Unravelling the Cognition of Coding in 3-to-6-year Olds

The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language

Eva Marinus

ARC Centre of Excellence in
Cognition and Its Disorders,
Department of Cognitive Science,
Macquarie University, Sydney,
NSW, Australia,
eva.marinus@mq.edu.au

Zoe Powell

ARC Centre of Excellence in
Cognition and Its Disorders,
Department of Cognitive Science,
Macquarie University, Sydney,
NSW, Australia,
zoe.powell@mq.edu.au

Rosalind Thornton

ARC Centre of Excellence in
Cognition and Its Disorders,
Department of Cognitive Science,
Macquarie University, Sydney,
NSW, Australia,
rosalind.thornton@mq.edu.au

Genevieve McArthur

ARC Centre of Excellence in Cognition
and Its Disorders, Department of
Cognitive Science, Macquarie
University, Sydney, NSW, Australia,
genevieve.mcarthur@mq.edu.au

Stephen Crain

ARC Centre of Excellence in Cognition
and Its Disorders, Department of
Cognitive Science, Macquarie
University, Sydney, NSW, Australia,
stephen.crain@mq.edu.au

ABSTRACT

There is growing interest in teaching children computer programming ("coding") to prepare them for the demands of our increasingly digital society. However, we do not yet understand what cognitive skills children need in order to learn to code. The aim of our research program is to identify the requisite skills, with the goal of building a cognitive model of coding. The present research used a wooden robot ("Cubetto", www.primotoys.com) to investigate coding ability in young children. Exp. 1 describes the development and evaluation of the assessment instrument, which was tested with 18 3-to-5-year-old children. The instrument ("Coding Development (CODE) Test 3-6") was used in Exp. 2 to investigate the relationship between coding skill and "cognitive compiling" - the ability to formulate mental action plans in natural language. Thirty 5-to-6-year-olds participated in Exp. 2. Using Bayesian statistics, we found evidence that cognitive compiling predicts coding performance above and beyond age and nonverbal intelligence. We evaluate the outcomes and reflect on whether cognitive compiling depends solely on maturation or might be a skill that can be trained, and if so, how this could be done.

CCS CONCEPTS

- Social and professional topics-Student assessment

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the Owner/Author.

ICER '18, August 13–15, 2018, Espoo, Finland

© 2018 Copyright is held by the owner/author(s).

ACM ISBN 978-1-4503-5628-2/18/08.

<https://doi.org/10.1145/3230977.3230984>

KEYWORDS

Assessment, programming, coding, natural language ability, cognitive compiling, nonverbal intelligence, children, preschool, kindergarten

ACM Reference format:

Eva Marinus, Zoe Powell, Rosalind Thornton, Genevieve McArthur and Stephen Crain. 2018. Unravelling the cognition of coding in 3-to-6-year olds. In *Proceedings of ACM International Computing Education Research (ICER) conference*, Espoo, Finland, August 2018 (ICER), 9 pages. DOI: 10.1145/3230977.3230984

1 INTRODUCTION

In recent years there has been a growing interest in encouraging and teaching students to learn computer programming ("coding") in order to prepare them for the demands of our increasingly digital society. As a result, there has been an explosion of research in the field of computer science education (e.g., the steady increase in submissions to SIGCSE conferences), and there have been many global and local programming initiatives (e.g., Code.org [4] and Code Club [5]). Countries such as the UK, Estonia and Australia have started to introduce coding and computational thinking into their national K-12 curricula.

In order to optimize the instruction of coding and computational thinking in children, it is critical to understand the cognitive processes underlying these skills. In the curriculum context in particular, we need to know how coding skills and computational thinking develop over time and how they can be assessed [8,7,10,30]. This knowledge will help us to identify and support children who struggle to learn computational thinking and coding (i.e., dyscodia).

A recently commissioned Google survey of the field of pre-college computer science education (CSEd) concluded that CSEd research must become more rigorous and connect more with new and established knowledge in cognitive science, learning sciences and machine learning [1]. These fields of research seemingly have great potential to boost the study of coding and computational thinking, building on the strong foundation that has been established by the field of Computer Science Education over the last decades. The field of Cognitive Science in particular can expand this knowledge via the development of cognitive models of coding and computational thinking using techniques that have produced valid cognitive models in other domains, like reading [6], language, [20], and belief formation [19].

In this research paper, we focus on coding in 3-to-6-year-old children. Our aims were to develop an assessment instrument (Experiment 1), and to determine if there is a relationship between coding ability and cognitive compiling of natural language phrases (Experiment 2). The outcomes of these studies represent early steps towards the ultimate goal of building a valid cognitive model of the acquisition of coding and computational thinking.

2 EXPERIMENT 1: DEVELOPING A CODING ASSESSMENT TOOL FOR 3-6-YEAR OLDS

2.1 Background

The aims of our first experiment were to develop a standardized cognitive assessment of coding for children aged from 3 to 6 years, and to use this assessment to determine if there are individual differences in children's coding skills at this young age (see [3] for a similar approach). We were also interested to find out if these skills improve with development. We developed our assessment tool based on "Cubetto" [24] - a small wooden robot who needs help to find its way on a map. Cubetto is a simplified version of the turtle LOGO programming task developed by Seymour Papert [23] that is suitable for children from 3 to 7 years of age according to its developers [24]. To the best of our knowledge, we are the first research team to develop such an assessment using a tangible robot, and the first to develop an assessment for such a young age group. Previously developed tests typically focus on older children and have used digital environments (e.g., the Computation Thinking test [27], Fairy Assessment in Alice [32], Computational Thinking Patterns in AgentSheets [18], Dr. Scratch [22], Foundations for Advancing Computational Thinking [9], classifying programming responses with SOLO [29]). The only comparable effort with young children that we are aware of was that of Bers [31]. However, she used portfolio and curriculum style-assessments, not a direct measurement of programming in action.

2.2 Methods

2.2.1 Participants. We obtained active consent from the parents of 18 children (11 boys). Before assessment, we also obtained verbal assent from the children. The sample consisted of eight 3-year-olds, eight 4-year-olds and two 5-year-olds. Children ranged from 3 years and 3 months to 5 years and 10 months in age (average age

= 4 years and 1 month). Sixteen participants were recruited at a childcare centre on our university campus. The other two children were recruited via an online recruitment website developed at our research centre. All procedures in this study were approved by the Ethics committee at our university (Reference number: 5201600786).

2.2.2 Materials.

2.2.2.1 Coding Development (CODE) Test 3-6 – version 1. The materials for Cubetto are a 1m² grid mat with 6x6 squares, Cubetto (a cube-shaped wooden Robot; see Image 1), and a "remote control" that children control by inserting small wooden blocks into holes (see Image 2). We only used the green (one step forward), yellow (90-degree left turn), and red (90-degree right turn) "instruction blocks" (see Image 2). We developed an assessment protocol that asked children to use these blocks to complete 13 tasks ("items") of increasing difficulty. For every item, children were asked to move Cubetto to a picture on the mat (e.g., the water).

Image 1: Left: Cubetto on the map with pictures. Right: Cubetto with hat and tail, friend, and fruit

We piloted this initial coding assessment with a 3-year-old and a 5-year-old. This pilot revealed four issues. First, for the 3-year-old child, we needed to improve her understanding of our instructions by facilitating her sense of direction. To this end, we (1) gave Cubetto a hat and tail to make clearer which way he was facing (see Image 1, right side); (2) gave all instructions while the child was sitting behind Cubetto (Note: the child and instructor had to move around the mat with all the materials to ensure that the child kept facing Cubetto's tail); (3) put yellow and red stickers on each child's left and right hand respectively, since many 3-6-year olds do not know left from right [19]. Note: the red and yellow colors corresponded with yellow (left) and red (right) instruction blocks in the remote control.

Second, instead of asking the child to move Cubetto to certain pictures on the map, we introduced the children to a small toy who was Cubetto's friend to make the task seem more dynamic and interactive. Children were asked to join the friend in different activities (e.g., tree climbing) or to 'look at' the friend if the instruction required the child to move Cubetto on a coordinate next to the friend (see Image 1 at the right for a demonstration). We also had some toy fruits at hand that we would ask the children to give to Cubetto, either when we felt they needed a little break or as a smooth way to end the testing session ('reward Cubetto for his hard work') (see also Image 1 at the right).

Third, we adjusted the Cubetto remote control. This control comprises 12 holes for regular instruction plus four holes for a function (see Image 2 on the left). After the fourth hole, the sequence line makes a 180 degree turn, which effectively reverses the direction of the colored blocks. This is very difficult for 3- to 4-year-olds to understand. Thus, we could only use the first four holes to create our items. Fortunately, we were able to generate enough items of increasing difficulty using the first four gaps only. To avoid distractions, we covered all the other gaps (see Image 2 on the right).

Fourth, our younger pilot child struggled to fit the blocks into the holes in the remote control. The delay caused by this led to the child forgetting the instructions for the items. We therefore attached a small transparent handle to each block to make it easier for the child to fit the block in the hole. This had an added benefit of providing the child with an extra cue about which direction the block had to face in order to fit into a hole.

It also became clear that children needed more instructions for each item and needed more than one attempt per item. We revised all the items in the assessment accordingly.

Image 2: Left: Instruction blocks in 'remote control'. Right: with cover and handles on blocks.

2.2.2.2 Coding Development (CODE) Test 3-6 – final version.

The final version of the coding assessment began by familiarising the child with Cubetto and the workings of the remote control. We also assessed if the child could count to four, and whether s/he knew left from right. We assessed each child with the CODE Test 3-6, irrespective of their fail or pass on these assessments.

After the instructions, the child moved on to the test items. The first and easiest item required the child to make Cubetto make one step forward (i.e., insert one green block). The final and hardest item asked the child to make Cubetto collect his friend and bring him to the mountains. This item involved one step forward, turning 180 degrees (i.e., two yellow or red blocks) and another step forward.

There were two types of items: "build" items (10 in total), in which a child was given an empty remote control and asked to create Cubetto's movements from scratch; and "debug" items (3 in total), in which a child was asked to fix Cubetto's existing movements (i.e., debugging). A child had three chances to get each item correct. If their first attempt was correct, then the child was assigned 3 points for that item (Step 1). If a child failed, s/he would be invited to try again. If successful, s/he would earn 2 points for that item (Step 2). If a child failed, s/he was given additional standardized instructions to help the child to solve the item (Step

3). If a child still could not solve the item, s/he was given zero points, and the instructor demonstrated how to do the item correctly (Step 4). This "three-step" approach for each item increased the range of scores from 0-13 to 0-39, allowing for more fine-grained measuring of coding ability. After testing 5 participants, we noticed that they never earned any points after failing on 3 consecutive items. We therefore started using 3 errors in a row as our stopping criterion to reduce the length of the assessment for the child. The final protocol and scoresheet, which include all items and the instructions for the three Steps for both Build and Fix items, can be found on the Open Science Framework (osf.io). See Section 3.2 of Experiment 2 for a link.

2.2.3 Procedure. Sixteen participants were tested at the childcare centre, and two were tested in the language acquisition lab at our university. The sessions in the childcare centre took place in June and July 2017. The other two children were tested in May 2017 and January 2018. The first and second author conducted the testing sessions, which each took between 20 and 40 minutes.

2.3 Outcomes

The raw data of Experiment 1 can be found on the Open Science Framework. See Section 3.2 for the link.

Our first step was to look at the distribution of scores. As can be seen in Figure 1, the scores followed a normal distribution. The average score for our sample was 14.4 with a standard deviation of 7.5. The scoring range was 0-28.

Second, we looked at the relationship between coding ability and age. We found a significant positive correlation, $r = .75$, $p < .001$. Only one child scored at floor (i.e., 0), who was one of the youngest participants (See Figure 2).

Figure 1: CODE Test 3-6 score distribution Experiment 1, 3-5-year olds

Finally, we examined the scoring patterns of the children across the items and examined whether they benefitted from the three-step approach for each item. As can be seen in Figure 3, accuracy diminished gradually over the items, which suggests that the order of difficulty was appropriate. This was supported by an analysis of the three-step approach for each item (see Figure 3). For example, 10 children needed just one step for Item 1, six

needed two steps, and one child needed three steps. In later items (e.g., Item 5), fewer children needed just one step (e.g., five children for Item 5), and more children needed two steps (seven children for Item 5) or three steps (three children for Item 5). Again, this suggests that the order of difficulty was well chosen.

Figure 2: Scatterplot CODE Test 3-6 score and age

2.4 Discussion

We succeeded in creating a preliminary cognitive assessment for coding ability for 3-5-year olds. The results show a good distribution of skill and no ceiling effects. The scores increased with age, and the three-step procedure for each item worked appropriately. The fact that none of the children - not even the 5-year-olds - scored at ceiling suggested that the assessment instrument would be appropriate for a slightly older population. And it would be appropriate for use in studies that aim to identify the cognitive skills that underpin coding ability in children aged from around three to six. Limitations will be discussed in the General discussion.

Figure 3: Scoring patterns across items and steps, y-axis represents number of children who used a Step

3 EXPERIMENT 2: THE RELATION BETWEEN CODING, AND COGNITIVE COMPILING ABILITY IN 5-6-YEAR OLDS

3.1 Background

To the best of our knowledge, the only studies that have attempted to identify the cognitive processes that underpin the ability to code in children between 3 and 6 years of age have been conducted by Kazakoff and colleagues [17, 16]. These studies targeted sequencing skills as measured by assessing performance on picture sorting. They used an intervention design. Children were measured on sequencing ability before and after a week of programming training involving robots. The results were mixed. The 2012 study reported a larger improvement of sequencing skill by the trained group as compared to the control group, whereas the 2013 study failed to replicate this finding.

One limitation of these pioneering studies is that they did not measure coding ability per se and hence could not investigate the strength of the relationship between coding and sequencing skill. In Experiment 2, the coding assessment that we developed and trialed in Experiment 1 was used to measure the potential relationship between coding ability and cognitive compiling in natural language. With cognitive compiling we refer to the formulation of mental plans to execute actions that are denoted by natural language instructions such as “Point to the third green ball.”

We focused on natural language phrases such as “third green ball” for several reasons. First, 5- to 6-year-old children have been found to formulate non-adult action plans in response to instructions that contain such phrases. Despite knowing the meanings of the individual words “third” and “green” and “ball”, children in this age range often fail to point to the third green ball in an array; rather, they point to the third ball in the array, if it is green. Similarly, we found in Experiment 1 that children often used the correct blocks, but in the wrong order. In addition, many of the children, especially the younger ones, preferred to take a sequential approach (i.e., using one block at the time) to bring Cubetto to the target location on the map. We hypothesize that these strategies, or preferences, indicate variability in ability to accurately represent all the different steps in a cognitive action plan. This hypothesis is supported by previous research reporting that children struggle to keep track of events, variables and states that cannot be seen [2, 11]. Programming a robot to make specific moves over a map requires the formation of a specific mental plan involving all the steps and moves in the correct order. The ability to successfully form such plans is the essence of cognitive compiling.

To develop an independent measure of cognitive compiling, we turned to the literature on linguistic skills in young children. In the 1980s, Hamburger and Crain investigated the acquisition of cognitive compiling in natural language [12,13]. Their results showed that young children (ranging in age between 4;4-6;2) are far more able to formulate complex action plans than had been found in previous research [21]. Hamburger and Crain showed that children’s successful performance dramatically increased

when specific steps were taken to simplify the cognitive load that is required to build an appropriate algorithm or plan for phrases such as “third green ball.” In addition to behavioral data, Hamburger and Crain also provided pseudocode ‘programs’ that visualized the different ‘compiling’ stages or phases at which a child could make ‘errors’ in building or executing correct, adult-like plans.

Our aim was to determine if there are individual differences in cognitive compiling in a slightly older sample of children. Although the children in the present study were older than those in the Hamburger and Crain studies, we assumed that some child participants would nevertheless struggle to go through all the steps of cognitive compiling. In addition, instead of relieving the mental load for the children, we chose the most complex task compiling task and offered it without the scaffolding that reduced cognitive load in the Hamburger and Crain studies. We know from previous research that the complexity of a plan significantly increases if elements (e.g., color, shape and number) are combined [12, 13]. Therefore, our task involved asking a child to point to objects presented in arrays that included three elements (position, color and object type) that had multiple operations between each element. For instance, a child was asked to point to the third yellow star that was presented in an array of items. To get this item right, the child has to “compile” third and yellow and star together into a single representation. This involves finding the set of stars, check that they are yellow and then pick the third one of this set (see Image 3).

We used our coding assessment and our cognitive compiling assessment to test three hypotheses. First, based on Experiment 1, we hypothesized that coding ability would be positively correlated with age. Second, based on previous work of Hamburger and Crain, we predicted a positive correlation between cognitive compiling ability and age. Third, we hypothesized that coding performance could be predicted by cognitive compiling despite controlling for both age and non-verbal intelligence. We controlled for the latter since coding could simply depend on general cognitive ability that increases with age and non-verbal intelligence.

3.2 Methods

Prior to conducting this experiment, we preregistered it on the Open Science Framework (OSF). Please see here: osf.io/e3rku. By preregistering we committed to a research plan in advance, before collecting data. This allows for separating hypothesis-generating (exploratory) and hypothesis (confirmatory) research. Making an explicit distinction between the two improves the quality and transparency of research and helps others to build on it. In particular by also making the data and materials available. Therefore, the raw data for Experiments 1 and 2 and the materials for the CODE Test 3-6 and the cognitive compiling test can also be found on the same OSF page.

3.2.1 Participants. The sample consisted of 20 5-year-olds and 10 6-year-old children (16 boys). Children ranged from 5 years and 0 months to 6 years and 10 months in age (average age = 5 years and 8 months). The participants were recruited via an online

recruitment website developed at our research centre. The parents of the participants were sent an email including an information letter and a consent form. The lead researcher arranged a time for testing with the parents who gave their written consent. The testing sessions were held on-site at our university. Parents received \$15 to cover the costs of travel, and children received their choice of either \$10 or a toy of equivalent value in compensation. All procedures in this study were approved by the Ethics committee at our university (Reference number: 5201600786).

3.2.2 Materials.

3.2.2.1 The Coding Development Test 3-6. We used the final version of the CODE Test 3-6.

3.2.2.2 Cognitive compiling task. The cognitive compiling task aimed to determine the child’s ability to ‘compile’ pieces of syntactic information. To do this, we asked the children to point at an object (e.g., ‘the second yellow star’) in an array presented on a laminated piece of white paper. For all items we made sure that the children were presented with the array of objects before we asked the question. There were 3 pre-test arrays and 16 test arrays.

The pre-test arrays were used to ensure the child could name the colors and shapes used during the testing session, to familiarise the child with the process of counting from left to right and to establish that they knew the meaning of the ordinals ‘first’, ‘second’, ‘third’ and ‘last’. Two participants did not successfully complete the pre-test and were thus excluded from the analyses that involved the cognitive compiling task.

Each of the 16 test arrays consisted of 7 objects that could be one of two shapes and one of two colors (e.g., blue squares, blue circles, red squares, and red circles). We created 8 target arrays and 8 filler arrays (i.e., items that will not be analysed). Therefore, the maximum score for this task was 8. The target arrays were designed to distinguish between the adult (from here on, ‘correct’) answer, and errors that we expected 5-6-year-old children to make.

The most common error we expected was the intersective error. This type of error occurs when the answer focuses foremost on the positioning of the object (‘third’), and secondarily on whether the object matches the description (e.g. ‘blue square’). So, in the instance of the ‘third blue square’, the intersective error is the third object from the left in the array and will be selected if the object is a blue square. Each of our target arrays included a valid (i.e., meets the descriptive criteria, in this instance a ‘blue square’) intersective and correct answer. The other previously identified error was a ‘head-set’ error, which would occur if the answer is based predominantly on the noun (e.g., find the third square in an array, and select it if it is blue). We also explored the possibility that children may decide on answers based predominantly on the adjective (i.e., the color), in an ‘adjective-set’ error (e.g., find the third blue object in an array and select it if it is a square). For all of our target arrays, the correct answer could be distinguished from each of these three error types. See Image 3 for examples.

The eight filler arrays had the same format (i.e., 7 objects varying in shape and color). Four of the filler arrays were designed to be easy (e.g., ‘the first red star’) to make sure that all children

felt successful on at least some of the items, and an additional four arrays that were slightly harder, but not informative to what kind of strategy the children were using.

Image 3: Examples of target arrays. I = Intersective, H = Headset, A = Adjective, C = Correct solution.

3.2.2.3 Nonverbal intelligence task. The Matrices subtest of the Kaufman Brief Intelligence Test (2nd Edition) [15] was used to assess non-verbal intelligence. The Matrices subtest consists of 46 items ordered in increasing difficulty. For each item, a child is presented with picture printed at the top of a page that also illustrates five small pictures at the bottom of the page. One of the small pictures "matches" the picture at the top of the page. Items 1 to 9 present a complete image (e.g. a picture of a shoe, with the correct answer being an image of a foot). From item 10 onwards, the image is missing a part (e.g., a picture of a pattern with a missing section, with the correct answering being the missing section). The Matrices subtest of the KBIT-2 has demonstrated acceptable internal consistency ($\alpha = .78$ in 4- and 5-year olds, $\alpha = .87$ in 6-year olds) and acceptable test-retest reliability (adjusted $r^2 = .76$ over 4 weeks in 4- to 12-year-olds). The administration and discontinuation rules (4 incorrect answers in succession) outlined in the manual were followed. The raw scores were used for further analysis.

3.2.3 Procedure. Testing was carried out by the first two authors and an additional research assistant in the language acquisition lab at our university. The additional research assistant was trained by the first author. In addition to the active consent obtained from the parents, we first briefly explained the nature of the tasks to the children and obtained verbal assent of each child. Following this, the children completed the CODE Test 3-6, which took approximately 25 minutes. With the consent of both parent and child, the administration of the CODE Test 3-6 was filmed for the purposes of scoring and qualitative analysis. They were then given a 5-minute break where they had the opportunity to play with Cubetto. After this, the cognitive compiling task was administered (about 5 minutes), followed by the matrices subtest of the KBIT-2 (about 10 minutes). The entire duration of testing was therefore approximately 45 minutes. The children were tested between mid-December 2017 and early March 2018.

3.3 Outcomes

The raw data for Experiment 2 can be found on the Open Science Framework. See Section 3.2 for the link.

The mean scores of our 30 5- to 6-year-olds was 26.4 on the CODE Test 3-6, with a standard deviation of 7.0. None of the participants scored at floor or at ceiling. The scoring range was from 12 to 38. Two participants failed on the pre-test of the cognitive compiling task. Therefore, all planned analyses that include the cognitive compiling test will have an $n=28$ instead of 30. The mean score of our sample on the cognitive compiling test was 4.75 with a standard deviation of 3.57. The scoring range was 0-8.

As specified in our preregistration, we used a Bayesian approach to analyse our data. Since we are the first to test correlations between coding ability and cognitive compiling, it was hard to conduct a power analysis to estimate the required sample size. We therefore decided to use Bayesian statistics, which allow for analyzing after collecting data with small sample and analyzing again after increasing the sample size.

All our tests were one-tailed as we had specific predictions about the (positive) direction of correlations between factors. In our dataset, Bayes Factors below 0 indicate evidence for the Null Hypothesis (e.g., there is no positive correlation between CODE Test 3-6 performance and age). Bayes Factors above 0 indicate evidence for the Alternative Hypotheses (e.g., there is a positive correlation between CODE Test 3-6 performance and age). However, Bayes factors between 0 and 3 are typically considered to be equivocal evidence. Bayes factors between 3 and 10 are considered to give some evidence, with BFs above 10 to pose strong evidence and above 30 very strong evidence [28].

Our first hypothesis was that we would replicate the positive relationship between coding ability and age. Indeed, we found a positive correlation, $r = .57$, with a Bayes factor of 75.6 ($n=30$). This means we found very strong evidence for a positive correlation between coding ability and age. See Figure 4.

Figure 4: Scatterplot CODE Test 3-6 score and age

Our second hypothesis was that we would replicate the finding of Hamburger and Crain that demonstrates a positive relation between cognitive compiling skill and age. Indeed, we found a positive correlation between age and cognitive compiling, $r = .44$, with a Bayes factor of 6.4 ($n=28$). This means that there is some

evidence that the older children tend to use to pick the correct solution, whereas the younger children tend to pick in the intersective solution. See Figure 5.

As can be seen in Figure 5, our cognitive compiling test did not result in a continuous outcome. Instead, the children could be divided into three different categories based on their strategy. First, 12 children obtained a full score. This group clearly mastered the complex compiling strategy needed to properly execute the request. Second, 7 children exclusively picked the ‘intersective’ solution. As explained above this meant that when asked to point to the second yellow ball, they would focus on the second object in the array and point at this when it was a yellow ball, even if it was not the second yellow ball in the array. Finally, there was a group of 9 children that scored somewhere in between. These children sometimes picked the correct item, but not consistently. Some children made a mix of intersective and correct decisions, others picked a range of different answers.

Figure 5: Scatterplot Cognitive compiling score and age

Our third hypothesis was that cognitive compiling would predict coding performance whilst controlling for both age and non-verbal intelligence. As specified in our preregistration, we conducted Bayesian regression analyses and we started by running the regression analysis on the first 20 participants. As we are using a Bayesian approach, the outcome will tell us whether we find more support for our hypothesis that cognitive compiling significantly predicts performance on the CODE Test 3-6 over and beyond nonverbal intelligence and age or that it is more likely that this predictive relationship does not exist. A third option is that more data is needed to judge which possibility gets more support.

We ran our statistical analyses in JASP [14] using the Bayesian Linear Regression option. CODE Test 3-6 score was added as the dependent measure and Age (using one decimal) and Nonverbal intelligence (raw score) were added to the Null model before adding Cognitive compiling as the predictor. The Bayes factor for this model was 1.77. This means that there is about 1.8 times more evidence for a model that also includes cognitive compiling skill than a model that only includes nonverbal intelligence and age as

predictors. Bayes factors between 0 and 3 imply that we either have to collect more data or should develop a more sensitive measure. As specified in our preregistration we added our next 8 participants. The Bayes factor now amounted to 4.7, providing some evidence that cognitive compiling is a meaningful predictor of coding ability, over and beyond age and nonverbal IQ.

4 GENERAL DISCUSSION

In Experiments 1 and 2 we discovered that it is possible to measure coding ability in 3-6-year olds and that coding ability in this age group increases significantly with age. Confirming the predicted positive correlations between nonverbal intelligence and programming ability, and between children’s performance on the assessment of cognitive compiling and programming ability provided the requisite initial steps in validating our coding assessment task. Obviously, however, there is a need for further replication and validation [25]. In this context it is important to realise that the field is still working on establishing clear definitions of coding skill and its underlying cognitive skills in (young) children. As a result, further validation will need to go hand in hand with developing more specific definitions of the skills involved.

Our research clearly shows that, with appropriate scaffolding, children as young as 3 years of age can successfully use programming tools like Cubetto. We simplified the task in several ways, most notably by improving children’s sense of direction. This was an important simplification for children younger than the age of 5, because they do not typically distinguish right from left [26]. Our results show that coding performance depended on children’s ability to correctly compile the multiple components of a program. An interesting question is whether cognitive compiling of multiple programming components is a skill that can be trained or that it is a matter of maturation. If children’s abilities are determined by maturation, it may turn out to be necessary to wait until children for children to mature (e.g., around the age of 5) before confronting them with tasks that involve compiling larger movement sequences. Concretely, they could be instructed to work sequentially, with one block at the time, to make Cubetto move over the map.

Another approach would be to train children in the requisite skills that are involved in cognitive compiling. As described in Section 3.1, Hamburger and Crain [12,13] found that the ability of children to formulate the kinds of mental plans that are required in order to execute instructions in natural language (e.g., “Point to the third green ball”), dramatically increased when specific steps were taken to simplify the cognitive load required to build an appropriate algorithm corresponding to such instructions. In their study they discovered two possible ways to facilitate children’s compiling of mental action plans. First, they found that children’s ability to formulate adult-like plans improved by providing the linguistic instructions first, before revealing the array. This ‘phrase before display’ manipulation helped children to forestall the premature execution of the piece of the plan that corresponds to the ordinal “third.” As a result, the children opted less often for the intersective solution. Second, they found that performance could be improved by asking the children to sort and count sets of

objects of the same color or shape. The idea is that pre-handling sets of objects helped the children to become aware that ‘second’ or ‘third’ is not necessarily tied to a position in an array but could be used to refer to one of the subsets of objects, for example, the green balls.

So what kind of similar strategies could be undertaken to improve cognitive compiling in programming a robot? One way to scaffold children’s composition of complex mental action plans could be by encouraging children to verbalize the steps of their plans by thinking out loud. During Experiment 2 we made several observations of 5 and 6-year olds spontaneously verbalising their mental plans before inserting the blocks in the remote control. In parallel to the ‘phrase before display’ solution by Hamburger and Crain, the trainer could go one step further by taking away the remote control and blocks and ask the children to verbalise the type of actions (‘forward’, ‘turn left’, ‘turn right’) in the correct order before they put the blocks in the remote control. It is less straightforward to identify a similar parallel between the handling strategy for a task with arrays and a programming task with blocks and a remote control. One possibility is that children would benefit from systematic practice with the different types of blocks. For instance, children could be asked to use just one block at the time, pressing the blue button multiple times to reach their goal, before they progressed to putting in multiple blocks in the same program. Future training studies are needed to determine 1) how to effectively train cognitive compiling in children, 2) whether such training also leads to improved coding skills, and 3) whether potential improvements in coding skills hold in the absence of scaffolding strategies.

5 CONCLUSION

We developed a coding ability assessment (“Coding Development Test for children from 3-6 (CODE Test 3-6)”) for children from 3 to 6 years of age. The instrument shows a good spread of scores. Performance of the test is positively correlated to age, nonverbal intelligence and cognitive compiling of syntax in natural language. Our second experiment showed that there is evidence that cognitive compiling predicts coding ability in young children over and beyond age and nonverbal intelligence.

ACKNOWLEDGEMENTS

We would like to thank the director, teachers and children from childcare centre Gumnut Cottage at Macquarie University for participating in Experiment 1 and all the children who participated in Experiment 2 and their parents for bringing them to our language acquisition lab. Thank you, Marion Aitchison, for your help with the data collection of Experiment 2. This research was funded by the ARC CCD Cross Program Support Scheme. The first author’s salary was paid by funding from the ARC Centre of Excellence for Cognition and its Disorders (CCD) (CE110001021), www.ccd.edu.au.

REFERENCES

- [1] Paolo Blikstein. 2018. Pre-College Computer Science Education: A Survey of the Field. Mountain View, CA: Google LLC. <https://services.google.com/fh/files/misc/pre-college-computer-science-education-report.pdf>
- [2] Benedict du Boulay. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2, 1, 57–73. DOI: 10.2190/3LFX-9RRF-67T8-UVK9
- [3] Lucía Gabriela Caguana Anzoátegui, María Isabel Alves Rodriguez Pereira, e Mónica Del Carmen Solís Jarrin, 2017. Cubetto para pre-escolares: programación informática código a código. *Proceedings of the XIX International Symposium on Computers in Education*. 114-118. https://www.eselx.ipl.pt/sites/default/files/media/2017/siie-cied_2017_atas-compressed.pdf
- [4] Code.org. 2018. <https://code.org>
- [5] Code Club Australia. 2018. <https://codeclubau.org>
- [6] Max Colthart, Kathleen Rastle, Conrad Perry, Robyn Langdon, and Johannes Ziegler. 2001. DRC: a dual route cascaded model of visual word recognition and reading aloud. *Psychological review* 108, 204-256. DOI: <http://10.1037/0033-295X.108.1.204>
- [7] Katrina Falkner and Rebecca Vivian. 2015. Coding across the curriculum: Resource review. Australian Government: Department of Education and Training, 1-52. https://digital.library.adelaide.edu.au/dspace/bitstream/2440/95629/3/hdl_95629.pdf
- [8] Shuchi Grover and Roy Pea. 2013. Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, 42, 1, 38-43. DOI: <https://doi.org/10.3102/0013189X12463051>
- [9] Shuchi Grover, Roy Pea, and Stephen Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer science education*, 25, 2, 199-237. DOI: <http://dx.doi.org/10.1080/08993408.2015.1033142>
- [10] Shuchi Grover and Satabdi Basu. 2017. Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. *Proceedings of the 48th ACM Technical Symposium on Computer Science Education (SIGCSE '17)*. Seattle, WA: ACM. DOI: <https://10.1145/3017680.3017723>
- [11] Mark Guzdial. 2015. Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, 8, 6, 1–165. DOI: <https://doi.org/10.2200/S00684ED1V01Y201511HCI033>
- [12] Henry Hamburger and Stephen Crain. 1984. Acquisition of cognitive compiling. *Cognition*, 17, 2, 85-136. DOI:10.1016/0010-0277(84)90015-5
- [13] Henry Hamburger and Stephen Crain. 1987. Plans and semantics in human processing of language. *Cognitive Science*, 11, 101-136. DOI: <https://doi.org/10.1111/j.1551-6708.1987.tb00864.x>
- [14] JASP Team. 2018. JASP (Version 0.8.6) [Computer software]
- [15] A. S. Kaufman and N. L. Kaufman. 2004. Kaufman Assessment Battery for Children Second Edition. Circle Pines, MN: American Guidance Service
- [16] Elizabeth R. Kazakoff, Amanda Sullivan, and Marina Umaschi Bers. 2013. 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. DOI: <https://doi.org/10.1007/s10643-012-0554-5>
- [17] Elizabeth R. Kazakoff and Marina Umaschi Bers. 2012. Programming a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia*, 21, 4, 371-391. Waynesville, NC USA: Association for the Advancement of Computing in Education (AACE). Retrieved March 26, 2018 from <https://www.learntechlib.org/p/39512/>
- [18] Kyu Han Koh, Ashok Basawapatna, Hilarie Nickerson, and Alexander Repenning. 2014. Real time assessment of computational thinking. *Visual Languages and Human-Centric Computing*. IEEE Symposium, 49-52. DOI: 10.1109/VLHCC.2014.6883021
- [19] Robyn Langdon and Max Coltheart. 2000. The cognitive neuropsychology of delusions. *Mind & Language*, 15, 1, 184-218. DOI: <https://doi.org/10.1111/1468-0017.00129>
- [20] Willem Levelt. 1999. Models of word production. *Trends in cognitive science*, 3, 223-232. DOI: [https://doi.org/10.1016/S1364-6613\(99\)01319-4](https://doi.org/10.1016/S1364-6613(99)01319-4)
- [21] Edward H. Matthei. 1982. The acquisition of the prenominal modifier sequences. *Cognition*, 11, 301-332. DOI: [https://doi.org/10.1016/0010-0277\(82\)90018-X](https://doi.org/10.1016/0010-0277(82)90018-X)
- [22] Jesús Moreno-León, Gregorio Robles and Marcos Román-González. 2015. Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia* 46, 1-23. https://www.researchgate.net/publication/281714025_Dr_Scratch_Automatic_Analysis_of_Scratch_Projects_to_Assess_and_Foster_Computational_Thinking

- [23] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, New York, NY
- [24] Primotoys. 2018. <https://www.primotoys.com>
- [25] Miranda C. Parker, Mark Guzdial, and Shelly Engleman. 2016. Replication, Validation, and Use of a Language Independent CS1 Knowledge Assessment. *Proceedings of the 2016 ACM Conference on International Computing Education Research*. 93-101. DOI: 10.1145/2960310.2960316
- [26] Robert Rigal. 1994. Right-left orientation: Development of correct use of right and left terms. *Perceptual and Motor Skills* 79, 3, 1259-1278. DOI: 10.2466/pms.1994.79.3.1259
- [27] Marcos Román-González, Juan-Carlos Pérez-González, and Carmen Jiménez-Fernández. 2016. Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior* 72, 678-691. DOI: 10.1016/j.chb.2016.08.047
- [28] Jeffrey N. Rouder, Paul L. Speckman, Dongchu Sun, Richard D. Morey, and Geoffrey Iverson. 2009. Bayesian t tests for accepting and rejecting the null hypothesis. *Psychonomic Bulletin & Review*, 16, 2, 225-237. DOI:10.3758/Pbr.16.2.225
- [29] Linda Seiter. 2015. Using SOLO to Classify the Programming Responses of Primary Grade Students. *Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*. 540-545. DOI: 10.1145/2676723.2677244
- [30] Linda Seiter and Brendan Foreman. 2013. Modeling the Learning Progressions of Computational Thinking of Primary Grade Students. *Proceedings of the 2013 ACM Conference on International Computing Education Research*. 59-66. DOI: 10.1145/2493394.2493403
- [31] Marina Umaschi Bers. 2010. The TangibleK Robotics Program: Applied Computational Thinking for Young Children. *Early Childhood Research and Practice*, 12, 2 <http://ecrp.uiuc.edu/v12n2/bers.html>
- [32] Linda Werner, Jill Denner, Shannon Campe, and Damon Chizuru Kawamoto. 2012. The fairy performance assessment: measuring computational thinking in middle school. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, ACM Press, New York, NY, 215-220. DOI:10.1145/2157136.2157200