Yang, J. C. et al. (Eds.) (2018). Proceedings of the 26th International Conference on Computers in Education. Philippines: Asia-Pacific Society for Computers in Education

Generating Abstract and Real-World Coding Exercises with Adjustable Difficulty

Thomas James TIAM-LEE & Kaoru SUMI
Future University Hakodate, Japan
g3117002@fun.ac.jp

Abstract: We developed an approach for automatically generating abstract and real-world coding exercises with adjustable difficulty. Using our approach, we can produce exercises based on abstract computational operations for learning the syntax of a programming language, and exercises based on real-life contexts for learning abstraction and logic formulation. We present an initial evaluation of the exercises by students and teachers of programming. This work can pave the way for the development of intelligent programming tutors with adaptive and personalized feedback that can display content based on the state of the student.

Keywords: Programming, education, content generation

1. Introduction and Related Studies

Previous work on intelligent programming tutors display different types of adaptive feedback to students while learning programming. In Ask-Elle, hints are given as feedback based on the student's code (Gerdes, Heeren & Jeuring, 2017). In Java Sensei, empathetic responses are given based on the student's detected emotion (Cabada et al., 2015). In our previous work, we developed a system for programming practice that offers a guide and adjusts the difficulty of the problems based on the presence of confusion (Tiam-Lee & Sumi, 2018).

In this paper, we discuss an approach for generating coding exercises as adaptive feedback for learning programming, which few studies have explored. Previous work on this domain have either focused on generating tracing and debugging exercises, such as that found in the work of Wakatani & Maeda (2016) or relied on parameterized questions, such as those found in the work of Prados et al. (2005) and Hsiao, Brusilovsky & Sosnovsky (2009). In this paper, we instead discuss an approach for generating coding exercises that have flexibility to produce exercises that vary in terms of solution structure and difficulty.

2. Generation of Programming Exercises

We represent an exercise as a set of nodes arranged in a flowchart-like structure, which represents the sequence of operations in a solution for the exercises. We generate exercises based on abstract computations by procedurally combining nodes to form an exercise structure. First, the structure of the exercise is generated (Figure 1a). Then, an algorithm ensures that all operations are relevant to the output (Figure 1b). Finally, the remaining parameters are randomly assigned (Figure 1c). In this exercise, the difficulty can be adjusted by setting the number of nodes or operations to be performed in the exercise. The exercise text can be generated by mapping of each node configuration to text.

We generate exercises based on real-world computations by mapping real-world computations such as getting the area of a square or converting meters to feet to a flowchart structure. Each computation also has a set of requirements that need to be fulfilled by a simple plot planning algorithm. For example, if the chosen action is computing the area of a square in feet, then the plot planning algorithm should produce a story in which the length of the side of a square object is given in feet, and there is some intention to compute the area of that object. In this type of exercise, difficulty can be adjusted based on the type of the computation chosen, and difficulty can be further

increased by performing regression on the chosen computation. For example, when regression is applied to the action to compute the area of a square in feet in the action to covert meters to feet, then the problem should now give the length of the side of the square object in meters. The solution would now be to convert the value to feet first, and then compute the area. Once the computations have been determined, we use a backward state-space search planner on a domain commonsense knowledge base to produce a simple narrative that serves as the context of the computation, like a Mathematics word problem. Table 1 shows examples of exercises generated by our approaches.

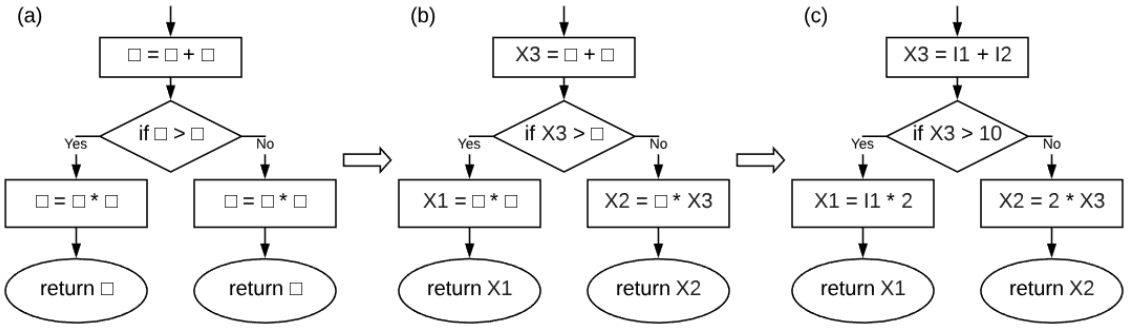


Figure 1. Exercise generation process. In (a), the exercise structure is generated. In (b), all nodes are ensured to be relevant to the output. In (c), the remaining parameters are assigned.

3. Initial Evaluation

We perform an initial evaluation of the generated exercises by seeking feedback from students and teachers of programming. For the student evaluation, the participants are 13 students in a Japanese university. The students were asked to rate each exercise on two criteria on a 5-point Likert scale: (1) how easy it is to understand and (2) how engaging it is to answer. Table 2 shows the results.

On average, the score for "how easy it is to understand" is 4.08 for the abstract level questions and 3.75 for the real-world level questions. A common feedback cited on real-world level questions is some student's unfamiliarity with some concepts such as "body mass index" and "feet". On average, the score for "how engaging it is" is 3.52 for the abstract level questions and 3.75 for the real-world level questions. Some students think that the exercises in the abstract level are repetitive.

We also sought the qualitative feedback of 7 programming teachers handling university programming courses. Most of the teachers have stated that the exercises on the abstract level are usable for teaching programming (5), but are simple and straightforward (6) so it is appropriate for beginners (3). Some teachers stated that the exercises do not require students to analyze the problem on a higher level (2). Some teachers stated that they perceive the exercises on the real-world level to be of higher quality (3). Reasons cited are because they are relevant to real world concepts (3) and they are not straightforward and thus require the student to do a higher level of analysis (1). However, some teachers stated that the types of computations in the examples given to them are simple and limited for practical use (2). Almost all the teachers have stated that the exercises in the abstract level are easy to understand (6). One of the teachers is concerned that the lack of an intention in the context of a real world may cause the exercises to be difficult to understand (1).

Some teachers have stated that the real-world context of the problems allow students to imagine and understand them better (3), but there are also concerns that the irrelevant story components that are not necessary for the problem could cause distractions in understanding (2). A common concern is that the grammar and wording of the sentences sound unnatural (3) and could affect understanding. For example, names are repeated instead of using pronouns.

The teachers perceived the exercises in the abstract level as a way for teaching students the syntax of the programming languages such as remember the syntax of an if-else statement or remembering how to write arithmetic expressions (5), while the exercises generated in the real-world are more appropriate for logic formulation and/or mapping real-world relationships to programming statements (4), although the types of computations might need to be diversified for it to become very useful. The two different levels of abstraction can potentially be used to target different aspects of coding skill.

Table 1

Examples of Exercise Generated with Abstract and Real-World Computational Operations

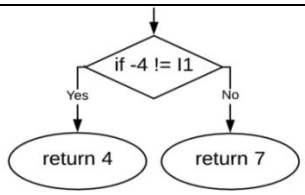
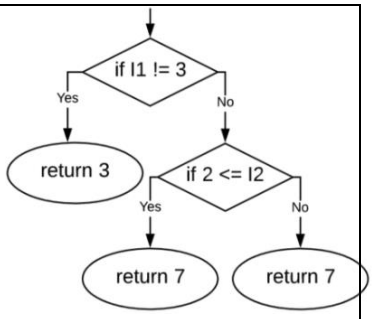
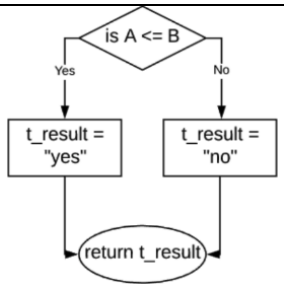
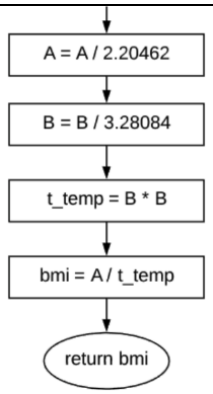
Complete the function. If -4 is not equal to I1, return 4. Otherwise return 7. `int func(int I1) { }`	
Complete the function. If I1 and -3 have different values, return 6. Otherwise do the instructions in A. A: If 2 is less than or equal to I2, return 7. Otherwise return 7. `int func(int I1, int I2){ }`	
Cloud is a doctor. Cloud wants to buy a new stethoscope for work. A stethoscope costs A yen. Cloud has a total of B yen. Complete the function which should return yes if Cloud has enough money to buy the stethoscope, or no otherwise. `String func(double A, double B) { }`	
Thomas is overweight. Thomas wants to get more fit. Thomas weighs A pounds. The height of Thomas is B feet. Complete the function that computes the body mass index (BMI) of Thomas. To compute the BMI, divide the weight in kilograms by the height in meters squared. 1 meter is 3.28084 feet. 1 kilogram is 2.20462 pounds. `double func(double A, double B) { }`	

Table 2

Results of Student Evaluation

		Abstract	**Real-World**
How easy to understand is it?	Avg. (Std. Dev.)	4.08 (1.57)	3.67 (1.42)
How engaging is it?	Avg. (Std. Dev.)	3.52 (1.44)	3.75 (1.37)

References

Cabada, R. Z., Estrada, M. L. B., Hernández, F. G., & Bustillos, R. O. (2015, July). An affective learning environment for java. *Advanced Learning Technologies (ICALT), 2015 IEEE 15th International Conference* (pp. 350-354). IEEE.

Gerdes, A., Heeren, B., Jeuring, J., & van Binsbergen, L. T. (2017). Ask-Elle: an adaptable programming tutor for Haskell giving automated feedback. *International Journal of Artificial Intelligence in Education*, 27(1), 65-100.

Hsiao, I. H., Sosnovsky, S., & Brusilovsky, P. (2009, September). Adaptive navigation support for parameterized questions in object-oriented programming. *European Conference on Technology Enhanced Learning* (pp. 88-98). Springer, Berlin, Heidelberg.

Prados, F., Boada, I., Soler, J., & Poch, J. (2005). Automatic generation and correction of technical exercises. *International conference on engineering and computer education: Icece* (Vol. 5).

Tiam-Lee, T. & Sumi, K. (2018). Adaptive feedback in a system for programming practice. *International conference on intelligent tutoring systems* (pp. 243 - 255). Springer.

Wakatani, A., & Maeda, T. (2016, August). Evaluation of software education using auto-generated exercises. *Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES), 2016 IEEE Intl Conference* (pp. 732-735). IEEE.