A Comparison of Preschool and Elementary School Children Learning Computer Science Concepts through a Multilanguage Robot Programming Platform

Cecilia Martínez
Facultad de Filosofía y Humanidades
Universidad Nacional de Córdoba/CONICET
Córdoba, Argentina
cecimart@gmail.com

Marcos J. Gómez
FAMAF, Universidad Nacional de Córdoba
Córdoba, Argentina
mgomez4@famaf.unc.edu.ar

Luciana Benotti
Logic, Interaction and Intelligent Systems Group
FAMAF, Universidad Nacional de Córdoba/CONICET
Córdoba, Argentina
benotti@famaf.unc.edu.ar

ABSTRACT

This paper describes a school intervention to teach fundamental Computer Science (CS) concepts to 3-11 year old students with a multilanguage robot programming platform (using drag and drop, Python and C++ languages) in Argentina. We analyze students' performance and learning process based on multiple choice test and classroom observations. Data show that all students can intuitively learn sequence, conditional, loops and parameters and that girls performed slightly better than boys. Older students can easily combine these concepts to write a program. The multilanguage platform promotes student spontaneous exploration of more sophisticated CS concepts and languages. These findings imply that introducing CS in mandatory schooling from an inquiry based approach is both achievable and beneficial.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer science education

General Terms

Education

Keywords

Computer science K-7 outreach, robots, experimental evaluation, iconic programming language

1. INTRODUCTION

There has been a lot of debate on whether preschool and elementary school children should use computers or are developmentally ready to learn programming [2]. On one side, children are using computers much earlier each decade [14].

On the other side, this intensive use that many children have of computers may not be contributing to early access of Computer Science as a discipline (CS) which includes notions of creating and developing technology, programming, designing, and automatizing actions. We could argue that children become software consumers very early but they do not learn some basics of how this technology works. Bergen [1] points out that re-programmable toys such as robots, give children the possibility of creating, imagining, programming and exploring rather than developing procedural digital competences.

Previous research suggests [2] that early introduction of some basic CS concepts benefits both children cognitive development and learning about CS. Nevertheless, we still are debating what kind of CS concepts should be taught in preschool and elementary school [15]. Some countries such as Estonia and the UK have recently introduced CS in these levels [5], but most others—including ours—are still deliberating when would it be appropriate to introduce CS in schools and what content is most suitable for the general basic education. While research on teaching CS in different educational levels is vast, and colleagues have proposed curricular designs [9, 3], comparisons among different age groups performance that inform curriculum selection and scope is not as common [5, 11].

With the purpose of both investigating how children learn basic CS concepts in schools using programmable toys and contributing to a CS curriculum selection and scope; we designed an exploratory study to compare how preschool children (ages 3 and 5), and elementary school children (ages 8 to 11) learn some basics CS concepts. We piloted CS lessons in a real school setting focusing on loops, variables, conditionals, sequence, and parameters; and their application to robot programming. We analyzed children' learning of CS using a multilanguage robot programming platform and compared boys and girls performance. The main contributions of this paper are: 1) Analyzing how different age group of children learn fundamental CS concepts. 2) Introducing a multilanguage robot programming platform that permits students to discover new CS concepts on their own, growing with the platform. 3) Evaluating gender and age

ITiCSE'15, July 6–8, 2015, Vilnius, Lithuania.
Copyright is held by the owner/author(s). Publication rights licensed to ACM.
ACM 978-1-4503-3440-2/15/07 ...$15.00.
http://dx.doi.org/10.1145/2729094.2742599.

differences in the acquisition of CS concepts in preschool and elementary school.

We begin the paper summarizing previous work. Then, we describe a multilanguage robot programming platform and its rationale. We address the study design followed by our findings. We close this paper with conclusions and implications for teaching CS in preschool and elementary school.

2. PREVIOUS WORK

Starting preschool, children can create, run and debug simple computer programs using specific platforms that are both challenging and attainable for most children [2, 10, 8]. The effects and implications of learning CS at such an early age have also been analyzed. According to Clements [2], children who use computer assisted programs have the opportunity to analyze a situation and reflect on the properties of objects they have to manipulate.

While exploring how to teach CS to little children some researchers have found that the difficulties in children programming laid in their immature motor skills and on syntax problems [12, 8]. Thus, there has been a vast development on specific programming platforms to address the developmental traits of preschool children (such as Toon Talk, Scratch Jr, CHERPS, etc). In this context, programming robots has been an interesting line of work to teach CS to little children.

Flannery and Berns [7] showed that as a result of robot programming in preschool, children imagine, plan its action, and construct a robot. In their study, the authors found that all 4-6 year old students program short challenges and explore robot's capabilities.

Although most interventions to teach programming with robots achieve high student engagement and task completion, we still need to understand more how the use of these platforms promotes learning specific CS concepts. Morgado and Kahn [12] have documented how preschool children learn competences and concepts such as syntax, parameter passing, compound procedures, parallelism and concurrency, communication channel, input and client and servers using Toon Talk platform. We also need to learn more about what CS concepts children can understand in different developmental stages to establish a school curricula content and scope. We found only a handful of studies that compare different age groups performance on similar CS teaching activities. Magnenat and his colleagues (2014) [11] taught CS with robot programming to different age groups of children using event handler language to program a robot action in different events. Comparing the groups performance with the same task, they found that most children understood and solved simple tasks such as moving a robot upon a touch of a button or identifying robot's instructions. However, older children performed better on complex programming that required several conditions or events. Dagiene et al [5] compared students from Finland, Sweden and Lithuania ages 7 to 12 performing similar algorithmic thinking tasks exercises. Using multiple choice questions, they evaluated concepts such as graphs, search algorithms, data structures, and executing sequences. They found no strong difference across age groups, but rather among countries. The authors suggest that educational context, academic quality and in particular, reading ability promoted by each school system may be strongly related to learning CS concepts. Thus, we want to highlight the value of conducting exploratory studies in purposely selected geographical context that may be transferable to similar places, to contribute to CS curriculum design appropriate for each region. In this paper we compare different age groups performance on robot programming tasks and analyze students learning of basic CS concepts in a real school context in Argentina.

3. A ROBOT PROGRAMMING LANGUAGE

The `UNC++Duino` programming environment is an extension of blocklyDuino[1], a platform based in blockly, for programming Arduino[2] boards supporting Grove System. Code org uses Blockly in their Hour of Code initiative [16]. We extended blocklyDuino to adapt it to a multiplo N6[3], an educational robot platform created in Argentina and selected for our pilot study (illustrated in Figure 1). UNC++Duino includes a drag and drop language, that allows students to focus in solving CS problems without thinking about syntax. The platform can also be programmed in other full programming languages such as Python and C++ with different levels of language difficulties and expressiveness. The simplest one is the iconic language, but the student can switch into a more complex one, being C++ the hardest and most expressive. Our iconic programming language, with no natural language, allows kindergartners to sixth graders to easily program the N6 robot. Each block represents an executable robot action. A set of arrows enables the robot to move forward (20 cm), turn 90 degrees left or right. We also created blocks for control structures such as loops, conditionals, parameters, among others.The program has a musical block (represented by a saxophone), allowing kids to choose different songs for the robot to play. Two images, one showing the robot in front of an object, and the other one with nothing in front of the robot, represent conditional (Figure 1e). We programmed the platform to translate each iconic block automatically into Python and C++ encouraging children to explore into the different languages, seek other robot functions and grow out of the iconic interface

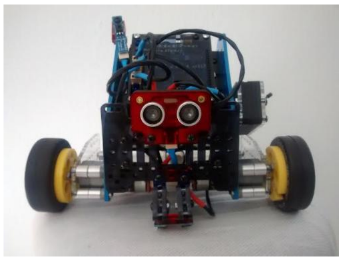


Figure 1: The N6 robot

Figure (1a), shows a program that allows children to move the robot 60 cm. Simply selecting arrows make it easy for all students to code. In contrast, coding with blocklyDuino (1b), requires working with numbers, time delay, engine like objects and calculating the relation between the defined speed and time to advance 60 cm. If we wanted the robot turn 90 degrees to the left, on UNC++Duino, students simply select the Turn left arrow block (1c). Image (1e) shows

[1]`https://github.com/gasolin/BlocklyDuino`
[2]`http://www.arduino.cc/`
[3]`http://www.robotgroup.com.ar/index.php/productos/131-robot-n6#especificaciones`

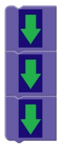
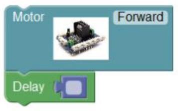
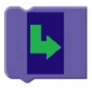
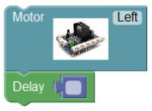
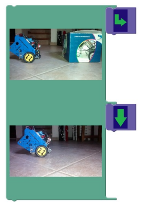
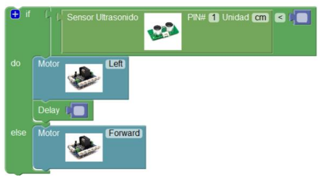
	UNC++Duino	BlocklyDuino
Advance	 (a)	 (b)
Turn left	 (c)	 (d)
Conditional	 (e)	 if Sensor Ultrasonido PIN# 1 Unidad cm do Motor Left Delay else Motor Forward (f)

Table 1: Comparison between UNC++Duino and blockly-Duino.

how students can program their own Object detector robot selecting the pictures representing the conditions (1f).

4. STUDY DESIGN

We describe the school context where we conducted the study outlining the different interventions in preschool and elementary school.

4.1 The Setting for the Study

We made the intervention in a privately run school receiving state public and students' tuition funds. There are no children below the poverty line and most children have middle class, professional parents. The school follows an experience based pedagogy and organizes its curriculum on problem, case and project based learning. The institution has strong links with the School of Education at Cordoba National University, and constantly organizes professional development events for their teachers. We decided that this context provided a unique opportunity to pilot discovery based CS teaching experiences. Our team provided materials and expertise on teaching CS and the school provided expertise on reaching elementary and preschool children.

4.2 The Preschool Intervention

University professors and preschool teachers designed an academic unit to program a robot following three stages. During the first stage, students acted as the robots who followed their peers commands. Teachers designed a floor game that consisted on a 5 by 5 square grid with obstacles and targets randomly sprawled in the grid. Children chose a sequence of arrows that took the robot (i.e., the child) to the matching target. There were three types of arrows: straight, turn right and turn left. Students made one to one correspondence (one arrow followed by one movement) to play this game. Also, children placed arrows in the specific square where the robot had to move making spatial correspondence as well.

The second stage consisted in replicating the floor game into a board game using a table size cardboard and toy robots. Similar to the floor game, children made one to one spatial and movement correspondence but moving a toy robot instead of their bodies. This was the first step to detaching their bodies from the robot's actions.

In the third stage children programmed the N6 robot in a computer with the described platform UNC++Duino. Children programed the robot to run on the floor squared grid using notions of *sequence* and *parameter*. Then, children programmed the robot to avoid objects using *conditionals* working freely on the floor without the grid. Finally, 5 year-old students learned *Loop*. The task required advancing the robot many times using only two lines of code.

4.3 The Elementary School Intervention

From May to December 2014 a university professor in CS and member of our research team taught CS lessons to 8-11 years old children for one hour a week. Before the professor took over, students have learned a mix of offimatics and LOGO. However, upon recalling previous knowledge, students remembered almost no CS concepts. Similar to the preschool experience, there were three different stages.

During the first stage, children worked with Code.org [16] tutorials using concepts such as *sequence*, *conditionals* and *loops*. In the second stage, students developed their own animations using the platform Alice [4, 6]. In the final stage, students programed the N6 robots with the *UNC++Duino* platform focusing on *sequence*, *conditionals*, *parameters*, and loops.

4.4 Data Collection and Analysis

In all, 190 students participated in this exploratory study. Table 2 describe participants' distribution.

School level	Pre	Pre	Elem	Elem
Number of students participating	25	30	70	65
Number of students tested	17	26	42	43
Student's age	3-4	5-6	8-9	10-11

Table 2: Participants Distribution(some students were absent on the test day).

All students programmed the robot employing sequences, parameters, conditionals, and loops in different tasks. Tasks required either using one concept (e.g. sequence), a combination of concepts (e.g. sequence and conditional) or applying concepts to two different programming situations (e.g. using the same program the robot must run in two different labyrinth). After each lesson, children took the same multiple choice test to assess how different age groups understood each CS concept. The test included 7 different multiple choice programming tasks. A "simple" task required applying one or two concepts (such as combining sequence and conditional to make an obstacle dodger robot). A "complex tasks" required combining concepts and selecting a program to solve two different problems. Thus, a higher level of abstraction.

We also conducted lesson observations during all of the robot programming classes. Observations allowed us to gather data on student engagement with programming and on transferring concepts learned with other platforms. In this paper, we only report on the robot programming stage to compare how preschool and elementary school children learned basic CS concepts and applied them to robotic programming. We do not present data on students learning other CS concepts with code.org or Alice.

Because the focus of this paper is understanding how different age group of children understand basic CS, we compared results of both preschool and elementary school interventions. We analyzed test results with descriptive statistics. We crossed preschool and elementary school data and compared gender differences in elementary school. We triangulated these results with qualitative data from observations that provided further indicators of emerging themes.

5. RESULTS

In this section we present age and gender differences in learning CS concepts at preschool and elementary school. Figure 2 summarizes the multiple choice test results showing performance by age group. Because of organizational issues, not all age groups were tested on all concepts.

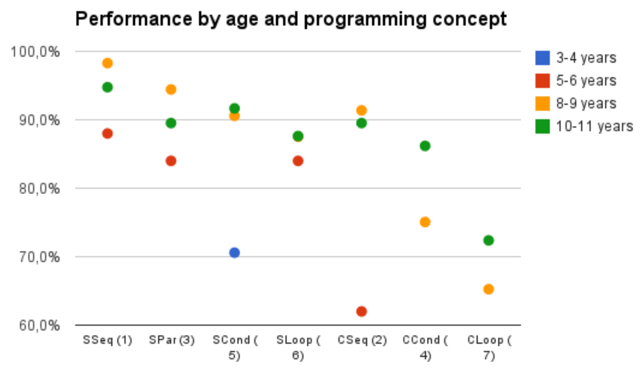


Figure 2: Percentage of correct test results per age group and CS concept (Capital S, stands for Simple y Capital C stands for Complex).

5.1 Learning Sequences

23 out of 26 preschool children selected the right sequence of instructions. Lesson observations indicated that most preschool children could provide a series a sequential instructions including both straight arrows interleaved with turn arrows. By the time children played the board game and programmed a robot on the computer they could clearly indicate the amount and orientations of arrows needed to make the robot get to target. Divergent thinking was also present as children suggested different paths for the robot to get to target. However, systematically, 3-4 years old placed one arrow only when the robot had to turn instead of selecting both an straight and turning arrow. This last group of preschool students also had difficulties realizing that robots run the sequence code written on the computer. Linking the virtual world of the screen with the concrete spatial movements of the robots was not a problem for 5 years old.

Most elementary school students performed high on both the simple and complex sequence multiple choice test. Observations showed that students solved the sequence challenge in five minutes and quickly moved into discovering other types of commands and robot functions. Only 2 groups of students took longer to program the robot because instead of reading the task they simply wrote the sequence they wanted the robot to run. Based on observation data, both simple and complex sequence were not challenging for elementary school students. However, most importantly we observed that the activity of programming a robot seemed to encourage exploration of other CS concepts. Elementary school children of all age groups immediately designed new robot challenges. For example, one observations in 4th grade (9 to 10 years old) indicated:

"Because the proposed challenge is to simple, students create their own circuit. The loudest and more 'active' group of the class, is the one thinking the most complex challenge where the robot has to run under a chair made tunnel. They walk the circuit with their bodies and go back and forth from the circuit to the computer to decide each line of code. Only 2 out of 30 children did not complete the task. One group wants the robot to run in circles, another is experimenting how the robot run into things. From this exploration children demand learning about sensors and turning upon an obstacle. One student asks how a proximity sensor is activated.

Excerpt such as this one showed that, thirsty for more, students started asking how to control speed, wheels and other functions. Children spontaneously opened the different UNC++Duino interfaces and made changes on the C++ lines of codes that was automatically generated from their first iconic program.

5.2 Learning Conditionals

In Preschool, children approached the concept of conditional creating their obstacle dodger robot. Classroom observations noted that the teacher simply asked them "What would it happen if a box is in the way of the robot?" Spontaneously and unanimously children replied "The robot has to turn". We showed students the block 1e that would allow them to create their own obstacle dodger robot. 3-5 year old children worked in groups of 5 to 6 children and each group programmed the robot using conditional. Upon showing them the block representing conditional all of them placed the turn arrow command when the block showed an obstacle, and an straight arrow command when the block showed no obstacles. While 5-6 years old read the code and predicted robot's actions, 3-4 years old could not realized that the robot would performed actions written in the computer.

12 of 17 preschool students solved the conditional multiple choice test correctly. The 5 students who responded incorrectly argued that they wanted the robot to crash into the box. When we asked: "Why did you chose this answer" they replied "because I want the robot to crash". Thus, we inferred that the wrong answers is not related to children' understanding of conditional, but rather of student's will.

Similar to the preschool children, elementary students applied conditional creating their obstacle dodger robot. Observations notes showed that one group of 10-11 years old students spontaneously called the conditional "decisions" and compared this function with the robot music block. Thus, students immediately transferred the notion of conditional learned with Hour of Code and Alice platforms. Learning to program an obstacle dodger robot was intuitively and fairly easy for students. However, while 8 to 11 year old children have similar performance on simple conditional, 10 to 11 years old students have better results on complex conditional. We believe that tasks that require combining different fundamental concepts such as sequence, loops and conditional demands deeper understanding and levels of abstraction.

5.3 Learning Loops

Because understanding loops requires some minimal comprehension of counting and multiplying, we only taught loops to 5-6 year old preschoolers. We simply told the students that the new block allowed writing the many times we wanted the robot to move forward instead of placing the amount of arrows we would need. Checking for understanding we asked:“So, if I write here 4 times what would happen?” Students responded “It will advance 4 times. We won’t get tired of writing so many arrows” (excerpt from observation notes). Children’ expressions are evidence that there is some understanding of loops as a repeated action that provides economy in programming. Moreover, each of the groups that programmed a loop selected different numbers of repetitions. One boy wanted the robot to move forward 10 times, so he selected a loop of 5 times and place 2 arrows inside. When we asked what would the robot do with that code, without hesitation he answered that the robot will move 10 times as it will advance 5 times each of the 2 arrows.

In elementary school, observation data showed that a group of students spontaneously asked the teacher where was the “repeat” instruction when they had to write a simple sequence code. Thus, children correctly “assumed” that the robot program had the loop function, transferring what they have learned with Alice and Hour of Code. Results on the loops multiple choice test showed the same performance pattern that we noted for sequence and conditional evaluation. All students, regardless of age and gender, effectively applied the concept of loop on simple tasks. However, when the task required combining concepts such as conditional and loops, older children performed better.

5.4 Learning Parameters

In order to introduce parameters in preschool, we asked students that upon completing the sequence they programmed, the robot must sing a song that matched the target. Within the block that allowed students to program the robot to sing, they had three options: sing any of two popular songs, or remain silent. All of the students were able to add the singing block at the end of the sequence selecting the appropriate song and they were extremely enthusiastic with the “singing” robot. In addition, about 85% of the students pick a song that matched a given picture in the test.

Elementary school students easily transferred notions of parameter previously introduced in Alice. They also enjoyed the idea of a singing robot, and added the song block every time they programmed. But in contrast to preschool children, because the platform allowed to see the code both in blockly and in C++, children spontaneously switched to the C++ or python interface, without having previous experience on them. Upon programming the robot, children soon wanted to change the robot speed and avoid the pause between each movement. They had two possibilities: uploading the block-code into the robot directly from the platform or they could use the arduino translator and copy the code into the arduino interface IDE and uploaded from it. They decided to read the arduino code, without any previous knowledge about it, and modify it to change speed. First, they observed that inside the loop function, there were tabs for the advance method. Upon this discovery, their first intention was to add a numerical argument to modify speed. When compiling the code, they observed an error, so they started to read the whole code and discovered the *avanzar()* method, without knowing what a method was. They also learned that in the body of avanzar(), there were many instructions such as *motor0.setSpeed(50)*,*motor1.setSpeed(50)* and *delay(1000)*. They realized if they changed the argument of the *setSpeed()* of both engines they could change the robot speed and they did it. But the robot still was moving for only 1 second. So they came back to the arduino code, and noticed that the instruction *delay(1000)*, was responsible of engine movement time. They erased the delays and the *setSpeed(0)* instructions, making the robot advance quickly and for a longer time. They continued playing with the arguments of the instructions, making the robot move backward or in circles, actions that would not be possible with the initial version of block code.

5.5 Gender Gaps in CS in Elementary School?

We analyzed test results by gender and identified that while both boys and girls have similar performance applying simple concepts, girls did systematically better on complex concepts (Figure 3). About 12% more of the girls were better at choosing options where loops, conditional and sequence were applied. In addition, the teacher noted that at the beginning of the lesson girls were not enthusiastic with programming a robot because they assumed it was similar to playing with toy cars. However, when noting that the robot responded to their orders or commands, they became progressively more interested. Boys enthusiasm remained constant One hypothesis explaining the different achievement is that girls are generally more focused on academic tasks while boys at this age are very playful. Previous work shows that female students have considerable lower marks in the first year of university than male students. Redmond et al (2013) [13] argues that their lower marks correlates with the fact that they have less exposure to computers and thus, lower confidence with CS. Based on our data, we suspect that the gap between boys and girls may occurs later during their teenage years.

6. CONCLUSIONS

Through a school intervention focusing on fundamental CS concepts we found that all children, regardless of their age group, could intuitively learn sequence, loops, parameters, and conditional, and were capable of applying these notions into robot programming. However, as expected, older children could combine these concepts to create a new program. As obvious as these results may seem, we still need strong empirical evidence about what CS concepts different age group can learn to inform the design of a CS school curricula. We also identified that 3-4 year old students could not correspond the written code in the computer with the robot actions. Girls did a slightly better than boys combining CS on robot programming suggesting that the “gender gap” may occur later in the teenage years. Students

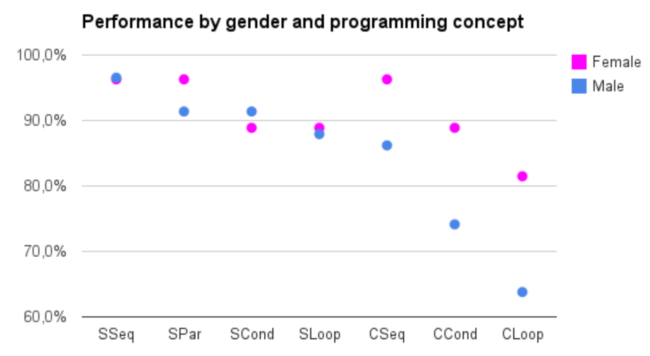


Figure 3: Percentage of correct test results per gender and programming concept

showed high engagement with robots and most importantly, our multilanguage platform triggered students exploration of other CS concepts and allowed children to easily switch from simple to complex languages focusing on concepts rather than on syntax. There are several implication of these findings. 1) Based on our research we suggest that it is not only possible to teach CS in K to 12 mandatory schooling but it is also beneficial in terms of developing CS literacy. 2) Programming robots, while it is costly, highly engage our students. 3) Inquiry based developmentally appropriate teaching strategies proved valuable because they allowed students to build intuitively on concepts, explore with their bodies and apply these concepts to multiple situations, platforms or environments (such as animations, games, board games, etc); and then transfer these ideas into robot programming. This is an important contribution to the field of CS education since we are still debating about pedagogical approaches to introduce CS in schools. 4) Our multilanguage platform designed for k to 12 students was a great tool to expand children exploration and knowledge on CS concepts. In general children grow out of computer platforms designed for a particular age group very quickly. Our platform encourage students to grow with it and because of it. We acknowledge that further research is necessary with multilanguage platforms, but this could be one direction to encourage children growth in CS.

7. ACKNOWLEDGMENTS

The authors wish to thank Science and Technology Secretary at Universidad Nacional de Cordoba, Google for Education, and Manuel Sadosky Foundation at the Argentinean Ministry of Science, Technology and Productive Innovation.

8. REFERENCES

[1] D. Bergen. Technology in the classroom: Learning in the robotic world: Active or reactive? *Childhood Education*, 77(4):249–250, 2001.

[2] D. H. Clements and J. Sarama. Teaching with computers in early childhood education: Strategies and professional development. *Journal of Early Childhood Teacher Education*, 23(3):215–226, 2002.

[3] Computing at School Working Group. *Computer Science: A Curriculum for Schools*. Computing at School Working Group, 2012.

[4] S. Cooper, W. Dann, and R. Pausch. Teaching objects-first in introductory computer science. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education*, pages 191–195, 2003.

[5] V. Dagiene, L. Mannila, T. Poranen, L. Rolandsson, and P. Söderhjelm. Students' performance on programming-related tasks in an informatics contest in finland, sweden and lithuania. In *Proceedings of the 2014 Conference on Innovation; Technology in Computer Science Education*, ITiCSE '14, pages 153–158, New York, NY, USA, 2014. ACM.

[6] W. Dann, S. Cooper, and D. Slater. Alice 3.1 (abstract only). In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, pages 757–757, 2013.

[7] L. P. Flannery and M. U. Bers. Let's dance the "robot hokey-pokey!" children's programming approaches and achievement throughout early cognitive development. *Journal of Research on Technology in Education*, 46(1):81–101, 2013.

[8] L. P. Flannery, B. Silverman, E. R. Kazakoff, M. U. Bers, P. Bontá, and M. Resnick. Designing scratchjr: support for early childhood learning through computer programming. In *Proceedings of the 12th International Conference on Interaction Design and Children*, pages 1–10. ACM, 2013.

[9] J. Goode, G. Chapman, and J. Margolis. Beyond curriculum: The exploring computer science program. *ACM Inroads*, 3(2):47–53, June 2012.

[10] E. Kazakoff, A. Sullivan, and M. Bers. The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, 41(4):245–255, 2013.

[11] S. Magnenat, J. Shin, F. Riedo, R. Siegwart, and M. Ben-Ari. Teaching a core cs concept through robotics. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, ITiCSE '14, pages 315–320, New York, NY, USA, 2014. ACM.

[12] L. Morgado, M. Cruz, and K. Kahn. Preschool cookbook of computer programming topics. *Australasian Journal of Educational Technology*, 26(3):309–326, 2010.

[13] K. Redmond, S. Evans, and M. Sahami. A large-scale quantitative study of women in computer science at stanford university. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, SIGCSE, pages 439–444, New York, NY, USA, 2013.

[14] V. Rideout. Zero to eight: Children's media use in america 2013. *Pridobljeno*, 11(1):2014, 2013.

[15] A. Tucker. A model curriculum for k–12 computer science: Final report of the acm k–12 task force curriculum committee. Technical report, New York, NY, USA, 2003. ACM Order No.: 104043.

[16] C. Wilson. Hour of code: We can solve the diversity problem in computer science. *ACM Inroads*, 5(4):22–22, Dec. 2014.