Computational Problem Solving with Plethora

Michal Armoni1[0000-0002-8988-9023], Judith Gal-Ezer2[0000-0002-3630-5733],

Michal Haskel-Ittah1[0000-0003-2626-5835], Rami Marelly3 and Smadar Szekely1

1 Weizmann Institute of Science, Rehovot, Israel
2 The Open University of Israel, Israel
3 Plethora Technologies Ltd., Israel
galezer@cs.openu.ac.il

Abstract. Many educational endeavors address the challenge of developing approaches, tools and platforms for introducing computational problem-solving skills, often referred to as computational thinking, into the school system. Plethora, a game-based platform, teaches the concepts and skills of computational problem solving, by means of graphically attractive logical challenges that are expressed in a natural language. It was developed at the Weizmann Institute of Science, in collaboration with The Israeli Center for Educational Technology. Plethora is used in schools across Israel and was selected as the leading tool for national school-level cyber competitions. In this panel, we will explain our approach towards computational problem solving, show how it is implemented with Plethora, present the scientific basis of Plethora, and finally show how Plethora can be extended to discuss computational models of scientific phenomena.

Keywords: Computational Thinking, Problem Solving, STEM.

1 Computational Problem Solving

Computational problem solving (CPS) constitutes the object of research and practice of computer scientists. They devise solutions to computational problems and study the attributes of computational problems and solutions. When doing that, they employ a set of ideas, strategies, skills, and thinking patterns. This set is nowadays mostly referred to as computational thinking [1], though the familiar term “algorithmic thinking” [2] expresses the same notion. We will refer to this set by the term CPS, to emphasize the context in which it is used.

Wing [1] viewed CPS as relevant to everyone, not just to computer science (CS) experts, and therefore called for teaching it to every student, thus helping them to acquire a powerful and useful set of tools. In fact, these concepts and skills have always been employed in areas and contexts outside CS. However, during its evolution as a discipline, CS enhanced and refined them, and hence learning them in the context of CS can foster their acquisition, so they can later be put to use in other, non-CS contexts. The latter is in fact the goal - achieving transfer of CPS from CS to other contexts.

A wide-spread CS context for teaching CPS includes various friendly programming environments. These may be based on different programming paradigms (for example, Alice and Scratch are based on the object-oriented and event-driven paradigms, respectively), but all of them share a common core of imperative programming, that is, sequences of commands combined with some control structures, which instruct an obedient agent (the computer) what to do, step by step. Moreover, the teaching of these commands and control structures often constitutes the main part of teaching CPS through these environments.

Plethora is based on a very different programming paradigm, in which one has to state what will happen rather than how it is done. It focuses on behaviors rather than on instructions that may serve to create these behaviors. Nevertheless, it employs the same general set of concepts and skills [5], and thus provides opportunities for learning and experiencing them in a context that is perceived as playing rather than programming, and hence may foster more meaningful transfer.

2 Computational Problem Solving with Plethora

Plethora is an online game-based educational platform where students solve challenges as well as create new ones by themselves. Challenges are organized by topics representing algorithmic concepts, each of which includes levels of varying difficulty.

A Plethora challenge is a small system with a set of possible behaviors and is composed of four parts: *An initial state* – a set of shapes present on the screen when the level starts; *a goal* – the set of shapes required to be on the screen at the end of the level; a set of *partially specified activation rules* – upon completion define the behavior of the system represented by this level; and a *halting rule* – specifies the condition that causes the level to end (see Fig. 1).

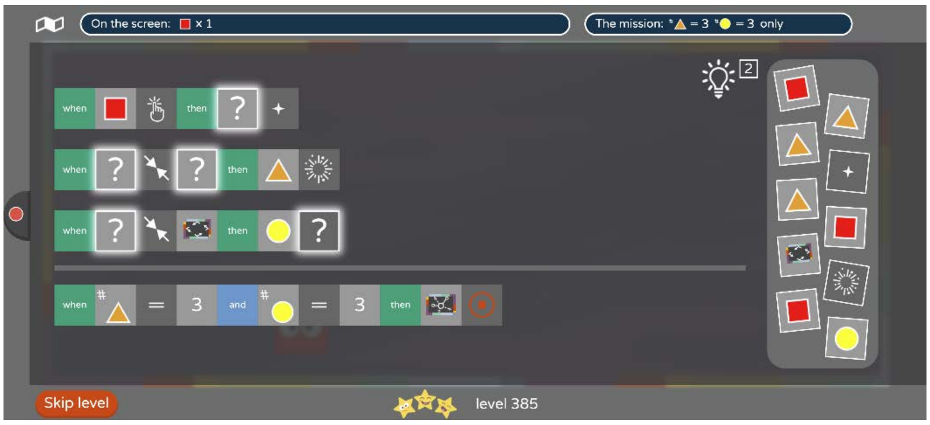

Fig. 1. A screenshot of Plethora

A challenge requires completing the partially specified rules so that, when run, they will cause the system to behave correctly and halt with the goal state holding. The rules themselves are depicted visually, using simple and intuitive icons, and are independent of conventional programming language syntax. The graphic visualization is accompanied by text in the students' native language. The user can also experiment with completing and activating only some of the rules, observing how these affect others, and determining the overall effect on the system's execution[1].

In addition to addressing challenges, Plethora offers a studio with which students can create new challenges and share them with the Plethora community. The combination of a closed approach, where given challenges are to be solved, and an open one, where users create challenges of their own, introduces an important benefit.

3 From LSC to Plethora Science

Plethora evolved from the innovative Scenario Based Programming approach developed at the Weizmann Institute of Science. This approach, together with an extended Life Sequence Charts (LSC) language [3,4] was developed for intuitively specifying and simulating complex systems' behavior without the need for actually coding the system. This approach and its supporting tools introduced an extensive and unique way of specifying requirements, analogous to a human's way of thinking when defining scenarios of a systems' behavior.

After more than a decade of experience with the LSC language and tools, Plethora was developed with the idea of utilizing a simple and intuitive, yet expressive, visual language, adapted for young students, to allow them practice CPS skills, by describing the behavior of abstract systems based on geometric shapes [5]. Plethora has been used by more than 200,000 students worldwide and proved its visual language to be intuitive and clear.

Following the directions, recently pointed out in the K-12 Science teaching framework, mainly the requirement to acquire and practice CT skills to develop, experience and discuss computational models of scientific phenomena [6], Plethora is now extended to Plethora Science to allow just that. It can now be used by students to develop scientific models using Plethora's enriched language, to explain and predict natural phenomena. In the next section we show briefly how Plethora Science can form a bridge between CPS and STEM education.

4 Science Aspects - Mechanistic Reasoning and Modeling

From high creativity to error findings, the benefits of interdisciplinary in problem solving are widely discussed [7]. Thus, interdisciplinary teaching and learning gained the attention of science educators in the past two decades. For example, the K-12 framework for science and engineering directs educators toward focusing on crosscutting concepts and common practices [6].

[1] Some video clips demonstrating Plethora in action can be found at: https://drive.google.com/open?id=1CEDOIfwxqZe9c5Sh5tLY1dUmrrd2SlJM.

When simultaneously teaching two fields with the aim of solving problems or explaining phenomena, the challenge is allowing the two fields to mutually inform each other [8]. Two highly connected practices from the fields of CS and natural sciences are CPS and mechanistic reasoning. Mechanistic reasoning is a central scientific practice in which phenomena are explained by describing the interactions between entities.

This type of thinking, in which a given input (cause) and output (effect) are explained by steps of functions and interactions between entities, shares similarities with CPS. Thus, teaching the use of both for explaining and simulating natural phenomena is an opportunity for deep interdisciplinary teaching. Such teaching links between the K-12 frameworks' crosscutting concepts "Using mathematics and computational thinking" and "Developing and using models" with the crosscutting concept "Cause and effect: Mechanism and explanation" and core ideas from the natural sciences (e.g., ecosystems or motion and stability).

Plethora Science is an example of such interdisciplinary teaching. It enables students to model and explain natural phenomena by building simulations. At the beginning of the task students can explore phenomena using a given simulation (e.g., a balanced ecological system). Then, they are provided with several pieces of evidence of input (cause) and output (effect) connections (e.g., when the number of flowers dropped, the number of birds dropped). Students use the evidence for constructing or refining rules. Then, they can run the simulation which is based on the rules they constructed. Once their model is stable and is supported by all the given evidence, students can use it for predicting the outcome of different changes, for example, predicting an ecological collapse as a result of a small change in temperature due to global warming. By using this interdisciplinary platform, students may concurrently develop CPS and mechanistic reasoning along with other STEM practices such as building models and reasoning from evidence in the context of natural phenomena.

References

1. Wing, J. M.: Computational thinking. Communications of the ACM 49(3), 33–35 (2006).
2. Knuth, D.: Algorithmic thinking and mathematical thinking. The American Mathematical Monthly 92(3), 170–181 (1985).
3. Damm, W., Harel, D.: LSCs: Breathing Life into Message Sequence Charts. Formal Methods in System Design 19(1), 45-80 (2001).
4. Harel, D., Marelly, R.: Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine. Springer-Verlag, Berlin (2003).
5. Gal-Ezer, J., Marelly, R., Szekely, S.: Plethora of Skills: Play-Learn-Practice-Invent-Share. Proceedings of the 25th Annual Conference on Innovation and Technology in Computer Science Education, pp. 15–19. ITiCSE 2020, ACM, New York, NY, USA (2020).
6. National Research Council: A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas. Washington, DC: The National Academies Press (2012).
7. Nissani, M.: Ten cheers for interdisciplinarity: The case for interdisciplinary knowledge and research. The Social Science Journal 34(2), 201-216 (1997).
8. Carter, L.: Interdisciplinary computing classes: worth the effort. In Proceedings of the 45th ACM Technical Symposium on Computer Science Education, pp. 445-450. SIGCSE 2014, ACM, New York, NY, USA (2014).