Chen, W. et al. (Eds.) (2016). Proceedings of the 24th International Conference on Computers in Education. India: Asia-Pacific Society for Computers in Education

The Interest-Driven Creator Theory and Computational Thinking

Bowen LIU, Ping LI*, Siu Cheung KONG* & Sing Kai LO
The Education University of Hong Kong, Hong Kong
*{ pli, sckong }@ied.edu.hk

Abstract: There is a growing interest for Computational Thinking (CT) for the last decade. Most studies focus on teaching CT skills in K-12 level. In higher education, applying CT methods over all disciplines still needs cross institute movement, proper teaching tools and assessment procedure. In this paper we discuss the potential of applying CT methods in the mathematics courses at university level. With the inspiration of the Interest-Driven Creator (IDC) theory, we suggest that applying CT methods in mathematics benefits students' understanding of concepts and overcomes the drawbacks of traditional pedagogy. We study the case of introducing the limit of a sequence, which is a fundamental concept in calculus. An algorithm, inspired by the $\boldsymbol{\varepsilon}$-$\boldsymbol{N}$ definition, is designed to find suitable $\boldsymbol{N}$ given a specific $\boldsymbol{\varepsilon}$ with exhaustion methods. Based on the algorithm designed, a game that help to introduce the $\boldsymbol{\varepsilon}$-$\boldsymbol{N}$ definition of the limit of a sequence is presented as an example. The game would be developed on mobile devices for easy accessibility and for catering the trend of mobile device.

Keywords: Computational Thinking, Higher Education, Interest-Driven Creator (IDC)

1. Introduction

Computational Thinking (CT) is considered as one of the fundamental skills in today's society. Computational Thinking is not specified to computer scientists, but a universal skill set including logic, algorithmic thinking, recursive thinking, multi-level abstraction, parallel thinking, pattern matching and related processes. Besides writing, reading, speaking and arithmetic, Computational Thinking should be taught to students in all disciplines to prepare for future challenges. Increasing interests are emerging in teaching Computational Thinking in K-12 level. Research of CT in K-12 level mainly focuses on developing environments and tools that foster CT and relevant assessment. Most programming platforms and tools are on desktop computers. However with the prevalence of smartphones and tablet computers worldwide, mobile apps provide a good juncture of CT and other disciplines. In higher education, research interests concentrate on teaching CT skills to early stage computer science undergraduates. But since CT is universally applicable to every students, teaching CT methods in various disciplines at college level is necessary and beneficial. In this paper, we will discuss the application of CT in mathematics courses and the potential benefit for comprehension of mathematical concepts with CT methods.

2. Background of Study

The concept of Computational Thinking dates back to 1960s. Alan Perlis, the first recipient of ACM Turing Award, suggested that "information theory" should be included as a part of the education for students in all disciplines (Guzdial, 2008). In the 1980s, Papert put forward the idea of fostering children's procedural thinking with LOGO programming language (Papert, 1980; Papert & Idit, 1991). The term "Computational Thinking" is first used to refer to expressing powerful ideas with computational representation (Papert, 1996). A more recent definition on computational thinking came from Wing's articles (2006). Wing proposed that computational thinking represents "a universally applicable attitude and skill set everyone, not just computer scientists would be eager to learn and use"

(Wing, 2006). Computational Thinking exploits recursive thinking, abstraction and decomposition, debugging and prevention, and heuristic reasoning. Wing's definition of CT is widely acknowledged and applied across multiple fields by researchers and organizations. The National Research Council (2010) conducted a workshop focusing on the scope and nature of computational thinking. More than 20 high-level skills and practices related to CT are included in the workshop report. In 2012, Royal Society also offered a tangible and concise definition that reveal the essence of CT-"Computational Thinking is the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer sciences to understand and reason about both natural and artificial systems and processes" (Furber 2012).

A useful three dimensional (3D) framework on CT was proposed by Brennan and Resinick in 2012. The framework is mainly developed for design-based learning activities. Students who participate in these activities are regarded as a designer. The framework includes three components: concepts, practices and perspectives. Computational concepts refer to the concepts that designers are exposed to when they are programming. Computational practices are the experiences when designers engaged with the concepts. Computational perspectives are about how designers transfer their thinking pattern formed in the computational activities to the world around them. This framework is valuable in exploring the CT issues under computer-aid or game-based circumstances.

3. The Computational Thinking in Higher Education and Its Application in Mathematics Classrooms

3.1 Computational Thinking in Higher Education

The idea of Computational Thinking have received much attention since the publication of Wing's essay (2006). Research studies are devoted to incorporating computational thinking skills into K-12-level curricula (Barr & Stephenson, 2011). Grover and Pea (2013) review the state of CT in K-12 level and point out that much work mainly focuses on definitional issues, and tools for CT development while empirical study is scarce. In higher education, studies focus on teaching CT skills to computer science undergraduate in initial phrase of study. Practical research on teaching CT skills largely takes place within the fields of computer science and the science, technology, engineering and math (STEM). Miller and Settle (2011) make a comparative study of the learning process of tree traversal methods between computer science and non-computer science students. Kilpeläinen (2010) proposed analogies based on the metaphor of traveling to illustrate the concepts of reduction in computer science. A trend of teaching Computational Thinking in game-based scenarios is emerging. An advantage of applying game-based teaching method is the connection between abstraction, interactive computation and constructivist pedagogy (Berland, 2011).

However, outside the computer science and STEM fields, there exists a lack of cross-institute cooperation to add CT into fundamental skill-set. There are various potential reasons for this phenomenon. One may be the potential obstacle around the conceptualization of CT. Different from the situation in the K-12 level, higher education is research-oriented and a more precise definition on CT is needed in different disciplines. Moreover, the distinction between applying CT method and simple application of computers to problems is not apparent and required well-understanding for most educators and students. Last but not least, even within the computer science, not all the problems are computable. As Aho (2012) mentioned, there are emergent models of computing that beyond the traditional Turing machine computation, but extending the scope of CT to these models requires great efforts. Lacking a computable structure makes it difficult to apply CT methods.

3.2 Computational Thinking in Mathematics Classrooms and IDC Theory

In traditional mathematics courses, students acquire mathematical concepts largely through lecturers' explanation and consolidate their comprehension through exercises. In class, lecturers have to face the trade-off between conciseness and rigorousness in their teaching. Lecturers' expertise play an important role in the pedagogic process. Lectures who are also experienced researchers, which is common in colleges and universities, tend to skip fundamental details which are important for students'

understanding. After class, students usually spend lots of time in assessments. However, doing mathematics exercises is time-consuming and the process could be painful. Students may lose interest in math due to the failure in assessments and mistakenly equal the mathematical ability to correctly doing mathematics exercises only. Mathematics courses in higher education try to equip learners with profound logical thinking abilities and adequate understanding of mathematical concepts and approaches but not simply an ability of doing exercises. In fact, mathematical thinking, such as abstraction, algorithmic thinking, shares the same base with CT. This make it possible for students to grasp mathematical idea while developing CT. We suggest that with proper computational tools and pedagogy, students can study mathematics in a concrete scenario and can accomplish the transition from figurative to abstract.

Game-based learning strategies is considered an effective method to teach complex computational thinking skills. They enable students to accomplish certain tasks in games and acquired knowledge in playing. The process of playing stimulates students' interest and can cultivate learning habits with adequate repetitions. The Interest-Driven Creator (IDC) theory provides a more precise description of the nature behind this process (Chan, Looi and Chang, 2015). Designed by a group of researchers in Asia, the Interest-Driven Creator (IDC) is a novel theory for technology enhanced learning in the future. The IDC theory hypothesizes that with the development of technology, driven by interest, students can participate in the creation activities and by repeating this process to cultivate learning habits, our future generation will become lifelong interest-driven creator. One of the anchor concepts of IDC theory is the creation loop which consist of imitating, combining and staging. Imitating is the first creation component which refer to emulation through observation. Combining, the second component, is to synthesizing the thoughts or things of others and individual views to form something new. Staging, the third creation component, is about presenting new thoughts or achieved outcome to others. Inspired by this theory, we suggest that if we could provide students a platform that can help to apprehend mathematical concepts through creating (such as a numerical simulation program, algorithm design or 3D models), combining mathematics study with the CT skills acquisition is possible.

4. Introducing Mathematics Concepts with CT Methods: An Example

The idea of limit is one of the fundamental concepts in mathematics. Students who participate calculus courses (or real analysis courses) are commonly exposed to this concept at the beginning. Lecturers or textbooks usually start the introduction with "limit of a sequence" then expand the definition to other situations. Here we present an example of how to introduce the "Limit of a sequence" with CT methods. Before applying CT methods, reviewing the definition of "limit of a sequence" can help to understand why it is applicable to CT methods. The modern calculus is developed by Issac Newton and Gottfried Wilhelm Leibniz in 17th century Europe. But the modern definition of a limit was given by Bernhard Bolzano and by Karl Weierstrass in the 1870s (Grabiner, 1983). In the real numbers, a number $\boldsymbol{L}$ is defined as the limit of the sequence $\boldsymbol{X_i}$ if the numbers in the sequence are getting closer and closer to $\boldsymbol{L}$ than any other number. In formal definition, the limit of a sequence are described with $\boldsymbol{\varepsilon}$-$\boldsymbol{N}$: $\boldsymbol{L}$ is defined as the limit of a sequence $\boldsymbol{X_i}$ if for each real number $\boldsymbol{\varepsilon > 0}$, there exists a natural number $\boldsymbol{N}$ such that for every natural number $\boldsymbol{i > N}$, we have $\boldsymbol{| X_i - L | < \varepsilon}$.

In the situation of a monotonous bounded sequence, with the inspiration of the $\boldsymbol{\varepsilon}$-$\boldsymbol{N}$ definition, we design an algorithm applying exhaustion method to find a suitable $\boldsymbol{N}$. The pseudo code of the algorithm is shown below. In the circumstance of a decreasing sequence, the limit exists if for each $\boldsymbol{\varepsilon}$ the user input, the $\boldsymbol{N}$ could be found with finite iteration. The limit of a sequence is expressed in an algorithmic way. This algorithm provides a recursive view on the $\boldsymbol{\varepsilon}$-$\boldsymbol{N}$ definition.

Initialize:
Given an monotonous bounded sequence X_i
A possible limit L
$i=1$
Step 1. Input an ε
Step 2. Calculate $|X_i - L|$
Step 3. Compare $|X_i - L|$ with ε
Step 4. If $|X_i - L| < \varepsilon$:
Then let $N = i$,
Print N
Stop;
Else let $i = i + 1$
Go back to Step 2.

The ε-N definition can also be compared to a game: given a sequence and a limit L, one player A provides an ε, and the other player B have to find out an N that meets the requirement to win the game. If player B has a sure-win strategy, then the sequence has the limit L. This analogy make it possible for educators to design a game for the concept introduction. With the algorithm designed, we develop a computer program for this game that can used for computational experiment and pedagogy. The flowchart of the program is shown in Figure 1. A player tries to win the game by entering an ε that makes the computer cannot find an N that satisfies the condition. Considering the situation that the algorithm maybe trapped in an endless loop or the ε entered beyond the accuracy and ability of a computer, we introduce a timer for prevention. It is worth to point out that the failure of the algorithm could also prevent with proper selection of the sequence by teachers. Teachers can decide students' win or lose by setting different sequences, limits or timer and conduct students to think about the determinants of the game.

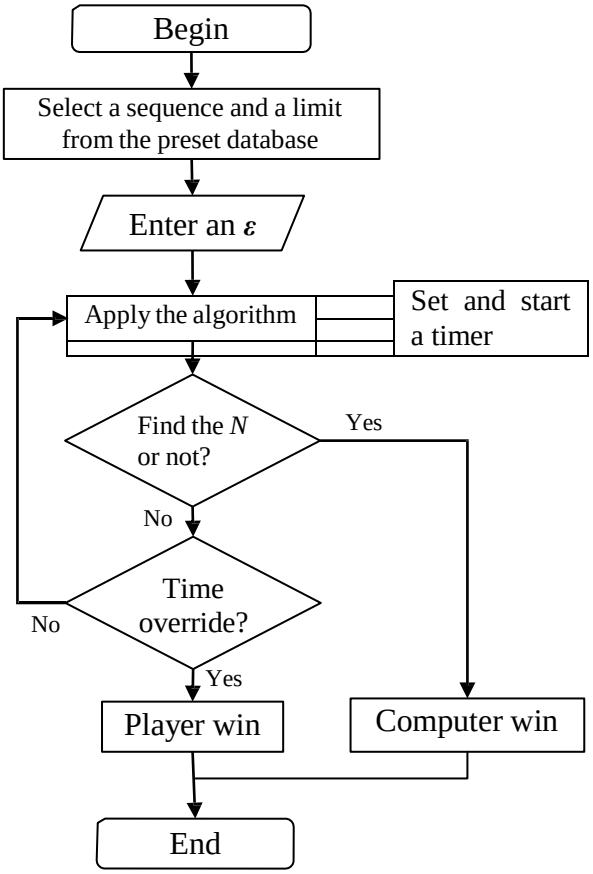

Figure 1. The program of a game to find out a suitable N of a sequence.

If a player always lose and get the suitable N with the algorithm, then the L may be the limit of the sequence. If one enter an ε that can make the algorithm loop infinitely, he/she may reject L as the limit. This algorithm provides opportunities for students to execute the ε-N definition with computers. Students can have an intuitive impression of limit and infinite with the repetitive enter of ε or the long waiting for a non-exist N in an infinite loop. Students can also have opportunities to use proper loop statement to take place of repetitive enter of ε and hence are exposed to recursive thinking.

5. Summary and Future Work

In this paper, we discuss the Computational Thinking in mathematics education at university level. With the inspiration of the IDC theory and the inherent connection between mathematics and computational thinking, we suggest that applying CT methods in mathematics study at university level benefits the comprehension of mathematical concepts and the cultivation of CT skills. We provide an example illustrating how to introduce limit of a sequence with an algorithmic program. However, there are still limitations. Firstly, a better user interface is needed. Most programming platforms for CT have graphical interface and pre-design code modules. Students can easily write programs with drag and drop. We would develop our game on mobile platform so that students can easily have access to the game with their mobile devices. The game will become a reliable learning system simultaneously. Secondly most definitions in mathematics are rigorous, it may not be possible to express all of them in computable forms. Finally, whether the method is effective needs empirical study. We are going to conduct experiments in universities and attempt to figure out whether the designed tools and pedagogy is helpful for students' understanding in calculus. A comparative experiment between students in mathematics and other major would be valuable to measure the effectiveness of our pedagogy.

References

.

Aho, A. V. (2012). Computation and computational thinking. *Computer Journal, 55*(7), 832-835.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?. *ACM Inroads*, *2*(1), 48-54.

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning, 1*(2), 65-81.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. *Proceedings of the 2012 Annual Meeting of the American Educational Research Association,* Vancouver, Canada. Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Chan, T., Looi, C., Kit, & Chang, B. (2015). The IDC theory: creation and creation loop. *Workshop Proceedings of the 23rd International Conference on Computers in Education,* 814-820.

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK schools.* London, UK: The Royal Society. Retrieved from http://royalsociety.org/education/policy/computing-in-schools/report/

Grabiner, J. V. (1983). Who gave you the epsilon? Cauchy and the origins of rigorous calculus. *American Mathematical Monthly*, *90*(3), 185-194.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: a review of the state of the field. *Educational Researcher, 42*(1), 38-43.

Guzdial, M. (2008). Education: paving the way for computational thinking. *Communications of the ACM, 51*(8), 25-27.

Kilpeläinen, P. (2010). Do all roads lead to Rome? ("or" reductions for dummy travelers). *Computer Science Education, 20*(3), 181-199.

Miller, C. S., & Settle, A. (2011). When practice doesn't make perfect: effects of task goals on learning computing concepts. *ACM Transactions on Computing Education*, *11*(4)

National Research Council (2010). *Report of a workshop on the scope and nature of computational thinking.* Washington, D.C.: The National Academics Press.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. Sussex, U.K.: Basic Books.

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Papert, S., & Idit, H. (1991). Situating constructionism. In S. Papert, & I, Harel (Eds.). *Constructionism* (pp. 1-11). Norwood, NJ: Ablex.

Wing, J. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.