Journal for STEM Education Research
https://doi.org/10.1007/s41979-021-00058-y

RESEARCH ARTICLE

Students' Use of Computational Thinking Practices in an Undergraduate Biology-Engineering Course

Anna F. DeJarnette[1] · Corey Larrison[1] · Stephanie M. Rollmann[2] · Dieter Vanderelst[3] · John E. Layne[2] · Anna E. Hutchinson[2]

Accepted: 25 June 2021

Abstract

The construct of Computational Thinking (CT) first emerged to describe problem solving in the context of computing environments, but it has expanded to serve as a set of practices that can be applied across disciplines with or without the use of computers. We recorded students' work during two lab sessions in an undergraduate, biology-engineering course to answer the question, how did students' participation in CT practices vary according the disciplinary contexts and the demands of a biology lab compared to the engineering lab? We found that students applied some of the same CT practices, but these practices were used for different purposes across the two labs. The biology lab allowed for more varied CT practices, in part because students did not have to do any programming. These findings indicate how a biology lab that used a computational model, but not a computer, to understand a biological phenomenon led to meaningful engagement in CT. Documenting the ways that different types of tasks from different disciplines elicit aspects of CT in students can make the construct more useful for designing learning experiences and documenting student learning.

Introduction

Computational Thinking (CT) has become increasingly popular as a descriptor of the habits and practices involved in problem solving since it was first introduced by Wing (2006). When Wing proposed CT as a construct, she characterized it as, "problem solving, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33). Since then, CT—and computation more generally—has been taken up by researchers to capture students' work not only in computer science contexts (Chao, 2016; Lye & Koh, 2014; Wu et al., 2019) but also in science (Peel et al., 2019; Wilensky & Reisman,

✉ Anna F. DeJarnette
dejarnaa@ucmail.uc.edu

Extended author information available on the last page of the article

Published online: 02 August 2021

2006), mathematics (Lockwood et al., 2019), and engineering (Vieira et al., 2019). CT is included in the *Next Generation Science Standards* (NGSS) as a practice that K–12 students should develop (NGSS Lead States, 2013). The term's growing popularity reflects an interest among educators, researchers, and policymakers in capturing the range of skills, habits, and dispositions necessary for science, technology, engineering, and mathematics (STEM) professionals. This is particularly salient as the power of modern technology reduces the need for more rote aspects of STEM work that can now be done by computers.

Although there is a clear interest across disciplines to incorporate and develop CT (e.g., Grover & Pea, 2013), how CT is enacted across disciplinary settings is much less clear. Adding to this ambiguity is that there is little consensus around precisely what CT is (National Research Council, 2010). It has alternately been described as a set of concepts and capabilities (Barr & Stephenson, 2011); as a collection of concepts, skills, and perspectives (Brennan & Resnick, 2012); and as a set of applied practices in a variety of disciplinary contexts (Weintrop et al., 2016). In the present study we adopt Weintrop et al.'s notion of CT as a collection of practices, and we use "practice" to refer to an activity that is shared among a cultural community (Gutiérrez & Rogoff, 2003). Scientific disciplines such as biology or engineering are made up of communities who establish certain norms for activites such as collecting data, constructing models, or solving problems.

Because different disciplines have distinct norms around the practices associated with CT, the learning experiences that are afforded to students likely create varied opportunities to participate in these practices. The degree to which students can learn CT largely depends on the specific demands of a task, the more general design of a learning environment, and the norms of a discipline. With this study we describe how students practiced CT in an undergraduate biology-engineering course, with particular attention to the differences that surfaced between students' work in a biology lab compared to an engineering lab. This study provides a contrast of CT activities between a lab that incorporates computer programming and one that does not. We provide a case for the range of ways that students can engage CT, and a case for how decoupling CT from computers can foster creative spaces for students to engage in meaningful forms of CT.

Review of Literature

Defining CT

Wing's (2006) initial definition of CT, which emphasized solving problems using concepts of computer science, was gradually refined to describe a process of formulating problems "so their solutions can be represented as computational steps and algorithms" (Aho, 2012, p. 832). Many applications of CT in computer science come in connection to students learning computer programming skills, in K–12 contexts (Atmatzidou & Demetriadis, 2016; Bers et al., 2014; Brennan & Resnick, 2012; Lye & Koh, 2014; Repenning et al., 2010), and to a lesser extent, in higher education (Czerkawski & Lyman, 2015; Hasni & Lodhi, 2011; Wu et al., 2019). In

a review of research on CT published between 2006–2016, Ilic et al. (2018) found that the most frequent topic of empirical studies of CT was the development of particular skills, and this was most frequently done through programming instruction, the use of computer software, and robotics. Because computer programming itself encompasses elements of design, problem solving, and reflection (Chao, 2016), it is reasonable that studies of CT in computer science use computer programming as a vehicle for developing this mode of thinking.

Part of the effort to decouple CT from computer programming has been to characterize it in a way that acknowledges how CT extends beyond the use of a computer. Brennan and Resnick (2012), whose work was situated primarily in the realm of agent-based programming environments with young learners, characterized CT as a collection of concepts, practices, and perspectives. While Brennan and Resnick's use of concepts and practices seemed to refer more squarely to computer programming and coding (e.g., concepts included conditionals and operators, and practices included debugging and abstracting), their description of CT perspectives included broader problem-solving activities such as making connections and questioning. Barr and Stephenson (2011), who sought to operationalize CT in a way that was relevant across school subjects including social studies and language arts, described it in terms of concepts such as algorithms, and capabilities such as data collection and problem decomposition.

Weintrop et al. (2016) developed a taxonomy of CT practices based on an analysis of classroom materials from a variety of disciplines and interviews with STEM professionals, teachers, and researchers. Their taxonomy is divided into four categories of practices, including data practices, modeling and simulation practices, computational problem-solving practices, and systems thinking practices (Table 1). Data practices incorporate the range of practices used to create, collect, interpret, and represent empirical data. Modeling and simulation practices represent the practices associated with modeling real-world phenomena. Computational problem solving practices are those most closely aligned with computer programming and computer science—translating problems into computational language, developing solutions, and troubleshooting. Systems thinking practices refer to ways of approaching problems alternately through macro or micro lenses. Weintrop et al.'s taxonomy shares several features with other efforts to characterize CT, including focused activities such as programming and debugging in addition to broader activities such as thinking in levels and using models to understand concepts. All of these characterizations acknowledge that work at a computer is but one aspect of the range of activities that constitute CT.

Application of CT to STEM Learning

In the United States, CT is included as one of eight science and engineering practices for K–12 education (NGSS Lead States, 2013). Perhaps the strongest connection between CT and school science comes from the use of computer programming environments to model scientific phenomena. The use of agent-based environments, where learners control an "agent" on the screen, requires learners to align multiple

Table 1 CT Practices described by Weintrop et al., 2016 (p. 135)

Data	Modeling and simulation	Computational problem solving	Systems thinking
Collecting data	Using computational models to understand a concept	Preparing problems for computational solutions	Integrating a complex system as a whole
Creating data	Using computational models to find and test solutions	Programming	Understanding the relationships within a system
Manipulating data	Assessing computational models	Choosing effective computational tools	Thinking in levels
Analyzing data	Designing computational models	Assessing different approaches/solutions to a problem	Communicating information about a system
Visualizing data	Constructing computational models	Developing modular computational solutions	Defining systems and managing complexity
		Creating computational abstractions	
		Troubleshooting and debugging	

points of view—including their own, the perspective of a moving object, as well as a computational agent—to create computational models (Farris & Sengupta, 2014). Alternating among these points of view to refine computational models can support students' careful interrogation of the relationships between time and displacement of a physical object (Farris et al., 2020). Computational modeling can also be productive for learning about biological systems such as the spread of disease (Swanson et al., 2021), evolution (Horn et al., 2014), predator–prey population dynamics, and coordinated behavior among a species (Wilenksy & Reisman, 2006). In all these cases, creating and then refining models provoked students to delineate the impacts of events on different levels within a system.

There are fewer studies of how students might develop CT practices distinct from computer programming, because most studies are situated in computer programming settings (Ilic et al., 2018; Kalelioğlu et al., 2016). Peel et al.'s (2019) study of how secondary biology students used CT to learn about natural selection is one important exception. In Peel et al.'s study, students learned CT principles and applied them to develop "unplugged" (i.e., without using a computer) algorithmic explanations of natural selection in various contexts. As they learned about the concept of natural selection, students also developed the use of CT principles, including branching (i.e., choosing a path based on a condition) and iteration. This study marks an important example of how students created and used computational models—by which we mean "nonstatic representations of a phenomena that can be simulated by a computer" (Weintrop et al., 2016, p. 137)—without the actual use of a computer. Unplugged activities such as this can be viewed as an extension of embodied modeling, in that they allow learners to consider the behaviors of actors within a system rather than only specifying relationships between variables (e.g., Swanson et al., 2021; Wilensky & Reisman, 2006). That students stop at the point of creating algorithms, without encoding them on a computer, does not preclude such activity from being characterized as CT. Efforts to broaden the applicability of CT as an educational construct are supported by explorations of the range of computing activities that complement and precede writing a computer program.

Setting of Study and Research Questions

This study documents students' work in an undergraduate, transdisciplinary biology-engineering course. We define "transdisciplinary" teaching and learning as combining the knowledge and skills of multiple disciplines in order to apply the knowledge and skills to open-ended, real-world tasks or projects (Vasquez, 2015). In this course, the summative project required students program a robot that could autonomously navigate an arena of students' design. Throughout the semester, students studied computer programming as well as sensory-guided behavior in biological organisms, with the intent that students would draw upon biological principles of object detection and avoidance to complete their final project.

For this study we recorded students' work during two labs—one biology lab where students modeled (with their bodies) kinesis and taxis as mechanisms for orientation, and one engineering lab where students programmed robots equipped

with sensory mechanisms to avoid a specified obstacle. Our study was guided by the following research question: How did students' participation in CT practices vary according the disciplinary contexts and the demands of the biology lab compared to the engineering lab? This study illustrates how CT can be applied by learners in generative ways that support learning and inquiry, particularly when a task allows students to explore a new phenomenon.

Data and Methods

The Biology-Engineering Course and the Two Focal Labs

The learning objectives of the biology-engineering course included describing properties of natural stimuli, explaining how biological and human-made sensors process stimuli, and comparing human-made sensors with biological sensory organs. The course was taught by the coauthors of this paper, who bring varying expertise in sub-disciplines of sensory biology, robotics, and psychology. The course met once per week, and the instructors used direct instruction as well as labs for students to gain understanding and hands-on experience of the relevant concepts. Early in the semester, class sessions were typically organized to include instruction on biology, engineering, and computer programming. Toward the end of the semester, students needed to integrate these different skills to complete the final robotics challenge. To compare students' use of CT according to the disciplinary demands of different contexts, we selected one biology lab from the first third of the semester and one engineering lab that took place toward the middle of the semester. These two labs were designed to develop the knowledge and skills that students would ultimately use to complete their final project in the course. The labs were not designed specifically to teach CT. Thus, this study documents the ways that CT surfaced as part of students' work across the two settings rather than the result of a specific intervention.

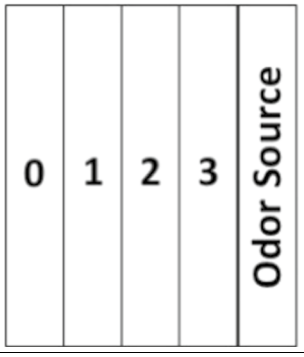
The biology lab. The biology lab was entitled "Chemokinesis versus Chemotaxis". These terms refer to specific biological mechanisms by which organisms orient themselves to external stimuli (e.g., avoiding or approaching an odor, Fraenkel & Gunn, 1961). Kinesis is an undirected movement in response to the local intensity of an external stimulus. In contrast, taxis is directed movement toward or away from a source of stimulation. Prior to the lab students learned about the biological principles by which animals identify and discriminate odors in their environment, how robots can be used to understand odor-tracking in animals, and how engineering has exploited the biological principles underlying odor coding to construct electronic noses for the detection of chemical stimuli. Immediately before the lab, students learned about kinesis and taxis.

Students modeled the principles of kinesis and taxis through a four-part lab modeled after Williams (2010). In this lab, students moved around the classroom according to different sets of rules. They compared what percent of students reached a stimulus (a cookie, in the case of the lab) under the various rules. A large section of classroom space was cleared of desks and other furniture. In part 1 of the lab, students began by spreading themselves "randomly" around the room and performed

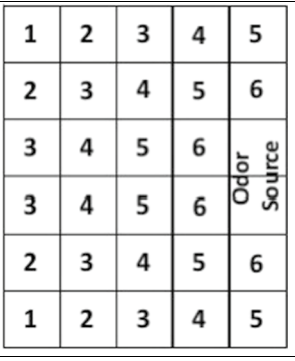
a random walk, defined as a sequence of alternating tumbles and runs. A "tumble" was a change in direction determined by spinning a hand-held spinner; a "run" was a sequence of 2 steps in whatever direction the student was facing. In this first part of the lab, students performed 6 tumble-run combinations to create undirected movement relative to the stimulus. Following their observations of the outcomes of this random walk, they had time to discuss, in pairs, alternative mechanisms that could improve the use of tumbles and runs for reaching the stimulus.

In part 2 of the lab, the classroom was divided into segments by putting tape on the floor, forming bands that spanned the width of the room and numbered from 0 to 3 (Fig. 1a). The random walk protocol was modified to make it biased toward reaching the goal. Instead of taking 2 steps for each run, students varied the number of steps, depending on the floor section where they stood. In the section farthest from the odor source, they took 2 steps, and nearer to the odor source, they took 3, 4, or 5 steps, respectively. This altered procedure made students travel farther per run if they were closer to the source, a form of kinesis.

Parts 3 and 4 of the biology lab were designed to model chemotaxis, or movement directed relative to a stimulus source. In part 3, students again performed tumbles and runs along the numbered bands as in Fig. 1a. Students altered each run's length depending on whether they were oriented toward or away from the stimulus—as determined by comparing the numbered strip they faced to the strip where they were located—rather than only based on their current location. This mimicked taxis as employed by various organisms. In part 4 of the lab, the classroom floor was divided into a grid of numbered squares, 1 foot by 1 foot, with the number pattern on the floor representing the concentration pattern of an odor plume originating at



a A model of the classroom floor during parts 2 and 3 of the biology lab. In part 2, students took more steps if they were nearer the odor source. In part 3, students took steps if they were oriented towards the odor source.



b A model of the classroom floor during part 4 of the biology lab. Students held a pair of sensors and moved towards the higher number, which indicated moving towards the odor source.

Fig. 1 Diagrams indicating floor layout for modeling kinesis and taxis. **a**. A model of the classroom floor during parts 2 and 3 of the biology lab. In part 2, students took more steps if they were nearer the odor source. In part 3, students took steps if they were oriented toward the odor source. **b**. A model of the classroom floor during part 4 of the biology lab. Students held a pair of sensors and moved toward the higher number, which indicated moving toward the odor source

a stimulus (Fig. 1b). Each student held a pole with a "sensor" at each end used to measure the local "odor intensity". These sensors were two laser pointers pointing straight down. Students held their poles in front of their bodies. After each run, they noted the right and left lasers' locations on the numbered grid on the floor. Students changed their direction based on a comparison between the two sensors: if the right sensor detected a higher concentration than the left, students turned 90° to the right; likewise for the left. To test whether different distances between two sensors affects the ability to localize the odor, some students had short poles while others had poles twice as long. This procedure derived some directional information from comparing sensors. Therefore, this procedure mimicked biological taxis.

At the conclusion of part 4 of the biology lab, students had time to discuss and respond in writing to three prompts. Prompt 1 asked whether the different mechanisms for reaching the odor source differed in their effectiveness. Prompt 2 asked whether the spatial segregation of the sensors in part 4 of the lab impacted odor localization. Finally, prompt 3 asked students to consider the implications of the lab for robotics design. The questions at the end of the lab were designed for students to reflect on the nature of odor localization, how it is best accomplished, and how these principles might be applied to robotics design.

The engineering lab. The engineering lab was an "obstacle avoidance" lab, which was students' first experience automating a set of robots. The robotic platform used in the class was the iRobot Create 2®, an educational version of the Roomba vacuum. These circular robots had six built-in infrared sensors, arranged across the front of the robot, that could detect the presence of obstacles. The sensors recorded numerical values ranging from 0 (immediate obstacle) to 1 (no obstacle within range). Prior to the lab, students learned about the working principle of the sensors. The students also learned about the code required to control the robot from Python, building on earlier classes on Python programming. This lab's overarching goal was for students to program the given robots to move autonomously and respond to obstacles within an arena (Fig. 2).

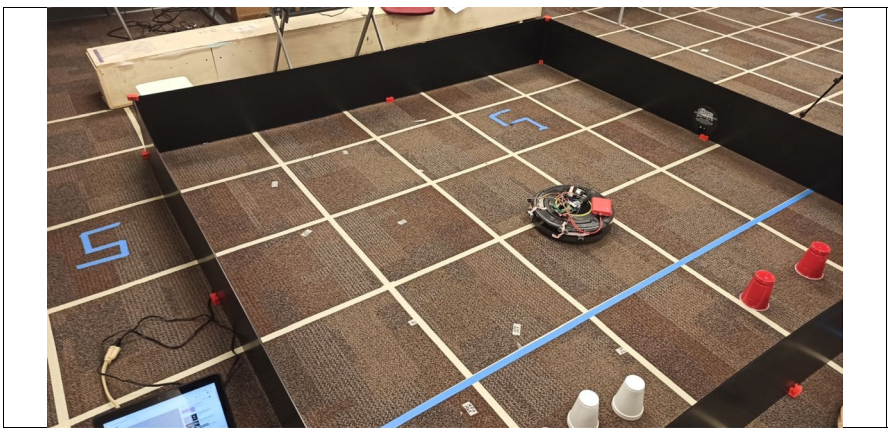


Fig. 2 An iRobot Create 2 and an "arena" used by a pair of students

The engineering lab was divided into three parts. In part 1 of the lab, students had to establish a set of commands that would move the robot forward if no obstacle was detected and stop the robot if an obstacle was detected. Students were provided with a set of paper strips that included commands and logical connectors written in nontechnical language. Students worked in pairs to organize their commands; then, the instructor displayed a correct solution, and students could compare their own solutions.

In part 2 of the lab, the robot's goal was to move forward in the absence of an obstacle and turn right or left if an obstacle was detected. Students were again given a set of paper strips. These consisted of a solution program, in Python, cut up line by line. Similar to part 1, students needed to organize the paper strips (lines of the program) in a logical order that would allow the robot to complete the task. Once they did this, they were able to compare their code to a working program.

In part 3 of the lab, the robot's objective was to slow down if an obstacle was detected and turn right or left to avoid the obstacle. Students needed to write their own programs using Python and test them on the robots at this stage of the lab. Students could refer to commands for turning in place, moving forward, collecting sensor data, and setting linear and angular velocity that had been introduced earlier in the class. Students also could have adapted the code that had been provided to them in part 2 and incorporated additional commands to slow down the robot. Students needed to piece these commands together and adjust quantities as necessary—for example, collecting sensor data, determining whether it was under a particular threshold, slowing down, and turning away from any sense obstacle. At the conclusion of the engineering lab, each pair of students shared their programs, and the instructor facilitated a conversation to compare and contrast features of the different programs.

Participants

Thirteen students participated in the undergraduate biology-engineering course. These included four high school seniors (Grade 12, 17–18 years old) who had participated in the previous summer program, and nine undergraduates from a range of majors across the university. Twelve students consented to participate in the study, which was reviewed and approved by the university's research ethics board, including all four high school students. Students were presented with the details of the study at the start of the semester, and all students had the opportunity to read and sign a consent form. High school students who were under 18 years old also received parent permission to participate. Due to some absences five pairs of students participated in each lab (Table 2; all names are pseudonyms). Students in the course were allowed to choose their own partners, and most students worked with the same partners consistently throughout the semester. Each pair of students participated in both labs, with the one exception that Omar replaced Opal (due to Opal being absent) in the engineering lab.

Table 2 Student Pairs

Kara and Avni
Marie and Nathan
Natalie and Melody
Noelle* and Opal (Omar* replaced Opal in Engineering Lab)
Sadie* and Camila*

Note: Asterisk (*) indicates a high school student

Data and Analysis

We placed an audio recorder with each pair of students to record students' conversations during pair-work portions of each lab. Prior to systematically analyzing the data, the second author listened to each recording to make notes about students' work throughout each lab. From this initial review, we eliminated off-task sections of audio prior to transcribing. This included segments in which a pair of students had completed part of a lab and were waiting for others to move on. It also include segments early in the engineering lab when students were having trouble connecting their computers to the robots, and the instructor helped each pair connect the devices. Once we had eliminated the superfluous audio, we had the audio records transcribed for coding. Transcripts of students' conversations during the labs constituted the primary data that we analyzed to document students' CT practices, supported by reviewing their written lab responses to help interpret their conversations. We used students' discourse as markers of their participation in different practices, reflecting a primary focus on the process by which students engaged CT rather than on the final products they produced.

We applied Weintrop et al.'s (2016) taxonomy of CT practices to code students' discussions (Table 1). While there are several common features across many efforts to characterize CT, the Weintrop et al. taxonomy was appropriate for this study for several reasons. First, their attention to practices, rather than specific knowledge or concepts, allowed us to attend to the question of students' participation in CT in applied settings. This is consistent with a discipline-based approach to CT that emphasizes formulating solutions to problems (Li et al., 2020a). Additionally, the particular attention to "systems thinking" practices was relevant to understanding differences in the work we observed among students across the two labs. Beginning with the transcripts and notes of students' work, we used an iterative coding process (Strauss & Corbin, 1998) to document the CT practices that students applied. The first and second author used Weintrop et al.'s (2016) taxonomy of 22 CT practices as a priori codes to code the transcripts of students' conversations. We used the definitions and descriptions of these practices as described by Weintrop et al. to characterize students' turns of speech (see Appendix Table 3).

After documenting codes directly in the transcripts, we copied and pasted excerpts of the transcripts into another document to organize them in two ways. First, we grouped transcript excerpts according to individual CT codes. This grouping allowed us to investigate how a single CT practice, such as "thinking in levels", might have looked similar or different in the biology lab compared to the

engineering lab. The first and second authors shared transcript excerpts with the other authors—who had taught the lessons—at this stage to check the validity of our interpretations of students' CT practices against the disciplinary expertise of the course instructors. In our second grouping, we created a table for each lab where we pasted excerpts and codes chronologically from beginning to end of lab, with a column for each pair of students. With this organization, we looked for patterns in how students' CT practices were grouped together, noting the ordering of practices throughout each lab and which pairs or groups of practices most frequently appeared chronologically next to each other in transcripts.

Findings

The purpose of this study was to answer the question, how did students' participation in CT practices vary according the disciplinary contexts and the demands of the biology lab compared to the engineering lab? Through the phases of our analysis we identified two major ways in which students' participation in CT practices varied. First, we noted some differences in how students sequenced or grouped CT practices, largely related to the different structures and demands of the two labs. Second, we noted instances when the biology lab elicited different ways of engaging students in CT compared to the engineering lab. We present our findings in two subsections, according to these two insights.

An Overview of the CT Practices That Students Applied in Each Lab

Figure 3 provides an overview of the CT practices we documented during the engineering lab. The lab, which required students to program a robot to autonomously avoid obstacles based on sensor data, primarily elicited computational problem solving practices. Because the lab scaffolded students' work so that each consecutive exercise built on the previous, each exercise elicited the same CT practices as the one before as well as some additional. In exercise 1 students needed to organize and sequence paper strips with commands written in nontechnical language. This work involved preparing a problem for a computational solution, creating a computational abstraction, and understanding the relationships within a system. Exercise 2, which required students to sequence a set of commands that were written with Python programming language, elicited the same practices as exercise 1 in addition to students assessing different approaches to the problem (through considering different ways of organizing the code) and troubleshooting. Exercise 2 also elicited the systems thinking practice of thinking in levels as students shifted between the specific line-by-line details of the program and the robot behavior the program specified. In exercise 3 students needed to program their robots. In addition to the range of practices that they had previously applied in exercises 1 and 2, students collected sensor data that informed the action of the robot. The engineering lab did not elicit any modeling practices, because the lab was not designed as a model of a specific biological behavior or concept.

	Data Practices	Computational Problem-Solving Practices	Systems Thinking Practices
Exercise 1		Preparing problems for computational solutions	Understanding the relationships within a system
		Creating computational abstractions	
Exercise 2		Assessing different approaches to a problem	Thinking in levels
		Troubleshooting and debugging	
Exercise 3	Collecting data	Programming	

Fig. 3 A summary of students' application of CT practices during the engineering lab. Because the exercises scaffolded students' work toward writing a program, each exercise incorporated the practices of the previous exercise as well as some additional practices

Figure 4 illustrates the CT practices that students applied during the biology lab. The biggest difference between the two labs was that students applied a range of CT practices throughout the four parts of the biology lab, with no clear delineation in how these practices were sequenced. This difference is largely related to the design of the two labs—with the engineering lab carefully scaffolded to develop students' programming skills, and the biology lab more open-ended. During the biology lab,

	Data Practices	Modeling and Simulation Practices	Computational Problem-Solving Practices	Systems Thinking Practices
Parts 1-4	Creating data	Using computational models to understand a concept	Preparing problems for computational solutions	Understanding the relationships within a system
	Collecting Data	Designing computational models	Assessing different approaches to a problem	Thinking in levels
				Investigating complex systems as a whole

Fig. 4 A summary of students' application of CT practices during the biology lab. Students applied the range of practices throughout the four parts of the lab

students spent time toward the beginning of the lab anticipating how they might improve a random walk. They spent time at the end of the lab reflecting on their use of different mechanisms for odor localization. Students applied some practices from each of the four categories of CT practices during the biology lab. As students modeled the behavior of organisms and recorded their findings, they created and collected data. They prepared problems for computational solutions through proposing mechanisms by which to reach an odor source; they also compared different approaches to the problem of reaching the odor source. Throughout the lab students practiced systems thinking by alternately considering the relationships between individual actors (i.e., the students moving around the room and their rules for doing so) and the overarching outcomes of the group.

Because the biology lab was designed as a model of chemokinesis and chemotaxis, students also applied modeling and simulation practices. Although the simulations of movement were run by students themselves, and not on a computer, we consider the lab to be a computational model because it represented odor localization through sets of rules that could have been translated to a computer. The biology lab can be considered an "unplugged" model, and students used the model to understand how biological organisms perform chemokinesis and chemotaxis. Additionally, as students proposed alternative mechanisms to reach the stimulus, they engaged in designing computational models through the rules they proposed. In the following section, we will expand upon some of the ways that students applied CT in the biology lab and how they compared to the engineering lab.

Differences in How Students Applied CT Practices

There were five CT practices that reflected important differences in students' work between the engineering lab and the biology lab: (1) preparing problems for computational solutions; (2) using computational models to understand a concept; (3) assessing different approaches to a problem; (4) understanding the relationships within a system; and (5) investigating a complex system as a whole.

Preparing problems for computational solutions. Weintrop et al. (2016) described the practice of preparing problems for computational solutions as "reframing problems into forms that can be solved, or at least progress can be made, through the use of computational tools" (p. 139). Both labs created opportunities for students to prepare a problem for computational solutions, but this was more obvious in the engineering lab. All pairs of students engaged in this practice during the engineering lab as they translated the objective of the task—make the robot avoid obstacles—into an automated program. A conversation between Natalie and Melody during the first exercise illustrates this. The robots used sensors to determine distance from surrounding objects; the sensors collected values ranging from 0, indicating an immediate obstacle, to 1, indicating no obstacles in range. Natalie and Melody, working with existing commands that they needed to sort, had the following conversation about using these sensors.

> **Natalie:** So I have like, three different possible things, but I only have like, two actions, I feel like.

Melody: Right. So, okay, talking this out, 1 is as far away as it could possibly be.
Natalie: Yeah.
Melody: So if it's greater than 0.75, you could keep going because it's not close enough to want to stop and move.
Natalie: Yeah, okay.
Melody: But if it's less than, it's close enough where you want to maybe try and avoid it.

With the support of the printed commands, Natalie and Melody interpreted the meaning of the sensor data and reframed it through an "if/else" statement. Their comments about how to react to the sensor data according to the numerical value marked an important step toward translating the problem into a solution that could be read by a computer. All of the students applied this practice with some success in the first part of the engineering lab.

The biology lab did not require the use of technology tools because students modeled odor localization with the movement of their own bodies. But the lab, which asked students how they might improve upon the random walk method for reaching the odor source, provoked some pairs of students to begin considering systematic rules or algorithms that they could apply. For example, Camila and Sadie had the following discussion about alternative mechanisms:

Sadie: So clearly what happened was with the tumbling and just taking two steps obviously when we're spinning this arrow, it's way too many chances. So, I think we need to come up with a rule that can, I wanna say somehow, so we don't go in the wrong direction.
Camila: But how are you gonna do that?
Sadie: Exactly my point of what we're supposed to find out.

Although Sadie initially suggested they come up with "a rule", the pair did not immediately know what such a rule would look like. However, after further discussion, the pair's written work indicated that they had begun to articulate options for the rule they proposed in conversation. They noted in writing that an alternative mechanism for reaching the odor source could be "smaller steps when not toward the source and larger steps when it is". Their written work illustrated the formation of conditional logic that would precede solving the problem with a computer.

Similar to Sadie and Camila, Marie made the following suggestion to Nathan about an alternative mechanism to reach the odor source:

Marie: So, I was kind of thinking, if you're not, you just still do the random spins. But if you're not facing the cookies, then only take one step but if you are, then take like five steps, four steps.

Like Sadie and Camila, Marie's idea reflected the use of a conditional statement that took the context of the spinner and cookie and translated it into a repeatable algorithm. While these ideas mark subtle shifts in students' talk, they

also do the work of taking a complex problem—reaching an odor source from any point on a large map—and translating it to a logical set of statements that could be interpreted by a computer.

Using computational models to understand a concept. Using computational models to understand a concept is described as, "recreating phenomena in environments that support systematic investigation" of a concept from the natural world (Weintrop et al., 2016, p. 137). The engineering lab was not directly connected to a real-world context of obstacle avoidance (although the class did have discussions at other times about how robotics can be used to understand animal obstacle avoidance), so this practice did not clearly surface in students' talk. The biology lab was presented as a case of locating a cookie and was not explicitly tied to a real-world context, but some students recognized the lab as a model of the related concepts of chemokinesis and chemotaxis in living organisms. Natalie and Melody, in response to the prompt about how the random movement could be improved upon to reach the stimulus, had the following conversation.

> **Natalie:** Well, I think when it comes to senses…
> **Melody:** I mean, you have to have some sort of stimulus. We had no stimulus.
> **Natalie:** Yeah. So our stimulus, I'm assuming it's gonna be olfaction since that's what all of our lectures were about.

After their initial experience spinning spinners and walking around the room, Natalie and Melody used that experience as a lens to discuss olfaction. They noted that the random movement represented the absence of a stimulus. Although this was not exactly correct, because a stimulus was present, the students acknowledged that their behavior did not respond to the stimulus. They anticipated that to improve upon the tumbles and runs, they would need to sense and subsequently respond to a stimulus. Some groups, who initially approached the biology lab by preparing the task for a computational solution, also reflected on the lab as a model of a biological phenomenon. Sadie and Camila, who first proposed the idea of a rule to direct movement, later connected that idea to the biology context of their activity:

> **Camila:** So, whoever is doing that spinning and looking at their spinner can answer the question. Is my spinner pointing closer to the cookies or not? Right? Animals that can do taxis can do that. Okay, so that's correct, and that's a good example.

Camila extended her idea about the spinner beyond the lab to describe its connection to living organisms applying taxis. Camila and Sadie used the lab as a bridge between a biology concept and a computational model by approaching the biology lab from multiple perspectives. In the case of the biology lab there were several instances in which students used the unplugged model to make sense of chemotaxis. That connection did not surface when students used sensors to make a robot avoid an obstacle, and there are two reasonable explanations for this. First, the engineering lab was not situated in the same class period as a discussion

of sensory biology in the way that the biology lab was. It is likely that biological concepts were more at the forefront of students' minds, because the discussion had surfaced earlier in the class period. Second, and related to the first point, the engineering lab was designed primarily for students to learn how to program the robot, rather than to understand sensory behavior. Because the demands of learning to program were so high, this this became the primary focus during certain parts of the course.

Assessing different approaches to a problem. Assessing different approaches to a problem is a problem solving practice defined as, "mak[ing] an informed decision about which route to follow….based on the requirements and constraints of the problem and the available resources and tools" (Weintrop et al., 2016, p. 139). The primary difference between the two labs was in the scale of the alternatives that students considered. In the engineering task, students made small-scale assessments related to aspects of programming their robots. For example, when Omar and Noelle were trying to write code to use the sensor data, Omar posed a question to an instructor about different logic commands that he might use:

> **Omar:** So if I say, "if *mn* is greater than 0.75"–wait.
> **Instructor:** I think he was using "else".
> **Omar:** Wait, no, not "or", I want to put an "and".
> **Instructor:** Wasn't he using, "else"?
> **Omar:** Well, "else" is not going to work in this. Wait. No, "else" will work in this. Yeah it will.

Omar's was making sense of the different implications of connectors and logical statements and how they would allow him to achieve a goal of responding to the sensor data. Omar wrestled with the decision of writing a program following a pattern of "if…and…" or "if…else…" and how that choice would direct the robot's movement. Such considerations are critical to learning to program and are a natural extension of preparing problems for a computational solution. In this case, Omar was focused on one narrow aspect of the overarching problem, which was to determine how to create a statement that would let the robot respond to sensor data. This example is reflective of the types of assessments many pairs made throughout the engineering lab, which were primarily related to the syntax of the programming language.

The biology lab provoked assessments of problem solutions at a broader scale. One of the reflection questions prompted students to compare the different mechanisms for reaching the odor source. The following example from Kara and Avni illustrates one case of how students approached this comparison.

> **Kara:** So, the different tests, they do differ in like their effectiveness, because like I said the chemotaxis is the most successful and then the least successful is just randomly walking, with the spinners determining how many ways you would go depending on where the spinner landed. So then, did you have anything to add to that?

Avni: So, the ones that seem to be most obvious—like, that would work—do not work. I didn't think the laser ones would work.
Kara: Yeah. And then number two, does the spatial segregation of the sensors impact odor localization?
Avni: Definitely.
Kara: Yeah, definitely. So then we saw that…it would be the longer lasers would be able to get to the food source more quickly because they cover more ground. And then the smaller ones would take twice that time to get to the food source.

Kara and Avni made several assessments of the different solutions to the problem of locating a stimulus. They first provided a general rank of the effectiveness of the different methods, and then Avni reflected on how the outcomes of their tests compared to her expectations. Finally, Kara and Avni moved beyond comparing the effectiveness of the solutions and clearly connected the outcomes to the design of the different mechanisms. Not all pairs of students were as thorough as Kara and Avni when they compared solutions, but the design of the lab and the reflection questions posed to students provoked some comparison among all students.

Understanding the relationships within a system. A system can be defined "as a single entity composed of many interrelated elements" (Weintrop et al., 2016, p. 141), and understanding the relationships within a system requires attending to how those individual elements interact. In both labs, students used aspects of systems thinking, including understanding the relationships within a system, although this practice looked different across the two contexts. In the engineering lab, the relationships that students attended to were those between the variables that they defined. Consider one example from Avni and Kara, when they were trying to modify their program to clarify the relationship between the obstacle sensor, the linear velocity of the robot, and its angular velocity. In the following conversation, *m* referred to the variable to detect the distance from an object; *mn_left* and, respectively, *mn_right* represented the sensors on either side of the robot.

Avni: So, okay, so if *m* is greater than 0.75, so if it's far away from something, then it does this. Then it increases speed. But if this is not true, it goes to this—So now if *mn_left* is less than *mn_right*, so it has sensors on its left and on its right.
Kara: Oh, okay.
Avni: So if these ones are closer to 0, and this one is closer to 1, that means you're gonna set the velocity to 25. So you're going toward the left. Wait, would it be the opposite way?
Kara: I don't know.
Avni: Oh, linear, angular. Linear, angular. So, I don't know. I'm lost. [To instructor] So we have a question about the angular velocity. So, wouldn't it be—25 if *mn_left* is less than *mn_right*?
Instructor: It should be negative, yes.

Avni: Okay. And then, if *mn_right*…[does not finish statement].
Instructor: So, are you, are you trying to make it turn? What are you trying to make your robot do?
Avni: Like, doing the same thing. I guess we're trying to make modifications.
Instructor: What kind of modifications?
Avni: I'm not sure.

To produce a program that would allow a robot to move around a space and avoid obstacles, Kara and Avni needed to account for several components of the system and how they impacted each other. First, Avni noted the value of the distance variable *m*, and how if that value were sufficiently large, the robot could move at a particular speed. However, if *m* was below a certain value, they determined they needed to check whether the obstacle was on the left or right side; depending on which sensor detected the obstacle, they needed to turn the robot in a particular direction. No single component of the program could be determined in isolation from the other components.

In the biology lab, understanding the relationships within a system surfaced when students reflected upon their actions as part of the postlab discussion question. One of the final prompts asked students to consider whether the spatial segregation of sensors would impact odor localization. Recall that sensors were mimicked in this lab by students holding onto poles—some longer and some shorter—with lasers attached to the ends of the poles. The floor was arranged in a grid with different squares marked to represent regions of a more or less intense stimulus. Melody and Natalie had the following conversation.

Melody: Okay, does the spatial segregation of the sensors impact odor localization? So, this was the laser one, right?
Natalie: Yeah. I'm gonna say, no it didn't.
Melody: Yeah it did. Whether you had—well it did with the speed but not with the efficiency. Wait that's the same thing. Cuz like the one—typically the people with the longer ones had more turns available. Cuz I, cuz you had less ability to turn because sometimes yours could be in the same box.
Natalie: Yeah but if I, I think like, because the sensors were the same, like if you were in the short one, you could just keep on going. And those were the ones that got there first.
Melody: No, but the sensors, as in like, the lasers.
Natalie: Yeah.
Melody: I think that it did affect it.
Natalie: I'm gonna say that it doesn't.
Melody: Okay.
Natalie: Cuz it depends on how you were facing. You know what I mean? Like if you had a longer one, you were more likely to get there, logically.
Melody: That's what I was saying—
Natalie: But if you had a small one and you were in the same row, then you just hit the jackpot, cuz you just had to keep walking.

Melody: Yeah.

In Natalie and Melody's conversation excerpted above, the students disagreed about whether the spatial segregation of the sensors would impact odor localization, but they agreed upon some key relationships within the system. Melody articulated how the length of the sensor impacted one's ability to move toward the sitmulus, which in turn made it more likely that a student would reach the odor source. Natalie acknowledged this relationship and also noticed that certain starting conditions within the grid made it so that a student with a short pole could quickly reach the source. Their conversation illuminated the complicated relationship between the starting condition, the spatial segregation of sensors, the ability to move, and the speed with which someone could reach the odor source.

Investigating a complex system as a whole. Investigating a system as a whole involves, "the ability to define and measure inputs and outputs" and "being able to black box the details of the underlying systematic interactions" (Weintrop et al., 2016; p. 141). We did not see clear evidence in the engineering lab of students investigating the system of robotics movement as a whole. Likely this was mostly due to the fact that the nature of the programming task required students to think very carefully about the specific interactions between elements of their programs. In the biology lesson, there were some instances when students applied the practice of investigating a system as a whole. The most obvious instance for this to occur was toward the beginning of the lab, after students practiced the 'random walk' of tumbles and runs, and when they were asked to anticipate how this random walk might be improved. In conversation with Avni, Kara noted, "I guess like, increasing the number of times we tumble." Kara's idea of how to improve the random walk did not depend on the relationships within the system, but rather on the overarching behavior of the system. She recognized that if the system continued in its current state (changing the time input), everyone would eventually reach the source (a corresponding change to the output). Her comment reflected an understanding that several groups within the class expressed about the behavior of random systems.

Discussion

We observed several differences in how students practiced CT in the applied contexts of biology and engineering. The engineering lab was by design a task for students to translate a problem of obstacle avoidance into language that a robot could use. The majority of CT practices that students used were related to computational problem solving. Because of the carefully scaffolded nature of the lab students progressed through a fairly sequenced set of CT practices—first translating the problem into computational terms, then assessing different approaches, and then programming. This sequence of practices is a meaningful way to engage CT, particularly when learning to program a computer, but it is not the only way.

The biology lab elicited more varied ways of engaging in CT, including problem solving practices as well as modeling and simulation practices. The "unplugged" nature of the biology lab allowed students to embody the phenomenon at hand and apply CT practices in a nonlinear fashion to make sense of it from different perspectives.

Assessing different approaches to problem solving was something that surfaced in both labs but in different ways. During the engineering lab, assessing different approaches occured on a smaller scale as students made decisions about the appropriate syntax to achieve a given outcome. This was an instance in which it was clear that the experiences of learners were an important factor in how they applied practices of CT (Berland & Wilensky, 2015). Students in this course were mostly novice programmers, which is fairly typical in the United States, where fewer than 50% of public high schools teach foundational computer science courses (code.org Advocay Coalition et al., 2020). Programming is only one of many CT practices. However, programming in any language requires learning a specific set of rules and conventions. There are specific syntactic challenges associated with learning to program that demand students' attention but do not necessarily support understanding beyond the use of a given syntax (Repenning & Ioannidu, 2006). Learning the practice of programming, while it is a core part of CT, may overwhelm students' opportunities to develop other CT practices.

The use of visual, block-based programming environments have helped to overcome some syntactic challenges (e.g., diSessa, 2000; Weintrop & Wilensky, 2017) and foster more embodied approaches to developing CT (Wilensky & Reisman, 2006). Additionally, efforts to reposition the role of coding, by framing it through mathematical or scientific practices that students already know, have potential to make it a more accessible practice for teachers and students (Dickes et al., 2020). Programming may also be better integrated with other CT practices, with less energy driven toward troubleshooting and debugging, if students plan their work before writing any code (Chevalier et al., 2020). These studies make clear that it is possible to engage students in programming in ways that foster other aspects of CT. But they also point to the central challenge of teaching CT through programming environments, which suggests the opportunity of creating meaningful experiences for students to learn CT separate from programming.

In contrast to the engineering lab, where students focused most of their attention on the details of their programming, the biology lab allowed students to assess different approaches at different levels of the system. This was related to students' consideration of how different rules for tumbles and runs would lead to disparate eventual outcomes. "Unplugged" computational activities are somewhat rare for older learners, but they have clear potential to support the learning of scientific concepts and the development of CT practices (Peel et al., 2019). Students engaged in more varied forms of CT during the biology lab than they did during the engineering lab, when they investigated multiple related systems, and they compared the efficiency of those systems according to the macro-level outcomes and the micro-level

processes. Moreover, because the lab was framed within the context of the natural world, students could use natural phenomena to make sense of computational solutions, and vice versa.

While it is helpful to characterize CT in a way that is not beholden to a specific discipline, there is nuance to be lost if the construct is entirely abstracted from the disciplines in which it is applied. Beheshti et al. (2017) found that researchers working in various STEM disciplines tended to draw on different categories of CT practices depending on the type of research they conducted, so it is clear that the findings of this study are not an anomaly. Applications of CT in biology contexts, for example, may be more likely to draw on modeling and simulation practices, since computation is often used within the discipline to better understand natural phenomena. These activities will look different from a mathematician modeling an abstract mathematical object (Lockwood et al., 2019) or an engineer using a known biological principle to solve a robotics problem. To develop well-rounded CT practices, students likely need more metacognitive instruction to understand what types of activities they are engaging in and how those activities align with disciplinary norms. Additionally, there is work to be done to document the relationships between aspects of CT and more discipline specific models of thinking such as scientific inquiry, mathematical problem solving, or engineering design.

Conclusion

In this study, we looked at two cases of students applying CT in an undergraduate biology-engineering course. We recorded students' pair-work during two labs—one engineering lab where students programmed robots to autonomously avoid obstacles through the use of sensors, and one biology lab where students modeled chemokinesis and chemotaxis. Overall, students' applied CT practices during the engineering lab in ways that were closely aligned with the process and technical details of learning to program. During the biology lab, which did not include any programming, students engaged in practices of CT that were more varied and that investigated the multiple levels of the biological system. This study illustrates how different disciplinary demands—learning to program compared to modeling a biological phenomenon, in this case—elicit different ways of engaging in CT.

For CT to be a useful construct to connect STEM subjects, there is a need for further empirical evidence of students engaging in CT across subject areas (Li et al., 2020b). Although there are certainly ways in which CT has long been used in the professional disciplines of science and math (Brodland, 2015; Chevalier et al., 2020; Garner et al., 2016; Lockwood et al., 2019), this does not always translate to the school subjects of science and math. CT has the potential to serve as a framework to articulate students' learning beyond the content of a single discipline to describe practices integral to modeling and problem solving. Taking a view of transdisciplinary learning as that which combines the knowledge and skills of distinct disciplines, embracing the disciplinary differences in how CT is applied can broaden the range of skills and activities that students bring to real-world problems.

Appendix

Table 3 Coding criteria for CT practices documented in our data

CT Practice	Definition, from Weintrop et al. (2016)
Data practices	
Collecting data	"Propose systematic data collection protocols and articulate how those protocols can be automated" (p. 136)
Creating data	"Define computational procedures and run siulations that create data" (p. 136)
Modeling and simulation practices	
Using computational models to understand a concept	"Advance their own understanding of a concept by interacting with a computational model" (p. 137)
Using computational models to find and test a solution	"Find, test, and justify the use of a particular solution through the use of a computational model" (p. 137)
Designing computational models	"Defining the components of the model, describing how they interact, deciding what data will be produced by the model, articulating assumptions being made by the proposed model, and understanding what conclusions can be drawn" (p. 138)
Constructing computational models	"Implement new model behaviors, either through extending an existing model or by creating a new model" (p. 138)
Computational problem solving practices	
Preparing problems for computational solutions	"Employ…strategies toward reframing problems into forms that can be solved, or at least progress can be made, through the use of computational tools" (p. 139)
Programming	"Understand, modify, and create computer programs" (p. 139)
Assessing different approaches to a problem	"Assess different approaches/solutions to a problem based on the requirements and constraints of the problem and the available resources" (p. 139)
Creating computational abstractions	"Conceptualize and then represent an idea or a process in more general terms by foregrounding the important aspects of the idea while backgrounding less important features" (p. 139)
Troubleshooting and debugging	"Identify, isolate, reproduce, and ultimately correct unexpected problems" (p. 140)
Systems thinking practices	
Investigating a complex system as a whole	"Pose questions about, design and carry out investigations on, and ultimately interpret and make sense of, the data gathered about a system as a single entity" (p. 141)

CT Practice	Definition, from Weintrop et al. (2016)
Understanding the relationships within a system	"Identify the constituent elements of a system, articulate their behaviors, and explain how interactions between elements produce the characteristic behaviors of the system" (p. 141)
Thinking in levels	"Identify different levels of a given system, articulate the behavior of each level with respect to the system as a whole, and be able to move back and forth between levels" (p. 141)

Acknowledgements This project is funded by the National Science Foundation, grant #1759150. Any opinions, findings, and conclusions or recommendations expressed in these materials are those of the author and do not necessarily reflect the views of the National Science Foundation. Partial funding from the UC Forward program at the University of Cincinnati.

Declarations

Competing Interests The authors declare no competing interests.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems, 75*, 661–670. https://doi.org/10.1016/j.robot.2015.10.008

Barr, B., & 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. https://doi.org/10.1145/1929887.1929905

Beheshti E, Weintrop, D., Swanson, H., Orton, K., Horn, M. S., Jona, K, & Wilensky, U. (2017). Computational thinking in practice: How STEM professionals use CT in their work. Retrieved July 15, 2021 from https://ccl.northwestern.edu/2017/Beheshtietal.pdf

Berland, M., & Wilensky, U. (2015). Comparing virtual and physical robotics environments for supporting complex systems and computational thinking. *Journal of Science Education Technology, 24*, 628–647. https://doi.org/10.1007/s10956-015-9552-x

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education, 72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper presented at the annual meeting of the American Educational Research Association, Vancouver, BC.

Brodland, G. W. (2015). How computational models can help unlock biological systems. *Seminars in Cell & Developmental Biology, 47–48*, 62–73. https://doi.org/10.1016/j.semcdb.2015.07.001

Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers & Education, 95*, 202–215. https://doi.org/10.1016/j.compedu.2016.01.010

Chevalier, M., Giang, C., Piatti, A., & Mondada, F. (2020). Fostering computational thinking through educational robotics: A model for creative computational problem solving. *International Journal of STEM Education, 7*, Article 39. https://doi.org/10.1186/s40594-020-00238-z.

Code.org Advocacy Coalition, Computer Science Teachers Association, & Expanding Computing Education Pathways Alliance. (2020). *2020 State of computer science education: Illuminating disparities.* Retrieved July 15, 2021 from https://advocacy.code.org/stateofcs

Czerkawski, B. C., & Lyman, E. W. (2015). Exploring issues about computational thinking in higher education. *TechTrends, 59*(2), 57–65. https://doi.org/10.1007/s11528-015-0840-3

Dickes, A. C., Farris, A. V., & Sengupta, P. (2020). Sociomathematical norms for integrating coding and modeling with elementary science: A dialogical approach. *Journal of Science Education and Technology, 29*, 35–52. https://doi.org/10.1007/s10956-019-09795-7

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. MIT Press.

Farris, A. V., Dickes, A. C., & Sengupta, P. (2020). Grounding computational abstractions in scienctific experience. *Learning and becoming in practice: The International Conference of the Learning Sciences (ICLS) 2020* (pp. 1333–1340). International Society of the Learning Sciences.

Farris, A. V., & Sengupta, P. (2014). Perspectival computational thinking for learning physics: A case study of collaborative agent-based modeling. In J. L. Polman, E. A. Kyza, D. K. O'Neill, I. Tabak, W. R. Penuel, A. S. Jurow, K. O'Connor, T. Lee, & L. D'Amico (Eds.), *Learning and becoming in practice: The International Conference of the Learning Sciences (ICLS) 2014* (Vol. 2, pp. 1102–1106). International Society of the Learning Sciences.

Fraenkel, G. S., & Gunn, D. L. (1961). *The orientation of animals: Kineses, taxes and compass reactions*. Dover.

Garner, G., Reed, P., & Keller, K. (2016). Climate risk management requires explicit representation of societal trade-offs. *Climatic Change, 134*, 713–723. https://doi.org/10.1007/s10584-016-1607-3

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Gutiérrez, K. D., & Rogoff, B. (2003). Cultural ways of learning: Individual traits or repertoires of practice. *Educational Researcher, 32*(5), 19–25. https://doi.org/10.3102/0013189X032005019

Hasni, T. F., & Lodhi, F. (2011). Teaching problem solving effectively. *ACM Inroads, 2*(3), 58–62. https://doi.org/10.1145/2003616.2003636

Horn, M. S., Brady, C., Hjorth, A., Wagh, A., & Wilensky, U. (2014). Frog pond: A codefirst learning environment on evolution and natural selection. In *IDC '14: Proceedings of the 2014 conference on interaction design and children* (pp. 357–360). Aarhus, Denmark: ACM.

Ilic, U., Haseki, H. I., & Tugtekin, U. (2018). Publication trends over 10 years of computational thinking research. *Contemporary Educational Technology, 9*(2), 131–153. https://doi.org/10.30935/cet.414798

Kalelioğlu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Modern Computing, 4*(3), 583–596.

Li, Y., Schoenfeld, A. H., diSessa, A. A., Grasser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020a). Computational thinking is more about thinking than computing. *Journal for STEM Education Research, 3*(1), 1–18. https://doi.org/10.1007/s41979-020-00030-2

Li, Y., Schoenfeld, A. H., diSessa, A. A., Grasser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020b). On computational thinking and STEM education. *Journal for STEM Education Research, 3*, 147–166. https://doi.org/10.1007/s41979-020-00044-w

Lockwood, E., DeJarnette, A. F., & Thomas, M. (2019). Computing as a mathematical disciplinary practice. *Journal of Mathematical Behavior, 54*, Article 100688. https://doi.org/10.1016/j.jmathb.2019.01.004

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

NGSS Lead States. (2013). *Next generation science standards: For states, by states*. The National Academies Press.

Peel, A., Sadler, T. D., & Friedrichsen, P. (2019). Learning natural selection through computational thinking: Unplugged design of algorithmic explanations. *Journal for Research on Science Teaching, 56*, 983–1007. https://doi.org/10.1002/tea.21545

Repenning, A., & Ioannidou, A., et al. (2006). What makes end-user development tick? 13 design guidelines? In H. Lieberman (Ed.), *End user development* (pp. 51–85). Springer.

Repenning, A., Webb, D., & Ioannidou, A. (2010). *Scalable Game Design and the Development of a Checklist for Getting Computational Thinking into Public Schools.* Paper presented at the SIGCSE 2010, The 41st ACM Technical Symposium on Computer Science Education, Milwaukee, WI.

Strauss, A., & Corbin, J. (1998). *Basics of qualitative research: Techniques and procedures for developing grounded theory* (2nd ed.). Sage.

Swanson, H., Sherin, B., & Wilensky, U. (2021). Refining student thinking through computational modeling. Paper to be presented at the 2021 Annual Meeting of the International Society of the Learning

Sciences, Bochum, Germany. Retrieved July 15, 2021 from http://ccl.northwestern.edu/2021/Swanson_Sherin_Wilensky_ISLS_2021b.pdf

Vasquez, J. A. (2015). STEM: Beyond the acronym. *Educational Leadership, 72*(4), 10–15.

Vieira, C., Magana, A. J., Roy, A., & Falk, M. L. (2019). Student explanations in the context of computational science and engineering education. *Cognition and Instruction, 37*(2), 201–231. https://doi.org/10.1080/07370008.2018.1539738

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education, 18*(1), Article 3. https://doi.org/10.1145/3089799

Wilensky, U., & Reisman, K. (2006). Thinking like a world, a sheep, or a firefly: Learning biology through constructing and testing computational theories—An embodied modeling approach. *Cognition and Instruction, 24*(2), 171–209. https://doi.org/10.1207/s1532690xci2402_1

Williams, A. H. (2010). Chemotaxis on the move—Active learning teaching tool. *Journal of Microbiology and Biology Education, 11*(2), 177–178. https://doi.org/10.1128/jmbe.v11i2.216

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wu, B., Hu, Y., Ruis, A. R., & Wang, M. (2019). Analysing computational thinking in collaborative programming: A quantitative ethnography approach. *Journal of Computer Assisted Learning, 35*, 421–434. https://doi.org/10.1111/jcal.12348

Authors and Affiliations

Anna F. DeJarnette[1] · Corey Larrison[1] · Stephanie M. Rollmann[2] · Dieter Vanderelst[3] · John E. Layne[2] · Anna E. Hutchinson[2]

Corey Larrison
larriscc@mail.uc.edu

Stephanie M. Rollmann
stephanie.rollmann@uc.edu

Dieter Vanderelst
vanderdt@ucmail.uc.edu

John E. Layne
john.layne@uc.edu

Anna E. Hutchinson
hutchiae@ucmail.uc.edu

[1] School of Education, University of Cincinnati, PO Box 210022, Cincinnati, OH 45221, USA

[2] Department of Biological Sciences, University of Cincinnati, PO Box 210006, Cincinnati, OH 45221, USA

[3] Department of Psychology, University of Cincinnati, PO Box 210376, Cincinnati, OH 45221, USA