Patterns and Loops: Early Computational Thinking

Marielle Léonard, Yvan Peter, Yann Secq

Patterns and Loops: Early Computational Thinking

Marielle Léonard[1], Yvan Peter[2], and Yann Secq[2]

[1] France-IOI
mariellelеonard59@gmail.com
[2] Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
{name.surname}@univ-lille.fr,

Abstract. This article presents a large scale quasi-experiment to introduce primary school pupils to Computational Thinking. The aim is to enhance their capability to spot repetitive patterns and to express them as loops. Unplugged and plugged-in activities are used to train the pupils. Trace analysis and pre and post questionnaires were used to measure the impact of the intervention. This article deals with the 2018 session involving 20 classes. The results show a positive impact of the activities and give information about the skills acquired.

Keywords: Computational Thinking, Computer Science Education, Elementary School, Repetitive patterns and loops, Pedagogical Sequence

1 Introduction

Computational thinking (CT) brings abstraction and problem solving skills that can be exploited in many contexts and subject matters at school and in broader contexts. While not being equal to computer science and programming, computational thinking skills definitively lays the ground to explore more computer science related topics such as algorithms. In her seminal article about Computational Thinking, J. Wing advocates for the introduction of CT to non-majors in Computer Science and pre-college students [12]. Since then CT has been considered in many domains like mathematics and experimental sciences [11], arts [6] and even language learning [8].

In this article, we present our work to bring CT skills to pupils in elementary school (8 - 10 years old) in France. Within this project, we have mainly considered the ability of pupils to abstract data, to recognize redundant patterns in data and to express them as a loop structure. This opens the way to systematic and repetitive treatment of data. Towards this end, we have devised a pedagogical sequence including unplugged and plugged-in activities. We will focus in this article on the result from year 2018 which involved 20 classes from 16 schools and 447 pupils. For this quasi-experiment, we have set up pre and post tests to

assess the capability to identify patterns and we have collected traces from their on line programming activities.

This article seeks to answer the following questions:

RQ1 Does the pedagogical sequence presented in this article (section 3) bring an improvement of the pupils cognitive skill of recognizing and expressing repetitive patterns as loop structure ?

RQ2 To what extent do the pupils manage to transfer their skills from one kind of language to the other while solving similar puzzles ?

In the next section, we review existing works about CT, its introduction in pre-college classes and the assessment of CT skills. Section 3 presents the pedagogical sequence we have devised and the experimental setting described in section 4. In section 5 we discuss the analysis of the experimental data before drawing conclusion and perspectives.

2 Related Works

In this section, we review general definitions about Computational Thinking before considering how it is introduced in schools. We then review existing frameworks to assess CT skills.

2.1 Introducing Computational Thinking Concepts

The first concepts of Computational Thinking date back to the work of Seymour Papert with Logo [7]. More recently, the article by Jeannette Wing advocating CT as a primary skill along reading, writing and arithmetic raised a great interest in the education and research community [12]. Wing stresses that CT is not equal to programming but rather the capability to manipulate abstractions and to solve problems that can be applied to many fields. She called for the introduction of CT to pre-college audience.

Since then, the research community has explored ways and means to introduce CT at school : what are the fundamental concepts to teach ? Which technology can support that learning ? Etc. These questions are even more important since many countries have started to update their curricula to introduce these topics at different school levels.

Different works try to organize CT concepts around taxonomies. Gouws *et al.* propose a CT framework that describe skills related to computational thinking [4]. The framework proposes different kind of CT skills learned through programming out of their literature review : *Processes and Transformations*, *Models and Abstractions*, *Patterns and Algorithms*, *Tools and Resources*, *Inference and Logic*, *Evaluations and Improvements*. They combine these skills with a level of mastery inspired by Bloom's taxonomy of learning : *Recognize*, *Understand*, *Apply*, *Assimilate*. The framework can be used as an analysis or design framework. Weintrop *et al.* consider the introduction of CT practices in maths and

science providing the ground for a definition of CT activities away from computer science [11]. The authors define a taxonomy of 22 CT practices grouped into the following categories : *Data and Information*, *Modeling and Simulation*, *Computation*, *Problem Solving* and *Systems Thinking*.

Ching *et al.* rather take a technological entry to the introduction of CT concepts [3]. They provide an analysis of existing readily available technologies for teaching computational thinking. They have identified *robot toys*, *robot kits*, *board games*, *augmented reality tools*, *(visual) programming applications/websites* and *animation/game development tools*. These categories vary by whether it uses physical manipulation or screen interaction and concrete (i.e., robot) or visual feedback. Concepts learned through these technologies range from sequence and loop to more advanced concepts and may imply creativity and problem solving for some of them.

These taxonomies do not necessarily provide insights about the order in which CT concepts should be introduced. Based on a literature survey, Rich *et al.* have started to work on *Learning Trajectories* to define the concepts that can be addressed depending on the grade level and at which level of details . A *Learning Trajectory* is formalized as a set of learning goals, an associated learning path to achieve these goals and illustrative activities. Their literature study shows that many research results focus on a single or independent learning goals. They observe that the same goals have been introduced at multiple grade levels since they usually address inexperienced learners. For this reason, they have relied on maths pedagogical approaches and curricula to propose an ordering of the concepts introduced (learning path). Their article illustrates their approach on three CT concepts : *Sequence*, *Repetition*, and *Conditionals*.

The notion of repetition is one of the fundamental concepts present in all these works. We also believe that pattern recognition and redundant patterns reduction constitute one of the atomic skill in computational thinking, which is why we have focused specifically on this aspect in this study.

2.2 Assessment of CT skills

The assessment of students' skills is an additional dimension of the introduction of CT at school. One can find different approaches in the literature. Brennan & Resnick articulate CT around three dimensions: computational concepts (programming level : loops, parallelism...), computational practices (iterative development, debugging...), and computational perspectives (expressing oneself, connecting to others...) [1]. They propose to assess these dimensions through portfolio analysis, artifact-based interviews and design scenario (projects).

The SRI report by Snow *et al.* considers the means to assess CT skills (problem solving, abstraction...) in the context of the year long high school course "Exploring Computer Science" (ECS) [10]. Towards this end, they propose design patterns to create sound assessments to measure knowledge and practices. The report covers the assessment of the following ECS units : HCI, Problem Solving, Web Design and Introduction to Programming. The assessments include quizzes, problems and code reading and tracing.

Grover *et al.* used formative and summative assessment in the context of a 6 weeks middle school module involving computational concepts [5]. The assessments relied on multiple choice quizzes many of them including Scratch code snippets. Some exercises involved reordering code blocks or code tracing / debugging activities.

Seiter *et al.* propose a framework to assess CT skills in primary grades (1 to 6) called Progression of Early Computational Thinking (PECT) [9]. The framework provides measures based on Scratch programs (use of specific instructions) in the scope of common design patterns (e.g., animation, collision management...). These patterns are then related to CT concepts. The framework has been evaluated against programs found on the Scratch web site.

This later work as well as the approach by Brennan & Resnick are rather time consuming since they involve the study of students' productions potentially in the context of open-ended activities. The other approaches are more tractable since they rely on different kinds of quizzes.

3 Pedagogical Sequence

The sequence we have designed is based on two main inspirations. The first one is the pioneering work of Seymour Papert with his work on Logo [7] and the importance of thinking about the way we think by describing procedures that have to be interpreted by a computer. The second inspiration comes from work done by Jerome Bruner [2] on stages of representation : enactive (action-based), iconic (image-based) and symbolic (language-based). The pedagogical sequence was designed along these stages to support knowledge construction by the pupils.

The pedagogical sequence is presented Fig. 1. This progression includes unplugged and plugged-in activities to support the identification and synthesis of repetitive patterns and their expression in the form of sequence of actions and loops. There are three different phases. The first two ones last one hour and half, the last one takes two hours. The pupils came to university for a whole day. They have the first two phases in the morning and the last one in the afternoon. The different phases are described hereafter.

3.1 Absolute Orientation

In this phase pupils have to move a character on a grid to a given square using absolute directions (North, South, East, West). They start with a board game (see Fig. 2). Pupils work by groups of four to six and take turn at different roles : defining a solution, *program counter* (telling the instruction to perform), and *processor* (executing the instruction). The activity evolves from simple paths to more complex ones with the addition of obstacles and bonuses. When the sequence of instructions starts to get longer, pupils usually start to express frustration. This is the right time to introduce the loop notion (*repeat n times*).

When main concepts of instruction, sequence, loop, execution (and bugs...) are (dis)covered, the pupils switch to a similar activity on tablets by groups of

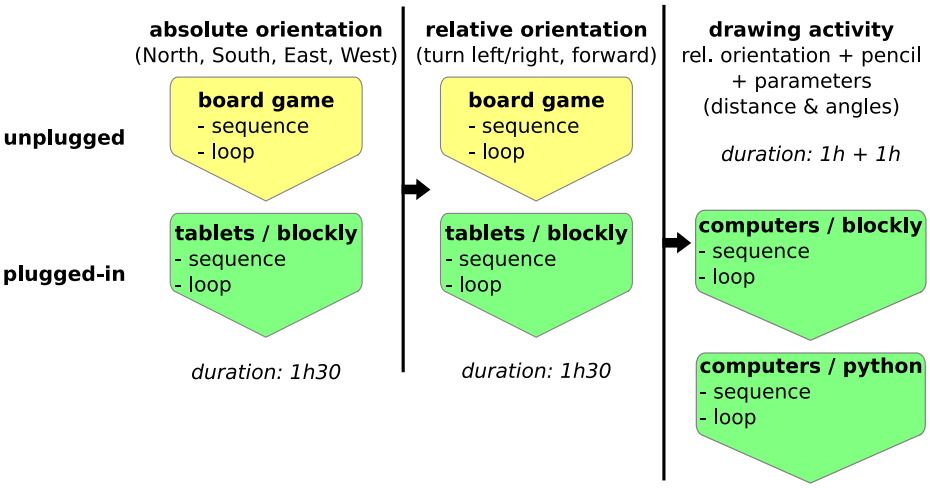

Fig. 1. Pedagogical sequence to train loop recognition and expression

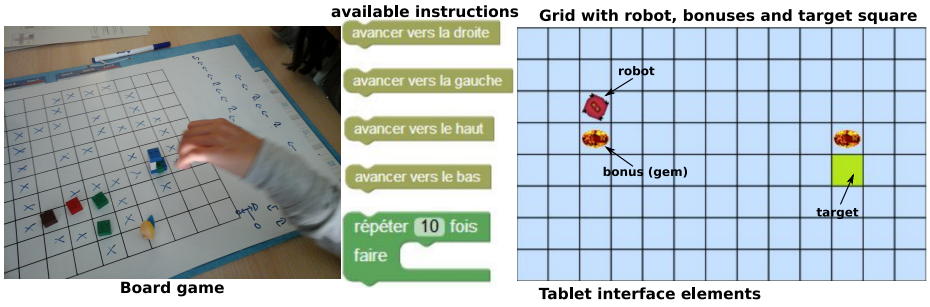

Fig. 2. Board game and tablet interface.

two. They use a visual block-based programming language (*Blockly*). Figure 2 shows one of the puzzles presented on tablets. For each puzzle, there is a specific instruction set provided (Fig. 2). The activities and instructions set evolve again from sequence to loop. The goal of this sequence on tablet is to reinforce learnings done through the unplugged activity and to lead slowly pupils towards autonomy by working by two instead of 4 to 6 in the first phase.

3.2 Relative Orientation

This phase follows a similar organization to the previous one. The main change is that by using an oriented character, the pupils have to handle a different instructions set (*turn left*, *turn right*, *forward*). This also implies remembering the character orientation when planning the moves. Turning is only by 90° and is not parameterized. It prepares the last phase where orientation is necessary for the drawing activity.

3.3 Drawing Activity : Back To Papert's Turtle

The last phase is done in a computer laboratory where each pupil is alone with a computer. The activities are oriented towards drawing with a turtle (in the spirit of Logo). The instructions set is similar to the previous one with addition of the pencil management (putting it up or down to draw) and parameterized functions (e.g. `forward(`*distance*`)` or `turn right(`*angle*`)`). The pupils use the same platform as on the tablets. In the first part, they continue to use the *blockly* block-based programming language. In the second part, we introduce some Python programming making them switch from a graphical to a textual notation within the same context (Fig. 3). To make it easier for the pupils, they use functions translated in French (e.g. `forward(10)` becomes `avancer(10)`), as it is their primary language. Nonetheless, they are introduced to the regular Python loop notation. This last part enables us to observe the transfer of competencies from block-based to textual programming

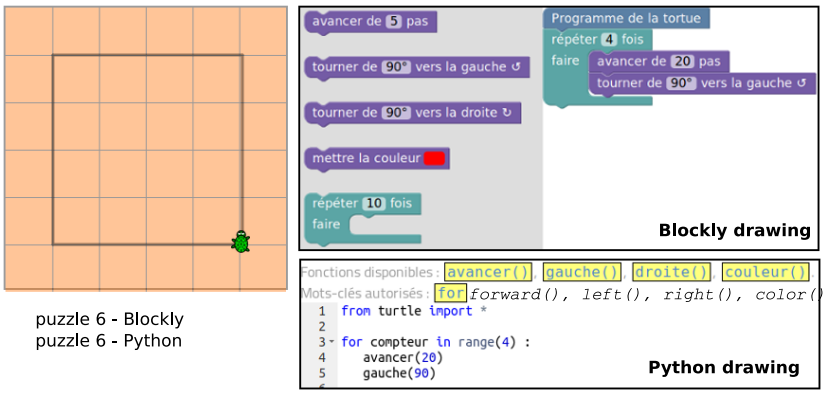

Fig. 3. From block-based to textual programming

4 Experimental Setting

4.1 Participants and Organization

The 2018 experiment involved pupils from 16 elementary schools around the university. Twenty classes participated for a total of 447 pupils. The age of the pupils is 8 - 10 years old and we had a balanced gender representation (49% girls). The experiment lasted for one week with 5 classes per day (excluding Wednesday). The classes came to the university for a whole day. To cope with the large number of pupils, they where supervised by second year computer science students with the support of their teachers. The students were presented with the pedagogical progression and learning activities beforehand so as to be able to manage the pupils and help them during activities.

4.2 Data Collection : Questionnaires And Online Activities Logging

Pre and Post Tests The pupils passed a test at the beginning and the end of the day to measure if there was a progress in their ability to spot repetitive patterns and to express them in a condensed notation opening the way to loop treatment. These tests are not intended to be a full assessment of the pupils' skills as presented in section 2.2 but rather to answer our first research question (*RQ1*).

The tests involved the coding of patterns as letters. The pupils were instructed they could use any notation they would see fit including shorthand notation. The pre test was presented as coding a graphical notation for music (Fig. 4, left - each color corresponding to a music note) while the post test involved coding pasta necklace crafting instructions (Fig. 4, right). We have chosen two different contexts to avoid pupils just remembering the patterns from the pre test. But it should be noted that patterns to be recognized and synthesized are strictly the same in the pre and post test.

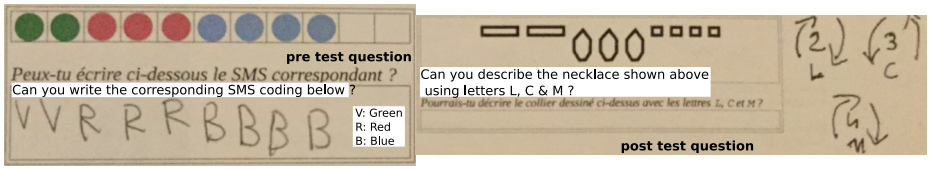

Fig. 4. Pre (left) and post (right) test patterns.

Table 1 presents patterns that were used ranging from a sequence based on a single instruction and up to three instructions for the most complex. The notation shown here corresponds to the pre test, but as stated before patterns are exactly the same in pre and post tests. The *Pattern* corresponds to what the pupils are given and the *loop notation* shows the kind of coding expected. Table 1 also presents the pattern types and correspondence. To answer RQ1, we look at the answers from the pupils. For instance for pattern type `1i` a pupils that has the notion of repetition would write something corresponding to `11R` (11 times R(ed)). In the other case, s.he would write all the letters.

Table 1. Patterns used in pre and post questionnaires

Type	Correspondence	Pattern	Loop notation
`1i`	1 instruction pattern	`RRRRRRRRRRR`	`11R`
`Nx1i`	N x 1 instruction pattern	`VVRRRBBBB`	`2V 3R 4B`
`2i`	2 instructions pattern	`BRBRBRBRBR`	`5x(BR)`
`3i+2i`	2 instructions + 1 instruction patterns	`VRRVRRBBBBB`	`2(VRR) 4B / 2(V 2R) 4B`

Programming Activities The plugged-in activities were realized on a France-IOI platform [3]. The four sequences include a set of puzzles of growing difficulty and build on each other. The study of the results from these activities will provide insights for our second research question about the transfer of skills from one language to the other (*RQ2*). As a whole, they include successively puzzles that can be solved by a sequence of instructions, a loop with one instruction, mixed sequence and loop, loop with multiple instructions and up to nested loops. The first sequence includes 24 puzzles and is very progressive so the pupils can build their skills. The next sequences provide between 15 and 18 puzzles. They all go through the easier puzzles (e.g. sequence) so that the pupils can transfer their skills to a new set of instructions. Then difficulty grows. Fig. 5 presents some of the puzzles that are further analyzed in the next section.

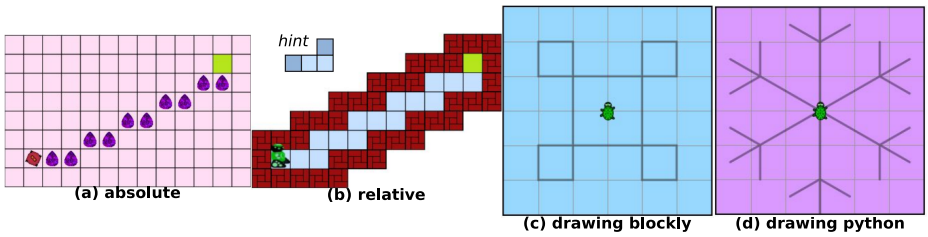

Fig. 5. Examples of tricky puzzles in each phase.

The platform progresses from one puzzle to the other upon success but it also allows to select a specific puzzle in a list. Each phase of the plugged-in activities lasted between 30 and 45 minutes depending on the groups. For this reason the pupils did not do the same number of puzzles depending on how quick they were and if they were stuck on some puzzles. When introducing new concepts or patterns, we have a tutorial puzzle with instructions or hints for resolution.

5 Results

5.1 Analysis of Pre and Post Questionnaires

The pre and post activity questionnaires (Fig. 4) have been coded to reflect whether the pupils have correctly coded patterns. This gave us a value between 0 and 1. For the 447 pupils, we have a mean $m = 0.147$, with a standard deviation $sd = 0.07$ for the pre test and $m = 0.241$, $sd = 0.12$ for the post test. A paired t-test gives us a value $t(446) = -6.76$ ($p < .0001$) which shows that the pedagogical sequence had a significant impact on pupils' capability to spot and code repetitive patterns.

[3] Association that organizes the Bebras computer science challenge in France (http://www.castor-informatique.fr)

Table 2. Successful coding of repetitive patterns

	`1i`	`Nx1i`	`2i`	`3i+2i`
pre test	118 (27%)	102 (23%)	29 (8%)	14 (4%)
post test	166 (38%)	140 (32%)	70 (17%)	56 (14%)

Table 2 presents the number (and percentage) of pupils that have used a shorthand notation (i.e. recognized the repetitive patterns) in the pre and post tests. It is interesting to first note that the awareness of 1 instruction patterns has significantly raised as well as sequences of several 1 instruction loop (+40% for both). But, the most interesting aspect is probably that more complex patterns (2 and 3 instructions) have increased even more. This could mean that after some training on short patterns the skill is generalized to more complex patterns quite rapidly by pupils. Fig. 4 shows a best case example of a pupil that did not use any notation for repetition in the pre test but successfully did in the post test.

5.2 Analysis of the Online Activities Logs

Traces of online activities on the platform were limited, since we could only get access to the last validation of each puzzle. We do not have an history of the trial and errors of pupils. This means that for this experiment we can only compute the number of successful vs. unsuccessful validation for the last trial.

For each sequence of puzzles we present a graph showing the success rate (number of successful validation / total number of trials) and the total number of trials for each puzzle. We also show the transitions between the levels of difficulty (e.g., from sequence to loop) which enables to spot at which point the pupils are in trouble. For the first two phases pairs of pupils share a tablet and take turn at resolving the puzzles. In practice, they would usually collaborate in the resolution even if they were not instructed to. This explains why we have a maximum number of trials around 200. For the last phase, pupils are alone in front of a computer giving a maximum of 447 trials (number of pupils). We have lost some trials on the first and the last sequence due to some technical problems which explains lower numbers of trials reported.

Absolute Orientation Fig. 6 presents results from the first phase. We have a very smooth progression in the puzzle difficulty giving a success rate above 90%. We observe a decrease in the number of trials when we enter the loop puzzles showing that some of the pupils start to get stuck. However, the real gap in success rate shows when we have loops with more than one instruction (i.e. longer patterns to identify) with the success rate going down to 66% for puzzle 19 (Fig. 5(a)) (puzzle 18 being a tutorial).

Relative Orientation Fig. 7 corresponds to the second phase. The success rate around or above 90% indicates that the pupils successfully managed to cope with

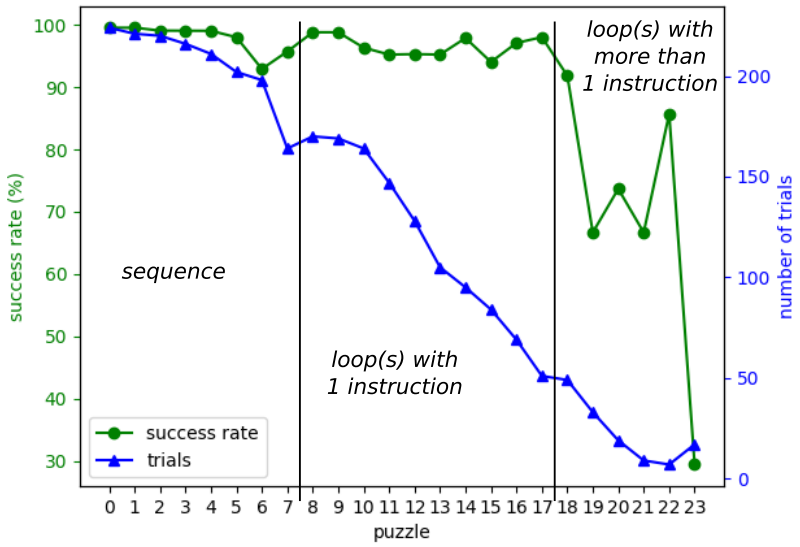

Fig. 6. Absolute orientation: success rate and trials.

a different set of instructions. There was also an improvement in their ability to handle loops with one instructions since we still have 139 trials for puzzle 7 (comparing to the 51 trials on puzzle 17 from previous phase) still with a good success rate. Again, moving to puzzles with more than one instruction is a major difficulty with puzzle 8 reaching a 56% success rate while having a significant number of trials. Fig. 5(b) shows the corresponding puzzle. It should be noted that being the first puzzle of this kind in the sequence there was a hint about the pattern to manage.

Drawing : Visual Syntax (Blockly) Entering the drawing phase introduces new challenges. First instructions are parameterized, second we start to use different kind of angles (i.e. other than 90°) that the pupils have not studied yet. Again, the number of trials and success rates for the first puzzles (including loops) indicate that the change of language is not a problem for pupils and they are still able to manage sequence and loop concepts (figure 8).

Mixing sequence and loops with more than one instructions seems to be quite difficult (puzzle 9 - 11, 9 providing hints) has we see the number of trials dropping. Fig. 5(c) shows puzzle 11 which still had 129 trials but a success rate of 58%. Few pupils did the nested loops puzzles but with more than 60% of success. This result could sustain the hypothesis that once patterns of length 2 are acquired, they are quickly generalized to more complex patterns.

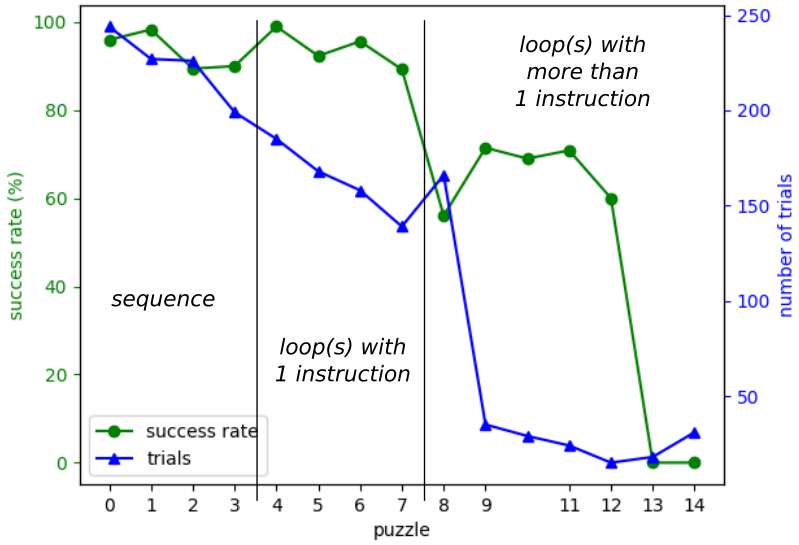

Fig. 7. Relative orientation: success rate and trials.

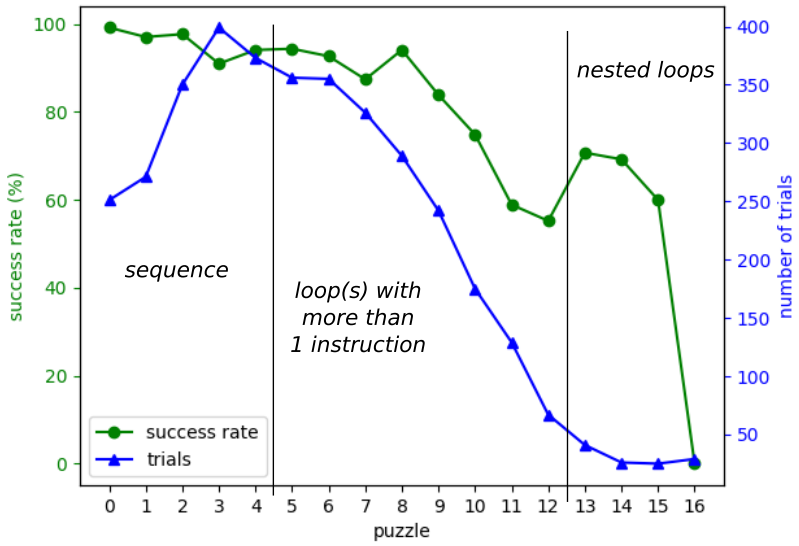

Fig. 8. Blockly drawing: success rate and trials.

Drawing : Textual Syntax (Python) As one can see on Fig. 3, pupils can use buttons corresponding to the instructions to avoid too much typing. Nonetheless, they still have to adapt the parameters to their needs. As can be seen on figure 9, the first sequence puzzles still achieve good results above 82% with a good participation[4]. This is an interesting result that shows that the pupils transferred well their skills from visual to textual language. Again mixed sequence and loops seem quite difficult (puzzles 8 and 9) with success rate barely above 50%. Nested loops are also a hard point. Puzzle 10 is a tutorial puzzle. Puzzle 11 shown Fig. 5(d) has only 18% success with very few trials. The last puzzle corresponds to a free activity where the pupils could draw what they want with no validation condition. The graph shows that a good number of pupils enjoyed it.

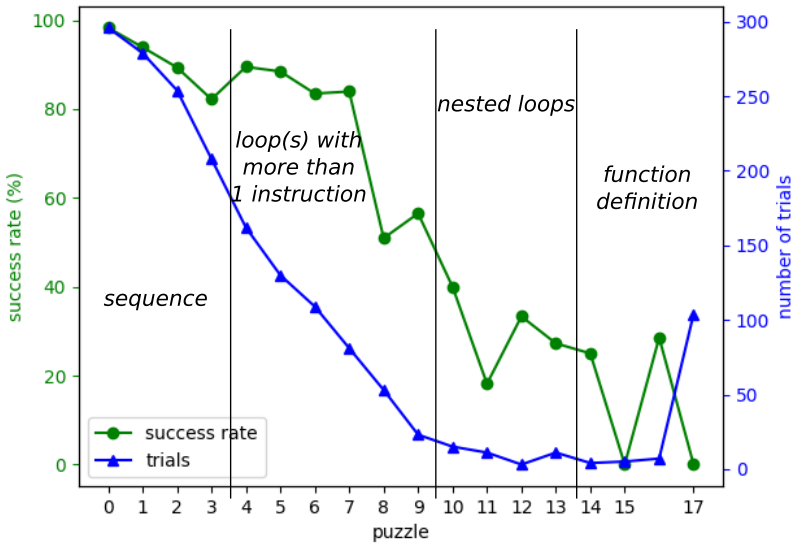

Fig. 9. Python drawing: success rate and trials.

6 Conclusion and Perspectives

This paper is focused on the learning of some fundamental Computational Thinking concepts and abilities by 8-10 years old pupils. We have designed a pedagogical sequence to initiate pupils to notions of instruction, sequence and loops, and to practice these concepts with several languages (free form (unplugged), block-based and textual language).

[4] we have lost some logs due to a technical problem.

The quasi-experiment reported in this article considered two research questions : whether the pedagogical sequence improves pupils' ability to recognize and express repetitive patterns as loops (RQ1) and to which extent they can transfer these skills to different languages (RQ2)

The statistical analysis of the questionnaires shows a significant impact of the sequence and we have a clear increase of pupils that identify repetitive patterns and are able to synthesize their description by using some notation to express a loop (RQ1). More interestingly it seems that when pupils acquire a pattern of length 2 (2 instructions) they quickly generalize it to longer patterns.

The results form the analysis of activities should be handled with more care since, being a learning session for the pupils, they, of course, get help from the students and even their professors and accompanying parents. However, having a student for 4 to 6 pupils, we hopefully get the results from their own thinking. The analysis shows that the pupils transfer quite easily their skills from one language to the other (RQ2). They manage well sequences and loops with one instruction then we have a gradual degradation of the results (number of trials) for loops with more than one instruction. Nested loops is a real hard point with very few trials and low success rate.

The pupils get a diploma which provides the address of the platform as well as their identifying code. This allowed us to see that around 300 of them did get back to the platform in the following days and up to two months later (by which we retrieved the data). All sequences were used by the pupils and we had 271 trials for the Python one which seems the hardest.

The results from this study can benefit to practitioners who could use the proposed activities. In terms of research, the questionnaires are a first step to assess the cognitive skill with a non-programming activity which, to our knowledge, is not so much explored as seen in section 2.2. There is still further work to quantify the relative contribution of the unplugged and plugged-in activities to the skills acquisition.

Future research should explore at what time the pupils acquire the concepts of repetitive patterns and loops and what level of practice is necessary for them to be able to transfer these concepts from one context or language to the other.

Acknowledgments

This work is partially funded by the EU Interreg Dig-e-Lab project. We thank the France-IOI association for providing the platform for the experiment.

References

1. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. annual American Educational Research Association meeting, Vancouver, BC, Canada pp. 1–25 (2012)
2. Bruner, J.: Towards a Theory of Instruction. Harvard University Press (1974)

3. Ching, Y.H., Hsu, Y.C., Baldwin, S.: Developing Computational Thinking with Educational Technologies for Young Learners. TechTrends (1980) (2018). https://doi.org/10.1007/s11528-018-0292-7, http://link.springer.com/10.1007/s11528-018-0292-7
4. Gouws, L., Bradshaw, K., Wentworth, P.: Computational thinking in educational activities. In: Proceedings of the 18th ACM conference on Innovation and technology in computer science education - ITiCSE '13. p. 10 (2013). https://doi.org/10.1145/2462476.2466518
5. Grover, S., Cooper, S., Pea, R.: Assessing computational learning in K-12. Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14 pp. 57–62 (2014). https://doi.org/10.1145/2591708.2591713, http://dl.acm.org/citation.cfm?doid=2591708.2591713
6. Knochel, A.D., Patton, R.M.: If Art Education Then Critical Digital Making: Computational Thinking and Creative Code. Studies in Art Education: A Journal of Issues and Research **57**(1), 21–38 (2015)
7. Papert, S.: Mindstorms: Children, computers, and powerful ideas. Basic Books, Inc. (1980)
8. Sabitzer, B., Jarnig, M.: Computational Thinking Through Modeling In Language Lessons. In: IEEE Global Engineering Education Conference (EDUCON). pp. 1913–1919 (2018)
9. Seiter, L., Foreman, B.: Modeling the learning progressions of computational thinking of primary grade students. Proceedings of the Ninth Annual International ACM Conference on International Computing Education Research pp. 59–66 (2013). https://doi.org/10.1145/2493394.2493403
10. Snow, E., Tate, C., Rutstein, D., Bienkowski, M.: Assessment Design Patterns for Computational Thinking Practices in Secondary Computer Science. Tech. Rep. December, SRI International (2017)
11. Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U.: Defining Computational Thinking for Mathematics and Science Classrooms. Journal of Science Education and Technology **25**(1), 127–147 (2016). https://doi.org/10.1007/s10956-015-9581-5
12. Wing, J.M.: Computational Thinking. COMMUNICATIONS OF THE ACM **49**(3), 33–35 (2006). https://doi.org/10.1145/1118178.1118215