Conceptual Checks for Programming Teachers

Luca Chiodini[1 [0000−0002−2712−9248]], Matthias Hauswirth[1 [0000−0001−5527−5931]], and Andrea Gallidabino[1 [0000−0003−4191−7766]]

Software Institute - Università della Svizzera italiana
https://luce.si.usi.ch/

Abstract. Learning to program and learning a new programming language is difficult because it requires learners to undergo conceptual change. Research on conceptual change has shown that instructors' awareness of their students' misconceptions can significantly affect learning outcomes. In this demo we present "conceptual checks", a web-based tool that allows instructors and teaching assistants of programming courses to quickly get an overview of the misconceptions that might come up at a given point in their course. Based on the idea of refutation texts, it asks users to assess the correctness of statements about programming language concepts. We implemented conceptual checks on top of progmiscon.org, an educational repository of programming language misconceptions observed in students learning to program. The inventory currently catalogues more than 200 misconceptions. This demonstration illustrates conceptual checks as an efficient and effective means for instructors to access the relevant information in the large body of misconceptions.

Keywords: Misconceptions · Programming Languages · Refutation Texts · Self Assessment.

1 Pedagogical and Technological Background

Conceptual Change Based on constructivism, conceptual change theories have been applied to many fields in science education. The main observation is that learners always possess some prior knowledge, and thus learning does not always occur by accumulating facts, but by revisiting and changing wrong conceptions. For this reason, prior works have examined the role of *misconceptions* in the learning process, positing that learning occurs precisely by overcoming those wrong conceptions. This also justifies why learning (to program, in this case) is so difficult: students tend not to abandon their beliefs unless there are good reasons for doing so [5].

Refutation Texts Two decades of research on the topic of refutation texts [7] have shown that, among text-based instruments, they stand out as a powerful mechanism to overcome (wrong) knowledge ingrained in learners. Showing one next to the other the "statement of a commonly held misconception, and an explicit refutation of that misconception with an emphasis on the currently accepted scientific explanation" [7] has been proven to be more effective than just stating the correct conception. According to studies, using a clear expository

format should be preferred over a longer narrative description that can be perceived as less rigorous. A good refutation text should start by clearly stating the misconception, marking the transition with a "refutation cue" (e.g., "but this is wrong"), and finally presenting the correct conception. Textbooks, however, do not seem to employ them enough [7]. A variation on the theme that can more directly engage learners are the *conceptual change texts* [1]. Before presenting the correct conception, they actively involve the reader by asking to make a prediction, stimulating thinking before reading the scientifically correct fact. Going beyond text as a medium, *concept cartoons*, a visual version of refutation texts, have been used in science education since the early 1990s [3].

Conceptual Checks In contrast to the existing kinds of refutation texts, the purpose of the idea of *conceptual checks* introduced in this paper is not the teaching per se, but the preparation of teachers. This different purpose has a significant impact on the wording of the texts: misconceptions' statements are written using expert terminology instead of using a vocabulary more accessible to novices. The resulting texts have then minimal ambiguity as they use proper domain-specific terminology. In the domain of programming languages, one has the advantage of being able to refer to their authoritative specifications.

progmiscon.org As an effort to collect and properly document misconceptions about programming [4], we have built and we maintain progmiscon.org, a curated inventory of programming language misconceptions [2]. Besides a unique, memorable name, each misconception is characterized by an unambiguous statement that describes, entirely in terms of the syntax and the semantics of the relevant programming language, what is the wrong belief students hold (even though they might not express it in those very words). Moreover, misconceptions are accompanied with information about the possible origin (where it might come from) and common symptoms (artifacts produced by students who hold the misconception), with the goal of making educators aware of the rich body of wrong conceptions their students might develop in their journey of learning to program.

2 Conceptual Checks in progmiscon.org

Our inventory contains, due to its own nature, precise statements about the wrong conceptions novice learners have about how a certain language feature works. We have augmented this data with correct statements that contrast the incorrect ones and precisely describe what is the truly correct behavior, according to the authoritative programming language specification.

As an example, consider the Java misconception AssignmentCopiesObject. The statement which might be spoken by a student and describes the wrong conception is "assignment copies the object". Instead, the correct statement is "assignment copies the reference pointing to the object". By juxtaposing the two texts, one can easily obtain a refutation text such as "Some people believe that an *assignment copies the object*. That is not true: an *assignment copies the reference pointing to the object*".

However, we deemed that a list with just the refutation texts would probably not be appealing as a reading. Taking conceptual change texts one step further,

we built an interactive system in which visitors are presented with both statements, the wrong and the correct one, and are challenged to make a choice and select the one they believe to be true for a specific programming language. Only after thinking and selecting one of the two claims, visitors can click a button to "solve the mystery" and check whether their prediction was correct or not.

We call this process *conceptual check* (see Figure 1), as it can serve the purpose of self-assessing the knowledge of a programming language, or a subset of it, through the means of selecting what is true at a conceptual level.

Performing these checks on the whole body of misconceptions (which at the time of writing consists of more than 200 misconceptions) is unfeasible and also not very useful for revising. Instead, the checks can be conveniently configured so that they target a coherent set of misconceptions. In particular, one can select to do a check on misconceptions that pertain to a certain concept or a set of concepts, since all misconceptions are tagged with one or more concepts derived from the programming language field (e.g., Constructor, Expression, Type and many more); or a check on misconceptions that can be potentially induced by reading specific sections of a textbook, since they are also indexed by popular books used for teaching (only Java is supported at the moment).

3 Use Cases and Next Steps

Out Of Scope Asking novices to assess the correctness of conceptual statements such as "references can point to variables" can be problematic: novices do not just lack the expert terminology (e.g., the term "variable"), but for domains they are entirely unfamiliar with, they may not have any vocabulary at all (e.g., no word to denote the idea of a "variable"). Thus, when assessing the conceptual understanding of novices, conceptual questions are often too abstract and devoid of meaning, and there may be no rephrasing of the questions in the students' own words. To best assess novices, questions based on concrete examples and situations, like the kinds of questions used in concept inventories, are necessary. Thus conceptual checks are not targeted at students, but at teachers. Teachers, unlike their students, usually already possess the necessary domain vocabulary.
Teacher training Teachers are often confronted with students who struggle to grasp certain concepts. It has been shown that timely feedback is a key element for overcoming students' difficulties and wrong conceptions. It is even better if one could prevent those wrong conceptions from forming in the very first place, since once a knowledge element becomes familiar for a learner, it becomes difficult to replace it with the correct knowledge. For these reasons, pedagogical content knowledge includes knowledge about misconceptions [6]. When teachers know which misconceptions are likely to be developed, they can quickly recognize them and put in place a variety of strategies to deal with them, including devoting classroom time to address common issues, preparing or selecting extra material, and tailoring assignments. The explicit discussion of misconceptions during courses to train current or future teachers can thus be highly beneficial.
Instructor preparation for a lecture Instructors benefit from remembering what misconceptions might pop up in a specific lecture of their course, as they

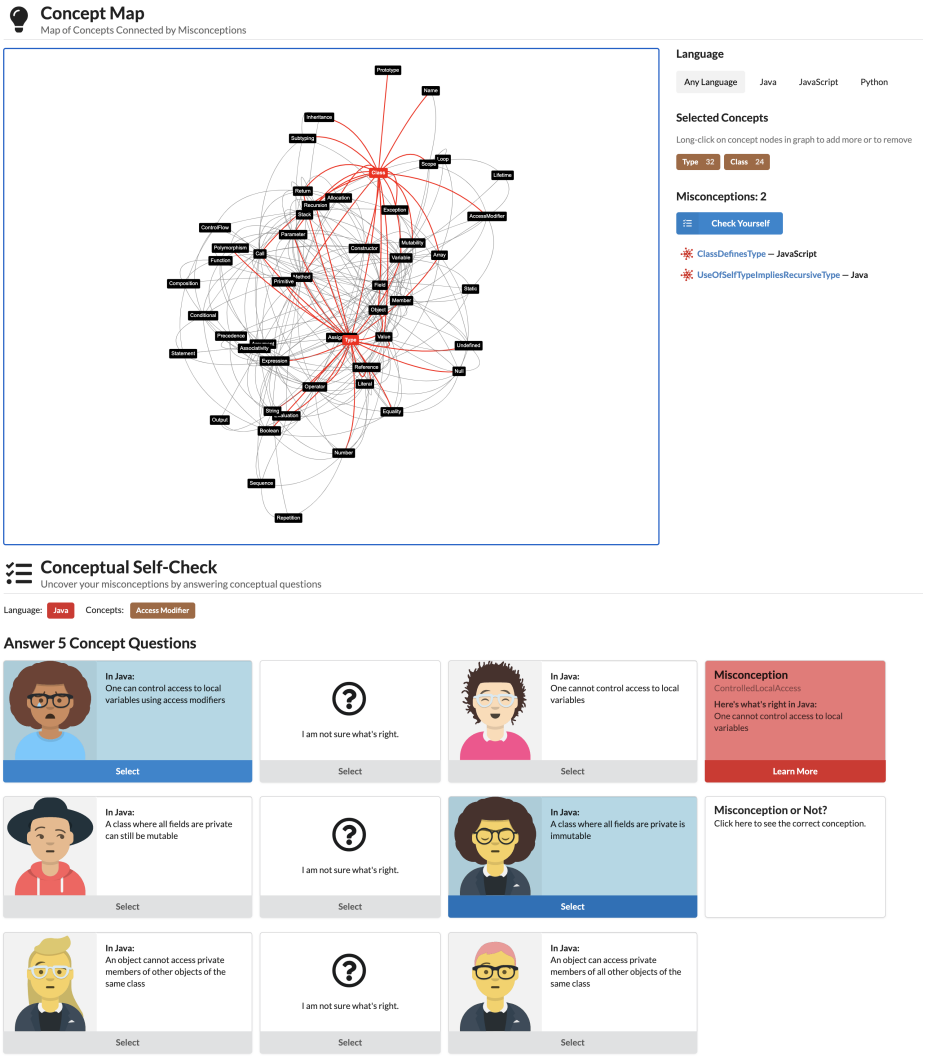

Fig. 1. The Concept Map allows teachers to configure a conceptual check by selecting a combination of concepts on which to be checked. The conceptual check then presents conceptual change text questions to the teachers. This provides them with an effective and efficient way to assess their own conceptual understanding and reminds them of the potential misconceptions students could hold about the chosen concepts.

can exploit tactics to challenge them or even anticipate them. When lectures are synchronized with a specific chapter of a book, instructors can take advantage of checks that focus exclusively on the contents of that chapter.

TA preparation for a course Many universities employ students as teaching assistants (TAs) to help instructors in the courses they teach. Conceptual checks offer a quick and effective opportunity to revise the course material for being better prepared to answer students' questions and being able to remedy their incorrect understandings. Moreover, given external constraints, it can happen that students are assigned to courses outside their specific area of expertise (for example, a doctoral student whose research interests lie in machine learning could be needed in an undergraduate course in introductory programming). In such cases, learning about misconceptions is even more important.

We hope that this tool will further increase teacher's awareness of the multitude of programming language misconceptions that have been the subject of research studies in the last decades. The goal is that this increased awareness about specific misconceptions can then translate into improved teaching. We explicitly welcome suggestions and new contributors to progmiscon.org, to improve the quality and the quantity of conceptual checks, foster the community around it, and increase the value for educators all around the world.

References

1. Chambers, S.K., Andre, T.: Gender, prior knowledge, interest, and experience in electricity and conceptual change text manipulations in learning about direct current. Journal of Research in Science Teaching **34**(2), 107–123 (1997). https://doi.org/10.1002/(SICI)1098-2736(199702)34:2¡107::AID-TEA2¿3.0.CO;2-X
2. Chiodini, L., Moreno Santos, I., Gallidabino, A., Tafliovich, A., Santos, A.L., Hauswirth, M.: A Curated Inventory of Programming Language Misconceptions. In: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1. pp. 380–386. ITiCSE '21, Association for Computing Machinery, New York, NY, USA (Jun 2021). https://doi.org/10.1145/3430665.3456343
3. Keogh, B., Naylor, S.: Concept cartoons, teaching and learning in science: An evaluation. International Journal of Science Education **21**(4), 431–446 (Apr 1999). https://doi.org/10.1080/095006999290642
4. Qian, Y., Lehman, J.: Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. ACM Transactions on Computing Education **18**(1), 1–24 (Oct 2017). https://doi.org/10.1145/3077618
5. Sawyer, R.K. (ed.): The Cambridge Handbook of the Learning Sciences. Cambridge Handbooks in Psychology, Cambridge University Press, Cambridge, second edn. (2014). https://doi.org/10.1017/CBO9781139519526
6. Shulman, L.S.: Those Who Understand: Knowledge Growth in Teaching. Educational Researcher **15**(2), 4–14 (Feb 1986). https://doi.org/10.3102/0013189X015002004
7. Tippett, C.D.: Refutation Text in Science Education: A Review of Two Decades of Research. International Journal of Science and Mathematics Education **8**(6), 951–970 (Dec 2010). https://doi.org/10.1007/s10763-010-9203-x