Towards Improving Introductory Computer Programming with an ITS for Conceptual Learning

Franceska Xhakaj(✉) and Vincent Aleven(✉)

Human-Computer Interaction Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
{francesx,aleven}@cs.cmu.edu

Abstract. Computer programming is becoming important in almost every profession. However, programming is still difficult for students to learn. In this work, we focus on helping students acquire strong conceptual and procedural knowledge of programing. We propose to create a new Intelligent Tutoring System (ITS) that will support students in two types of conceptually-oriented activities: code tracing and code comprehension. Further, we propose to run a study to evaluate whether the ITS can support students' conceptual learning and transfer to procedural learning of computer programming.

Keywords: Conceptual learning · Procedural learning
Intelligent Tutoring Systems · Computer Science education

1 Introduction

Computer programming is a key skill set in many professions and STEM domains [8]. However, learning programming is hard, and typical introductory programming instruction may leave substantial room for improvement [5, 14]. Prior research in Computer Science education suggests that practice with conceptually-oriented activities (e.g., activities that emphasize knowledge of code constructs and code execution) can be very helpful in learning procedural knowledge (e.g., skills of generating code) [2, 9, 11]. Although this prior work shows promise, it does not take full advantage of current insights in the cognitive science and mathematics education literature on transfer between conceptual and procedural learning and how they mutually influence each other [10]. Nor have the conceptually-oriented approaches tested in prior CS education research taken advantage of the capabilities of advanced learning technologies, such as Intelligent Tutoring Systems (ITSs) [13]. We propose a program of research that capitalizes on insights from cognitive science and on advanced learning technologies to facilitate the learning of programming. The proposed research has two strands of work. First, we will create a new ITS building on our existing infrastructure (CTAT [1]). The new ITS will support students in two types of conceptually-oriented learning activities: code tracing and code comprehension. Second, we will conduct an experimental study that will test whether and how such an ITS can support conceptual learning and transfer to procedural learning in the area of computer programming.

C. Penstein Rosé et al. (Eds.): AIED 2018, LNAI 10948, pp. 535–538, 2018.
https://doi.org/10.1007/978-3-319-93846-2_101

2 The Intelligent Tutoring System: TiPs

Traditional instruction in programming targets conceptual knowledge with (video) lectures, textbook reading, programming exercises and with a "stepper" tool for stepping through code execution line-by-line. However, more highly interactive and adaptive instruction supported by ITSs may be more effective at helping students develop conceptual knowledge. In Fig. 1 we show an initial design of TiPs (Transfer in Programming system), the proposed ITS that will support two types of conceptually-oriented activities: code tracing (left) and code comprehension (right). TiPs will target common challenging topics in introductory computer programming [e.g., 3, 12] including variables, the assignment operation, conditional statements, loops, etc.

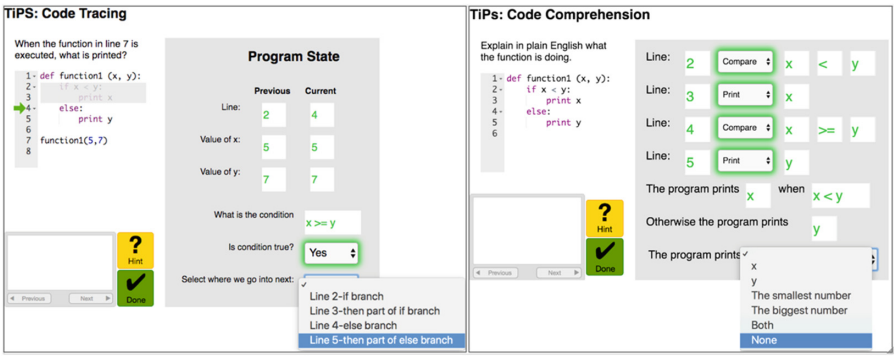

Fig. 1. Mockups of TiPs, the proposed ITS. Code tracing (left), code comprehension (right).

In code-tracing activities, the student tracks a program's changing internal state line-by-line (Fig. 1, left). The student generates and fills in the program state at each step of code execution, with support of feedback and hints from TiPs. TiPs simulates code execution by means of (1) a green arrow pointing to the current line, and (2) a greyed out area for code that is not relevant to the current step. TiPs will remove this scaffolding gradually as the student gains more competence. For each executed line, the student enters the line number, the values of the variables, and answers questions at the bottom. Once the student has correctly entered the current program state, TiPs dynamically changes the interface to let the student fill in the next state. This type of activity is likely to support acquisition of conceptual knowledge of code constructs and code execution, as it involves direct application of this kind of knowledge in the context of a piece of code. Past research suggests that code tracing can have beneficial effects for students, although it does not provide a fully rigorous demonstration of such effects on conceptual and in particular transfer to procedural knowledge of programming, as we plan to test in this work [4, 7, 15].

In code-comprehension activities, the student is asked to explain at a high level what the function of a given piece of code is (Fig. 1, right). In contrast to code tracing, the emphasis here is on being able to understand code at a higher level, without

simulation of code execution. For each line, the student chooses from a drop-down menu what the construct in that line is doing. Once an instruction is selected, the appropriate values appear on the right for the student to fill in. The student then is prompted to explain what the code is doing, with the aid of the tutor as needed. Code-comprehension exercises may help students develop conceptual knowledge of how the individual constructs can work together to realize desired functionality. We know of no rigorous studies that showed that code-comprehension practice can foster conceptual or procedural knowledge. [6] found that tracing, explaining and writing code are statistically significantly correlated, but these results are correlational, not causal.

3 Proposed Study

We propose to run a study to find out whether three forms of conceptually-oriented activities (code tracing, code comprehension, or a combination), supported by an ITS, (1) enhance students' conceptual knowledge of code and (2) transfer to students' work on code-generation exercises (e.g., enhances the learning of programming skill). The study will examine transfer from conceptually to procedurally-oriented activities.

The study will have a 2×2 design with experimental factors code tracing and code comprehension. Students will be randomly assigned to conditions. Students in all conditions will do two blocks of activities. In the first block, students will engage in (1) code tracing, (2) code comprehension, (3) a combination, or (4) neither (they will engage in code generation). The code-tracing and code-comprehension activities will be supported by the proposed ITS. The second block will involve a sequence of code-generation exercises for all conditions. Before and after the first block, students will complete a pre- and post-test, to assess both conceptual and procedural knowledge of programming. The conceptual items will be based on existing literature and will contain code-tracing and code-comprehension exercises designed to measure conceptual transfer [3, 12]. The procedural items will involve writing code in the language that students are studying. We will create new problems for conceptual and procedural items that will contain known constructs, alone and combined in new ways. In addition, we will collect ITS log data that we will use to extract measures of the students' performance and knowledge growth in the tutor activities. We will conduct analyses to investigate how code-tracing and code-comprehension exercises affect students' conceptual and procedural knowledge of programming. We hypothesize that both these activities positively affect conceptual knowledge and outcomes in procedural learning from the code-generation activities. We hypothesize further that code comprehension is more effective in the presence of code tracing, as code tracing may be a step along the way to code comprehension [6].

4 Expected Contributions

If successful, the proposed research will generate new scientific knowledge about whether (1) an ITS that supports conceptually-oriented activities (code tracing and code comprehension) can enhance conceptual knowledge and transfer to procedural knowledge of

computer programming, (2) which kind of conceptually-oriented activities are more effective in this regard. As a practical contribution, the research will yield a novel ITS that will support conceptually-oriented activities for learning programming. The results and findings could inform the design of other ITSs for introductory programming and of instruction beyond ITSs.

References

1. Aleven, V., McLaren, B.M., Sewall, J., van Velsen, M., Popescu, O., Demi, S., Ringenberg, M., Koedinger, K.R.: Example-tracing tutors: intelligent tutor development for nonprogrammers. Int. J. Artif. Intell. Educ. **26**(1), 224–269 (2016)
2. Bayman, P., Mayer, R.E.: Using conceptual models to teach BASIC computer programming. J. Educ. Psychol. **80**(3), 291–298 (1988)
3. Caceffo, R., Wolfman, S., Booth, K.S., Azevedo, R.: Developing a computer science concept inventory for introductory programming. In: Proceedings of the 47th ACM Technical Symposium on Computing Science Education, pp. 364–369. ACM (2016)
4. Kumar, A. N.: A study of the influence of code-tracing problems on code-writing skills. In: Proceedings of the 18th ACM Conference on Innovation and Technology in Computer Science Education, pp. 183–188. ACM (2013)
5. Lahtinen, E., Ala-Mutka, K., Järvinen, H.M.: A study of the difficulties of novice programmers. ACM SIGCSE Bull. **37**(3), 14–18 (2005)
6. Lister, R., Fidge, C., Teague, D.: Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. ACM SIGCSE Bull. **41**(3), 161–165 (2009)
7. Nelson, G.L., Xie, B., Ko, A.J.: Comprehension first: evaluating a novel pedagogy and tutoring system for program tracing in CS1. In: Proceedings of the 2017 ACM Conference on International Computing Education Research, pp. 2–11. ACM (2017)
8. Orsini, L.: Why Programming is the Core Skill of the 21st Century. https://readwrite.com/2013/05/31/programming-core-skill-21st-century/. Accessed 12 Dec 2017
9. Pennington, N., Nicolich, R., Rahm, J.: Transfer of training between cognitive subskills: is knowledge use specific? Cogn. Psychol. **28**(2), 175–224 (1995)
10. Rittle-Johnson, B., Siegler, R.S.: The relations between conceptual and procedural knowledge in learning mathematics: a review. In: The Development of Mathematical Skill, pp. 75–110. Psychology Press, Hove (1998)
11. Shih, Y.F., Alessi, S.M.: Mental models and transfer of learning in computer programming. J. Res. Comput. Educ. **26**(2), 154–175 (1993)
12. Tew, A.E.: Assessing fundamental introductory computing concept knowledge in a language independent manner. (Doctoral dissertation), Georgia Institute of Technology, Georgia, USA (2010)
13. VanLehn, K.: The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. Educ. Psychol. **46**(4), 197–221 (2011)
14. Watson, C., Li, F.W.: Failure rates in introductory programming revisited. In: Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, pp. 39–44. ACM (2014)
15. Xie, B., Nelson, G.L., Ko, A.J.: An explicit strategy to scaffold novice program tracing. In: Proceedings of the 49th ACM Technical Symposium on Computer Science Education, pp. 344–349. ACM (2018)