Demonstrating the Ability of Elementary School Students to Reason about Programs

Ashish Aggarwal
Computer & Info. Science
& Engineering
University of Florida
Gainesville, FL 32611
ashishjuit@ufl.edu

David S. Touretzky
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213
dst@cs.cmu.edu

Christina Gardner-McCune
Computer & Info. Science
& Engineering
University of Florida
Gainesville, FL 32611
gmccune@ufl.edu

ABSTRACT

Over the last decade, CS Education researchers have developed different curricula, resources, and strategies to foster computer science learning in K-12 education. However, there is a lack of research about how elementary school students develop the ability to reason about programs. Reasoning about programs consists of a student's ability to read, write, debug, trace, and predict program behavior. This paper presents results from a think-aloud study of fourth and fifth grade students learning to program in Kodu. The goal of this study was to track students' understanding of how Kodu interprets and executes rules of a program. To understand students' reasoning of program execution, we explicitly taught them the Laws of Kodu computation which govern the decision making and execution process of Kodu rules. We collected students' responses on pre- and post-assessments, and we conducted think-aloud interviews with students where students explained their answers to assessment questions. We found that explicitly teaching students how Kodu rules are interpreted significantly improved their ability to understand the execution of programs and to explain program behavior. The results of this study provide insight into how elementary school students reason about simple programs, and how this ability can be scaffolded.

ACM Reference format:

Ashish Aggarwal, David S. Touretzky, and Christina Gardner-McCune. 2018. Demonstrating the Ability of Elementary School Students to Reason about Programs. In *SIGCSE '18: 49th ACM Technical Symposium on Computer Science Education, Feb. 21–24, 2018, Baltimore, MD, USA.* ACM, NY, NY, USA, 6 pages. https://doi.org/10.1145/3159450.3159488

1 INTRODUCTION

Visual programming environments such as Alice [4] and Scratch [13] have been successful in introducing programming to younger students and overcoming the barriers inherent in learning text-based programming languages. Often curricula associated with visual programming environments focus on engaging students in the development of creative artifacts as a mechanism for teaching CS concepts to students [13]. These programming environments scaffold students' abilities to learn programming and support artifact design by minimizing the syntactical complexities of programming and hiding how programs are compiled and executed [7] [12].

As K-12 computer science teaching becomes more common, the number and range of programming environments will continue to expand. Thus, there is a growing need to help students and teachers transfer their knowledge of programming across these environments. One way to support programming knowledge transfer across environments is to develop students' ability to reason about programs. Reasoning about programs requires students to develop the ability to read, write, debug, mentally simulate (trace), and predict program behavior [2] [8] [9]. Underlying this ability is an inherent understanding of how programs are compiled/interpreted and executed. Mastery of these skills will help students increase their programming proficiency and better understand how programs work. Despite the importance of cultivating program reasoning ability, there is a lack of research on the development of students' understanding of how computers execute program instructions.

In this paper we describe the results of a think-aloud study conducted to track the development of elementary students' program reasoning ability in Microsoft's Kodu Game Lab. The findings describe students' abilities to read programs and to explain and predict program behavior. This paper aims to address the gap in the literature about how elementary students reason about programs.

2 NOVICE PROGRAMMERS

Over the past three decades, CS Education researchers have studied the skills novice programmers need in order to become proficient in programming and the challenges novice programmers have in understanding and reasoning about programs [3] [11] [15]. Several researchers have suggested that composing and coordinating different pieces of code and "putting the pieces together" is a major problem for novice programmers [17][18]. Thus, Deimel [5] argued that code

ACM ISBN 978-1-4503-5103-4/18/02…$15.00
DOI: https://doi.org/10.1145/3159450.3159488

reading is as important as code writing. Sheard et al. [14] found that a student's ability to explain programs positively correlated with their ability to write code. Moreover, they found that program explanation plays an important role in the development of novices' conceptual understanding of program construction. Research by Lister et al. highlighted that students' code tracing ability is dependent on their code explanation ability and confirmed that "students who cannot trace code usually cannot explain code" (p. 161) [10]. Conversely, they found that "students who tend to perform reasonably well at code writing tasks have usually acquired the ability to both trace code and explain code" (p. 161) [10].

Perplexed by the challenges that novice programmers face, Soloway [16] suggested that they need to be taught effective reasoning strategies and that "learning to program amounts to learning how to construct mechanisms and how to construct explanations" (p. 851) [16]. Most recently, Guzdial [7] suggested that development of a robust "notional machine", e.g, mental model of how programming environments work [6], will help students understand how programs work and improve their ability to read, write, trace, and understand programs. This paper aims to provide examples of how elementary school students reason about programs in a visual rule-based language, Kodu.

3 KODU

3.1 Microsoft's Kodu Game Lab

Kodu Game Lab is a visual programming language made specifically for 3D game development. It is designed to be accessible to children and enjoyable for anyone. It provides students with a 3D world to visualize the behavior of their programs and a rule editor to design and rapidly iterate on their programs. Kodu uses WHEN-DO semantics where the WHEN-part represents the predicate, or the condition and the DO-part represents the action. Kodu rules are conditional statements which are represented in sequences of tiles, e.g., objects, perceptions, and actions. Fig. 1 shows the first design pattern students learn in the Kodu curriculum which is Pursue and Consume (P&C). This is the 'Hello World' program of Kodu. The P&C design pattern programs a character to move toward the closest object that satisfies the rule (e.g., "WHEN see apple DO move toward"), and to consume it upon contact (e.g., "WHEN bumped apple DO eat it") [22].

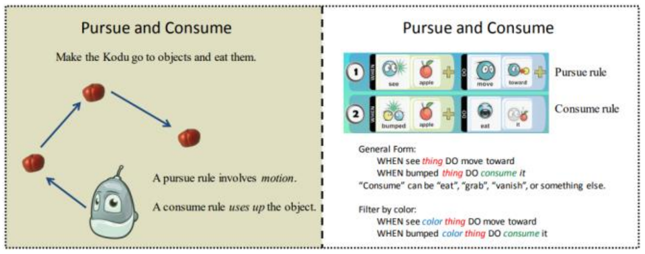

Pursue and Consume
Make the Kodu go to objects and eat them.
A pursue rule involves *motion*.
A consume rule *uses up* the object.
Pursue and Consume
WHEN
see
apple
DO
move
toward
Pursue rule
bumped
apple
eat
it
Consume rule
General Form:
WHEN see *thing* DO move toward
WHEN bumped *thing* DO *consume it*
"Consume" can be "eat", "grab", "vanish", or something else.
Filter by color:
WHEN see *color thing* DO move toward
WHEN bumped *color thing* DO *consume* it

Figure 1: Flashcard showing the Pursue and Consume idiom/design pattern.

Kodu simplifies learning to program by narrowing control structures available to students to evaluation of conditional statements. Mastery of Kodu programming thus can help students to better understand conditionals and if-then and while programming constructs. Moreover, since Kodu is an event-driven language, students are exposed to even-driven concepts of programming in a simplified manner [19][23].

3.2 Kodu Curriculum & Laws of Kodu

Building off simple Pursue & Consume programs, the Kodu curriculum developed by Touretzky [20] focuses on helping students to understand, reason about, and predict program behavior through explicitly teaching students the first three Laws of Kodu computation. The curriculum, provides students with visual representations of the Laws of Kodu on refrigerator magnets, animated videos, and activity worlds to help students explore and reason with the Kodu laws.

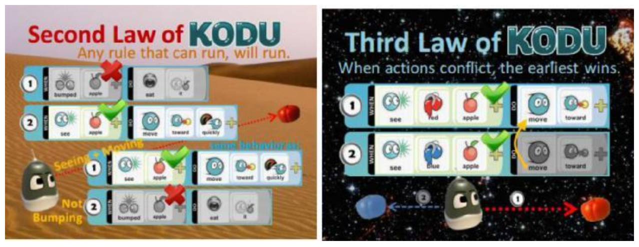


Figure 2: The refrigerator magnets showing the 2nd (left) and the 3rd (right) Laws of Kodu

Overall, the graphics on the refrigerator magnets of these laws are designed to build students' understanding of how these laws will be applied in Kodu [20]. The first Law of Kodu is one of the simplest ways we introduce students to program behavior and prediction. The 1st Law of Kodu says that "*Each rule picks the closest matching object.*" In the context of a pursue and consume program, this mean that if the kodu can see more than one apple, it will pick the closest one.

The 2nd Law of Kodu (Fig. 2-left) explains how the execution of rules takes place in Kodu. It says, "*Any rule that can run, will run*" which means that rules in Kodu run in parallel. In Kodu, if a rule's WHEN-condition is true and evaluated, it will run. This is because every rule in Kodu is evaluated 50 to 100 times per second to check if it can run or not [21]. The 2nd Law of Kodu helps students learn how to trace programs that have parallel execution, which is similar to Scratch program where multiple objects/events execute in parallel. The 3rd Law of Kodu helps students resolve conflicts that arise between rules. Conflicts occur when their WHEN-conditions are simultaneously true but their corresponding DO-action statements are incompatible. When rules conflict in this way, students are expected to use the 3rd Law of Kodu (Fig. 2-right) which says, "*When actions conflict, the earliest wins*" to understand which action will run. Thus, when conflict arises, the rule which is earlier (lower-numbered) will be executed. The 3rd Law of Kodu help students develop skills for reading programs and recognizing when rules might conflict and then use the law to interpret program execution and behavior while tracing the program.

Through Touretzky's [20] Kodu curriculum, students initially discover the Laws of Kodu by reasoning about the behavior of kodu characters that are executing simple pursue and consume programs in various Kodu worlds [1]. For example, a program facilitator might ask students to program a kodu character in a simple world filled with apples and ask students to guess which apple the kodu will pursue and why. The facilitator might then ask students to move the kodu around and to predict which apple the kodu will eat. Overtime, students start discovering patterns in how the character selects which apple to eat first. Then, the students are given the relevant Law of Kodu magnet and discuss a range of applications of the laws through watching animated instructional videos for the 1st and 2nd Laws. As students practice applying the laws they improve their ability to understand the execution of programs and their ability to explain program behavior. Our goal by the end of the curriculum is for students to be able to refer to, state and apply the laws correctly.

4 RESEARCH STUDY

In this paper, we focus on demonstrating how students use laws to reason about simple 2-4 rule Kodu programs. Critiques of prior Kodu work focus on the age-appropriateness of teaching students to reason about programs. Thus, in this paper we explore the following research question: To what extent are elementary students able to reason about programs? Consistent with a constructivist theory of learning [24], we believe that all students are capable of learning to program and reason about programs when given appropriate tools. Therefore, we hypothesize that students who understand the laws will be able to refer to, state and apply the laws correctly when explaining program execution and behavior. Thus, in this study, we examine the ability of students to read code, explain the meaning and the behavior of the code and trace it. Based on this evidence we evaluate the overall state of elementary school students' reasoning ability.

4.1 Study Design

This study has two main components: (1) an instructional intervention where we taught students the Kodu curriculum and observed their reasoning during class sessions and (2) a think-aloud interview where students either retrospectively explained their reasoning on assessments or verbally walked through their reasoning as they solved problems with the interviewer. The results of this study will allow us to (1) evaluate the influence of explicit teaching of laws in developing students' correct reasoning, (2) gain a better understanding of students' ability to predict and explain program behavior, and (3) examine the overall capability of students to reason about programs.

4.1.1 Instructional Intervention-Session Overview. In this study, we created a 4-session curriculum adapted from Touretzky's [20] Kodu curriculum focused on scaffolding students' learning to reason about 1-4 rule program variations of the pursue and consume design pattern. The intervention was conducted over 4 consecutive weeks. Program participants attended one 90-minute instruction session each week. Each session introduced one of the first three Laws of Kodu and evaluated students' reasoning based on their understanding of the law and its application in combination with previous laws.

Session #1, introduced students to the semantics of Kodu rules and the 1st Law of Kodu. In the think-aloud study, we measured students' ability to use the 1st law to simulate and predict program behavior. Session #2 introduced students to the 2nd Law of Kodu. Session #3 introduced students to the 3rd Law of Kodu. By the end of the third session we expected students to be able to recognize and use the Pursue and Consume design pattern. We also expected students to be able to state the laws, recognize appropriate times to apply the laws, reference laws when reasoning about the execution of a program, use the laws to mentally simulate 2-rule and 3-rule programs, and predict program behavior. In session #4, students were asked to use the P&C design pattern to build a game of their own choice.

4.2 Methodology

We conducted four ninety-minute sessions after school with three groups of six participants each. In each session, participants completed paper-based pre & post assessments based on the content covered in the session. The assessments were focused on recognition of laws and idioms, students' understanding and simulation of the rules, and prediction of program behavior. After each student completed their assessment, students were asked to participate in a think-aloud interview. In each of the think-aloud interviews, students were asked to read the question aloud, to explain the meaning of the question, and then to reason about their answer. The interviewer asked students to explain why they chose an answer option or rejected other options. The interviewers were also instructional intervention facilitators, so students were comfortable in sharing their ideas.

4.3 Participants

Eighteen 4th and 5th grade participants were recruited from a local elementary school. Interested students were given flyers and completed interest forms to voluntarily sign up for the study. An information session was conducted for parents and students to provide them with details about the study and Kodu. We received parental consent and student assent for each student to participate prior to the start of study. The eighteen recruited students were divided into one of three groups based on their schedule availability. By chance, every group ended up having six students with four boys and two girls. Seven students were from fourth grade and eleven students were from fifth grade. 16 out of 18 students indicated that they had prior programming experience while the remaining two did not. All eighteen students indicated that they completed Hour of Code activities, while varying number of students indicated using other programming environments such as Code Monkey (n = 1), HTML and JavaScript (n = 5), Kodu (n = 1), Minecraft (n = 3), Python (n = 1), Scratch (n = 3), Vex Robotics or Lego (n = 3), Other (n = 3).

4.4 Data Collection and Analysis

Data was collected using pre-and post-assessments and think-aloud interviews. During session 1, students completed post-activity assessments and think-aloud interviews. During session 2 & 3, students were given pre- and post-assessments where they were asked questions on the focal Kodu law of the day. The pre-assessments allowed us to collect data on how students reason before they were taught the law and helped us understand the default reasoning patterns of students. Audio and video recordings of the think-aloud study were captured, transcribed, and analyzed by the researchers.

We qualitatively analyzed student responses to questions from the pre- and post-assessments and think-aloud interviews. We listened to the audio of all the participants' think-aloud interviews and then noted down three aspects of students' responses: (1) whether the participants correctly answered the questions; (2) evidence the participant referred to, stated, or applied one or more Laws of Kodu in solving the problem or explaining their solution to the problem; and (3) whether the participant used the laws correctly when explaining their reasoning. We then calculated the number of participants who correctly answered the questions, and the number of students who referred to, stated, and/or applied the laws. We transcribed audio from a representative sample of the think-aloud conversations between researchers and students to highlight how students exhibited their knowledge and application of the laws as evidence of their reasoning.

5 FINDINGS

Overall, our analysis of students' think-aloud interview data shows that students can refer to, state and apply the laws correctly when reasoning about programs. In this section, we will discuss evidence for this claim using examples from the think-aloud interviews that followed session 1, 2 and 3. These examples highlight how students directly referred to or stated the 1st, 2nd and the 3rd law while explaining their reasons for selecting answers when asked to predict the behavior of a program.

Q10. Look at the three kodus below. One of them is obeying the First Law of Kodu when it runs the following rule. Which one is it? Circle the <u>obeying</u> Kodu.

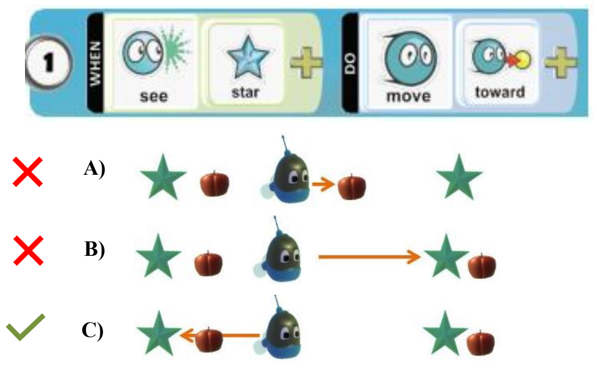


Figure 3: **Session 1, Q10 think-aloud question.**

5.1 Session 1 – 1st Law of Kodu

During session 1, students were asked questions related to the execution of simple pursue and consume rules. Below is an example from the think-aloud conversation where a student describes his/her understanding of the 1st law while solving Q10 (Fig. 3**).** Q10 asks students to consider the pursue rule provided and three possible scenario options and asks them to select the option where kodu is obeying the 1st Law of Kodu. Option C is the correct answer because kodu always goes to the closest matching object in the rule, which in this case is a star.

Interview Transcript 1: Student A explained the meaning of the 1st Law of Kodu

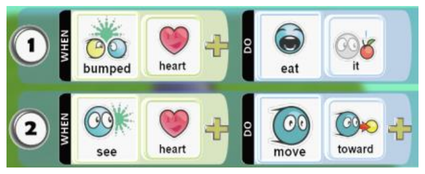
Context: Student indicates that the kodu will go to the closest object (star) because that is the "rule"
Interviewer: What rule [law] is that? Do you remember?
Student A: *"I don't remember, but I know what it means"*
Interviewer: So, what does it mean?
Student A: *"It means to always go to the nearest, well kodu is supposed to go to the nearest star, apple, heart whatever, so yes it is supposed to go to the nearest one not the farthest one"*

This transcript shows that Student A is able to correctly explain the 1st law and what it means. Student A is also able to generalizes the application of the law to multiple objects not just apples which is the example case used to teach students the 1st law and how it works. Overall, 13 out of 18 students marked the correct option C on Q10. During the think-aloud interviews 10 out of the 13 students directly stated or referred to the 1st law in their explanations. This indicates that students are able to refer back to and apply the 1st law. This transcript is representative of the capability of 10 out of 18 students to correctly reason about the choices they made when reading the Kodu programs.

5.2 Session 2 - 2nd Law of Kodu

5.2.1 Session 2, Think-aloud Question 3. During the session 2 think-aloud interview, students were given Q3, a reverse pursue and consume program (Fig. 4), and asked to consider "what will these rules do?" Q3 was designed to assess students' conceptual understanding of the program's behavior.

Q3. What can these rules do?



✓ **A) They can pursue and consume all the hearts**
✗ **B) They cannot do anything as the pursue rule is below the consume rule**
✗ **C) These rules make no sense**
✗ **D) They can do random stuff**

Figure 4: **Session 2, Q3 reverse pursue and consume think-aloud question.**

The correct answer to this question (Fig. 4) is option A- "they can pursue and consume all the hearts." This option is correct because the 2nd Law of Kodu says that "any rule that can run, will run." which suggests that the order of pursue and consume does not matter (Fig. 2, left). 14 out of the 16 students, marked the correct option A.

Interview transcript #2 provides Student B's explanation for choosing option A and shows the student directly referencing and stating the 2nd Law of Kodu while explaining their answer selection (Fig. 4).

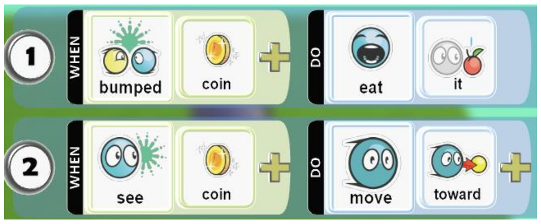
Interview Transcript 2: Student B directly referred to the 2nd Law of Kodu

Interviewer: Why did you mark option A?
Student B: "*because even though this [pursue rule] is below the top one [consume rule], it [pursue rule] can still follow that rule [consume rule] because that's the second rule [law] of kodu*"

This transcript shows that Student B is able to point to the pursue and consume rules individually, comment on their current order in the program, and attribute the working program to the 2nd Law of Kodu. This suggests that Student B knows that even though the order of the rules is switched in the program, it will still work as intended. Student B's response is consistent with 11 out of 14 students who also correctly answered this question and directly stated or referred to the 2nd Law of Kodu while explaining their reason for selecting option A. The remaining 3 explained their reasoning by correctly applying 2nd Law of Kodu but did not explicitly refer to the 2nd Law of Kodu.

5.2.2 Session 2, Think-aloud Question 5. In Q5 (Fig. 5), a subsequent question about reverse pursue and consume rules was given in context of a kodu world and students were asked to predict the kodu's behavior. The correct answer is option C which states that "kodu will pursue and consume all the coins as the order of pursue and consume does not matter." 15 out of the 16 students answered correctly by marking option C.

Q5. In the world above, what would the Kodu do with the given rules?



✗ **A) Kodu will not move as the consume rule is above the pursue rule**
✗ **B) Kodu will bump the coin first and then pursue the nearest coin**
✓ **C) Kodu will pursue and consume all the coins as the order of pursue and consume does not matter**
✗ **D) Kodu will do random stuff**

Figure 5: **Session 2, Q5 think aloud question**

Interviewer Transcript 3: Student C directly referred to the 2nd Law of Kodu and explained it

Context: At the beginning of a week two interview
Interviewer: What did you learn today?
Student C: "*I learned today that it doesn't matter which rule is first, it matters which rule works, so if the first rule is WHEN bump apple eat it and the second rule is WHEN see apple move toward, the second one will work instead of the first one*"
Interviewer on Q5: What is the correct answer?
Student C: "*I think it is C...because it is the Law of Kodu number 2, the second Law of Kodu is whatever can run will run so it (kodu) would go when it would see coin and when it sees coin it will move toward it and then the second law will go in and then when it bumps the coin it will eat it*"

This transcript demonstrates that Student C understood the 2nd Law of Kodu and that he/she was able to apply it when reasoning about the behavior of a kodu character in a game context. This transcript is consistent with 9 out of 15 students who correctly answered this question and also directly referred to or stated the 2nd Law while explaining their reason for choosing this option. The other 6 students said that rule ordering will not matter which is a correct application of the 2nd Law.

5.3 Session 3 – 3rd Law of Kodu

During the session 3 think-aloud interview, students were given **Q14** (Fig. 6), a 3-rule program: (1) consume apple rule, (2) pursue blue apple rule, and (3) pursue red apple rule. Students were asked to trace the path of kodu given 2 red and 2 blue apples scattered in front of kodu. Application of the 3rd Law of Kodu results in the kodu eating all the blue apples and then the red apples.

Q14. Here is another Kodu program for eating apples. Draw the path and write a number next to each apple to show the order in which the apples will be eaten.

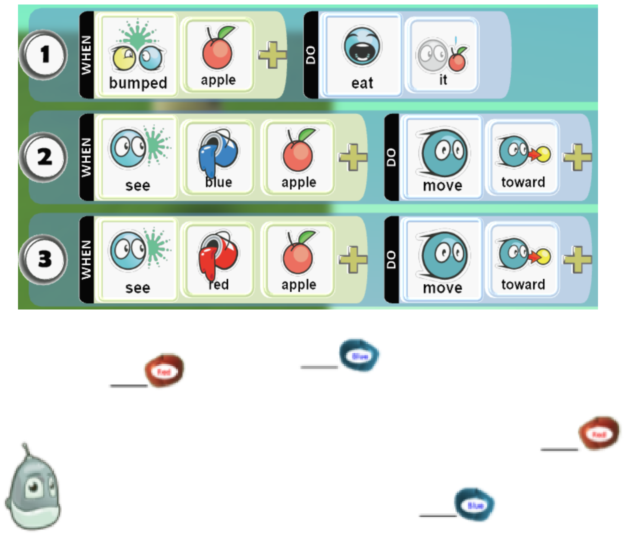


Figure 6: **Session 3, Q14 think aloud question**

Interview Transcript 4: Student D refers and states the 3rd Law of Kodu

Context: Student D explains why he/she marks a path in which the kodu first eats all the blue apples and then the red apples.
Interviewer: Why did you choose the blue apple first even if the red one is closer?
Student D: "*Yea, but the second one [pursue-blue-apple rule] is true and the first one [consume-apple rule] is not. [Pointing to second and third rules] these are both true, and like the 3rd Law of Kodu, it follows when the actions conflict, the earliest one goes*"

This transcript shows that student D was able to evaluate each of the three rules in the program, describe when a rule was eligible to run, and understood how to correctly apply the 3rd law to predict the path of kodu.

6 DISCUSSION

The results of this paper demonstrate that 4th and 5th grade students were able to refer to, state, and apply the three Laws of Kodu when reasoning about 1-3 rule programs. In addition, these results demonstrate that students were frequently using laws to predict program behavior, justify their answer choices, and to explain their reasoning. This suggests that the Laws of Kodu provided students with the conceptual framework and vocabulary to explain the behavior of rules in a program and to predict the resulting behavior of the program when executed. We believe this is due to the concise and simple language of each rule and the use of the animated simulations of the laws to explain when to apply them. These results also indicate that teaching 4th and 5th grade students to understand how programming statements are interpreted and executed is age-appropriate and easily accomplished when scaffolded.

7 CONCLUSION

As elementary school students are increasingly engaged in learning to program, it is important to help them understanding how to read and understand programs written by themselves and others. In addition, it is important to help them understand the underlying mechanism by which programs are interpreted and executed by computers. Our findings show that when elementary school students are explicitly taught the Laws of Kodu computation, they can read and understand programs written by others and predict the behavior of these programs. Reasoning about programs in this way has the potential to help students improve their ability to read and comprehend programs and transfer this knowledge across programming environments. While the Kodu environment and its semantics may be different from other environments, we believe curriculum designers for other environments can scaffold students' abilities to reason about programs by providing explicit instruction on the computational laws underlying the programming frameworks.

ACKNOWLEDGEMENTS

We would like to thank Joseph Isaac and Ashley Cahill for their assistance in conducting the study and all the participants for their time. This work was supported by a gift from Microsoft Research.

REFERENCES

[1] Aggarwal, A., Gardner-McCune, C., & Touretzky, D. S. (2016). Designing and Refining of Questions to Assess Students' Ability to Mentally Simulate Programs and Predict Program Behavior. *In Proceedings of SIGCSE '16*, (pp. 696-696). Memphis, TN: ACM

[2] Chmiel, R., & Loui, M. C. (2004). Debugging: from novice to expert. *ACM SIGCSE Bulletin*, 36(1), 17-21.

[3] Clancy, M. (2004). Misconceptions and attitudes that interfere with learning to program. In S. Fincher & M. Petre (Eds.), *Computer Science Education Research* (pp. 85-100). Abingdon, UK: Taylor & Francis.

[4] Dann, W. P., Cooper, S., & Pausch, R. (2011). *Learning to Program with Alice* (w/CD ROM). Prentice Hall Press.

[5] Deimel Jr, L. E. (1985). The uses of program reading. *ACM SIGCSE Bulletin*, 17(2), 5-14.

[6] Du Boulay, B., O'Shea, T., & Monk, J. (1981). The black box inside the glass box: presenting computing concepts to novices. *International Journal of Man-Machine Studies*, 14(3), 237-249.

[7] Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8(6), 1-165.

[8] Kollmansberger, S. (2010). Helping students build a mental model of computation. *In Proceedings of the fifteenth annual conference on Innovation and technology in computer science education* (pp. 128-131). ACM.

[9] Kumar, A. N. (2013). A study of the influence of code-tracing problems on code-writing skills. *In Proceedings of the 18th ACM conference on Innovation and technology in computer science education* (pp. 183-188). ACM.

[10] Lister, R., Fidge, C., & Teague, D. (2009). Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *ACM SIGCSE Bulletin* 41(3), 161-165.

[11] McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B. D., Laxer, C., Thomas, L., Utting, I., & Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin*, 33(4), 125-180.

[12] Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education*, 23(3), 239-264.

[13] Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner., A., Rosenbaum, E., Silber, J., Silverman, B., & Kafai, Y (2009) "Scratch: Programming for everyone." *Communications of the ACM* 52(11), 60-67.

[14] Sheard, J., Carbone, A., Lister, R., Simon, B., Thompson, E., & Whalley, J. L. (2008). Going SOLO to assess novice programmers. *ACM SIGCSE Bulletin* 40(3), 209-213.

[15] Sirkiä, T., & Sorva, J. (2012). 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* (pp. 19-28). ACM.

[16] Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9), 850-858

[17] Spohrer, J. C., & Soloway, E. (1986). Novice mistakes: Are the folk wisdoms correct?. *Communications of the ACM*, 29(7), 624-632.

[18] Spohrer, J. G., & Soloway, E. (1986). Analyzing the high frequency bugs in novice programs. In *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers* (pp. 230-251). Ablex

[19] Touretzky, D. S. (2014). Teaching Kodu with physical manipulatives. *ACM Inroads*, 5(4), 44-51.

[20] Touretzky, D. S. (n.d.). Kodu Curriculum Modules. Retrieved July 26, 2016, from https://www.cs.cmu.edu/~dst/Kodu/Curriculum/

[21] Touretzky, D. S. (n.d.). Kodu Idiom Flash Cards & Tiles. Retrieved July 26, 2016, from https://www.cs.cmu.edu/~dst/Kodu/

[22] Touretzky, D. S., Gardner-McCune, C., & Aggarwal, A. (2016). Teaching Lawfulness With Kodu. *In Proceedings of SIGCSE '16* (pp. 621-626). Memphis, TN: ACM.

[23] Touretzky, D. S., Gardner-McCune, C., & Aggarwal, A. (2017). Semantic Reasoning in Young Programmers. *In Proceedings of SIGCSE '17* (pp. 585-590). Seattle, WA: ACM.

[24] Wadsworth, B. J. (1996). Piaget's theory of cognitive and affective development: *Foundations of constructivism*. Longman Publishing.