Presenting CS Concepts through Multiple Representations to Engage African-American Elementary School Children

Olivia M. Nche
School of Computing
Clemson University
Clemson, SC, USA
oncheey@g.clemson.edu

Elizabeth Colbert-Busch
Business Development
Clemson University
Clemson, SC, USA
ebusch@clemson.edu

Murali Sitaraman
School of Computing
Clemson University
Clemson, SC, USA
MSITARA@clemson.edu

Victor B. Zordan
School of Computing
Clemson University
Clemson, SC, USA
vbz@clemson.edu

***Abstract*— This experience report presents elements of an outreach program to elementary school children to broaden participation in computing. The program is based on a unique multi-faceted curriculum that facilitates the presentation of abstract computer science (CS) concepts within a summer camp setting. The curriculum exposes the same abstract content to children using different representations and helps them to learn and practice the concepts using a "hands-on" approach. Further, use of multiple representations supports the learning needs of diverse learners, considering the grades and ages of the participants. Specifically, we present CS concepts to students in three formats: 1) using motivated real-world and everyday examples, 2) elementary graphical programming, and 3) a custom video game designed to test and exercise concepts in a fun environment. Pretest/posttest analysis show promising trends, including positive changes in attitudes and learning of computational thinking and coding concepts.**

***Keywords*— broadening participation, integrative approach, computational thinking, problem solving**

I. Introduction

Research on learning with representations has shown that multiple representations can provide unique benefits when learning complex new ideas and that learners can enhance their performance if they interact with an appropriate representation [1]. Multiple representations can offer opportunities for repetition which can help to reinforce concepts. Further, multiple representations can also accommodate different learning styles and reach a broader set of learners. This paper describes how the idea of multiple format representations can be employed in the development of a curriculum for teaching CS concepts within the context of a summer camp. The curriculum is designed to give students ample opportunities to learn abstract concepts.

A lack of interest or proficiency in CS subjects among students is a prevalent problem in many school districts, yet early intervention can help to improve students' self-efficacy and increase motivation. This deficiency is more predominant among minorities, including African American and Latino students, who are often largely underrepresented in computing [2], [3], [4]. Hence, it is becoming increasingly critical to research and experiment with effective methods of engaging students of all ages, especially minorities and traditionally excluded groups [3]. The target students for our experiment are all African Americans and are amongst the least likely to have access to and benefit from CS without intervention. Considering the ages of our population and the notion that CS concepts tend to be abstract, we employed a novel multi-faceted approach which borrows from the principles of multiple representations. The curriculum also aims to highlight the benefits of learning CS concepts to the students so that they can see how computing can impact their lives. So in part to achieve this goal, the curriculum proposed also incorporates a "STEM talk" element, which exposes students to STEM career possibilities.

The rest of the paper is organized into the following sections: Section 2 discusses related work. Section 3 describes the experimental set up. Section 4 details the curriculum and methodology. Section 5 contains results of qualitative and quantitative assessments, and analysis. Sections 6 and 7 contain a discussion and a conclusion respectively.

II. Related Works

In the experiment of Ainsworth, pre-algebra students learned about functions using a curriculum that presented the information in multiple formats [1]. The transition from arithmetic to algebra is especially difficult for some students probably because the concept of functions, which is a central topic in pre-algebra mathematics, is typically presented in an abstract format rather than in a concrete context. Post-test results from this research indicate that students who were taught using multiple formats performed better at solving word function problems than their counterparts who did not receive this treatment.

Geometry is also another topic that is difficult for students to grasp due in part to the abstract method of presentation. Wong et al. used a computer-assisted learning environment called *Mr. Geo* with the help of multiple representations, to aid students learn theorem proving [5]. The representations used included a problem description, a static figure, a dynamic geometry figure, a formal proof and a proof tree. Their empirical results indicated that students found this method helpful in learning geometry proofs. Multiple representation of concepts in this case was more than the context of the instruction—a gaming environment, a lecture environment, and a problem-based environment. It also included mathematical approaches to representing the same relationship in different ways. However, the emphasis here is the fact that multiple representations can help students learn difficult concepts. The above-mentioned works are all math related and multiple representations have been widely used in math. The curriculum for this exploration employs multiple representations for computer science concepts building the hypothesis that, like math concepts, computer science tends to be abstract in nature and so the approach may transfer. This work is therefore similar to the mentioned works in that it also attempts to impute knowledge of difficult and abstract concepts to students by presenting it in multiple formats.

With respect to the difficulty associated with novice programming, some studies suggest that students who complete introductory programming courses are not as competent at developing computer programs to solve straightforward problems as might be expected. Prior work indicates that students may lack an understanding of fundamental programming concepts and that learning to program is difficult for many students [1], [6], [7]. As a result, CS educators have tried a variety of instructional methods to assist beginning programmers [7]. Furthermore, results from other studies to discover why some students have difficulty learning to program point towards the fundamentals of programming [8]. From the findings of these studies, we can conclude that experimenting with teaching methods that encourage attention to fundamental concepts of computing might help alleviate some of the problems encountered by novice programmers. This outreach therefore focuses on CS concepts that are applicable to all programming languages without burdening the students with learning the syntax of any particular language. To this end, we use a pseudo code in our lectures and video game [9], complemented by visual programming on SNAP [10].

III. EXPERIMENTAL DESIGN

The subjects of the present project are amongst those who are largely excluded when it comes to opportunities to acquire knowledge in computer science. So our intervention served as an opportunity to expose them to CS concepts in particular and include them in the quest for computing knowledge in general. The students were enrolled in the Metanoia community-centered summer program (Freedom School). The school provided us with the opportunity to recruit participants while our project served as part of the enrichment activities for the students. Buses were available to bring the students to a nearby facility with a custon computer classroom, conveniently located less than 10 minutes away from the Metanoia Freedom School. The facility serves as the Clemson University campus in Charleston and this is where our summer camp took place. There were a total of 40 students when the camp started and 30 students when it ended (14 boys and 16 girls). All the students were African American. There were two cohorts of students. The first cohort comprised of students from the third and fourth grades, while the second cohort included students from the third and fifth grades. The camp for each cohort involved meeting twice a week for four hours. The students took a pretest, a pre-survey, a posttest and a post survey [11]. Some questions tested computational thinking and were designed in the form of Bebras' challenge style questions [12]. Other questions were similar to code snippets used in the lectures as seen in Fig 1 and 2). The pretest and the posttest were designed to assess if our intervention helped the students learn CS concepts and positively influence their attitude towards computing. The posttest was very similar to the pretest. We maintained the same questions in the posttest but changed the numbers and names involved. This was done to discourage memorization of answers although the chances were slim, considering that this was a five-week summer camp. The tools we used for the experiment include SNAP and the Unity 3D gaming engine. SNAP is a blocks-based programming tool from University of California, Berkeley, which is an extension of Scratch [10]. Unity 3D is a gaming engine that allows users to create virtual applications.

IV. CURRICULUM AND METHODOLOGY

In this project, basic computing concepts—sequencing, variables assignments, operators (arithmetic and relational), conditionals, iteration—are introduced beginning with day-to-day examples with which students are familiar and can grasp. These are later reinforced through multiple graduated exposures. Following this methodology, we prepared a stack series of modules that are each designed to offer the students motivation, by way of a provocative STEM-talk introduction, followed by multiple exposures to a single focal CS concept. In the end, we opted for a two-day module length due to practical logistics. The students were prepped at the beginning of each module, by listening to the motivational career-oriented STEM talk. In this talk we emphasized the connection between excellence in STEM and a great career in the future. This was intended to make students aware that they had great opportunities before them if they focused on STEM and coding. We brought in expert guest speakers from various fields to deliver the talks. The speakers typically talked about the path to their current career, what they do in their jobs and specifically how they use computers to do their jobs. In every discussion, we ensured that a parallel was drawn between what the students were learning in the camp, and the different career options they were exposed to.

This was closely followed by content exposure of a specific CS concept in a traditional lecture style. Then we extrapolated

this concept to SNAP to showcase how it is explored in a visual block-based programming interface and then categorically merged the two together. This segment was followed by a game session in which the students got to explore the same content in a video game [9]. Another interesting feature which we incorporated into the curriculum was a STEM activity segment. During this session, the students interacted with various small robots which are programmable through a simple graphical user interface. This further emphasized the same CS concept that was being explored in SNAP. Finally, the students finished each module with a creative period that let them explore the new concept (with previous ones) to make designated programs.

In its construction, the module is designed to provide: 1) scaffolding within and across modules with repetitive cumulative concept reprises; 2) multiple and varied exposures of each CS concept; and 3) frequent opportunities for free experimentation. Considering the poor reading skills of some of the students, we added a short reading period to help the students who needed to improve their reading skills. During this time, the students watched an excerpt of a movie and then took turns reading the corresponding script out loud. The summer camp lasted five weeks. The first and last weeks were spent on introduction and conclusion. The introductory week was dedicated to administering a pretest and pre-survey. It was also used to introduce SNAP, and general concepts about computers, including key terms such as "algorithm", "program", "programming languages" and so on. In this phase, we used "making a peanut butter and jelly sandwich" as an example to emphasize the importance of precision in commands, sequence and order in executing computer programs.

The middle three weeks were devoted to three modules, each focusing on different concepts. During the concluding week, the students took post assessment tests, created their final projects, and participated in a graduation ceremony. The ordering of computing concepts roughly follows the learning trajectory for school children detailed and adapted from the Beauty and Joy of Computing course developed at the University of California, Berkeley [10]. Inspired also by the work of Rich, et al. [13] in formulating learning trajectories for students from K-8 grades, we created a taxonomy of concepts that influenced the order in which our curriculum was executed.

A. Module 1: Focus on Variables, Assignments, Operators

In this module, the first 35-minute block was dedicated to an interactive talk, which allowed the students to peek into programming using the terminal and webpage designing. After the break, we introduced the concept of assignments, operators and variables to the students using everyday examples like a piggy bank, which is a single holder that can contain a variable amount of money. This portion constituted the first presentation in this module (Fig. 1). In the second presentation, we explored these concepts in SNAP. We initially focused on creating, naming and using variables by dragging and dropping relevant blocks on to the script area. We then used several

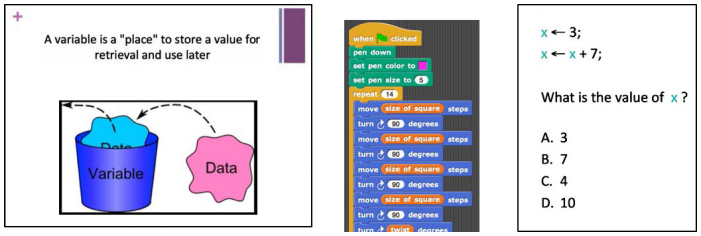

Fig. 1. Multiple representations Module 1: Left, lecture slide examples; Center: SNAP program; and right, video game pseudocode

SNAP examples to help the students understand variables, assignments and operators. The last portion of the class was dedicated to hands-on activities in SNAP and STEM. Day 2 started off with a concept reprise of variables, assignments and operators, followed by further extended examples in SNAP to solidify the topic.

Then the students spent the rest of the afternoon playing the video game developed in Unity 3D. The various game levels were based upon the content and concepts that were expounded upon earlier. It contained questions in the form of pseudocode snippets, covering the concepts taught in the respective modules. In the discussion in [9], benefits of the educational videogame, including its ability to pinpoint student obstacles and encourage student practice with concepts are noted. This was the third format in which the same CS concepts were presented (Fig. 1). During the last 20 minutes of this segment, the students who needed help in reading were released to participate in the reading activity for that week. The subsequent modules followed this same pattern but focused on differed in content.

B. Module 2: Focus on Conditionals

For Module 2, the students listened to a talk about LEDs in relation to how they are programmed and used. In the first presentation of this module, we introduced students to conditionals using examples like "if it is raining, then take an umbrella" to foster the concepts of decision making based on some condition (Fig. 2). They proceeded to create artifacts on SNAP using conditionals in the second presentation and for the third presentation students played the video game levels dedicated to conditionals.

C. Module 3: Focus on Loops

The third module focused on repetition. The module began with a talk from an architect. Along with other things, she showed the students how house plans were drawn in the past manually and how they are done today with the aid of computer applications. This allowed the students to see how critical computing has become in this career field. The first presentation followed this discussion. In this phase, the students were introduced to repetition and how computers handle them using loops and repeat patterns. In phase two, we used SNAP to foster the concept of loops and let the students practice with several examples. For the third presentation, as

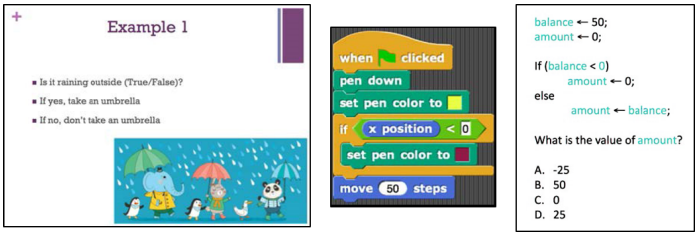

Fig. 2. Multiple representations Module 2: Left, lecture slide examples; Center: SNAP program; and right, video game pseudocode

usual, the students played the video game levels dedicated to loops.

V. IMPACT

Measuring student learning is critical to any educational effort [14]. Therefore, the main objective of our assessment was to determine the extent to which our instructional intervention impacted student learning. We used three forms of evaluation to accomplish this. The first was informal, yet revealing. The second was quantitative and aimed at assessing logical reasoning with core concepts. The last one was qualitative and an evaluation of students' attitudes.

A. *Engagement and Learning*

We assigned a final project to the students in SNAP and gave them a choice—to build a graphic or to build a conversational agent. We gave them loose guidelines but required them to implement at least three of the concepts they had learned. We encouraged the students to take the projects in their own direction. This was done partly to encourage student involvement and creativity. We observed that they were focused, engaged, entertained, and eager to collaborate and share their work.

A female student built a set of functions, each of which made a unique shape and color. Then she proceeded to use them like a custom tool palette to produce artwork; see Fig. 3. While her program elements were simple, she succeeded in expressing herself in a creative way by constructing an art piece out of coded elements. Another pair of students used a feature to put two sprites on the screen (something not covered in the class) and systematically unfolded a story through a conversation between the two characters. The story might not have made much sense, but they had a tremendously good time evolving the storyline and sharing it with their peers. Their final projects showed evidence of the application of CS concepts.

B. *Quantitative Assessment*

To assess the impact on students' ability to reason logically in the presence of computational concepts, we employed a pre- and posttest as described in the experimental design section. The questions were based on the content delivered to see if our intervention had any impact. A total of 30 students were present for both tests: There were 10 in the 3rd grade, 11 in the 4th grade and 9 in the 5th grade. A majority of all the students scored higher on the posttest (Fig. 4). A two sample

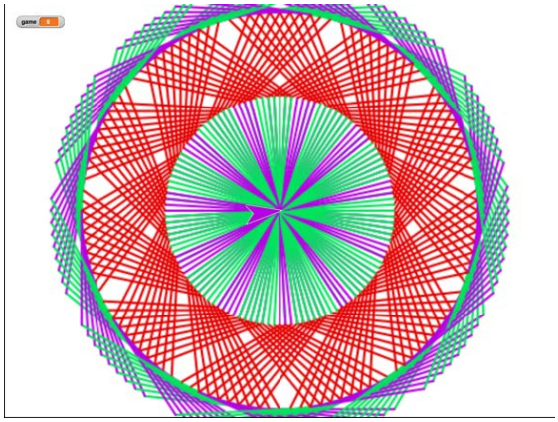

Fig. 3. Sample of student outcome

z test shows that students' posttest scores were statistically significantly higher than the pretest scores. z = -2.35, p = .001. We conducted a one-tailed t test to see if the difference between the pre- and posttest scores of the students in each grade and between the boys and girls was significant. The p values are as follows: fifth grades: p = 0.004, fourth grade: p = 0.0001, third grade: p = 0.5. Girls: p = 0.088, boys: p = 0004.

C. *Qualitative Assessment*

To assess the impact on student attitudes towards computing and to understand how much they enjoyed the camp, we administered surveys modeled on those used by the Georgia Tech Institute for Computing Education in their camps [11]. Two items were of particular interest to us as we embarked on this coding camp venture: 1) we wanted to know if our intervention caused the students to become more interested in computing; and 2) if it had any effect on their self-efficacy. Overall, we found that more students felt that people like them can do computing. We also saw that slightly more than half of the participants think that the camp increased their interest in computing. A majority found the camp to be fun.

VI. DISCUSSION

The results from our assessments show general cognitive and affective improvement for most of the participants. The cognitive assessment shows that there was an overall improvement from the pre- to post-test results for most students (Fig.4). Based on the p values of the statistical analysis of both the z and t tests, we can conclude that the results of our assessments are significant. All the participants from the 3rd to the 5th grades were exposed to the same treatment in order for us to assess how the students in the various grades would respond. We noticed that the greatest improvement in test scores was observed amongst the 4th graders, followed by the 5th graders. The 3rd grade students made some progress but not as much as the students in the other grades. We think that this is as a result of the age difference between the students. We noted however that there was a greater

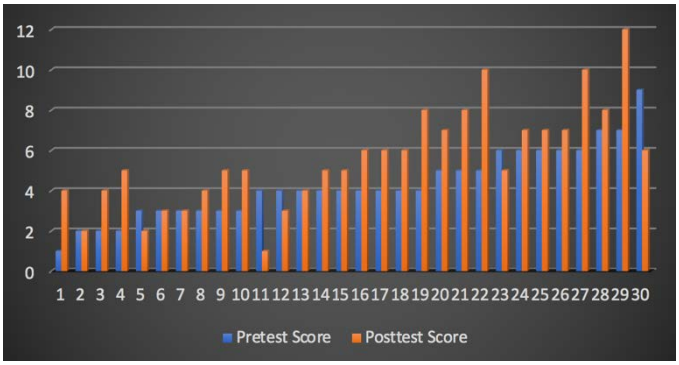

Fig. 4. Pretest vs. Posttest scores for all students

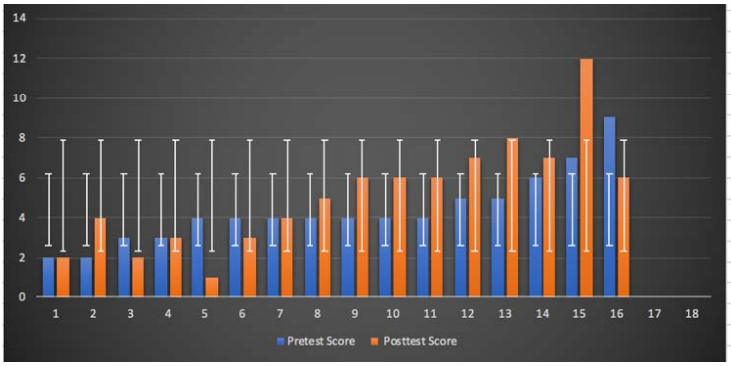

Fig. 5. Pre- and post-test scores for girls

difference between the tests scores of the students in the 3rd and 4th grades than there was between the 4th and 5th graders' scores. We also noticed (from the statistical analysis) that the boys demonstrated slightly higher improvements from pre to post-test than the girls (Fig. 5 & 6), although the student who scored the highest on the post-test assessment and our overall best student was female. This particular student was just exceptional and thoroughly enjoyed the camp. She distinguished herself in attitude and academic excellence.

In this project, we can only report the trends that we observed in the performance of the students from the data we collected. However, we are unable to provide sufficient explanations for the trends and what caused them. This will be a question for future work.

The students demonstrated in their final projects that they had indeed gained knowledge of CS concepts. The fact that they were able to apply at least three of the concepts learned during the camp to create a final artifact shows that they had at least learned a subset of the core concepts. We also noticed that some of the students took off on a tangent with their projects as they explored other aspects of SNAP that were not covered in the class. For example, the students who created the conversational agents did not only use concepts learned in class but also applied them in another context not necessarily covered during the lectures. This is further evidence that the students had gained content knowledge and were excited enough to push the bounds of the camp curriculum driven by their own creativity.

When operating a summer camp, one expects that there would be possible logistical unforeseen circumstances that could have impact on the camp. In our case, intermittent transportation failures caused us to make adjustments to the module on loops (Module 3). Similarly, we started off the camp with about 40 students but only retained 30 in the end. This was because we had no control over the circumstances surrounding the students' ability to come or not since we were working with another entity that facilitated the availability of the students. If a student exited their program in the middle, then we automatically lost that student which was the case with a few students.

The fact that this was in the summer also meant that family traveling time could interfere with student availability. For future references, providing some sort of incentive for the students to stay until the end is highly recommended.

The teachers and mentors who participated in this project served as facilitators in the hands-on sessions when the students created artifacts using the knowledge acquired in the respective modules. To this end, elementary-school teachers were hired from nearby schools. This project also served therefore as a professional development opportunity for the teachers, as they also acquired knowledge in CS concepts. The parents did not play any active role in this project but were present on graduation day to see what their children had accomplished.

VII. CONCLUSION

This paper has presented an approach that uses multiple representations to help African American elementary school children learn and enjoy computational concepts. Considering the fact that CS concepts tend to be abstract in nature and that abstract content can be difficult for students regardless of their ages, we attempted to overcome this challenge by employing a curriculum which facilitates the presentation of CS concepts in multiple representations. Another rational for our project is the fact that studies have shown that some of the problems associated with novice programming are related to a lack of understanding of CS fundamentals.

In this project, we therefore focused on exploring fundamental CS concepts which cut across different languages, in an effort to expose them to students and jump start their journey in computer science. The results from the qualitative assessments, the pre- and posttest show improvements in the students' performance. We conclude, based upon them, that our strategy was useful. We also observed diverse levels and trends of improvements across the different grades. We have no explanations for some of the patterns we observed except for the fact that age plays a role in a student's ability to grasp certain concepts. These ideas will be further explored.

Lastly, although this particular endeavor has focused on African American children, future projects will include other groups.

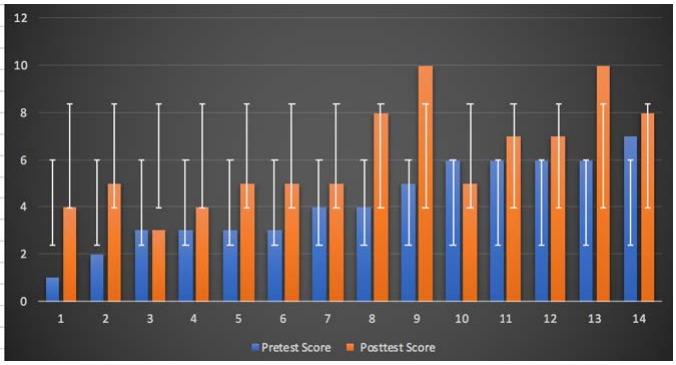

Fig. 6. Pre- and post-test scores for boys

ACKNOWLEDGMENTS

This research is funded in part by grants from Boeing South Carolina, Blackbaud, and NSF (1738760).

REFERENCES

[1] Shaaron Ainsworth. The functions of multiple representations. *Computers & education*, 33(2-3):131–152, 1999.

[2] Mark Guzdial, Barbara J Ericson, Tom McKlin, and Shelly Engelman. A statewide survey on computing education pathways and influences: factors in broadening participation in computing. In *Proceedings of the ninth annual international conference on International computing education research*, pages 143–150. ACM, 2012.

[3] Mark Guzdial and Barbara Ericson. Georgia computes!: an alliance to broaden participation across the state of georgia. *ACM Inroads*, 3(4):86–89, 2012.

[4] Jane Margolis, Jean J Ryoo, Cueponcaxochitl DM Sandoval, Clifford Lee, Joanna Goode, and Gail Chapman. Beyond access: Broadening participation in high school computer science. *ACM Inroads*, 3(4):72–78, 2012.

[5] Wing-Kwong Wong, Sheng-Kai Yin, Hsi-Hsun Yang, and Ying-Hao Cheng. Using computer-assisted multiple representations in learning geometry proofs. *Journal of Educational Technology & Society*, 14(3), 2011.

[6] Teemu Sirkiä and Juha Sorva. Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. In *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pages 19–28. ACM, 2012.

[7] Wanda M Kunkle and Robert B Allen. The impact of different teaching approaches and languages on student learning of introductory programming concepts. *ACM Transactions on Computing Education (TOCE)*, 16(1):3, 2016.

[8] Simon. Assignment and sequence: Why some students can't recognise a simple swap. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, pages 10–15, New York, NY, USA, 2011. ACM.

[9] Olivia M. Nche, John Welter, Megan Che, Eileen T. Kraemer, Murali Sitaraman, and Victor Zordan. Combining gaming, cs concepts, and pedagogy. In *Proceedings of the 4th RESPECT International Conference*. IEEE, 2019.

[10] Dan Garcia, Brian Harvey, and Tiffany Barnes. The beauty and joy of computing. *ACM Inroads*, 6(4):71–79, November 2015.

[11] Institute for Computing Education at Georgia Tech. Pre and post surveys. 2011.

[12] Valentina Dagienė, Gabrielė Stupurienė, and Lina Vinikienė. Promoting inclusive informatics education through the bebras challenge to all k-12 students. In *Proceedings of the 17th International Conference on Computer Systems and Technologies 2016*, pages 407–414. ACM, 2016.

[13] Kathryn M Rich, Carla Strickland, T Andrew Binkowski, Cheryl Moran, and Diana Franklin. K–8 learning trajectories derived from research literature: sequence, repetition, conditionals. *ACM Inroads*, 9(1):46–55, 2018.

[14] Allison Elliott Tew. *Assessing fundamental introductory computing concept knowledge in a language independent manner*. PhD thesis, Georgia Institute of Technology, 2010.