Abstraction Through Multiple Representations in an Integrated Computational Thinking Environment

Aakash Gautam
Virginia Tech
Blacksburg, VA
aakashg@vt.edu

Whitney Bortz
Fresno Pacific University
Fresno, CA
whitney.bortz@fresno.edu

Deborah Tatar
Virginia Tech
Blacksburg, VA
dtatar@cs.vt.edu

ABSTRACT

We present reflections based on qualitative analysis of data from the CHEM+C Project which promotes computational thinking (CT) in classrooms through integration with science classes. The curriculum utilizes multiple representations, requiring students to work with physical phenomena, chemical equations, digital simulations, and modifiable code-based representations. Much CT focus on abstraction naturally emphasizes (1) extraction of a set of features from an object or process, and (2) finding commonality between objects and processes. But Rosen [38] encourages us to think about abstraction as also including the production of new concepts or actions. Integrating CT into science offers the possibility of enhancing this aspect of abstraction. Changing the representational affordances available to the students allows them to take their CT thinking beyond learning-to-abstract towards learning-through-abstraction. This perspective moves computation from an internally focused exercise into the expression of valued ideas in a computational medium.

CCS CONCEPTS

• **Social and professional topics** → **Computational thinking**; *K-12 education*; • **Theory of computation** → *Abstraction.*

KEYWORDS

multi-representation; science context; computational thinking; K-12

ACM Reference Format:
Aakash Gautam, Whitney Bortz, and Deborah Tatar. 2020. Abstraction Through Multiple Representations in an Integrated Computational Thinking Environment. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366892

1 INTRODUCTION

CT is commonly defined as the thought process that involves "formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" [14, pp. 1]. Research posits a range of skills as part of CT including abstraction, decomposition, evaluation, and automation [16, 19, 29, 44]. These definitions and properties represent progress but do not yet reflect a precise canon of generalized knowledge and capabilities. The part of the issue we address here is (by analogy) like the different elements in learning to read a natural language. The purpose of reading is not just learning-to-read, but rather reading-to-learn. Yet these are not entirely distinct. Part of learning-to-read is gaining an understanding of the larger system of communication beyond the mechanics while reading-to-learn is partaking more fully in that system. This is one of the reasons that *being read to* is such a critical part of learning-to-read [9]. Even in early stages, the larger purposes of reading are exposed to the learner in a way that is integrated with the mechanics. By analogy, integrating CT into science affords a similar opportunity to make meaningful entities out of computing representations.

Many approaches utilize *contexts* to motivate elements of programming, including the Media Computation approach [20], games (e.g., [4, 37, 45]), narrative storytelling (e.g., [11, 46]), and a focus on meaningful data (e.g., [3, 23]). This work adds to these, but more particularly creates an opportunity for students to use the medium of computing to encounter and express thoughts about the structure of the world, such as those learned in science. We extend diSessa's approach [15] to teaching physics and computing in an aligned way, and build on earlier sets of insights [24, 35, 42]. We begin with the position that CT provides tools to think with, that is, it is a skill that can facilitate learning of diverse domains. In particular, analysis of the current project shows that details of how students struggle to understand 7^{th} grade chemistry can interact appropriately and usefully with ideas of abstraction relevant to computing.

The work reported here is part of a larger multi-year, multi-school study of integrating CT into pre-existing U.S. middle school science classes. It included three integrated replacement units for 7^{th} and 8^{th} grade science in low to moderate socio-economic-status (SES) schools, each of which combined teacher professional development and specially designed technological environments implemented in NetLogo [47]. The curriculum followed the Texas Essential Knowledge and Skills (TEKS) for science [1] and the Next Generation Science Standards (NGSS) [43] and covered chemistry concepts which have been found to be challenging for students such as the nature of matter and chemical equilibrium [22, 34].

Our curriculum utilized multiple representations of a scientific phenomenon, some of which were implemented in computational form. We reflect on our observations of students' practices as they transitioned between those representations. We observe that these transitions seemed to require thinking in terms of the computational abstraction that implemented the (abstract) scientific construct. That is, CT created meaningful accounts of science phenomenon and the science provided access to how computation embeds ideas.

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366892

In computing, teaching abstraction normally begins with the ideas of (1) extracting important features and ignoring unimportant ones, and (2) finding commonalities across contexts. Abstraction is thus presented as hierarchical. In contrast, from the first, abstraction in science requires students to move laterally across different representations of the concepts or actions — in this case, the chemical phenomena. Abstraction in computing too requires lateral movements such as in choosing representations and — at the core of computing — in using abstract data types, but these are typically presented only as advanced concepts. By using multiple representations in a science context (rather than in a stand-alone computing class), we make an advanced idea of computing more easily attainable. We argue for abstraction-based thought as a critical process *towards* learning, i.e., analogous to the way that being-read-to plays a role in the progression from learning-to-read to reading-to-learn. Thus, encounters with multiple representations indicate the path beyond *learning-to-abstract* towards deeper *learning-through-abstraction*.

2 RELATED WORK

Wing's influential paper [48] brought wide attention to CT; however, the idea of learning computation and using it to learn other concepts pre-dates Wing's paper and includes influential work such as LOGO [35] and varied systems influenced by it (e.g., [6, 13]). In fact, the phrase "computational thinking" can be seen in Papert's 1980 book Mindstorms [35, pp. 182]. diSessa [15], building on his experience of using the Boxer programming environment in Physics learning, argued for "computational literacy" which presents computation as an infrastructure to understand concepts in varied subject domains, including computation itself.

In recent years, there has been a significant movement to differentiate CT from computer science. One such move has been in trying to integrate practices of CT in other domains rather than it being isolated and under the purview of computer science (e.g., [5, 7, 16, 21, 31, 33, 36, 41, 44]). There are pragmatic reasons for integrating CT with another domain such as science. These include lack of time and space in the curriculum, the importance of CT for a variety of life outcomes, and the importance of providing exposure to students who might not take a dedicated class. There are also theoretical, learning-based reasons to integrate CT. Several vital science learning practices such as representational competence [28], problem identification, solution design and verification [17, 26], and use of models in deepening understanding [12] also appear to be important components of CT [19, 44, 49].

Abstraction is central in all these definitions and practices of CT. This can be heard in Wing's claim that "abstractions are the 'mental' tools of computing" [49, pp. 3718]. Weintrop *et al.* [44], based on empirical data, propose a taxonomy defining CT in science and mathematics that includes modeling and abstraction. Similarly, Lee *et al.* [29] propose CT as the use of abstraction, automation, and analysis to solve problems in domains such as robotics and game design. Likewise, in developing CT modules for training pre-service K-12 teachers, Yadav *et al.* [51] presented abstraction as one of the "five basic CT concepts". Similarly, in science learning, abstraction has been found to help in problem-solving [26] and transfer [32].

Despite abstraction being held in common across science and CT, the relationship between abstraction in science and CT remains underspecified and perhaps even avoided. Weintrop *et al.* [44] reserve the word "abstraction" in the particular context of "Creating Computational Abstractions", which appear as a sub-component of the high-level category "Computational Problem Solving Practices". Cuny *et al.* [14] and Weintrop *et al.*'s [44] ideas are mutually reinforcing: the high-level definition of CT is to formulate a problem in a way that solutions can be carried out by an information-processing agent and computational abstractions are defined as precisely those things that can solve the problem. But Weintrop *et al.* [44] sidestep using the word when many scientists would, as a component of "modeling and simulation practices", particularly under "Using computational models to understand a concept."

To unpack this, consider abstraction more generally. Rosen [38] defines abstraction as a "mental process in which new ideas or conceptions are formed by considering several objects or ideas and omitting the features that distinguish them". He highlights three major aspects of abstraction: (1) Extracting a set of features from an object or process while ignoring other features of those objects or processes, (2) Finding commonality between objects or processes encountered through different contexts and transcending those contexts, and (3) Resulting in new concepts or actions.

Most focus on abstraction in CT has been on the first two aspects of abstraction i.e., ignoring the details and finding commonalities [49]. These are what Weintrop *et al.* [44] mean when they locate "creating computational abstractions" as a sub-component of "computational problem solving practices". Abstraction in computing includes "data abstraction", which supports expressing how a computer scientist wants to use data, and "procedural abstraction", which supports hierarchical composition of what the computer scientist wants the computer to "do". Both require extracting meaningful features at any given level of abstraction and leaving the underlying details below that level.

An example is that we devise computers to create a correspondence between a certain concrete, electronic state and the binary number "01000001"; we then form a programming structure that allows us to see "01000001" as the decimal number "065" and then we create a layer of abstraction in which, if "065" is used in a text, it appears as "A". Each hierarchical layer of abstraction is more removed from the concrete machine and more tied to human purposes, yet each is deemed to signify "A" in some form.

An equal, if not more, important skill is to use such understanding to formulate new concepts i.e. abstract laterally. Continuing the "A" and "065" example, the learner could (and does) build laterally on the differences between the two layers to acquire the concept of data types ("A" is a character and "065" is an integer) and use this new concept in computer programs. The reification of a new concept outside the implemented hierarchy corresponds to Rosen's third kind of abstraction. Here, abstraction is *used* to learn a concept. This is an advanced element of CT, but it is a focal essence of science learning. For example, Kozma [27] differentiates between experts and novices by the fluidity in which they connect multiple representations to construct meaning. In chemistry, students strive to understand chemical formulas that can only be understood with lateral movements to references of abstract entities such as atomic structures, stable energy states, and conservation of matter. When we integrate CT into chemistry learning, we should not lose sight of the importance and difficulty of these concepts.

However, it is not clear how curricular activities can be designed in an integrated classroom setting such that they support teachers and students learning and using abstraction. We start with simpler, more pragmatic reflections related to student experience and achievement of drawing upon abstraction-based thoughts to deepen their understanding when they encounter multiple representations afforded by our integrated CT and science curriculum.

3 STUDY CONTEXT

Our intervention included curriculum, technology, and teacher participation and preparation. We presented a multi-representation system that used (1) physical, macroscopic phenomena along with (2) standard scientific representations of microscopic phenomena, (3) macroscopic and microscopic phenomena in relation to one another as shown through animated, modifiable simulations, and (4) modifiable code-based representations. The computing environments had two faces. The first allowed students to experience and investigate a relevant scientific situation through an animated simulation with interactors (sliders, buttons) and instruments (readouts, graphs). The second supported students to explore and change the simulation by reading and revising the code model.

The teacher initially knew little about CT and had participated in two sessions each of week-long professional development training that introduced concepts of CT, NetLogo, and walked her through student experience. Teacher materials included a scope and sequence document for the Computational-Chemistry Task (CCT or replacement unit) that specified the standards addressed, provided a "driving science question", a "driving CT question" and identified related skills [8]. Interaction with the CCT was supported by a number of student materials such as their drawings interpreting the simulation, a structured design critique document, and a code planning document. The data presented here is from the third CCT (CCT3); the first two CCTs had been taught six months earlier.

3.1 CCT3: The Natural Carbon Cycle

The natural carbon cycle was modeled in the context of macro-elements in the environment. The default view shows only the macro-phenomena, but students may choose to "show" the micro-elements which was hugely magnified and overlaid on the view (see Figure 1). They experienced the simulation first with only macro-elements before bringing the micro-elements into view. The code modeled photosynthesis and cellular respiration.

The objective was to help students see how chemical processes that they had learned previously (e.g. photosynthesis) took place in context, as a system. Looking at the carbon cycle as a system that could be modeled was hypothesized as beneficial to CT because it encouraged students to think about systems in a comprehensive way that included feedback and because it invited discussion and argumentation about the nature of models [8, 30, 44]. The teacher was encouraged to engage students in discussions about, for example, what aspects of the model in the simulation were accurate or not accurate and scientifically important or unimportant. These discussions were hypothesized to help students gain access to the kind of abstraction necessary to create models.

Students could also gain additional perspectives by looking at the code that implemented the system. They did this after significant, semi-structured exploration of the simulation. Simulations and code ran in their browsers on Chromebooks in a slightly modified version of NetLogo [47], an agent-based programming environment. Normally, students encounter NetLogo starting with simpler programs and learn the structure of the code gradually; however, our desire to prioritize science at the edge of student learning led us to provide a more complex simulation and code from the beginning. We scaffolded students' focus on parts of code. They were asked to look at the code in CCT3 from two perspectives: as investigators of the model and simulation via reading the code and as changers of the model and simulation via writing new code.

3.2 Participants

Twenty-one 7^{th} grade students in honors science participated in this part of the study during five 50-minute classes. Demographic data were provided on a questionnaire by 20 students. 11 reported as male and 9 as female. Reported ethnicities were 12 white, 4 Hispanic or Latinx, and 4 of mixed race. When asked about prior experience with computing, 11 students reported none, 7 reported some, and 2 reported a lot. 18 students expected to earn an "A"; 2 expected to earn a "B". The students had worked on two CCTs earlier and were familiar with the tool. However, during the first day of CCT3, many struggled to recall operational aspects of the system and such concepts as "turtles" covered previously.

3.3 Data Sources and Analysis

Data collected include pre/post assessments, student artifacts, logs, and video recordings. The recorded classroom videos provided details about how students used the simulation and model, and the ensuing discussion when they encountered different representations. The first two authors transcribed the videos using Transana [50], capturing details of students' actions and interactions with the instructor, classmates, and the researcher (e.g., interventions). We supplemented the video data with the students' written responses.

Following Saldaña [39], the three authors discussed observable actions in the videos and identified critical incidents of learning in the transcripts. These were then discussed in multiple meetings to identify salient practices. They were further triangulated with students' written responses. This paper reports on several critical incidents that exemplify students' applications of abstraction while working with multiple representations.

4 FINDINGS

We noticed several student practices that, when coded and reflected over, pointed towards students formulating abstract scientific thinking. The representations presented through the curriculum among macro and micro-levels. In the particular data reported here, they involved thinking through the meaning of sequential instructions, conditions, and different kinds of models.

4.1 Limitation in Science Representation

In the pre-assessment, we asked students if the chemical equation: "H_2 + O_2 = H_2O" was balanced. 11 out of the 19 who submitted their pre-assessment response correctly said that the equation was not balanced mentioning concepts of conservation of matter. However, 8 of the 11 struggled to balance the equation correctly with

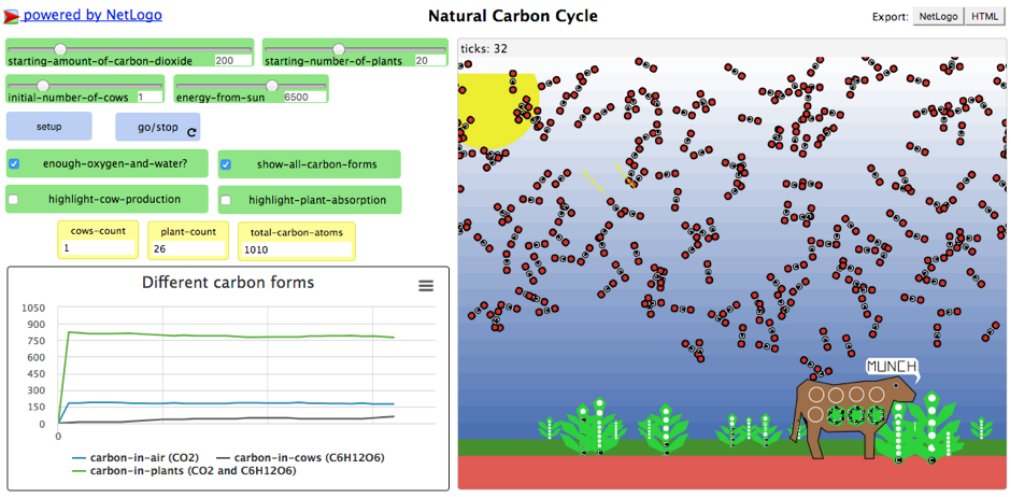

Figure 1: The interface and simulation showing magnified microscopic elements overlaid on the macroscopic phenomenon

responses such as "because it says $H_2 + O_2 = H_2O$ that would = H_2O_2" and "The chemical equation is showing $H_2 + O_2 = H_2O$ when it should be $H_2 + O = H_2O$".

These answers show that there were gaps in students' knowledge and skills. Rote memorization of the idea that H and O cannot exist as single atoms could lead students to seek the right answer from an algebraic, distributive point of view; however, deeper answers lie in the way the chemical formula refers to important abstractions in chemistry such as the idea of a chemical equation. A chemical equation is a symbolic representation of abstract ideas about atoms, molecules and stable energetic states, and summarizes the process without being explicit [25]. This kind of lateral abstraction with implicit assumptions make it hard for students to use such symbolic representation and connect it to their understanding of the principle of conservation of matter.

4.2 Encountering Science by Pulling Meaning from the Code

In contrast, computational representations such as codes have to be more explicit and thus can afford opportunities to students to connect their understanding with the representation. In our project, the students encountered multiple representations and were frequently asked to translate between those representations, drawing meaning from one and applying to another. One such encounter occurred when the instructor conducted a discussion about photosynthesis that culminated with her writing the reactants (H_2O and CO_2) and the products ($C_6H_{12}O_6$, and O_2) on the whiteboard. Reactants were connected to products via an arrow ; however, the formula was not balanced. She then asked students to work in groups to study the code-based model that simulated photosynthesis.

The students read a snippet of the code which was a function that modeled photosynthesis, a micro-level process. The code did multiple things in a lateral rather than a hierarchical relationship to one another, folding together the chemical formula, knowledge about stable energy states, and conservation of matter. It modified several properties of plants that represent the reactants in a photosynthesis reaction (water, carbon dioxide, and energy), and the product (glucose and oxygen). The instructor asked the students to discuss in groups (of 2 to 4), write what they think was happening in the code, and eventually write the balanced chemical equation for photosynthesis. To complete the task, the students had to translate between representations which required understanding the abstractions involved in both the chemical equations and the code.

Reflecting on two group's discussions, we see how each "pulled science meaning out of the code". Group 1 immediately identified that the code models photosynthesis (first remarked by S3) and drew upon their science understanding to decipher the code, noting that it means "photosynthesis":

> S21: So, if plant water level is greater than or equal to six and plant carbon dioxide level is greater than or equal to six and plant energy level is greater than equal to 1 so, umm, what does that mean?
> S3: *Photosynthesis*
> S21: Or things, if all that [conditions mentioned in the code] are right then photosynthesis happens. Like all 3 [conditions] are what they are supposed to be
> S3: Yeah, if you have the right amount of everything then really six oxygen are released I guess
> S21: Then photosynthesis will happen or if this [the condition] is right then oxygen hatches
> S3: Yeah. So, if you have the right amount of carbon dioxide, plant energy, and water, release six oxygen molecules

The phrase "release six oxygen" implied that there is a physical location, as in the macroscopic representation of plants, from which the microscopic entities emanate. This exemplifies the parallel they draw between their scientific knowledge and the code. By drawing parallels between their prior science knowledge and the code snippet, they were able to see the *code as embodying the science.*

In contrast, Group 2 did not initially see the connection between the code and the science concept. They struggled and required the instructor's help in disambiguation:

> S15: To photosynthesize plant water level greater than or equal to six and plant carbon dioxide level is greater

Code statement	Group 2's explanation
if plant-water-level >= 6 and plant-carbon-dioxide-level >= 6 and plant-energy-level >= 1	If this is true this happens [curly braces indicating the lower part of the code]
hatch-oxygen 6	creat [create] oxygen 6
set size 0	not visible; not seable [seeable in] real life
set heading 0 – random 10	molecule float to top
set ycor ycor + 1 + random size	where it happens has to do with height; lets the oxygen move around on its own
set plant-glucose-level plant-glucose-level + 1	adds stored glucose
set plant-water-level plant-water-level – 6	takes water from plant
set plant-carbon-dioxide-level plant-carbon-dioxide-level – 6	takes carbon dioxide from plant
set plant-energy-level plant-energy-level – 1	takes energy from plant

Table 1: Group 2's explanation of the photosynthesis code

than ... So, that's telling you that if the plant water level or plant carbon dioxide level is greater than or equal to six and plant energy level is greater than or equal to one that would happen [pointing to the lower part of the code]
S14: Write that down
S15: Hatch oxygen six. Umm, so, what should we do. If this [the conditions on the top] is true, then this happens. Do you guys agree?
S2: What does it say?
S14: Ok, hatch oxygen six [reads rest of the code]
S14: This does not make sense, this one is not like this one at all.
S15: Because they are different things
I: Listen y'all, if it says hatch oxygen, what do you think it means to hatch?
S2: There are oxygens
I: Yeah, it's just letting it out, it's just creating it. So then you would say, create six oxygen.

Group 2's effort was placed on understanding the code as it is – symbols that "did not make sense". They struggled to understand statements like "hatch oxygen 6", which neither groups had previously encountered. Through inquiry and the instructor's scaffold, they understood the symbols and sequentially worked on the code, tying parts of the code to the science concept that the instructor had suggested. Their attempt reflected in what they wrote next to each line of code in their later response (Table 1). Here, students dealt with the code statement-by-statement, drawing upon parts of their science knowledge to understand and explain bits of the encountered code. Unlike Group 1, they did not see the code as a single unit but were able to grasp parts of the reaction through the code. The code-based model helped the students understand the implicit aspects of the chemical reaction. Their discussion also pushed them to explore how variables were used to define properties of the plants and the overall processes.

As the students progressed through the activity, they built upon their understanding. They delved deeper into the meaning of the last four lines of code. In particular, when a group encountered the code "set plant-water-level plant-water-level – 6", the students moved beyond saying that "the water level is minus six" to "water level goes down six" or "takes water from plant", and to conclude that it is "because we are talking about, like, the photosynthesis process and then it [the code snippet] says if the plant water level is greater than or equal to six". These students moved from understanding local meaning based on the context where they encountered the representation to abstracting it to a higher-level concept where they forgo the specificity, merge it with the concept(s) obtained from other representations (chemical equation and prior science knowledge in this case), and reinforce their understanding of the phenomenon. Several groups took longer to get to this idea, but were eventually brought closer to understanding through the teacher's efforts in the whole class discussion:

I: ... set plant water level plant water level minus six. S2, what do you think that means?
S16: *You subtract six*
I: Because in reaction, how many did we use?
S16 (and some others): Six
I: Six. So, just like in normal chemical reaction these six waters are gone because they have now created a new substance. So, the plant loses those water molecules. The chemical reaction has taken place and it doesn't have those water molecules anymore because of the fact that we have our products, our reactants are gone, ok? This is important because, one, in the real science world that is true because you wouldn't have those six H_2Os, you wouldn't have those six CO_2 anymore because we have new substances, ok? And this is important in the program because it stops the plants from just continuously going through photosynthesis. It is saying that plants now need to gain six more H_2O molecules and six more CO_2 molecules and one radiant energy in order for photosynthesis to take place.

S16's response was limited to their understanding of the code and to the context (the code snippet) in which they encountered it. They may not have yet grasped that the "subtract six" implied the reactants being used in the chemical reaction. By discussing the commonalities between "normal chemical reaction" and the code, the instructor pushed students towards higher-level abstract thinking. By further discussing how part of the code snippet conveyed relevant information in seemingly two different representations - "the real science world" and "the program" - she pushed the students up the ladder of abstraction [40] where scientists, including chemists and computer scientists, generally revel [28].

4.3 Encountering CT by Pulling Meaning from the Science

Students encountered and interpreted sequentiality and branching without apparent difficulty. For example, both the above groups successfully interpreted conditionals in the form of "if-then" statements and the logic inherent in the "and" statements by drawing upon their understanding of the science, as we heard in S15's comment.

Students were also beginning to learn about procedural abstraction, at least to the extent of understanding blocks of code within the procedure. This occurred once in relation to the sequence of

"and" and again with respect to setting the values for the newly "hatched" oxygen objects. While discussing with Group 2, the instructor asked about the block of code within "hatch oxygen". Later, she went on to ask the entire class why the plant water level goes down, hinting at data abstraction by building on the students' existing knowledge that photosynthesis happens in plants. In the discussion that ensued, the code afforded opportunities to the instructor to highlight the differences between computational and science models. In a computer model, unlike a science one, we must subtract water explicitly so that it "stops the plants from just continuously going through photosynthesis".

5 DISCUSSION AND CONCLUSION

We presented a series of activities aimed to help students learn CT and science through integration. Students dealt with multiple representations through abstraction-based thought that allowed them to pull meaningful information from those representational forms to deepen understanding.

When Group 1 pulled meaning out of the photosynthesize code, they implicitly "agreed" to the assumptions in the code because they saw the parallels and could consolidate their understanding of the science and the code statements. Thus, "plant-water-level" and "plant-carbon-dioxide-level" became meaningful entities for them. Since the students could "see" those lines of code as photosynthesis i.e. Rosen's second aspect of abstraction, they were able to reify their understanding to conceive one entity (code statements) as processes and re-conceive those as objects at a higher level.

When the other group encountered the statement "set plant-water-level plant-water-level − 6", it did not run parallel to their understanding of the science. Chemical equations, which they were familiar with, implicitly mention such transformation whereas the code had to mention it explicitly. This difference caused the students to struggle to consolidate their understanding which led them to discuss, explore, and eventually see the similarity between the code statement and the chemical equation. We saw students move from a lower-level idea of the code representing "subtract six" to a higher-level concept of it representing the fact that to form a glucose molecule we need six water molecules. The differences in what is explicit and implicit, and how they are expressed in different representations meant that the students had to work to reconcile them and then to see them as mutually constitutive i.e. build towards the third aspect of abstraction that Rosen [38] posits.

Our study involved 21 students from an honors science class which is not representative of the population. We focused on critical incidents from the classroom and do not present summative performance analysis. Despite these limitations, our findings strengthen our belief in the value of integrating CT in the science curriculum, allowing us to present multiple representations in context. Reflecting on the intervention, the following aspects of our intervention helped encourage abstraction-based thoughts in the integrated CT and science environment:

- **Present multiple representations in context:** The varied representational choices encouraged students to think about what is being represented and how they can be represented in multiple forms. The computational medium afforded variation. This provided opportunities for students to see that phenomena can be expressed in multiple ways and then use abstraction-based thoughts while transitioning between those representations.
- **Follow pedagogical goals while making representational choices:** The photosynthesize code emphasized precisely the pedagogical point that the instructor wanted to make rather than being a more scientifically extensible model. With the instructor's scaffolds, students were able to extract key features, generalize between the multiple representations, and build new concepts.
- **Order the encounter with the representations:** Encountering the simulation before the code helped when they encountered the code model. The lack of experience with the macro representation has been identified as a problem in literature [18]. Their earlier encounter with the macroscopic representation helped in understanding newer code representation. We argue, along with earlier research [2], that the order of representations is critical in facilitating students to abstract and deepen their understanding.
- **Ensure cross-representational coherence:** The students could see photosynthesis being simulated, the whiteboard had a chemical equation representing photosynthesis (albeit unbalanced), and the code formulated the process of photosynthesis explicitly. The coherency facilitated in movement across these representations, encouraging the students to see the similarities and differences and pull meaning from one to understand the other.
- **Allow friction during encounters with representations:** In Group 2's discussion, we see that friction arose due to the difference in what was explicit and implicit in the representations that they encountered. The friction encouraged them to explore different ways of looking at the representations, revisit their understanding of the domain and draw upon it to understand the new representation, and negotiate the elements with one another.

A focus in scientific abstraction is on lateral relationships, as we saw with the case of chemical equations and the assumptions they hold. The notion of abstraction in computing starts with hierarchies of data and procedures. Ultimately, all three aspects of abstraction are important. Places in the curriculum where they align offer important learning and teaching opportunities.

Looking to the future, finding such places will be an on-going challenge. In particular, scientific abstraction may pull for different kinds of pedagogical design than computational ones. From a CT point of view, we want students to be able to see a set of code as being part of a single unit, as Group 1 did. This is a practice of procedural abstraction, involving Rosen's first two elements [38]. It is aligned with CT practices such as modularizing and reusing existing code [10]. But focus on this does not necessarily create encounters with lateral abstractions. Seeing the code as a single unit of the process does not by itself cultivate speculation about such abstractions in the chemical equation as the stability of energy states. While we see this approach as promising and important, more research must help us understand the theoretical and practical prospects and limitations in the depth of abstraction practices in an integrated environment.

ACKNOWLEDGMENTS

We wish to thank Ayaan Kazerouni, Kemper Lipscomb, and the teachers who participated in the project. This work was supported by National Science Foundation Grant #1543022.

REFERENCES

[1] Texas Education Agency. 2018. Texas Essential Knowledge and Skills for Science. Retrieved April 12, 2019 from http://ritter.tea.state.tx.us/rules/tac/chapter112/ch112b.html

[2] Shaaron Ainsworth. 2008. The educational value of multiple-representations when learning complex scientific concepts. In *Visualization: Theory and practice in science education*. Springer, 191–208.

[3] Ruth E Anderson, Michael D Ernst, Robert Ordóñez, Paul Pham, and Ben Tribelhorn. 2015. A data programming CS1 course. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 150–155.

[4] Ashok Basawapatna, Kyu Han Koh, Alexander Repenning, David C Webb, and Krista Sekeres Marshall. 2011. Recognizing computational thinking patterns. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 245–250.

[5] Satabdi Basu, Anton Dukeman, John S Kinnebrew, Gautam Biswas, and Pratim Sengupta. 2014. Investigating student generated computational models of science. Boulder, CO: International Society of the Learning Sciences.

[6] Paulo Blikstein, Dor Abrahamson, and Uri Wilensky. 2005. Netlogo: Where we are, where we're going. In *Proceedings of the annual meeting of Interaction Design and Children*, Vol. 23. Citeseer.

[7] Whitney Wall Bortz, Aakash Gautam, Kemper Lipscomb, and Deborah Tatar. 2019. Integrating Computational Thinking into Middle School Science: A Search for Synergistic Pedagogy. *2019 ASEE Southeastern Section Conference* (2019).

[8] Whitney Wall Bortz, Aakash Gautam, Deborah Tatar, and Kemper Lipscomb. 2019. The Availability of Pedagogical Responses and the Integration of Computational Thinking. *Integrating Digital Technology in Education: School-University-Community Collaboration* (2019), 81.

[9] John D Bransford, Ann L Brown, Rodney R Cocking, et al. 2000. *How people learn*. Vol. 11. Washington, DC: National academy press.

[10] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association, Vancouver, Canada*, Vol. 1. 25.

[11] Quinn Burke and Yasmin B Kafai. 2012. The writers' workshop for youth programmers: digital storytelling with scratch in middle school classrooms.. In *SIGCSE*, Vol. 12. 433–438.

[12] Carol KK Chan and Ivan CK Lam. 2010. Conceptual change and epistemic growth through reflective assessment in computer-supported knowledge building. In *Proceedings of the 9th International Conference of the Learning Sciences-Volume 1*. International Society of the Learning Sciences, 1063–1070.

[13] Vanessa Stevens Colella, Eric Klopfer, and Mitchel Resnick. 2001. *Adventures in Modeling: Exploring Complex, Dynamic Systems with StarLogo*. ERIC.

[14] Jan Cuny, Larry Snyder, and Jeannette M Wing. 2010. Demystifying computational thinking for non-computer scientists. *Unpublished manuscript in progress, referenced in http://www. cs. cmu. edu/˜ CompThink/resources/TheLinkWing. pdf* (2010).

[15] Andrea A DiSessa. 2001. *Changing Minds: Computers, Learning, and Literacy*. MIT Press.

[16] Yihuan Dong, Veronica Catete, Robin Jocius, Nicholas Lytle, Tiffany Barnes, Jennifer Albert, Deepti Joshi, Richard Robinson, and Ashley Andrews. 2019. PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 906–912.

[17] Tamar Fuhrmann, Shima Salehi, and Paulo Blikstein. 2014. A Tale of Two Worlds: Using bifocal modeling to find and resolve "Discrepant Events" between physical experiments and virtual models in Biology. Boulder, CO: International Society of the Learning Sciences.

[18] John K Gilbert and David Treagust. 2009. *Multiple representations in chemical education*. Vol. 4. Springer.

[19] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational researcher* 42, 1 (2013), 38–43.

[20] Mark Guzdial. 2003. A media computation course for non-majors. In *ACM SIGCSE Bulletin*, Vol. 35. ACM, 104–108.

[21] Susanne Hambrusch, Christoph Hoffmann, John T Korb, Mark Haugan, and Antony L Hosking. 2009. A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin* 41, 1 (2009), 183–187.

[22] Cari F Herrmann-Abell and George E DeBoer. 2011. Using distractor-driven standards-based multiple-choice assessments and Rasch modeling to investigate hierarchies of chemistry misconceptions and detect structural problems with individual items. *Chemistry Education Research and Practice* 12, 2 (2011), 184–192.

[23] Dennis Kafura, Austin Cory Bart, and Bushra Chowdhury. 2015. Design and preliminary results from a computational thinking course. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 63–68.

[24] James Kaput, Richard Noss, and Celia Hoyles. 2002. Developing new notations for a learnable mathematics in the computational era. In *Handbook of international research in mathematics education*. Routledge, 63–88.

[25] Bas Kolloffel, Tessa HS Eysink, Ton de Jong, and Pascal Wilhelm. 2009. The effects of representational format on learning combinatorics from an interactive computer simulation. *Instructional Science* 37, 6 (2009), 503–517.

[26] Janet L Kolodner, Paul J Camp, David Crismond, Barbara Fasse, Jackie Gray, Jennifer Holbrook, Sadhana Puntambekar, and Mike Ryan. 2003. Problem-based learning meets case-based reasoning in the middle-school science classroom: Putting learning by design (tm) into practice. *The journal of the learning sciences* 12, 4 (2003), 495–547.

[27] Robert Kozma. 2003. The material features of multiple representations and their cognitive and social affordances for science understanding. *Learning and Instruction* 13, 2 (2003), 205–226.

[28] Robert Kozma, Elaine Chin, Joel Russell, and Nancy Marx. 2000. The roles of representations and tools in the chemistry laboratory and their implications for chemistry learning. *The Journal of the Learning Sciences* 9, 2 (2000), 105–143.

[29] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *Acm Inroads* 2, 1 (2011), 32–37.

[30] Richard Lehrer and Leona Schauble. 2006. *Cultivating model-based reasoning in science education*. Cambridge University Press.

[31] Line Have Musaeus and Peter Musaeus. 2019. Computational Thinking in the Danish High School: Learning Coding, Modeling, and Content Knowledge with NetLogo. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 913–919.

[32] National Research Council. 2000. *How people learn: Brain, mind, experience, and school: Expanded edition*. National Academies Press.

[33] Kai Orton, David Weintrop, Elham Beheshti, Michael Horn, Kemi Jona, and Uri Wilensky. 2016. Bringing computational thinking into high school mathematics and science classrooms. Singapore: International Society of the Learning Sciences.

[34] Haluk Özmen. 2004. Some student misconceptions in chemistry: A literature review of chemical bonding. *Journal of Science Education and Technology* 13, 2 (2004), 147–159.

[35] Seymour Papert. 1980. *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc.

[36] Lori Pollock, Chrystalla Mouza, Kevin R Guidry, and Kathleen Pusecker. 2019. Infusing Computational Thinking Across Disciplines: Reflections & Lessons Learned. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, 435–441.

[37] Alexander Repenning, David Webb, and Andri Ioannidou. 2010. Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on Computer science education*. ACM, 265–269.

[38] Gideon Rosen. 2017. Abstract Objects. *The Stanford Encyclopedia of Philosophy* (2017). https://plato.stanford.edu/archives/spr2017/entries/abstract-objects/

[39] Johnny Saldaña. 2015. *The coding manual for qualitative researchers*. Sage.

[40] A Truman Schwartz. 2006. Contextualized chemistry education: The American experience. *International Journal of Science Education* 28, 9 (2006), 977–998.

[41] Amber Settle, Baker Franke, Ruth Hansen, Frances Spaltro, Cynthia Jurisson, Colin Rennert-May, and Brian Wildeman. 2012. Infusing computational thinking into the middle-and high-school curriculum. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*. ACM, 22–27.

[42] Elliot Soloway. 1986. Learning to program= learning to construct mechanisms and explanations. *Commun. ACM* 29, 9 (1986), 850–858.

[43] NGSS Lead States. 2003. *Next generation science standards: For states, by states*. National Academies Press.

[44] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147.

[45] David Weintrop, Nathan Holbert, Michael S Horn, and Uri Wilensky. 2016. Computational thinking in constructionist video games. *International Journal of Game-Based Learning (IJGBL)* 6, 1 (2016), 1–17.

[46] Linda Werner, Shannon Campe, and Jill Denner. 2012. Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 427–432.

[47] Uri Wilensky et al. 1999. Center for connected learning and computer-based modeling. In *NetLogo*. Northwestern University.

[48] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[49] Jeannette M Wing. 2008. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 366, 1881 (2008), 3717–3725.

[50] D Woods and C Fassnacht. 2018. Transana (Version 3.02). https://www.transana.com

[51] Aman Yadav, Ninger Zhou, Chris Mayfield, Susanne Hambrusch, and John T Korb. 2011. Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education*. ACM, 465–470.