Assessing Bootstrap:Algebra Students on Scaffolded and Unscaffolded Word Problems

Emmanuel Schanzer
Bootstrap/Brown University
schanzer@bootstrapworld.org

Kathi Fisler
Brown University/WPI/Bootstrap
kfisler@cs.brown.edu

Shriram Krishnamurthi
Brown University/Bootstrap
sk@cs.brown.edu

ABSTRACT

Bootstrap:Algebra is a curricular module designed to integrate introductory computing into an algebra class; the module aims to help students improve on various essential learning outcomes from state and national algebra standards. In prior work, we published initial findings about student performance gains on algebra problems after taking Bootstrap. While the results were promising, the dataset was not large, and had students working on algebra problems that had been scaffolded with Bootstrap's pedagogy. This paper reports on a more detailed study with (a) data from more than three times as many students, (b) analysis of performance changes in incorrect answers, (c) some problems in which the Bootstrap scaffolds have been removed, and (d) an IRT analysis across the elements of Bootstrap's program-design pedagogy. Our results confirm that students improve on algebraic word problems after completing the module, even on unscaffolded problems. The nature of incorrect answers to symbolic-form questions also appears to improve after Bootstrap.

ACM Reference Format:
Emmanuel Schanzer, Kathi Fisler, and Shriram Krishnamurthi. 2018. Assessing Bootstrap:Algebra Students on Scaffolded and Unscaffolded Word Problems. In *SIGCSE '18: SIGCSE '18: The 49th ACM Technical Symposium on Computer Science Education, February 21–24, 2018, Baltimore , MD, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159498

1 INTRODUCTION

Integrating CS education with existing disciplines is appealing for both intellectual and logistical reasons (the latter particularly in K-12 contexts). Effective integration helps advance student learning in both computing and the host discipline. However, given the known challenges of transferring skills between disciplines [2, 5], claims that computing curricula foster learning in other disciplines require validation through research.

Bootstrap:Algebra (henceforth BS:A) is a curriculum that has been designed to integrate introductory computing into math classes. Roughly, the curriculum has students build a videogame, where each game feature requires both a new mathematics concept (which already needs to be taught in the (pre)-algebra class) and a corresponding programming construct. Given that math is a high-stakes subject (which is heavily tested in the USA), math teachers are under pressure to use curricula that positively impact students' math scores. Understanding the impact of programs like BS:A on student performance is essential for their long-term adoption.

In 2015, we published a preliminary study showing statistically-significant impacts of BS:A on student performance on function composition and word problems [7]. This paper confirms and extends those results in several ways: we remove some of the problem scaffolds (while retaining performance gains), show that students often make progress even while getting wrong answers, demonstrate performance gains in BS:A skills beyond symbolic form questions, and apply IRT-based analysis to get a richer picture of what is going on across the different elements of the BS:A pedagogy.

2 RELATED WORK

Our work attempts to confirm and extend our initial assessment of student performance gains from BS:A [7]. Our original paper looked at both function-composition and word problems, using paired t-tests to compare performance deltas from pre- to post-tests. The current paper focuses on word problems, adding unscaffolded versions of these questions. Our analyses are now also more nuanced: we explore deltas in students' incorrect answers, compare performance across different parts of the BS:A pedagogy, and include an IRT analysis. Wright, Rich, and Lee [8] studied the impact of BS:A on students' understanding of variables, finding some positive effects. Variables are beyond the scope of this study.

BS:A's curricular design builds on theories about how to achieve transfer of skills from one discipline to another. Both explicit instruction and alignment of problem-solving processes are currently believed to be essential for transfer [2, 5, 6]. BS:A embraces both of these issues, as described in section 3. NCTM (the USA's National Council of Teachers of Mathematics) has raised concerns about casual claims that learning CS enhances learning of math [4]. Our studies leverage questions from state math exams to ground our analysis in performance measures designed for mathematics.

While other projects use computing or software tools to teach algebra concepts, we are not aware of ones that deeply align the *problem-solving process* across math and computing as Bootstrap does (see section 3). Space precludes a review of these other projects.

3 AN OVERVIEW OF BS:A

BS:A is a 20–25 hour curricular module designed to integrate introductory computing and specific learning objective from middle- and high-school algebra [1]. The module is designed to embed into a pre-algebra or algebra 1 class; such courses tend to be offered within the age range 12–15 (in the USA). The module has students design and build a video-game featuring three characters, each of which moves in a single dimension. There is a user-controlled

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
https://doi.org/10.1145/3159450.3159498

player moving in the y-axis (in response to key-presses), a *target* that the player is trying to catch, and a *danger* that the player is trying to avoid. Both the target and the danger move left-to-right across the screen (in the x-axis, at a fixed y coordinate that changes each time the character wraps around to restart from the left edge of the game). Students customize their games by selecting images for the three characters as well as a background image. This leads to wide variety in the game themes, from chasing credit cards in the mall to avoiding spiders in the desert.

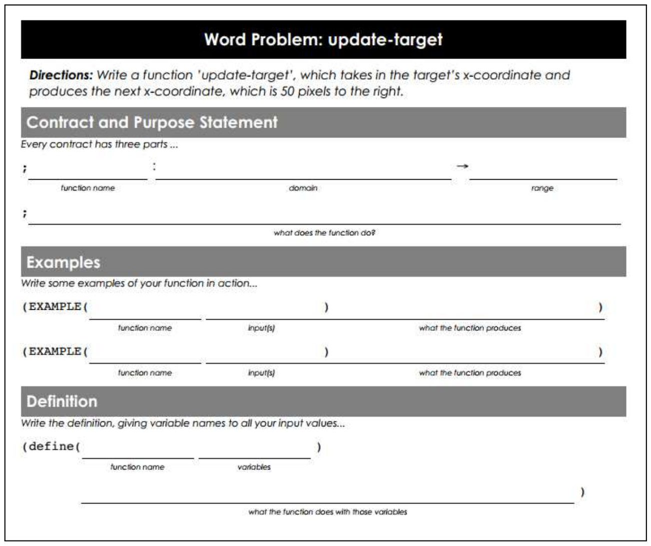
BS:A views animations and games as filmstrips consisting of a sequence of frames. Each frame captures one instant of the animation, just as in the filmstrips used to create movies. For each character in a game, students write a function describing how that character's position changes from one frame to the next. A function such as $f(x) = x + 5$, for example, captures moving a character 5 pixels to the right in each successive frame. Unlike in many popular early programming platforms (such as Scratch or Alice), students do not write loops to generate movement: they focus on the algebraic functions that describe how movement changes between frames. BS:A has students solve a series of word problems, each corresponding to some game feature (moving different characters, wrapping characters around after they leave the screen, responding to key presses, and detecting collisions to compute scores). This alignment in style of problems is a key component underlying the integration of computing and algebra in BS:A.

Why Might BS:A Impact Algebra? Reducing game programming to developing algebraic functions is only one aspect of how BS:A integrates computing and algebra. The algebra connection also lies in a *step-by-step process for solving word problems* that leverages *multiple representations of functions*. Both topics feature in state mathematics standards, the NCTM's (National Council of Teachers of Mathematics) guidelines for math outcomes, and the (USA) Common Core. Algebra standards typically emphasize four representations of functions: symbolic form (like $f(x) = x + 2$), domain and range, input/output tables, and graphs; the standards ask students to relate all four representations across the same conceptual function. BS:A exercises the first three (presently omitting graphs).

Our *Design Recipt Worksheet* (fig. 1) embodies how BS:A connects these representations. Given a word problem, students are taught to first articulate the domain and range of the problem, to write at least two examples of the input/output relationship described in the problem, and only then to write the symbolic form of a function to solve the word problem. The curricular materials explain how to leverage each step when attempting the next step, thus modeling ways in which the different representations depend on one another.

The curriculum teaches a specific way to approach input/output tables, which is designed to help students abstract over examples to derive the symbolic form. Consider the word problem in fig. 1, which asks students to produce a function that computes a new x-coordinate 50 pixels to the right of the input x-coordinate. Students could either write the expected output of examples as the final result of the function or as an *expression that computes the final result*. The difference is illustrated in the table below:

10	60	final result only
5	5 + 50	expression that computes result
8	8 + 50	another expression that computes result



Word Problem: update-target

Directions: *Write a function 'update-target', which takes in the target's x-coordinate and produces the next x-coordinate, which is 50 pixels to the right.*

Contract and Purpose Statement

Every contract has three parts ...

```
; ____________ : ______________________ → ____________
  function name        domain                 range
; ____________________________________________________
              what does the function do?
```

Examples

Write some examples of your function in action...

```
(EXAMPLE( ____________ ____________ )  ____________________ )
          function name   input(s)     what the function produces
(EXAMPLE( ____________ ____________ )  ____________________ )
          function name   input(s)     what the function produces
```

Definition

Write the definition, giving variable names to all your input values...

```
(define( ____________ ____________ )
         function name   variables
         ______________________________________ )
         what the function does with those variables
```

Figure 1: A design-recipe worksheet, showing the process for solving word problems.

The second two examples show the *computation* that would result in the answer. To create a function that solves the word problem, the student needs to abstract over the two expressions, replacing the varying (input) value by a parameter name. The curriculum explicitly teaches this practice: students circle differences between expressions, label the circles with names, then use the names when writing the symbolic form:

$$update\_target(xpos) = xpos + 50$$

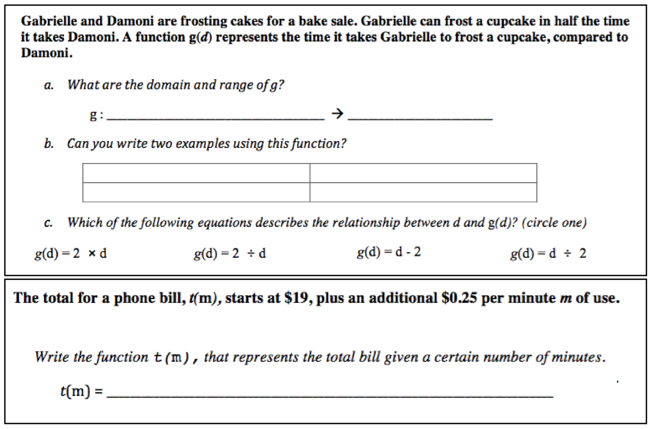
This practice of writing output expressions and labeling their differences is not part of conventional mathematics instruction for developing symbolic forms of functions.

The curricular materials provide teachers with several design-recipe worksheets for conventional algebra word problems, in addition to those for the word problems that yield code for creating the videogame. Thus, students are explicitly taught how to transfer the design process from computing to algebra. The transfer literature (section 2) establishes that explicit instruction is generally required before students will transfer skills from one domain to another.

4 STUDY DESIGN

In our original study [7], students solved both function composition and word problems. This new study focuses on word problems alone, with an intent to study them more deeply. (We also lacked a control group; see section 6 for more discussion on this.)

All of the word problems in the original study were scaffolded by the design recipe. This meant that students were asked to explicitly give the domain/range, examples of use, and symbolic form for each word problem. This scaffolding is useful for testing whether students are able to apply the BS:A design steps to algebra problems (rather than game-design problems). It also provides insight into where students might have gone wrong if their symbolic form answer is incorrect. However, typical standardized mathematics


Gabrielle and Damoni are frosting cakes for a bake sale. Gabrielle can frost a cupcake in half the time it takes Damoni. A function g(*d*) represents the time it takes Gabrielle to frost a cupcake, compared to Damoni.

a. *What are the domain and range of g?*

g : ____________________ → ______________

b. *Can you write two examples using this function?*

c. *Which of the following equations describes the relationship between d and g(d)? (circle one)*

g(d) = 2 × d g(d) = 2 ÷ d g(d) = d - 2 g(d) = d ÷ 2

The total for a phone bill, *t*(m), starts at $19, plus an additional $0.25 per minute *m* of use.

Write the function `t(m)`, *that represents the total bill given a certain number of minutes.*

t(m) = ____

Figure 2: Sample questions from our pre and post tests of algebra word problems. The top question includes the BS:A scaffolds, while the bottom question does not. The scaffolded example shown here uses multiple-choice format for the symbolic form; roughly half of our scaffolded questions instead provide a blank link as in the unscaffolded version.

exams (at least in the USA) ask students to provide only the symbolic form. Understanding how BS:A students perform without the scaffolds is therefore critical.

Rather than remove all of the scaffolds, however, we chose to remove the scaffolds on half of the post-test questions (leaving the pre-test fully scaffolded). This design allows us to check whether the same students perform differently on scaffolded vs unscaffolded questions. Naturally, students could be looking at the scaffolded problems for help in solving the unscaffolded ones. This ability adds value to this design: if students perform worse on the unscaffolded problems despite the scaffolding hints being on the same assessment (which was given on paper, so students could see the scaffolded questions alongside the unscaffolded ones), it would suggest that students have not understood the design steps as a process to approaching word problems. Figure 2 gives an example of each of a scaffolded and an unscaffolded question.

Study Logistics. We recruited teachers to participate through a message on the BS:A mailing list. In most cases, we gave teachers PDFs for the pre- and post-tests; the teachers printed the tests, matched the pre- and post-tests by student, and de-identified the tests with unique numbers. This past year, we offered to provide pre-numbered bundles of tests (to save teachers the overhead of matching them). In both cases, teachers mailed the hard/paper copies back to us. We graded all of the tests ourselves to ensure consistency. We asked teachers to allow the students 30 minutes for each test. Teachers gave the pre-test before they started teaching BS:A (or in some cases, within the first week before they got into any of the content on the test); teachers gave the post-test sometime after finishing BS:A (we did not stipulate how close or far the post-test should be given relative to the end of the BS:A unit). Participating teachers received a choice of incentives (Amazon gift cards, BS:A t-shirts for students, or BS:A teaching workbooks for a subsequent school year).

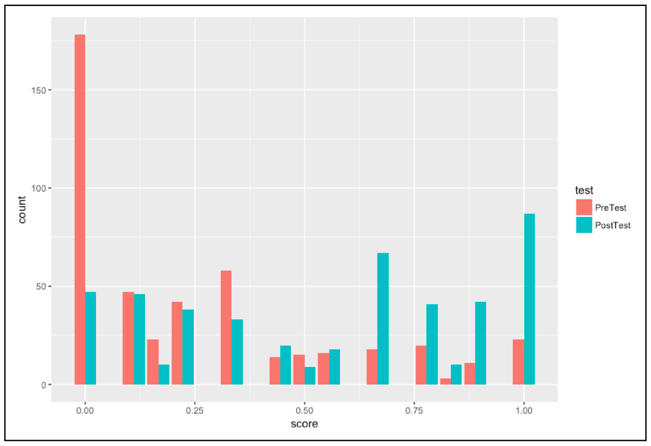


Figure 3: Histogram of Pre and Post Test scores [N=468]. Pretest scores are the left bar; post-test scores are the right bar.

Participants. Twenty-two distinct teachers provided data. We gathered data between 2014 and 2017. Teachers came from multiple states across the USA and represent several school types and district-level demographics. From across the 22 teachers, we received a total of 468 matched pre- and post-tests. Students who submitted only one of the two tests were dropped (and are not counted in the 468). Most students were in grades 7 through 10 (ages 12 through 15), though at least one teacher provided matched data from eight upper-level high school students (12th grade/age 17). All students (regardless of grade level) did the same problems. Not all teachers provided grade-level data, so we do not include that in our analyses. We did not collect student gender, race, or socio-economic data.

5 ANALYSIS AND RESULTS

In our analyses, we report students' *scores* on each test as the percentage of questions that a student answered correctly. We use percentages rather than question counts because different tests contained different numbers of word problems in different years (sometimes 6, sometimes 9). Questions were weighted equally. We did not give partial credit. Unless noted otherwise, our analyses treat blank answers as incorrect (rather than separate blank answers from incorrect ones).

5.1 Performance on Symbolic Form Questions

Figure 3 shows the distributions of pre and post-test scores on the symbolic form questions across the entire dataset. For scaffolded problems, the symbolic-form questions were part (c) as shown in the top sample in fig. 2; for unscaffolded problems, the symbolic form was the entire question (the blank line in the bottom sample in fig. 2). There is one data point per student (the percentage-based score across all symbolic form questions on the corresponding test).

Using a paired t-test, the differences between pre- and post-test scores are significant ($p < .0001$), with an effect size of .888 (by Cohen's d) and a 95% confidence interval of -0.32 to -0.26. This supports the findings of our original paper [7] showing performance gains on standard algebra assessments.

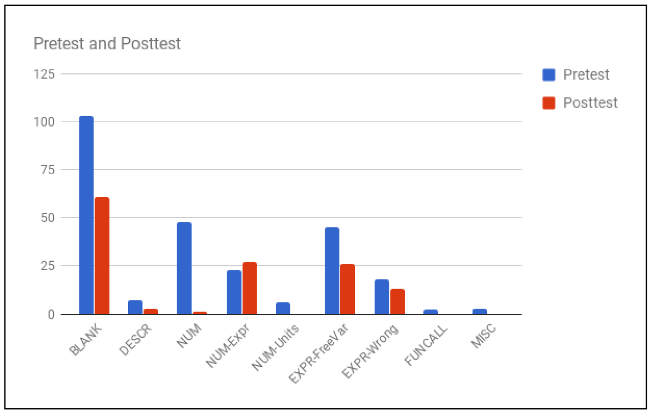


Figure 4: Distribution of types of incorrect free-response answer to symbolic form questions [N=113].

Changes in Incorrect Answers. Even students who answer incorrectly on both the pre- and post-tests could be making progress, depending on the *kinds of wrong answers* that they provide. Figure 4 shows the distribution across categories of wrong answers for questions that asked students to write the symbolic form free-hand (as in the lower problem in fig. 2, though some of these problems were scaffolded with parts (a) and (b) of the upper problem). The categories of mistakes we found were leaving the question BLANK, writing a text DESCRiption of the answer, writing just a constant NUMber, writing a variable-free numeric expression (NUM-Expr, such as "5 + 4"), writing a number and units but not using variables (NUM-Units, such as "3 cans"), writing an expression with a variable other than the given parameter (EXPR-FreeVar), writing an incorrect arithmetic expression over the correct parameter (EXPR-Wrong), writing a call to a function that wasn't named in the problem (FUNCALL, such as "f(10)" for an unknown "f"), and writing unrelated comments (MISC).

Students who used incorrect variables (EXPR-FreeVar) typically fell into two broad camps: those who used the intended function name as a variable, and those who used x or n as a variable (when these were not the given parameter name—these are typical variables used in writing equations and formulas). Typical answers in the EXPR-Wrong category used the wrong operation (+ instead of *), or combined the problem constants and parameter with the wrong operation (such as $5s + 10$ instead of $5 + 10s$).

Figure 4 shows a significant drop in the number of blank answers; the number of students simply entering single numeric answers drops to close to zero. Students make headway on putting only the correct parameter in formulas (EXPR-FreeVar), a discipline which one could expect that writing code would reinforce. The persistence of NUM-Expr errors, in which the symbolic form doesn't reference any variables, points to students who lack a fundamental understanding of functions as a concept. It would be interesting to see whether such students understand this concept in the programming context and haven't transferred it to algebra, or whether they fail to understand the concept in either domain. This is an important topic for future study.

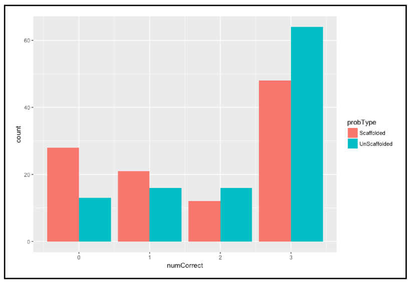


Figure 5: Histogram of scores on scaffolded (left bar) and unscaffolded (right bar) post-test questions [N=109].

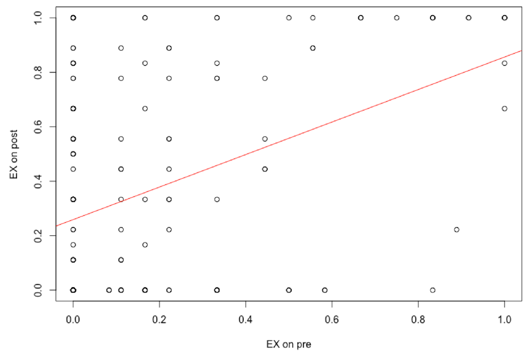


Figure 6: Pre and Post scores on writing examples [N=222].

5.2 Scaffolded Versus Unscaffolded Problems

Figure 5 compares student performance on scaffolded and unscaffolded word problems on the post-test, based on data from 109 students (we only collected data on unscaffolded problems in the last year of our study, after confirming that we were seeing performance gains on the fully scaffolded problems from the first two years). The graph actually shows students having a stronger performance on the *unscaffolded* problems, a result we didn't expect. A paired t-test of scaffolded and unscaffolded problems yields significance of $p < .0001$, but only at an effect size of .39. This result does not appear due to students leaving more scaffolded answers blank: there are 20 instances of a scaffolded symbolic-form question being left blank compared to 35 unscaffolded questions being left blank. It may be related to Heffernan and Koedinger's results that students struggle with expressing word problems in algebraic language more than with figuring out what computation a problem requires [3]. In future work, we need to gather more data and conduct different analysis to confirm and explain these data.

5.3 Performance on Writing of Examples

Writing examples of computations is a key component of the BS:A pedagogy, and aligns with students writing input/output tables in an algebra context. Part (b) of scaffolded word problems asked students to provide two examples for each word problem they were asked to solve (see fig. 2). We are interested in how students evolve

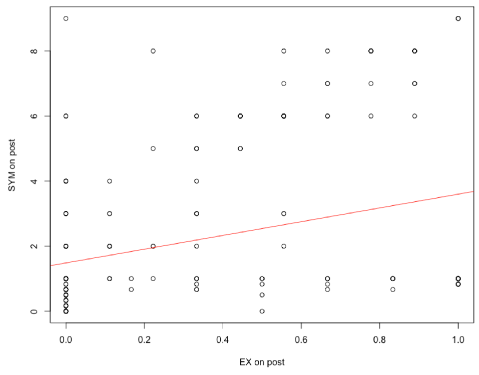


Figure 7: Relationship between examples and symbolic form scores on the post-test [N=222].

in this task, in part because we feel the ability to articulate examples is a valuable skill in both programming and in mathematics.

Figure 6 plots students' scores on writing examples in the pre-test versus the post-test (one data point per student). (We report this data only for students from the last two years of our study (N=222) because we had not tracked data at this detail in the first year of the study.) The graph shows that students are stronger on writing examples in the post-test. A paired t-test shows the differences are significant at p < .0001 (with a Cohen's d of .575). These computations, however, do not distinguish wrong answers from examples that are left blank. In 2016-17, we began distinguishing blank and incorrect answers in our grading; for the 140 students in that cohort, the percentage of blank answers does generally decrease from pre to post, but the number of blanks rises for 42 students. No particular variables (such as teacher or grade level) characterize the students in this group; they could have run out of time or not taken the test as seriously. We lack information to unpack this issue further.

Figure 7 plots the relationships between each student's scores on writing examples and answering symbolic form questions on the post test. Spearman's rho shows that writing examples and answering symbolic form are correlated (p < .0001, rho = .457). The graph illustrates that some students perform well on each of these two tasks while performing poorly on the other.

5.4 IRT Analysis Across Recipe Elements

For another perspective on how student performance evolves across the different design-recipe elements, we ran an Item Response Theory (IRT) analysis. IRT measures a population's performance on each question as a function of a latent trait θ (e.g., understanding of functions). For each item, IRT provides a measure of the question's difficulty (β) and discrimination (α). A larger α suggests that a particular item was a highly-discriminating measure of θ for that population. A β of 0.25 means that a student who is one quarter of a standard deviation above the mean is expected to have an even chance of generating a correct response, and a higher β implies that a correct answer on a particular item is associated with a higher level of θ (relative to the population).

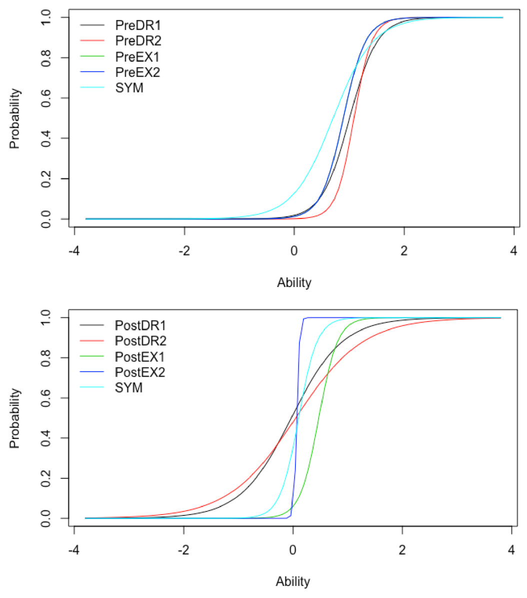


Figure 8: IRT graphs from each step of the design recipe, looking at similar problem from each of the pre- and post-tests. Each IRT plot shows curves for domain, range, two examples, and determining the symbolic form.

For each of the pre- and post-tests, we captured the correctness of students' answers on the domain, range, two examples in the input/output table (each example as a separate score), and the symbolic form of the function. This yields a total of 5 answers for each scaffolded problem and 1 answer for each unscaffolded problem. We ran a separate IRT analysis across all of the answers on each of the pre- and post-test data. Figure 8 shows the characteristic curves for each subpart of one question from each of the pre-test and the post-test (the analysis was done by constructing a latent-trait model using the `ltm` package and function in R). The questions from which these curves arise ask students to produce similar symbolic formulas, but the word problems use different contexts.

The symbolic form curve in the pre-test (SYM, α=2.732, β=0.704) has a gentler slope and is shifted slightly left of the curves for domain (α=3.976, β=1.005), range (α=6.173, β=1.085), and examples (α= 0.891, β=4.938 for both). This suggests that on the pre-test, students were more likely to correctly answer the symbolic form questions than others, regardless of their ability (θ). This is not necessarily surprising: most math classes spend considerable time on this form, but far less (if any) time on the terminology of domain and range, or the practice of writing examples. The higher difficulty and discrimination of these items merely shows that they evaluate specialized knowledge that few students possess during the pre-test.

Given that domain and range play a central role in BS:A's problem-solving strategy, we expect students' understanding to shift from being specialized knowledge (harder, more discriminating) in the pre-test to more general knowledge (easier, less discriminating) in the post-test. Indeed, this expectation in born out in the analysis, as the domain and range curves have a gentler slope and shift left

(β drops from 1.005 and 1.085 to -0.032 and 0.048), indicating that the relative proportion of the students who were able to correctly answer these questions increased from pre to post. α also decreases from pre- to post-test (from 3.976 and 6.173 to 2.139 and 1.619), suggesting that domain and range play a smaller role in discriminating students' level of θ in the post-test than they do in the pre-test.

Interestingly, the α values on the examples curves *increase* relative to the pre-test (from 4.938 on both to 5.85 and 40.690), suggesting that having students provide examples improved as an indicator of their understanding of functions on the post-test. The sharp difference in α between the two post-test example curves may indicate that *asking for a second example* is a more useful measurement of θ than asking for just one, a question worthy of future study.

This effect was even more pronounced for the distribution of correct answers to the symbolic form question. The measure of difficulty dropped from pre (β=0.704) to post (β=0.101), meaning that the likelihood of a correct response required that students be only 0.101 SD above the mean in the post-test, compared to nearly seven times that amount above the mean in the pre-test. The shift in difficulty implies that the portion of students who possessed sufficient θ also increased relative to the pre-test. Meanwhile, the usefulness of this type of question as a discriminator also improved from α=2.732 to α=5.909.

The trends from these particular two problems are similar to what we see across other problems on the pre- and post-tests.

6 THREATS TO VALIDITY

In general, we do not have information on what courses or lessons students were taking in parallel with BS:A. It is possible, for example, that some students were taking another math class in parallel to their BS:A course, and that some of the content tested on our post-test had been covered in the math class. We guard against this in part through having a large number of teachers participating in data collection. In addition, many (though not all) of the teachers who provided data are math teachers who were using BS:A in a math class, so students would not have been taking a math class in parallel. However, teachers use BS:A in different models: some do the curriculum in a concentrated chunk, some spread it over several weeks, and so on. Thus, there are almost certainly variations across the math teachers in how their BS:A curriculum interacted with other aspects of their courses. This paper again relies on a larger teacher population to ameliorate this issue in our analysis.

Unlike in our initial study, we did not have control groups. Our ideal control group would be another section of the same math course taught by a participating BS:A teacher. None of the teachers who provided data were able to provide a control group (for varying reasons). The lack of control groups mainly affects comparisons between pre- and post-test performance. It is not relevant, however, to comparisons of performance on scaffolded versus unscaffolded problems by the same student, or for the IRT analysis. Thus, key parts of our study are not affected by the lack of controls.

7 CONCLUSION AND FUTURE WORK

Our analysis goes beyond our initial paper in several ways. We confirm their findings that students improve significantly on algebra word problems after BS:A, but also go farther: based on the wrong answers to symbolic-form questions, we find students make certain (arguably more serious) errors less often on the post-test. This suggests that the impact of BS:A goes beyond what simply grading indicates, while also pointing to underlying skills that BS:A teachers and curriculum designers should drill more explicitly.

Our study also removes design-recipe scaffolds from half of the post-test questions. Students perform just as well (if not better) on the unscaffolded problems. Since some problems on the post-test were scaffolded, one should wonder whether students were copying scaffolding steps to solve unscaffolded problems. A sample of the actual test papers didn't reveal this behavior, but a future study should contain only unscaffolded problems.

Going forward, we want to explore whether errors that arose on the symbolic form questions manifest similarly in both the programming and mathematical contexts. We want to analyze our error data on domain and range questions (an analysis which did not fit into this paper). Overall, we want to continue to explore finer-grained impacts of the components of the BS:A design recipe, so that we might understand effective ways to integrate math and CS for the benefit of both disciplines.

To this point, our main instructional takeaway is that working with and relating multiple representations of functions—whether in math or computing—seems to positively impact students' abilities to express word problems in code and formulas. Articulating domain/range (types), writing examples (tests), and writing symbolic form (code functions) are important skills for programmers as well as algebra students. While we do not yet know the relative impacts of these skills on understanding and using functions, a multi-representational approach such as the BS:A design recipe seems a useful and lightweight method for developing them.

ACKNOWLEDGMENTS

We thank all the teachers who provided us data. We are deeply grateful to Leigh Ann DeLyser for suggesting and helping us set up the IRT. We are grateful to Peter Hahn for his outstanding work on the data analysis. This work is partially supported by the US NSF.

REFERENCES

[1] Bootstrap:Algebra [n. d.]. The Bootstrap:Algebra Curriculum. http://www.bootstrapworld.org/materials/courses/algebra. ([n. d.]).

[2] J.D. Bransford and D Schwartz. 1999. Rethinking transfer: A simple proposal with multiple implications. In *Review of Research in Education*. Vol. 24. American Educational Research Association, 61–100.

[3] N. T. Heffernan and K. R. Koedinger. 1998. A developmental model for algebra symbolization: The results of a difficulty factors assessment. In *Proceedings of the Twentieth Annual Conference of the Cognitive Science Society*. 484–489.

[4] National Council of Teachers of Mathematics. 2016. Should mathematics course requirements for high school graduation be satisfied by computer science courses? (an NCTM Position Statement). Available online, last accessed Feb 22, 2017 at http://www.nctm.org/Standards-and-Positions/Position-Statements/Computer-Science-and-Mathematics-Education/. (Feb. 2016).

[5] D Perkins. 2009. *Making Learning Whole*. Jossey-Bass, San Francisco.

[6] Peter J. Rich, Keith R. Leatham, and Geoffrey A. Wright. 2013. Convergent cognition. *Instructional Science* 41, 2 (March 2013), 431–453.

[7] E Schanzer, K Fisler, S Krishnamurthi, and M. Felleisen. 2015. Transferring skills at solving word problems from computing to algebra through Bootstrap. In *Symposium on Computer Science Education (SIGCSE)*. 616–621.

[8] Geoff Wright, Peter Rich, and Robert Lee. 2013. The Influence of Teaching Programming on Learning Mathematics. In *Society for Information Technology & Teacher Education International Conference*.