Evaluation of Childhood Computational Thinking with ScratchJR and Robot Toys: A Literature Review

Joseph Wilkes

Brigham Young University

Table Of Contents

Abstract

This literature review recognizes current research as it answers the following question: How effectively can children in kindergarten through third grade develop programming skills and computational thinking (CT) through ScratchJR and toy robots? This review considers the concept of computational thinking and associated criteria for CT assessment. Different experiment methodologies such as toy robots and ScratchJR will be evaluated and compared. Research regarding social factors that relate to CT will be described by various studies. One study proposes that social reasoning skills potentially enhance CT development. Significant findings compare ScratchJR and toy robots to conclude that tangible mediums like robots are more effective. There are contradictions between findings regarding children's capability to effectively decompose a problem. Primarily, research suggests that children can develop computational thinking capabilities with the help of age appropriate tools.

Evaluation of Childhood Computational Thinking with ScratchJR and Robot Toys: A Literature Review

This literature review describes the value and difficulty of children ages 3+ learning how to program. The literature is considered with the goal of answering the following question: How effectively can children, in grades K-3, develop programming skills and computational thinking (CT) through ScratchJR and toy robots? The importance of this topic is illustrated through the increase in demand for technical professionals who can think algorithmically. The primary consideration in meeting market demands for qualified professionals is to improve the education system in regards to computer science. Adjustments in many educational systems have increasingly focused on teaching programming to elementary school kids.

This review will consider the concept of computational thinking and associated criteria in recent research. Different experiment methodologies such as toy robots and ScratchJR will be evaluated and compared. Social factors as they relate to CT will be addressed. Some researchers incorporate social interactions as fundamental criteria for assessing CT while other research neglects social factors altogether. Children's abilities to decompose a problem will also be considered.

A critical gap in the research is the lack of a longitudinal assessment spanning several years of students in a CT focused curriculum. A future study could compare CT competence development through programming/CT specific activities as opposed to non-programming activities in academic settings.

It is hypothesized that children grades K-3 can begin to learn introductory programming concepts and develop aspects of computational thinking such as sequencing, debugging, and decomposing problems into segments.

Computational Thinking

Brennan and Resnick support the following definition for computational thinking: "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (2012, p. 2). The goal in defining criteria for computational thinking is to provide a framework for teaching and assessing children CT skills. Additionally, a CT framework supports research evaluation of children's cognitive development.

Criteria For Computational Thinking

Brennan and Resnick propose a three-part framework to organize the assessment of computational thinking. These parts are computational concepts, computational practices, and computational perspectives (Brennan & Resnick, 2012). Other sources share similar principles, but the computational perspective principle of Brennan and Resnick (2012) is unique. Brennan and Resnick (2012) provide definitions and examples for the core principles of their framework. They first define computational concepts as, "the concepts designers engage with as they program" (Brennan & Resnik, 2012, p. 1). Next, computational practices are, "the practices designers develop as they engage with the concepts, such as debugging projects or remixing others' work" (Brennan & Resnik, 2012, p. 1). Lastly, computational perspectives are, "the perspectives designers form about the world around them and about themselves" (Brennan & Resnik, 2012, p. 1).

Almost all of the sources have an intersection and form of cohesion that computational concepts and practices are inherent components in computational thinking. Some sources focus on debugging, sequencing/iteration, and decomposing a problem into segments as valuable points of observation. Although Brennan and Resnick do propose computational perspectives as an essential component of CT, the study recognizes the difficulties of gathering research around students' beliefs about themselves and the world around them (2012). The 2019 Chou study adopted the Brennan and Resnick CT framework but experienced difficulty in measuring change in relation to computational perspectives.

Alternative Frameworks For Computational Thinking

Opposed to Brennan, Resnick, and Chou, Pugnali, Sullivan, and Bers considered four primary aspects of computational thinking for the study criteria. In this study, researchers simply assessed children's ability to learn sequencing, debugging, conditionals, and repeats (Pugnali, Sullivan, & Bers, 2017). These four aspects of coding would fall under the first and second aspects of the Brennan and Resnick framework while neglecting their third component.

Bers, González-Gonzáles, and Armas-Torres also developed a framework called the Positive Technological Development (PTD) framework (2019). It focuses on six components, each starting with the letter C. These components are communication, collaboration, community building, content creation, creativity, and choices of conduct. Bers et al. recognize the value of social interaction in accordance with computational thinking. This framework, along with the Brennan and Resnick framework, has the end goal of helping students become effective programmers (Bers et al., 2019).

Experiment Methodologies

Experiments varied in the implementation of the study in order to assess children's development of computational thinking. A vast majority of the studies utilized a software called ScratchJR which allows children ages 5 to7 to create programs without having to read (Brennan & Resnik, 2012; Chou, 2019; Strawhacker & Bers, 2019; Pugnali et al., 2017). The programs are implemented by dragging colored blocks on the computer screen in a sequence that will instruct an animation to perform certain tasks. A precursor of ScratchJR was called KIDSIM (Rader, Brand, & Lewis, 1997). Both ScratchJR and KIDSIM are software interfaces that attempt to examine the development of computational thinking in the participants. An alternative methodology used toy robots. Kibo and Bee-Bot are robots that are made to help children learn how to program (Pugnali et al., 2019; Bers et al., 2019; Angeli & Valanides, 2019). Kibo depends upon physical blocks that have pictures on them. The child lays the blocks on a surface with the robot facing the blocks. Then, the robot will scan the blocks and perform actions accordingly. Bee-Bot has buttons that instruct the robot to move in various directions or rotate. Both methods, the software and the robots, attempt to help children develop computational thinking through introductory programming.

Kibo and Bee-Bot

Pugnali et al. (2017) utilized the Kibo robot in their studies. Alternatively, Angeli and Valanides used the Bee-Bot, utilizing two different scaffolding techniques to identify differences in computational thinking between 50 boys and girls ages 5 and 6. The Pugnali et al. (2017) experiment had a sample size of 28 students of ages 4 to7 and consisted of an intensive one week program. Instruction focused on sequencing, debugging, and decomposition as criteria for

computational thinking. Like Angeli and Valanides, Pugnali et al. also had sequencing, and debugging as a main assessment for computational thinking. However, Pugnali et al. also focused on student's ability to use repeats and conditionals which are recognized by the researchers as more advanced programming concepts (2017).

ScratchJR

Several studies evaluated computational thinking using ScratchJR. Brennan and Resnik uniquely attempted to quantify CT by evaluating differences between projects in students' portfolios over time. In a few cases, these researchers were able to examine longitudinal data since a few of the students had been using ScratchJR for several years (Brennan & Resnik, 2012). However, this was not a longitudinal study design, thereby leaving a gap for further research. Because ScratchJR is a digital platform, each block of code that is used in any project can be categorized thereby allowing for quantifying the projects of the students. Brennan and Resnik also recognize the limitations of the project portfolio analysis. One limitation is that project portfolio analysis yields no information about the development of the project as it relates to debugging, decomposition, or whether they got help on the particular project. Genres between projects is unavailable through automated means. Measuring genres would give insight into the student's ability to apply CT in different arenas (Brennan & Resnik, 2012).

Chou, Strawhacker, and Bers used the same methodology for assessing CT development in students. Both studies employed a visual examination where students would view programming questions and then respond on structured answer sheets. Chou focused on four categories: "fixing the program, circling the blocks, matching the program, and reverse engineering" (Chou, 2019, p. 8). The first three categories in the Chou study were evaluated

through multiple choice formatted questions, whereas the reverse engineering questions were open ended. Chou considered the tests to be highly reliable and valid. Compared to Strawhacker and Bers (2019), Chou had take-home assignments and instructor observation documents to report back the empirical progress of students (Chou, 2019). The Strawhacker and Bers assessments focused on two primary categories: "matching a command with its outcome and constructing a program with correct commands in the correct order" (Strawhacker & Bers, 2019, p. 550). These categorical assessments allowed the researchers to test the student's logic and understanding.

Comprehensive Comparison

Duncan, Bell, and Tanimoto created criteria for evaluating different tools to teach children how to program (2004). Each tool is referred to as an Initial Learning Environment (ILE). This study examines 47 different ILEs. This examination is beneficial when evaluating the effectiveness of these tools to successfully support CT development. One aspect of assessment is about the ease of adoption of the ILE for schools. The assessment criteria is defined in terms of levels. Level 0 is for children ages 2 to 7, and its tools primarily teach sequencing. These level 0 tools require no abstraction. Level 1 tools, however, require a minor amount of abstraction. Both level 0 and level 1 are ILEs with drag and drop methods like ScratchJR. The Bee-Bot is considered Level 0 whereas ScratchJR is considered Level 1 (Duncan et al., 2004). Thus, Duncan et al. (2012) supports the findings of Pugnali et al. (2017) that the robots are more effective for teaching young children to develop CT as opposed to ScratchJR.

Social Factors in Computational Thinking

Much of the research about computational thinking includes aspects of social interaction and reasoning. Some studies consider social intelligence to be an important companion to computational thinking in terms of helping students become prepared for the academic and vocational pursuits with technology. Brennan and Resnick's (2012) three part framework includes a social component called computational perspective. The Bers et al. (2019) study supports the Positive Technology Development framework, which has the goal of teaching children to develop CT in collaborative ways. Some studies have children work together but neglect the social component altogether in the research. Another study describes social reasoning as a precursor and companion to CT. The diversity of thought of the association between computational thinking and social skills provides unique opportunities for continued research and theoretical development.

Computational Perspective

Computational perspective is one of the essential three components in the Brennan and Resnick (2012) CT framework. This component describes the way individuals view themselves and the world around them. As individuals develop computational thinking, they will see themselves as creators in a technological world. They will begin to view the world around them as algorithmic processes that can be modeled and simplified through technology (Brennan & Resnik, 2012).

Positive Technology Development

Bers et al. (2019) advocates the PTD framework. Parts of the framework are social in nature: communication, collaboration, and community building. Unlike Brennan and Resnick

(2012), the Bers et al. study does not describe the changing perspectives of the student as they develop CT. The researchers observed the social nature of computational thinking development. The observations of Bers et al. were described in relation to communication, collaboration, and community building. Children practiced communication through describing their end result, as well as getting help on their project. Children collaborated with each other as they traded ideas and shared materials for the Kibo robots. Children practiced community building through discussing one another's projects as well as displaying their projects to family members.

Neglecting Sociality in Computational Thinking Development

The Rader et al. (1997) study, like the Bers et al. (2019) and Brennan and Resnick (2012) studies, had students work together; however, these interactions were never recognized by Rader et al. Since the students worked in pairs, there could have been several scenarios worth describing. The students could have collaborated effectively together to solve problems or, on the flip side, they could have distracted each other thereby inhibiting CT development. Rader et al. didn't consider the social factors associated with computational thinking development.

Social Skills Enhancing Computational Thinking

Although the Brennan and Resnick (2012) and Bers et al. (2019) studies recognize social factors in the CT framework, the Strawhacker and Bers (2019) uniquely highlights the initial evidence that social skills enhance computational thinking. Social reasoning, as termed by Strawhacker and Bers, results from deciphering the emotions of others, interpreting the intentions of others, fostering relationships, and imagining alternative perspectives. This research suggests that the development of social reasoning is correlated with, and may contribute to, programming logic and practices. The Strawhacker and Bers study postulates that causal, spatial,

verbal, and social reasoning appear to be related to the nature of programming, more so than even mathematics. These findings, about the positive relationship between social skills and computational thinking, are emphasized as primary results in the Strawhacker and Bers study.

Significant Results and Contradictions

There has been thorough research about children's capabilities to develop computational thinking. Several studies like Bers et al. (2019), and Angeli and Valanides (2019) used robots as the medium for CT development. Brennan and Resnik (2012), and Strawhacker and Bers (2019) used ScratchJR whereas Rader et al. (2019) employed other digital software as the medium for CT development. Two significant results to be recognized were the comparisons between mediums and contradictions that were found amongst studies in regards to CT development findings.

Comparison Between Mediums

Bers et al. (2019) and Pugnali et al. (2017) both utilized the Kibo robot. Brennan and Resnik (2012), Chou (2019), Strawhacker and Bers (2019), and Pugnali et al. (2017) all used ScratchJR to as tools to measure CT.

With the wide variety of tools and studies, Pugnali et al. (2017) were the only ones to compare robots and software for teaching computational thinking. Pugnali et al. (2017) found statistically significant results that students using Kibo scored higher across CT categories more so than students using ScratchJR. According to Pugnali et al. (2017), this finding supports the belief that children learn better through tangible means like the robots as opposed to a digital interface.

Contradictions in Findings

An ability that is expected of effective computational thinking is decomposing a problem into segments in order to solve the pieces. Two studies show polar conclusions regarding children's ability to perform such analysis. First, the Rader et al. (1997) study describes the difficulty that the 7 to 9 year olds experienced in the study. This study focused on a software called KIDSIM which allowed for world creation and design. The goal of the study was to teach the students to create simple scientific models in these imaginary worlds. The study identifies the student's lack of ability to decompose the general model into a set of objects with particular behaviors. On the other hand, Angeli and Valanides (2019) advocate for the young student's analytical abilities. Secondly, this research holds the student's decomposition skills as a paramount conclusion of the study. The Angeli and Valanides study concludes that children, even at a very young age, are capable of grappling with a complex learning task by decomposing it into subtasks that are easier for the students to tackle. It is important to note that the Angeli and Valanides (2019) study had 5-6 year old participants whereas the Rader et al. had 7-9 year olds. The study with the younger participants held higher esteem of children's analytical capabilities than the study with older participants.

Although both the Angeli and Valanides (2019) and Rader et al. (1997) studies varied in the methodology, they were both attempting to measure children's capability in programming environments. The Rader et al. research involved the KIDSIM software whereas the Angeli and Valanides study involved the Bee-Bot robots. Both studies had relatively the same amount of participants. Although the Rader et al. study didn't describe the distribution of gender of the participants, the Angeli and Valanides study had almost an even distribution of male and female

participants. Without many other explanations for this contradiction amongst the Angeli and Valanides (2019) and Rader et al. (1997) studies, the Pugnali et al. (2019) research provides one perspective regarding the tangibility of the Bee-Bot robots in the Angeli and Valanides study versus the KIDSIM digital interface of the Rader et al. study. The tangible robots provide context and concreteness for the students to explore computational thinking strategies like decomposition. Thus, age appropriate mediums should be considered for effectively developing computational thinking in children.

Conclusion

Because of increasing innovations in technology, it is beneficial for students and children to start developing cognitive capabilities while young. These cognitive capabilities are referred to as computational thinking. Although there are different criteria for measuring computational thinking, there is consistency amongst general definitions. There are different tools to assist in children's development of computational thinking. Through tools like toy robots and animated programming platforms on computers, children are able to develop understanding and capability with fundamental programming concepts and practices. Although these tools should be age appropriate, research identifies many of the developments that children can experience. Research also demonstrates the importance of tying social factors to computational thinking. Current research is optimistic in the potential of young children, ages 3 and up, learning to think computationally.

References

Angeli, C., & Valanides, N. (2019). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior*.

Bers, M. U., González-González, C., & Armas–Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education*, *138*, 130-145.

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada* (Vol. 1, p. 25).

Chou, P. N. (2019). Using ScratchJr to Foster Young Children's Computational Thinking Competence: A Case Study in a Third-Grade Computer Class. *Journal of Educational Computing Research*, 0735633119872908.

Duncan, C., Bell, T., & Tanimoto, S. (2014, November). Should your 8-year-old learn coding?. *In Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 60-69). ACM.

Pugnali, A., Sullivan, A., & Bers, M. U. (2017). The Impact of User Interface on Young Children's Computational Thinking. *Journal of Information Technology Education: Innovations in Practice*, *16*(1), 171-193.

Rader, C., Brand, C., & Lewis., C. (1997). Degrees of comprehension: Children's understanding of a visual programming environment. *Proceedings of the 1997 ACM SIGCHI Conference on Human factors in computing systems*.

Strawhacker, A., & Bers, M. U. (2019). What they learn when they learn coding: investigating cognitive domains and computer programming knowledge in young children. *Educational Technology Research and Development*, *67*(3), 541-575.