.58
	43	Recursive sets	-2.22	0.90	**44**	Number Theory	-2.03	0.88
	45	Number Theory	-1.91	0.92	**46**	Number Theory	-1.20	1.01
	47	Number Theory	-1.31	1.10	**48**	Probability	-1.31	0.93
	49	Probability	-0.70	1.11	**50**	Probability	1.95	1.11
	51	Probability	-1.29	0.95	**52**	Probability	-0.72	0.96
	53	Probability	-0.02	1.24	**54**	Series sums	0.22	1.47
	55	Series sums	0.46	1.49	**56**	Series sums	-0.18	0.82
	57	Series sums	0.38	1.84	**58**	Series sums	0.16	1.00
	59	Algorithm analysis	-1.39	1.03	**60**	Algorithm analysis	-4.12	0.43
	61	Algorithm analysis	0.42	0.87	**62**	Algorithm analysis	-0.06	0.79

distributions to justify using a standard t-test. The t-test shows that proof questions are significantly more difficult than Proof Blocks problems (p = 0.003). Proof questions had a mean difficulty of 0.64 (95% CI [0.025, 1.27]), meaning that students who had an ability level of 0.64 standard deviations above the mean had a 50% chance of receiving full credit on a proof problem, with students at mean ability level having a lower chance of receiving full credit. Proof Blocks problems had a mean difficulty of -0.68 (95% CI [-1.22, -0.134]), meaning that students with ability level 0.68 standard deviations below the mean had a 50% chance of receiving full credit on a Proof Blocks problem, on average.

A Shapiro-Wilk normality test showed that the distribution of discrimination parameters was also normal for both written proofs (W = 0.96, p = 0.77) and Proof Blocks questions (W = 0.96, p = 0.80). A t-test shows that the two distributions are indistinguishable (p = 0.40), so we do not reject the null hypothesis that written proofs and Proof Blocks problems measure knowledge with the same discrimination. The mean of the discriminations is 1.08 (95% CI [0.84, 1.32]) for written proofs, and 0.95 (95% CI [0.68, 1.21]) for Proof Blocks problems.

Figure 6 shows the relative information given by the types of questions, normalized by the number of questions in each category. In the information curves, the height and area under the curve are influenced by the discrimination of the questions (with more area meaning more information about student knowledge and a more accurate measurement), and the location of the peak of the curve shows the difficulty.

6.1.2 Discussion. The statistical evidence is clear: Proof Blocks problems were easier than proof problems, and on average, Proof Blocks problems provided a similar amount of information about student knowledge as did written proof questions. This makes Proof Blocks problems ideal test questions: they are straightforward to write, give substantial information about student knowledge, and can be graded fully automatically.

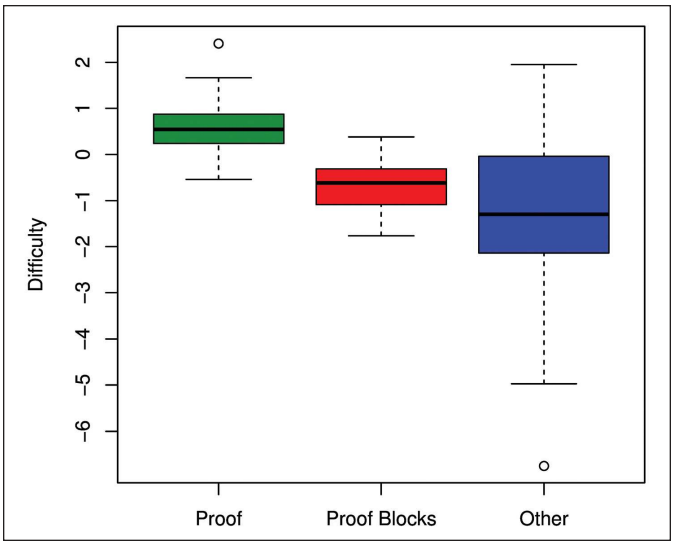

Figure 5: : Box and whisker plot showing the relative difficulty of Proof, Proof Blocks, and Other questions. There is a clear separation between the difficulty level of proof problems and Proof Blocks problems, with Proof Blocks problems being slightly easier (p = 0.003).

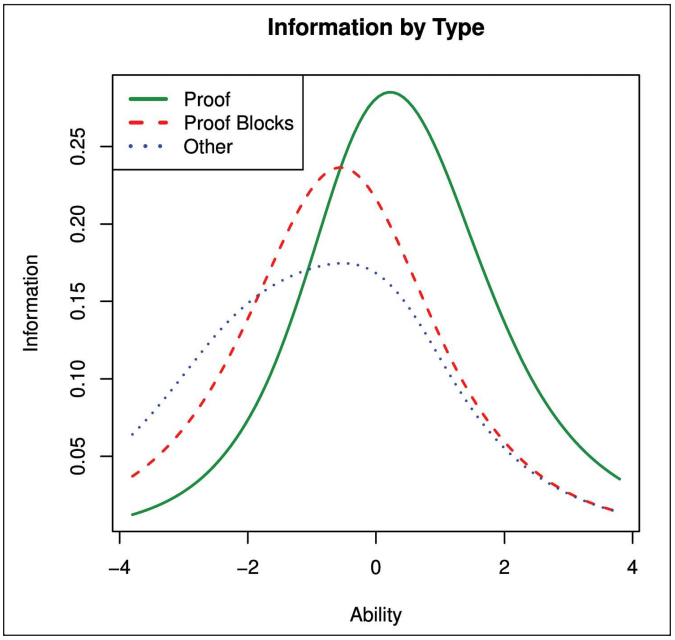

Figure 6: Information given by each type of test question, normalized by number of questions of that type. This plot can be viewed as a summary of the psychometric results: the large amount of area under the curve for both Proof Blocks problems and written proofs showed that they give a substantial amount of information about student knowledge, while the location of the peaks shows that Proof Blocks problems are easier than written proofs.

6.2 CORRELATIONS WITH OTHER QUESTIONS

6.2.1 Results. Table 3 gives the correlations between students' performance on different types of exam questions. All questions types were highly correlated.

Table 3: Correlations between question types. Student grades are highly correlated between all types of questions given to students on their exams. Each of the correlations is significant at $p < 0.001$.

	Correlation	Low. 95% C.I.	Up. 95% C.I.
Proof-Proof Blocks	0.65	0.58	0.71
Proof Blocks-Other	0.75	0.68	0.80
Proof-Other	0.72	0.65	0.77

6.2.2 Discussion. The high correlation between all types suggests that the types of skills assessed by the different types of questions are not dissimilar. By engaging students with Proof Blocks problems, which require similar skills to written proofs, but are easier, we hope to bridge the gap from students having the content knowledge required to understand proofs, to actually being able to write proofs.

6.3 SURVEY

6.3.1 Results. Only 51 of the 325 students included in the psychometric analysis responded to the survey (15.7%). The results of the Likert scale survey questions are show in Figure 7.

A Mann-Whitney U test fails to show significant difference (p =.087, W = 1058) between student agreement with to the statement "Proof Blocks accurately represent my understanding of how to write proofs" (mean = 3.67) and the statement "Written proofs accurately represent my understanding of how to write proofs" (mean = 3.98). As with all hypothesis tests, this could mean either that there is no difference, or that the effect size was small enough that our sample wasn't large enough to detect it.

A Mann-Whitney U test also shows no significant difference (p = 0.75, W = 1255) between student agreement with the statement "The assignment of partial credit for Proof Blocks was fair" (mean = 3.64) and student agreement with the statement "The assignment of partial credit for written proofs was fair" (mean = 3.75). No students disagreed that the user interface was easy to use.

6.3.2 Discussion. We find it very encouraging that 71% of respondents agreed that Proof Blocks problems *did* accurately represent their ability to write proofs, giving support to the authenticity of Proof Blocks—nearly as many as the 75% who believed that written proofs problems accurately represented their ability. It is difficult to have a scaffolded activity feel as authentic as the real thing. For example, some students have concerns over the authenticity of writing code using block based languages [34]. We also find it encouraging that students felt that the assignment of partial credit for Proof Blocks problems was just as fair as the partial credit assignment for written proofs.

Some students gave answers to the free response questions that helped give more meaning to the quantitative survey results. One student elaborated on the benefits of the scaffolding provided by Proof Blocks:

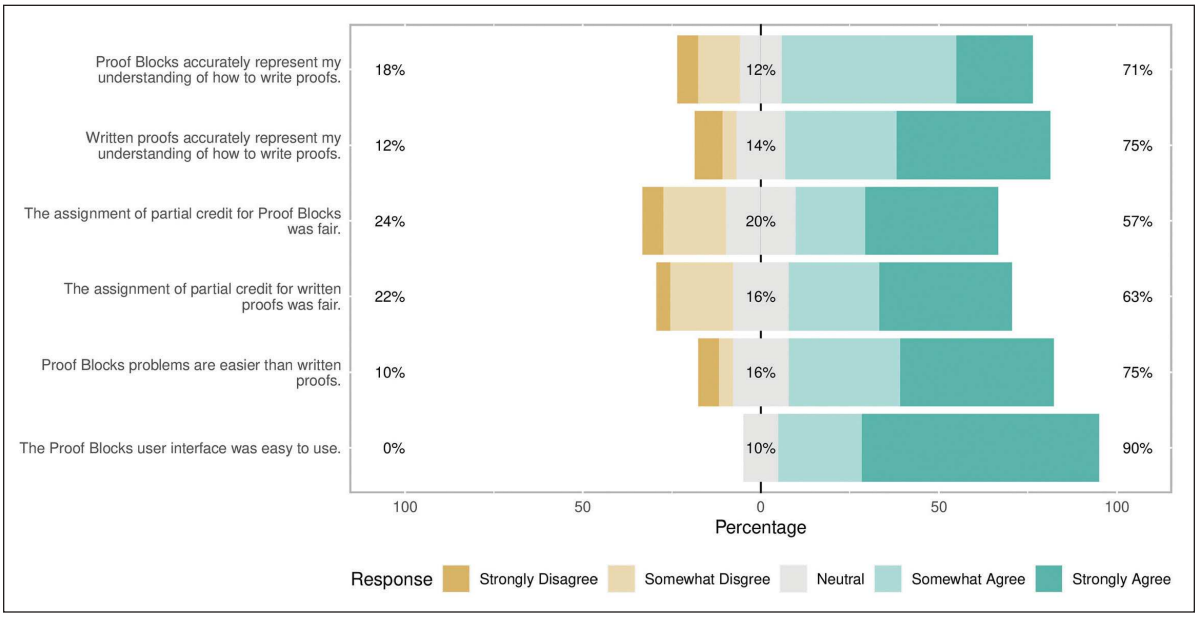

Figure 7: Responses to the Likert scale questions on the survey. Notable highlights of the survey are that no students disagreed that the user interface was easy to use, 71% felt that Proof Blocks accurately represented their understanding of how to write proofs (versus 75% for written proofs), and 57% felt that the assignment of partial credit for Proof Blocks problems was fair (versus 63% for written proofs).

> Usually my biggest struggle when it comes to writing proofs is finding a place to start and using concrete wording/reasoning to do so. With Proof Blocks, I get the skeleton and concrete wording given to me so I can focus on applying theorems and having a coherent train of thought.

Another student gave more insight into why they felt that Proof Blocks were easier than written proofs, a sentiment that most students seemed to share based on the Likert scale data:

> I think they're much easier than written proofs because of how much information the problem gives. There were a lot of proof block questions that I would have no clue how to do as a written proof but I got full credit on them through simple process of elimination. For example, some proofs have multiple sets of "consider" where you pick the function f and corresponding next steps based on which function was picked. It's very easy to tell which blocks go with which "set" of steps go together, which effectively makes the question multiple choice (with fewer choices) because the last step of the proof is obvious.

7.0 LIMITATIONS

The primary limitation of our study is the fact that our data set allows us only to answer certain questions about Proof Blocks problems and not others. For example, we are able to make a strong claim that Proof Blocks problems function well as test questions, assessing student knowledge of discrete mathematics in an accurate and useful way, but we are not yet able to comment on the usefulness of Proof Blocks problems for learning to write proofs. Since nearly all of the data we collected was quantitative, we are largely unaware of students thought processes and affect as they work through Proof Blocks problems. Furthermore, as distractors for questions were chosen in an ad-hoc manner, we are not able to comment on what types of distractor lines do or don't work well in Proof Blocks problems, or what their impact is on learning or assessment. Another limitation is that our survey sample was a small percentage of the course (15.7%), and because the survey was completely anonymous, we have no way of knowing any demographic information about those who chose to complete the survey.

The discrete mathematics course was taught by multiple instructors, some of whom had reservations about putting unproven problem formats onto the exams. Consequently, we could not include Proof Blocks problems and traditional proofs on every relevant exam, limiting the types of analyses we could perform. However, we believe that our study has very high ecological validity—we demonstrated that Proof Blocks problems are useful in flow of a normal discrete mathematics course, without special changes being made and without emphasizing Proof Blocks problems during instruction or assignments.

8.0 ADOPTING PROOF BLOCKS

Documentation, instructions, and more examples for Proof Blocks and PrairieLearn can be found online in the PrairieLearn documentation and example courses [28, 29]. PrairieLearn is integrated with Learning Tools Interoperability [25] to enable easier sharing of student data across learning platforms. Authors may be contacted with questions.

9.0 CONCLUSION

We have shown that Proof Blocks problems have many properties that instructors desire when writing tests. First, they have high discrimination and thus provide a substantial amount of information about student knowledge—comparable to written proofs. They are also easier than written proof problems, and thus may be appropriate for scaffolding students from content knowledge to writing proofs. Proof Blocks decrease the grading burden on course staff, allowing more time for office hours and other activities that help students learn. Furthermore, students felt that the Proof Blocks interface was easy to use, that the questions accurately represented their understanding of how to write proofs—almost as well as actually writing proofs. ❖

Acknowledgements

We acknowledge the Computers of Education research group at the University of Illinois at Urbana-Champaign for much helpful feedback on earlier versions of this work. We also acknowledge Benjamin Cosman and Patrick Lin for their work as instructors of the course we obtained data from, including writing many of the test questions. Mahesh Viswanathan was partially supported by NSF CCF 1901069 and NSF CCF 2007428.

References

1. William Billingsley and Peter Robinson. 2007. Student proof exercises using MathsTiles and Isabelle/HOL in an intelligent book. *Journal of Automated Reasoning* 39, 2 (2007), 181–218.
2. Richard Bornat and Bernard Sufrin. 1997. Jape: A calculator for animating proof-on-paper. In *International Conference on Automated Deduction*. Springer, 412–415.
3. Joachim Breitner. 2016. Visual theorem proving with the Incredible Proof Machine. In *International Conference on Interactive Theorem Proving*. Springer, 123–139.
4. Paul Denny, Andrew Luxton-Reilly, and Beth Simon. 2008. Evaluating a new exam question: Parsons problems. In *Proceedings of the fourth international workshop on computing education research*. 113–124.
5. Yuemeng Du, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Research on Parsons Problems. In *Proceedings of the Twenty-Second Australasian Computing Education Conference(ACE'20)*. Association for Computing Machinery, New York, NY, USA, 195–202. https://doi.org/10.1145/3373165.3373187
6. Douglas E Ensley and J Winston Crawley. 2005. *Discrete mathematics: mathematical reasoning and proof with puzzles, patterns, and games*. John Wiley & Sons.
7. Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th Koli Calling International Conference on Computing Education Research*. 20–29.
8. N. Fraser. 2015. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. 49–50. https://doi.org/10.1109/BLOCKS.2015.7369000
9. Ken Goldman, Paul Gross, Cinda Heeren, Geoffrey Herman, Lisa Kaczmarczyk, Michael C Loui, and Craig Zilles. 2008. Identifying important and difficult concepts in introductory computing courses using a delphi process. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education*. 256–260.
10. Geoffrey L. Herman, Craig Zilles, and Michael C. Loui. 2014. A Psychometric Evaluation of the Digital Logic Concept Inventory. *Computer Science Education* 24, 4 (2014), 277 – 303.
11. Mark Hodds, Lara Alcock, and Matthew Inglis. 2014. Self-explanation training improves proof comprehension. *Journal for Research in Mathematics Education* 45, 1 (2014), 62–101.
12. Association for Computing Machinery (ACM) Joint Task Force on Computing Curricula and IEEE Computer Society. 2013. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. Association for Computing Machinery, New York, NY, USA.
13. Sorin Lerner, Stephen R Foster, and William G Griswold. 2015. Polymorphic blocks: Formalism-inspired UI for structured connectors. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 3063–3072.
14. Frederic M Lord. 1980. *Applications of item response theory to practical testing problems*. Routledge.
15. John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 1–15.
16. The Joint Task Force on Computing Curricula. 2014. *Curriculum Guidelines for Undergraduate Degree Programs in Software Engineering*. Technical Report. New York, NY, USA.
17. Dale Parsons and Patricia Haden. 2006. Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses. In *Proceedings of the 8th Australasian Conference on Computing Education* - Volume 52 (Hobart, Australia) (ACE '06). Australian Computer Society, Inc., AUS, 157–163.
18. Leo Porter, Daniel Zingaro, Soohyun Nam Liao, Cynthia Taylor, Kevin C Webb, Cynthia Lee, and Michael Clancy. 2019. BDSI: A validated concept inventory for basic data structures. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 111–119.
19. Seth Poulsen, Geoffrey L. Herman, Peter A.H. Peterson, Enis Enis Golaszewski, Akshita Gorti, Linda Oliva, Travis Scheponik, and Alan T. and Sherman. 2021. Psychometric Evaluation of the Cybersecurity Concept Inventory. *ACM Transactions on Computing Education (TOCE)* In press (2021).
20. Seth Poulsen, Mahesh Viswanathan, Geoffrey L. Herman, and Matthew West. 2021. Proof Blocks: Autogradeable Scaffolding Activities for Learning to Write Proofs. arxiv:2106.11032 [cs.CY]
21. R Core Team. 2020. R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. https://www.R-project.org/
22. Dimitris Rizopoulos. 2006. LTM: An R package for latent variable modelling and item response theory analyses. *Journal of Statistical Software* 17, 5 (2006), 1–25. http://www.jstatsoft.org/v17/i05/
23. Annie Selden and John Selden. 1987. Errors and misconceptions in college level theorem proving. In *Proceedings of the second international seminar on misconceptions and educational strategies in science and mathematics*, Vol. 3. ERIC, 457–470.
24. Annie Selden and John Selden. 2008. Overcoming Students' Difficulties in Learning to Understand and Construct Proofs. In *Making the Connection*, Marilyn P. Carlson and Chris Rasmussen (Eds.). The Mathematical Association of America, Washington DC, 95–110. https://doi.org/10.5948/UPO9780883859759.009
25. Charles Severance, Ted Hanss, and Josepth Hardin. 2010. Ims learning tools interoperability: Enabling a mash-up approach to teaching and learning tools. *Technology, Instruction, Cognition and Learning* 7, 3-4 (2010), 245–262.
26. Andreas J Stylianides, Kristen N Bieda, and Francesca Morselli. 2016. Proof and argumentation in mathematics education research. In *The second handbook of research on the psychology of mathematics education*. Brill Sense, 315–351.
27. GJ Stylianides, AJ Stylianides, and K Weber. 2017. Research on the teaching and learning of proof: Taking stock and moving forward. In *Compendium for Research in Mathematics Education*, Jinfa Cai (Ed.). National Council of Teachers of Mathematics, Chapter 10, 237–266.
28. PrairieLearn Team. 2021. pl-order-blocks Documentation. https://prairielearn.readthedocs.io/en/latest/elements/pl-order-blocks-element
29. PrairieLearn Team. 2021. PrairieLearn Documentation. https://prairielearn.readthedocs.io/en/latest/
30. Association for Computing Machinery (ACM) The Joint Task Force on Computing Curricula and IEEE Computer Society. 2016. *Curriculum Guidelines for Undergraduate Degree Programs in Computer Engineering*. Technical Report. New York, NY, USA.
31. Lev Semenovich Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Harvard university press.
32. Keith Weber. 2001. Student difficulty in constructing proofs: The need for strategic knowledge. *Educational Studies in Mathematics* 48, 1 (Oct. 2001), 101–119. https://doi.org/10.1023/A:1015535614355
33. Keith Weber and Lara Alcock. 2004. Semantic and Syntactic Proof Productions. *Educational Studies in Mathematics* 56, 2 (July 2004), 209–234. https://doi.org/10.1023/B:EDUC.0000040410.57253.a1
34. David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th international conference on interaction design and children*. 199–208.
35. Matthew West, Geoffrey L. Herman, and Craig Zilles. 2015. PrairieLearn: Mastery-based Online Problem Solving with Adaptive Scoring and Recommendations Driven by Machine Learning. In *2015 ASEE Annual Conference & Exposition*. ASEE Conferences, Seattle, Washington, 26.1238.1–26.1238.14. https://peer.asee.org/24575.
36. David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of child psychology and psychiatry* 17, 2 (1976), 89–100.
37. Benjamin Xie, Matthew J. Davidson, Min Li, and Amy J. Ko. 2019. An item response theory evaluation of a language-independent CS1 knowledge assessment. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education(SIGCSE '19)*. Association for Computing Machinery, Minneapolis, MN, USA, 699–705. https://doi.org/10.1145/3287324.3287370

Seth Poulsen
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801
sethp3@illinois.edu

Mahesh Viswanathan
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801
vmahesh@illinois.edu

Geoffrey Herman
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801
glherman@illinois.edu

Matthew West
Department of Computer Science
University of Illinois at Urbana-Champaign, Urbana, Illinois, 61801
mwest@illinois.edu

DOI: 10.1145/3514213