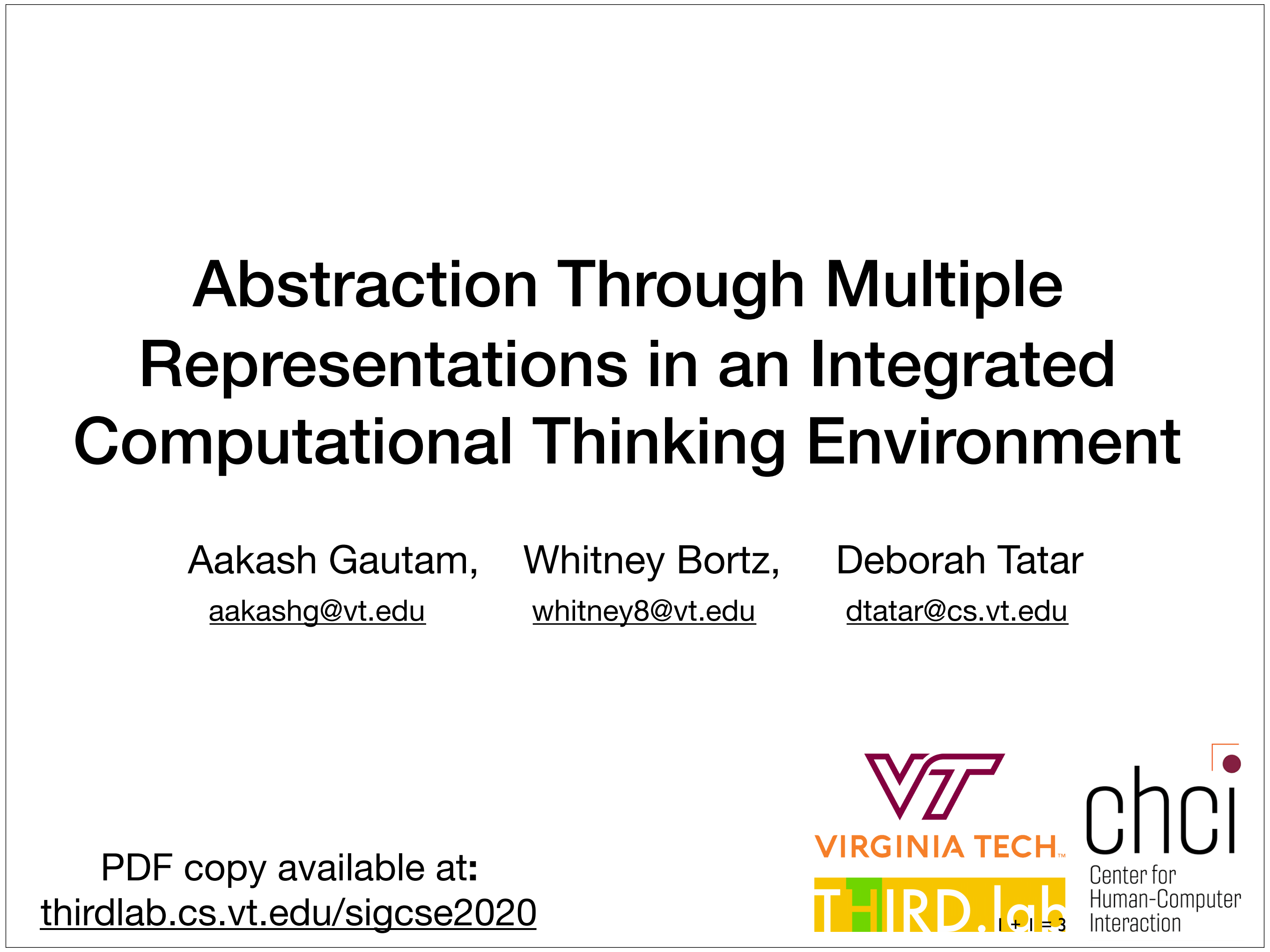


Abstraction Through Multiple Representations in an Integrated Computational Thinking Environment

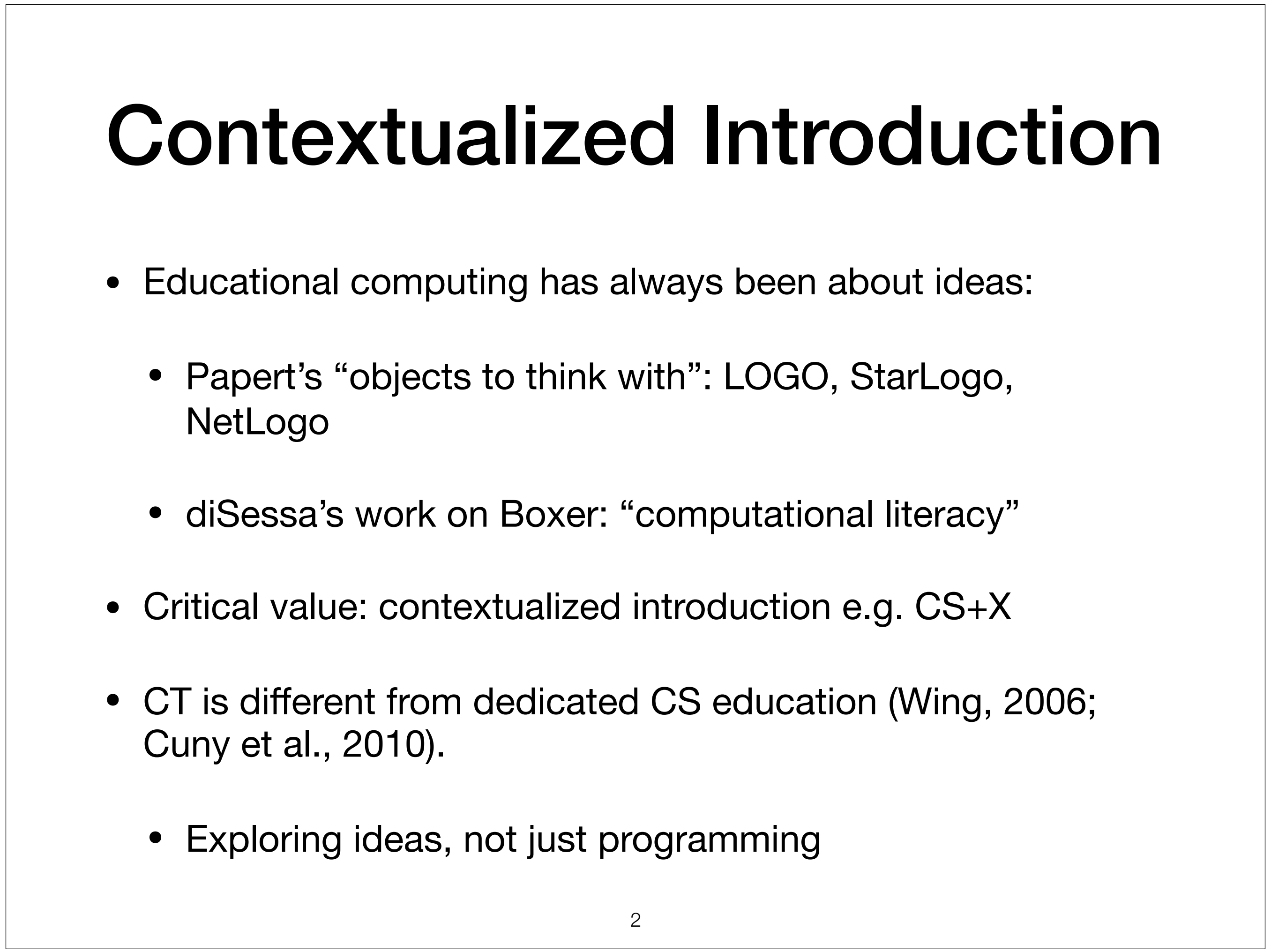
Aakash Gautam, Whitney Bortz, Deborah Tatar
aakashg@vt.edu whitney8@vt.edu dtatar@cs.vt.edu

PDF copy available at**:**
thirdlab.cs.vt.edu/sigcse2020

Hi Everyone!
Thank you all for attending this talk.
We are sorry that we couldn't attend the conference in person.
We have uploaded a PDF copy of the slide in the link you see here. It also includes a transcript of the audio recording.

Today, I want to talk about our work with middle school science students where we explored the possibility of integrating computational thinking or CT in existing science classes.

In particular, I will talk about the value afforded by computing systems to create multiple representations of a science phenomenon that we believe can help students dive deeper into learning abstraction and, more critically, learning through abstraction.



Contextualized Introduction

- Educational computing has always been about ideas:
    - Papert's "objects to think with": LOGO, StarLogo, NetLogo
    - diSessa's work on Boxer: "computational literacy"
- Critical value: contextualized introduction e.g. CS+X
- CT is different from dedicated CS education (Wing, 2006; Cuny et al., 2010).
    - Exploring ideas, not just programming

Educational computing has always been about ideas.

These build on theories from Bruner and Piaget and was brought to the forefront by Seymore Papert, who saw possibility of computers to be "objects to think with".

We see the influence of computing as a tool to explore ideas in a line of systems such as Logo, StarLogo, and NetLogo, which is what we used in this project.

Related to that, Andy diSessa introduced Boxer, a system that facilitated learning important ideas of physics as well as computation.

In all of the work, we find that the critical value is in providing meaningful context for the students to encounter computing.

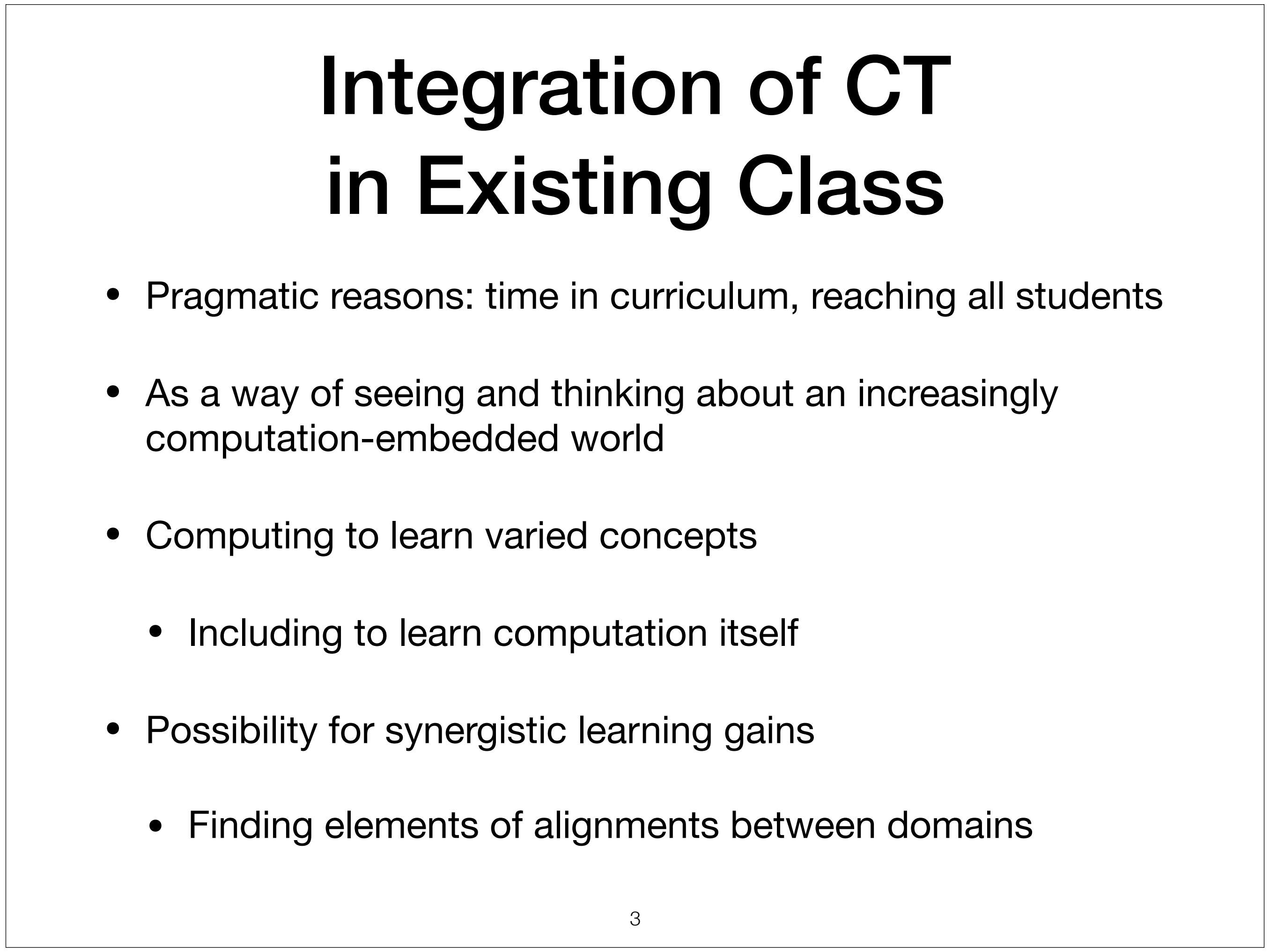
Such contextualized introduction have been argued for in teaching practices such as Guzdial, Ericson's Media Computation, Caitlin Kelleher's Story-telling Alice, and Dennis Kafura and team's Blockpy system at Virginia Tech.

I believe that this value that is very well reflected in this "CS + X" session as well.

Computational thinking, or CT, is well aligned to this idea of contextualized introduction.

CT is about exploring ideas through a computational medium.

Although critical ideas can be captured in programming exercises, CT is much broader than that.



Integration of CT in Existing Class

- Pragmatic reasons: time in curriculum, reaching all students
- As a way of seeing and thinking about an increasingly computation-embedded world
- Computing to learn varied concepts
    - Including to learn computation itself
- Possibility for synergistic learning gains
    - Finding elements of alignments between domains

In our project, we have tried to integrate CT into existing middle school science classes.

There are pragmatic reasons to do so.
There is limited room in an already overloaded K-12 curriculum to add dedicated computing class.
Also, many schools have a shortage of teachers and resources to teach dedicated computing classes.

Given that computing skills are increasingly becoming central in the modern workforce, there is a risk of widening inequity if we cannot provide students with rich learning environments that provides them with opportunities to learn fundamental concepts of computing.
Integrating CT can provide opportunities to students to see and think about an increasingly computation-embedded world.
This goes back to diSessa's argument for computational literacy, that is, computing is an infrastructure to understand concepts of varied subjects including computation itself.

Also, from a learning science perspective, integrating CT into classes can open possibilities for deeper engagement and lead to synergistic learning gains.
The key is to find elements of alignment between CT and the domain in which CT is integrated.

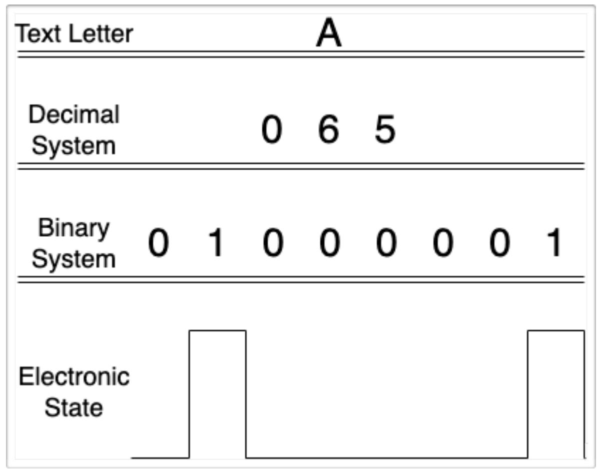
Our Approach: Integrate CT in Science Class

Abstraction is central in both science and CT

In our integrated approach where we brought CT into existing science classes, we find that Abstraction is one such common element in both the domains.
It is the focus of this work.

Abstraction in Computing

- Typically begins with a hierarchical perspective
  - Extract important features
  - Ignore unimportant features
  - Find commonalities across contexts
- Lateral movements between concepts or actions are more advanced
  - E.g. abstract data types: operations and constraints



Let’s unpack abstraction a little bit.

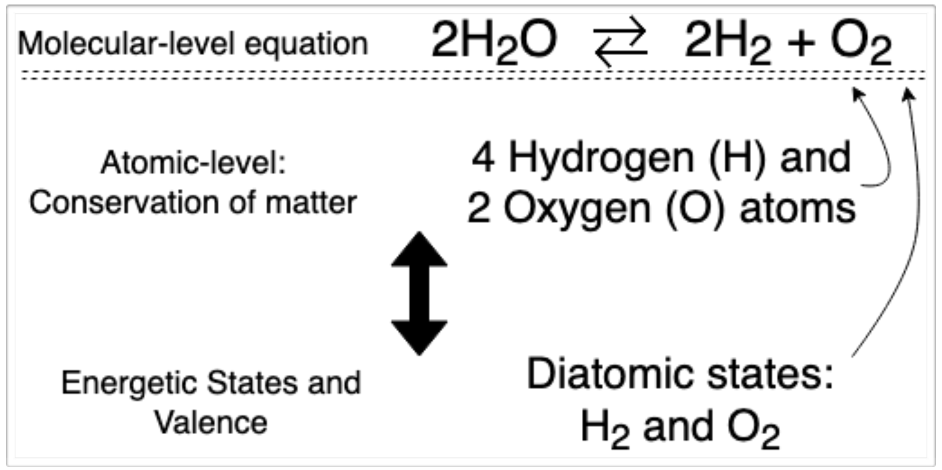
When we introduce abstraction to students in computer science, we typically begin by presenting it as a hierarchy.
The key skills that students encounter is to extract important feature, ignore unimportant ones, and find commonalities across contexts.

A simple example of this is the way we think about computing systems in general.
At the lowest level, we have the electronic states which can be on or off.
Moving up the ladder of abstraction, we represent those states in a binary form — 0s and 1s.
A further level above that, one can choose to represent it in a decimal system. For example, in the image that you see here, the binary number 01000001 can be represented as the decimal “065”.
A level above that, we can choose to take the decimal number as a representation of text. In this case, “065” can translate to the letter “A” if we follow ASCII encoding.
The takeway from here is that very rarely, beginners when working with text letter A have to think about the electronic states or the binary states.

Beginners are not introduced to the lateral hierarchy such as abstract data types, and the operations and constraints that come with it.
These are more advanced concepts in computing.

Abstraction in Science

- From the start, it requires lateral movements



- Why can't it be: $H_2O \leftrightharpoons H_2 + O$?

Whereas, when we look at abstraction practices in science, we find that students need to think about lateral movements right from the start.

Consider a simple chemical equation like the electrolysis of water where the equation is written as 2H20 -> 2H2 + O2 i.e. 2 water molecules split into 2 hydrogen molecules and 1 oxygen molecule.

The equation itself is represented at a molecular level.
We can move down a level and notice atomic elements too. In this case, there are 4 hydrogen atoms and 2 oxygen atoms in the left side, or the reactant side.
The law of conservation of matter dictates that the product side should also have the same number of atoms in the product side, or the right side.

But that concept alone is not enough.
For example, one, equipped with the conservation of matter can ask why can't the equation be H2O —> H2 + O. There are equal number of atoms in both sides here too!

However, the thing with the chemical equation is that it represents several implicit aspects of chemical phenomena.
In this case, it is important for students to understand energetic states and draw upon the fact that hydrogen and oxygen are diatomic, that is, they typically existing with two atoms together.
That is, H2 and O2.
This understanding is essential to be able to grasp the molecular level equation.
Students need to move laterally right from the start to understand the representations typically used in science.

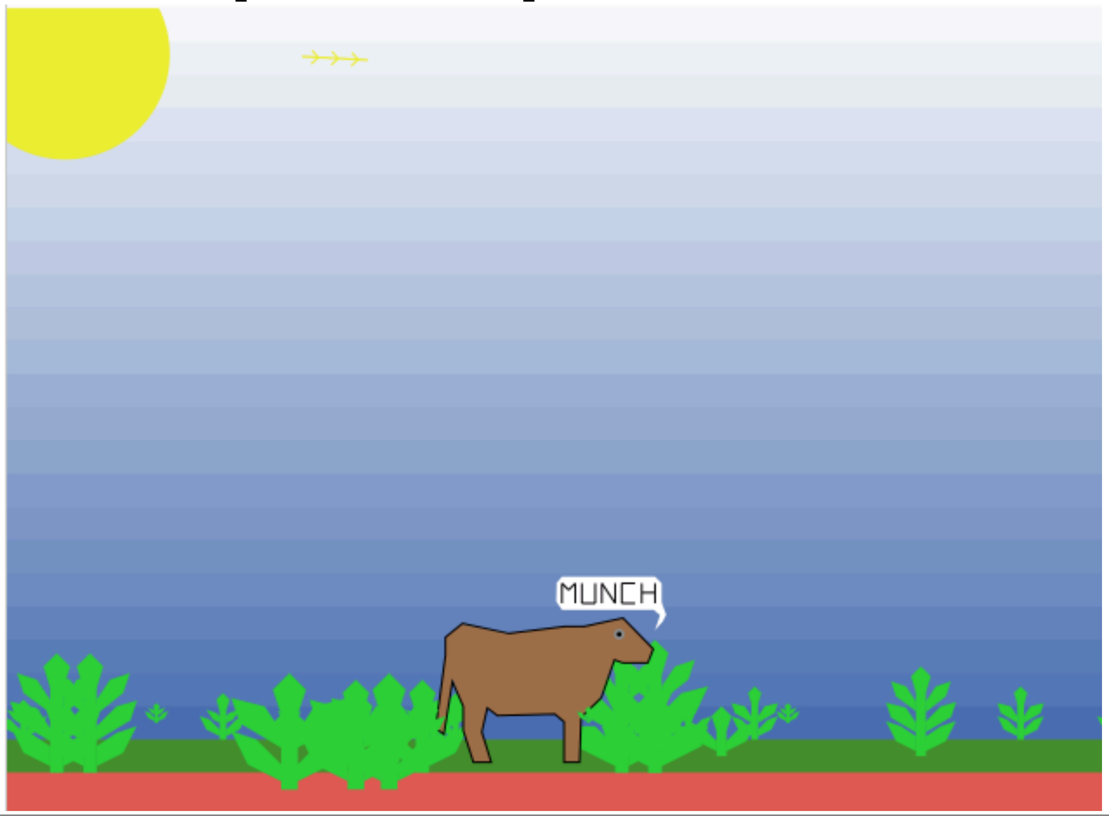
Multiple Representations

- Representing the natural carbon cycle

Middle school students from Texas and Virginia participated in our study.
In the paper, we report observations from one class of 21 7th grade students.

In the unit students encountered multiple representation of the natural carbon cycle.

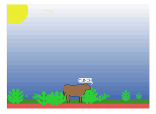
Multiple Representations



They began with a simulation of the view that they were quite familiar with.
The simulation showed cows grazing on grass under a bright sunlight.

Multiple Representations

1. Physical macroscopic phenomena

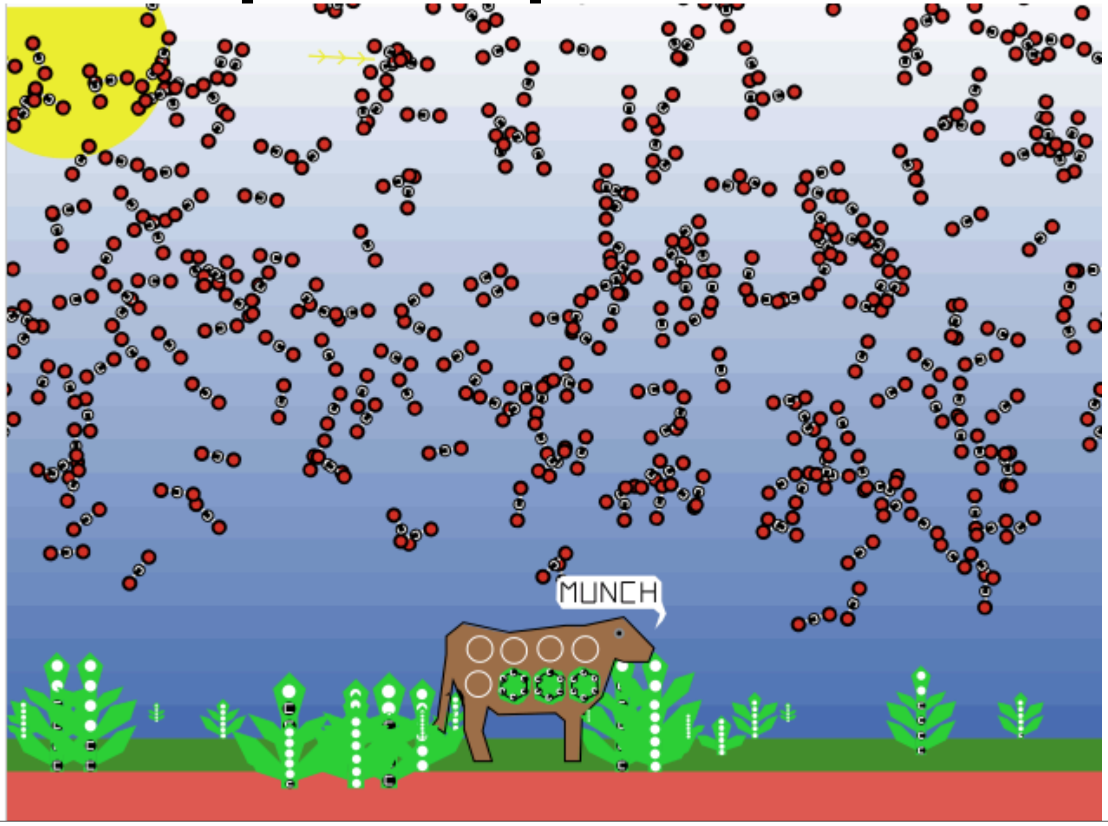


2. Standard scientific representation

$$2H_2O \rightleftarrows 2H_2 + O_2$$

After the simulation, the instructor led a discussion where the class encountered scientific representation of the phenomenon. This included chemical equations.

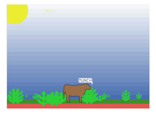
Multiple Representations



Following the discussion, students were encouraged to explore the simulation further.
But this time around, they explored a feature that we had built where microscopic representations of molecules were overlaid on the macroscopic view.
In the figure above you can see carbon dioxide molecules in the atmosphere, glucose molecules in the plants and inside the cow.

Multiple Representations

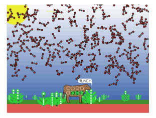
1. Physical macroscopic phenomena



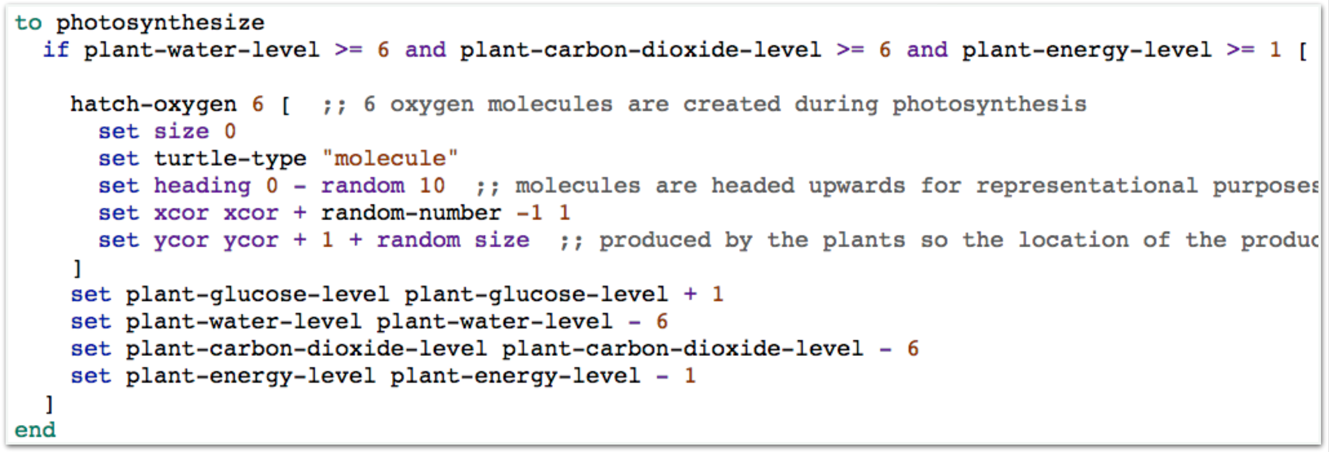
2. Standard scientific representation

$2H_2O \rightleftarrows 2H_2 + O_2$

3. Simulation: included microscopic phenomena overlaid on the macroscopic view



Multiple Representations



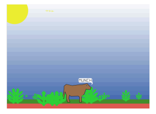
```
to photosynthesize
  if plant-water-level >= 6 and plant-carbon-dioxide-level >= 6 and plant-energy-level >= 1 [

    hatch-oxygen 6 [  ;; 6 oxygen molecules are created during photosynthesis
      set size 0
      set turtle-type "molecule"
      set heading 0 - random 10  ;; molecules are headed upwards for representational purposes
      set xcor xcor + random-number -1 1
      set ycor ycor + 1 + random size  ;; produced by the plants so the location of the produc
    ]
    set plant-glucose-level plant-glucose-level + 1
    set plant-water-level plant-water-level - 6
    set plant-carbon-dioxide-level plant-carbon-dioxide-level - 6
    set plant-energy-level plant-energy-level - 1
  ]
end
```

The students were then introduced to parts of the code that implemented the simulation.
An example of the code can be seen here where we implement the plant's photosynthesis process.

The introduction was heavily scaffolded with the help of the instructor.
We had written the code in a way to scaffold the encounter as well, some of which do not necessarily follow typical software engineering guidelines.
The code was verbose, repetitive so as to facilitate easy exploration and navigation, and ordered to make it easier for students to get a higher-level understanding of the science before diving deeper into the computational mechanics behind it.
In that sense, we prioritized learning science over writing efficient programs.

Multiple Representations

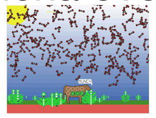
1. Physical macroscopic phenomena



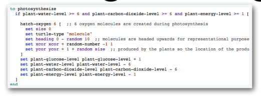
2. Standard scientific representation

$$2H_2O \rightleftarrows 2H_2 + O_2$$

3. Simulation: included microscopic phenomena overlaid on the macroscopic view



4. Modifiable code-based representations using NetLogo



```
to photosynthesize
  if plant-water-level >= 6 and plant-carbon-dioxide-level >= 6 and plant-energy-level >= 1 [

    hatch-oxygen 6 [  ;; 6 oxygen molecules are created during photosynthesis
      set size 0
      set turtle-type "molecule"
      set heading 0 - random 10  ;; molecules are headed upwards for representational purpose
      set xcor xcor + random-number -1 1
      set ycor ycor + 1 + random size  ;; produced by the plants so the location of the produ
    ]
    set plant-glucose-level plant-glucose-level + 1
    set plant-water-level plant-water-level - 6
    set plant-carbon-dioxide-level plant-carbon-dioxide-level - 6
    set plant-energy-level plant-energy-level - 1
  ]
end
```

It is worth noting here that the computing representations were interactive and highly modifiable, and students were encouraged to explore and modify the representations.

Science Representation

- Observed gaps in students' knowledge
- E.g. we asked if $H_2 + O_2 = H_2O$ was balanced

"… that would be H_2O_2" *"… it should be $H_2 + O = H_2O$"*

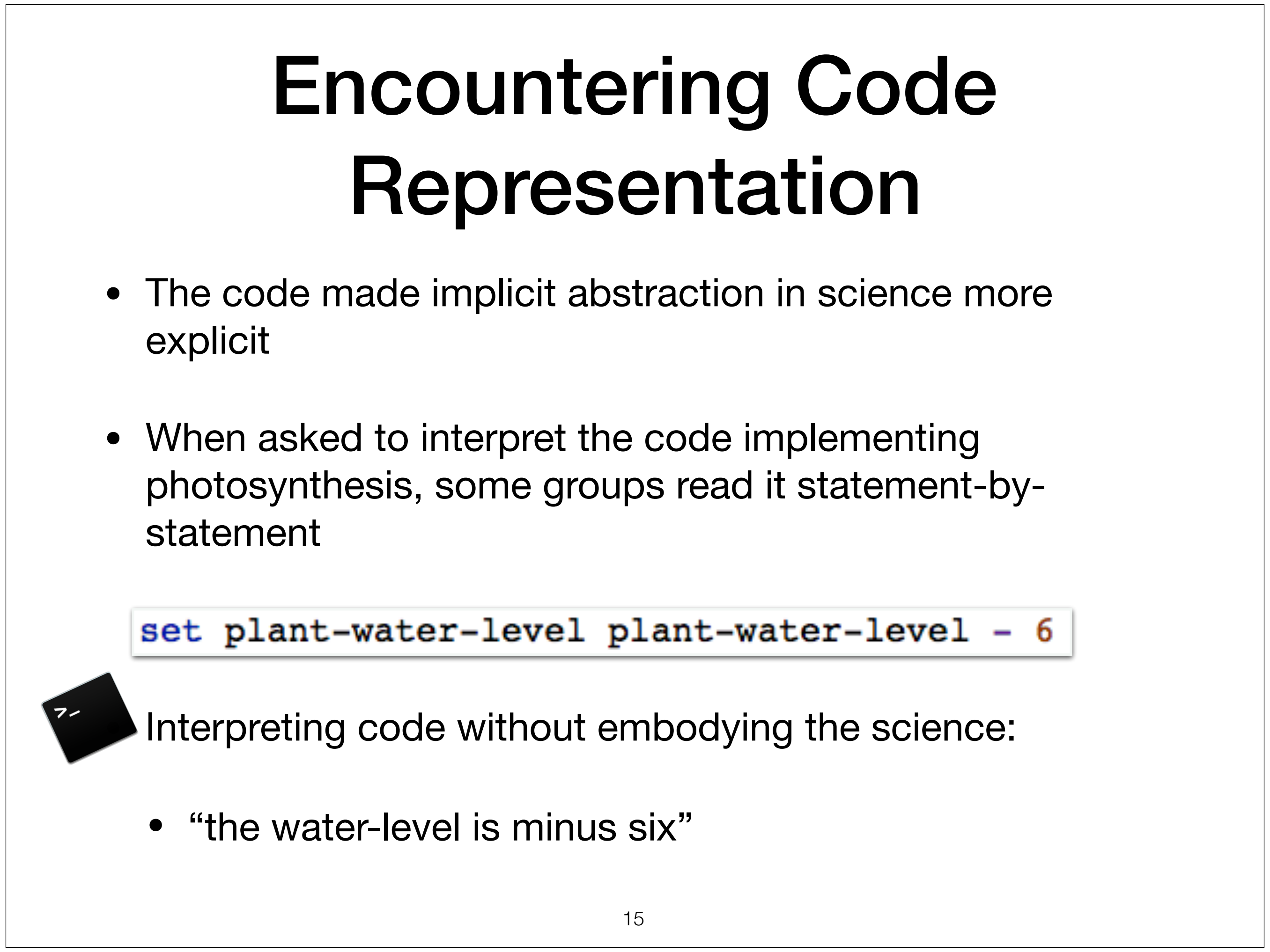
Students understood conservation of mass

- But not other implicit aspects represented in the equation about atoms, molecules, and stable energetic states

In the class, we observed that there were gaps in the students' knowledge.

For example, we had asked students if H2 + O2 equals to H2O was a balanced equation or not.
11 out of the 19 students who responded correctly mentioned that the equation violated the law of conservation of matter.
However, 8 of the 11 struggled to balance the equation correctly saying things like "that would be H2O2" or "it should be H2 + O = H2O".

It became clear that while students were familiar with the principle of conservation of matter, they had not imbibed the concepts of energetic states and molecular structure that is implicitly summarized in the chemical equation.



Encountering Code Representation

- The code made implicit abstraction in science more explicit
- When asked to interpret the code implementing photosynthesis, some groups read it statement-by-statement

```
set plant-water-level plant-water-level - 6
```

Interpreting code without embodying the science:

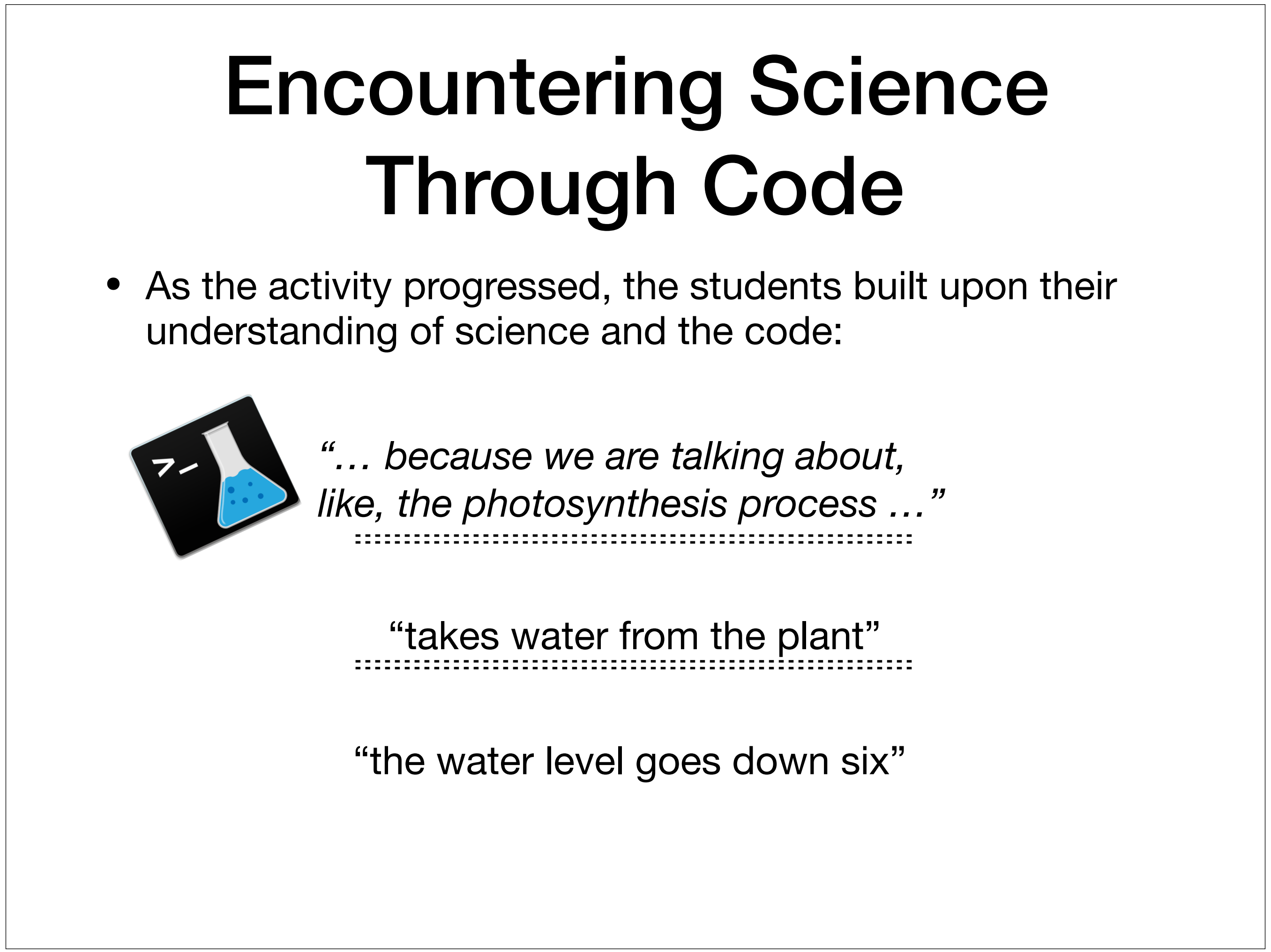
- “the water-level is minus six”

The implicit aspects of the equation became explicit when the students worked on the code.

Code requires statements that explicitly mention the change in state. This transformation is implicitly when students encounter chemical equation.

What we observed was that students were not able to draw from their science understanding to make sense of the explicit code statement.
That is, they were not able to move laterally.

For example, when a group of students when exploring the photosynthesis function encountered the statement “set plant-water-level plant-water-level - 6”, they grappled with the mechanics of the statement
The group interpreted it simply as “the water-level minus six” which is the notion associated with the abstraction common in computing.
They could not see the science embodied in the statement.



Encountering Science Through Code

- As the activity progressed, the students built upon their understanding of science and the code:

"… because we are talking about, like, the photosynthesis process …"

"takes water from the plant"

"the water level goes down six"

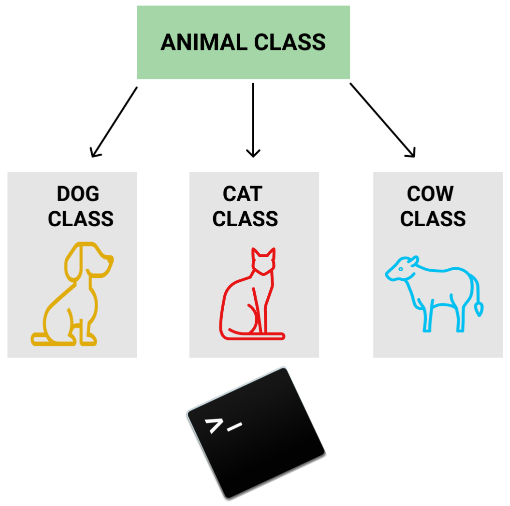
As the sessions progress, with the instructor's scaffolds, the students began to increasingly connect the science concepts with the code representation.

For the same group, we saw a movement where they increasingly saw the science where they moved from "the water level goes down six" to seeing that it represents taking of water from the plants to eventually seeing the entire code as a unit to represent photosynthesis.

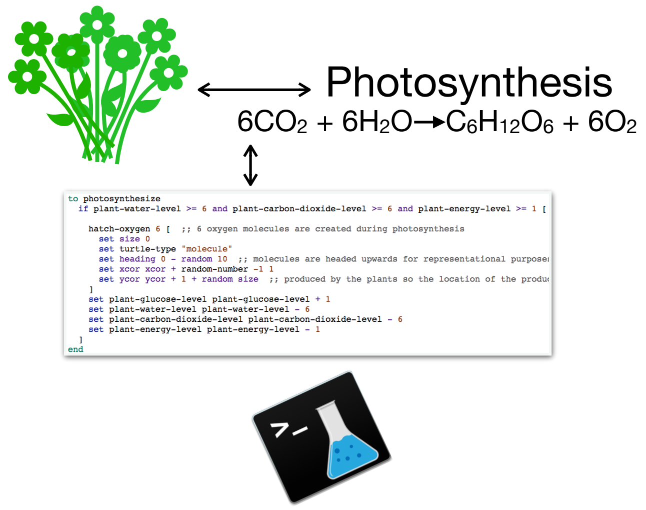
The students moved laterally between representations, connecting via abstraction-based thoughts to develop new understanding.

CT + Science

- Learning-to Abstract
- Learning-through-abstraction



Learning-to-read



$$6CO_2 + 6H_2O \rightarrow C_6H_{12}O_6 + 6O_2$$

```
to photosynthesize
  if plant-water-level >= 6 and plant-carbon-dioxide-level >= 6 and plant-energy-level >= 1 [

    hatch-oxygen 6 [  ;; 6 oxygen molecules are created during photosynthesis
      set size 0
      set turtle-type "molecule"
      set heading 0 - random 10  ;; molecules are headed upwards for representational purposes
      set xcor xcor + random-number -1 1
      set ycor ycor + 1 + random size  ;; produced by the plants so the location of the produc
    ]
    set plant-glucose-level plant-glucose-level + 1
    set plant-water-level plant-water-level - 6
    set plant-carbon-dioxide-level plant-carbon-dioxide-level - 6
    set plant-energy-level plant-energy-level - 1
  ]
end
```

Reading-to-learn

To summarize, in computing, we want students to learn to abstract.
We see this in programming exercises such as generalizing class structures, and thinking about class relations.

What we find in our study is that when students encounter multiple representations
Such as a macroscopic representation of the thing that they have seen in their day-to-day life
Which is then explained through a standard scientific representation
and engagement with it deepened further by presenting modifiable and interaction representation through computers,
students have to move laterally across representation,
abstract key features from it,
and build upon their understand to make sense of the new representations.
This is what we saw when the students moved from the mechanics of the code to realize the embodied science in the code.
That is, students learned through the abstraction.

These two differences are analogous to how students begin by learning to read and then progress where they start reading to learn.
Here too, we argue that the students learn to abstract, but we can provide scaffolds to move students further to learn through abstraction.

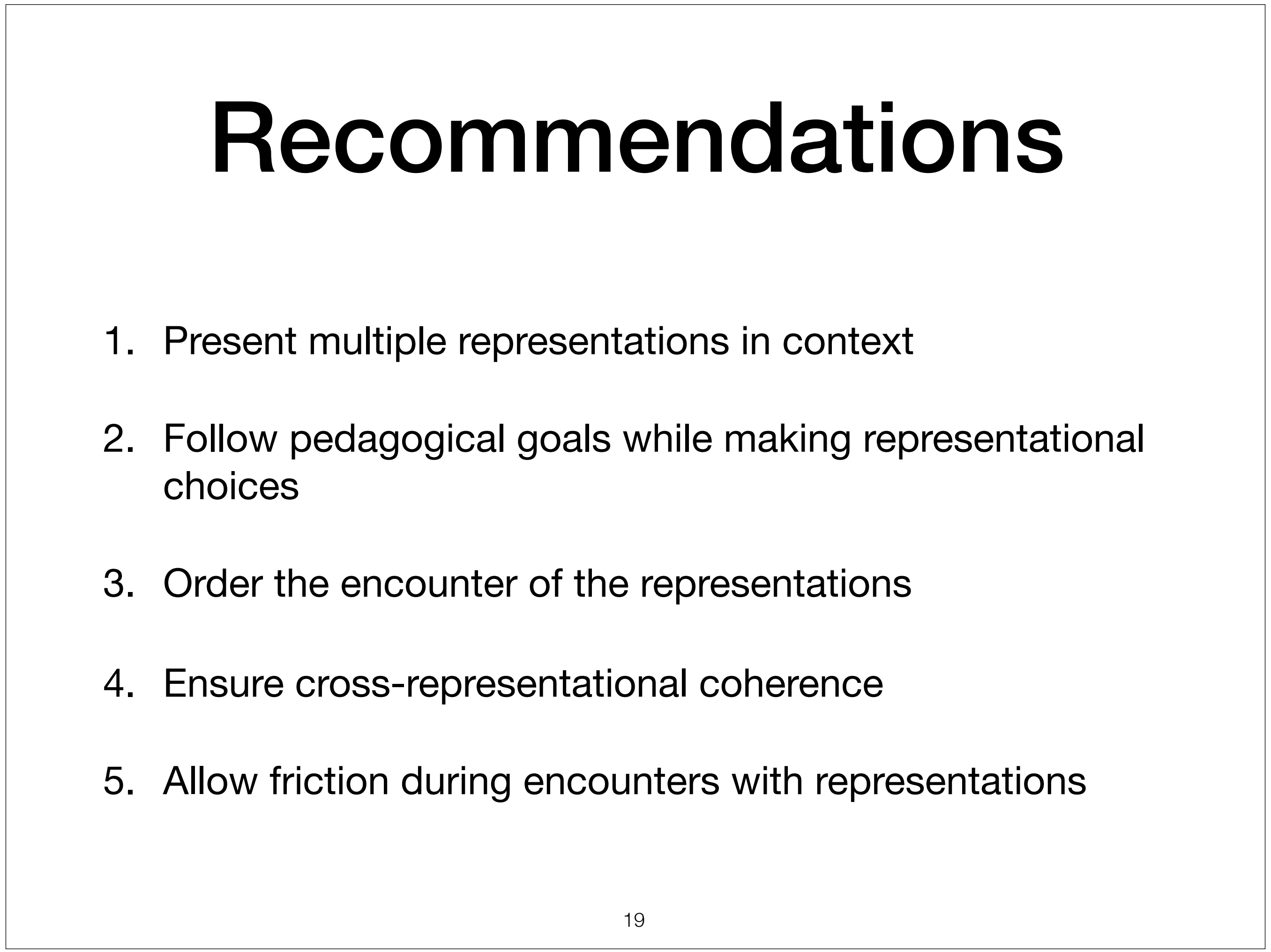
This, in sum, is what we argue in the paper.

Need for Careful Consideration

- Seeing code as being part of a single unit
  - Procedural abstraction: modularizing and reusing
- But it may not necessarily always work hand in hand:
  - Focus on hierarchies may not create encounters with lateral abstraction
  - Focus only on lateral abstraction may not suffice to learn about hierarchies

Seeing the code as being part of a single unit requires lateral movements.
From a computing perspective, this is part of procedural abstraction, critical to practices like modularizing and reusing.

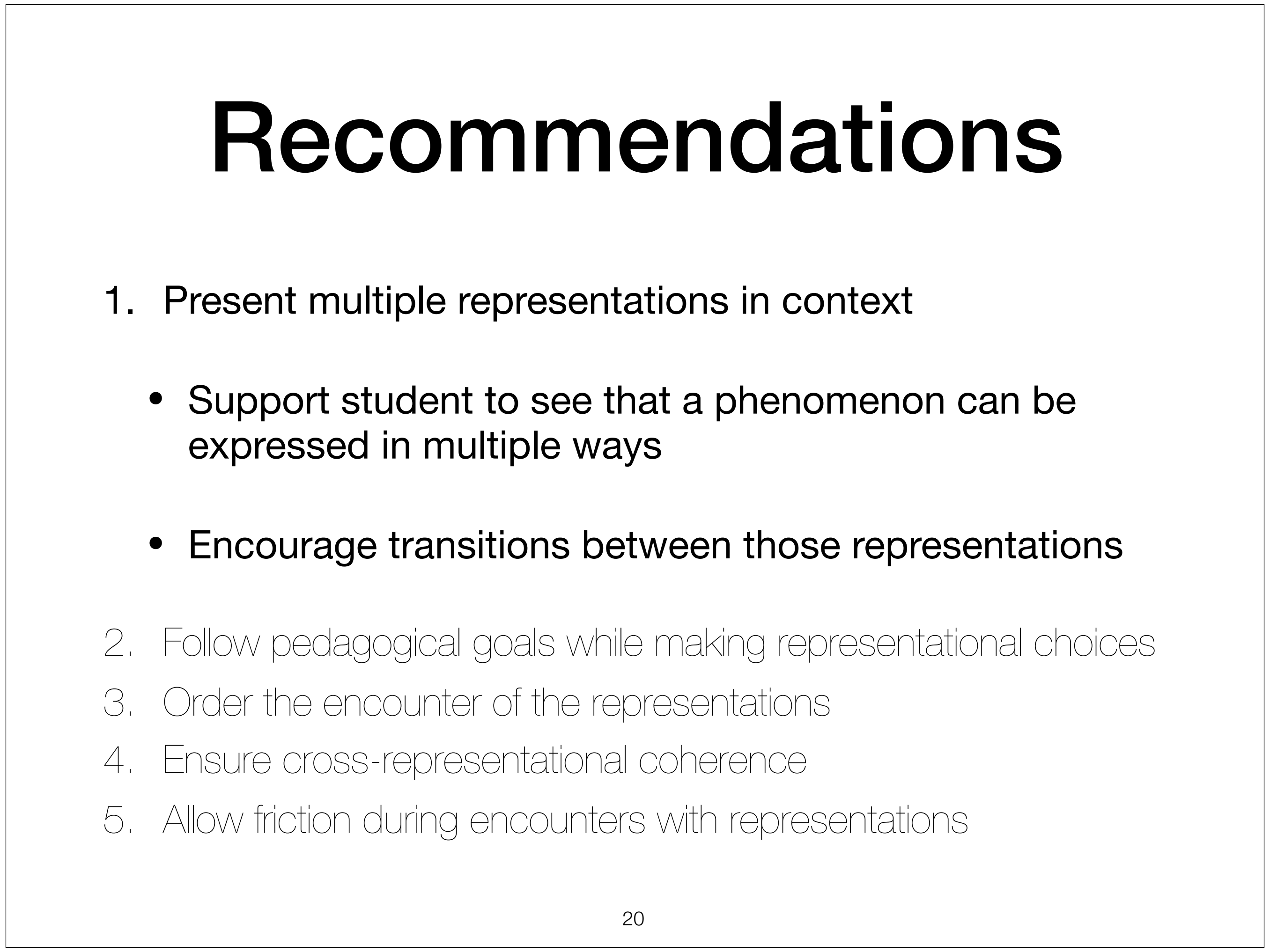
But this may not necessarily work hand in hand.
Focusing on hierarchies does not necessarily create encounters with lateral abstraction as we discussed with the binary-decimal-text example earlier.
But similarly, focusing on lateral abstraction alone may not be sufficient for the students to grasp ideas of hierarchical abstraction.
This, we believe is a limitation which requires further exploration.



Recommendations

1. Present multiple representations in context
2. Follow pedagogical goals while making representational choices
3. Order the encounter of the representations
4. Ensure cross-representational coherence
5. Allow friction during encounters with representations

Reflecting on our experience, we recommend the following to promote students' deep engagement with abstraction and learning through abstraction.



Recommendations

1. Present multiple representations in context
   - Support student to see that a phenomenon can be expressed in multiple ways
   - Encourage transitions between those representations
2. Follow pedagogical goals while making representational choices
3. Order the encounter of the representations
4. Ensure cross-representational coherence
5. Allow friction during encounters with representations

First, we believe that presenting the multiple representation in context was beneficial for students to move between the representation and draw upon their understanding.
The students were familiar with some of the aspects and the familiarity supported in seeing how it could be expressed in different ways.

Recommendations

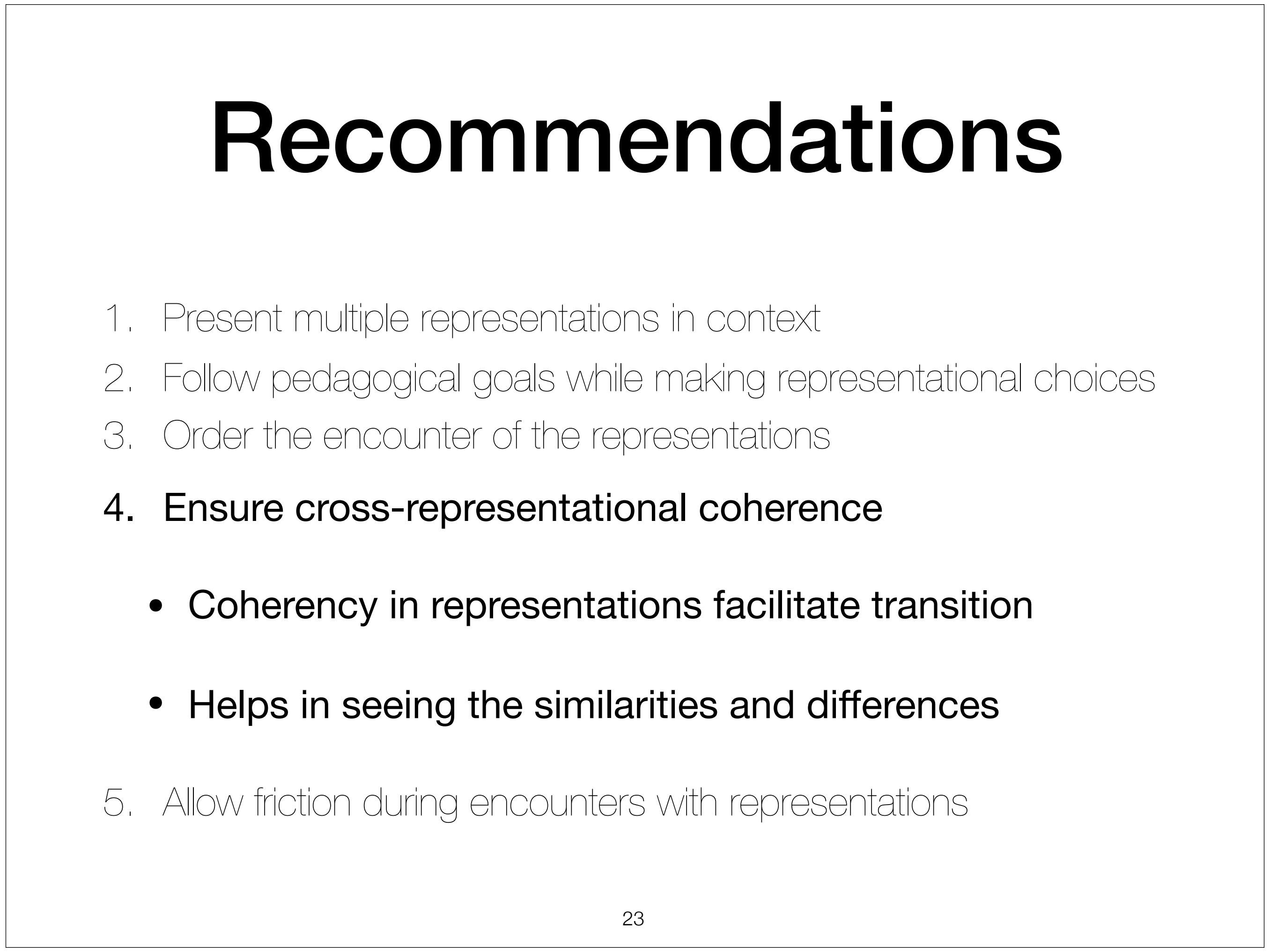
1. Present multiple representations in context
2. Follow pedagogical goals while making representational choices
   - Instructors are critical
   - The photosynthesis function emphasized the pedagogical point that the instructor wanted to make
3. Order the encounter of the representations
4. Ensure cross-representational coherence
5. Allow friction during encounters with representations

Technology is a tool that can facilitate the teaching process but can do so only if the instructors find it to be of value in furthering their pedagogical practice.
For example, the photosynthesis function emphasized exactly the pedagogical point that the instructor wanted to make and that afforded opportunity for the instructor to scaffold students' encounter with the representation.

Recommendations

1. Present multiple representations in context
2. Follow pedagogical goals while making representational choices
3. Order the encounter of the representations
    - Presenting the simulation before the code-based model helped contextualize the students' encounter (e.g. Ainsworth, 2008)
4. Ensure cross-representational coherence
5. Allow friction during encounters with representations

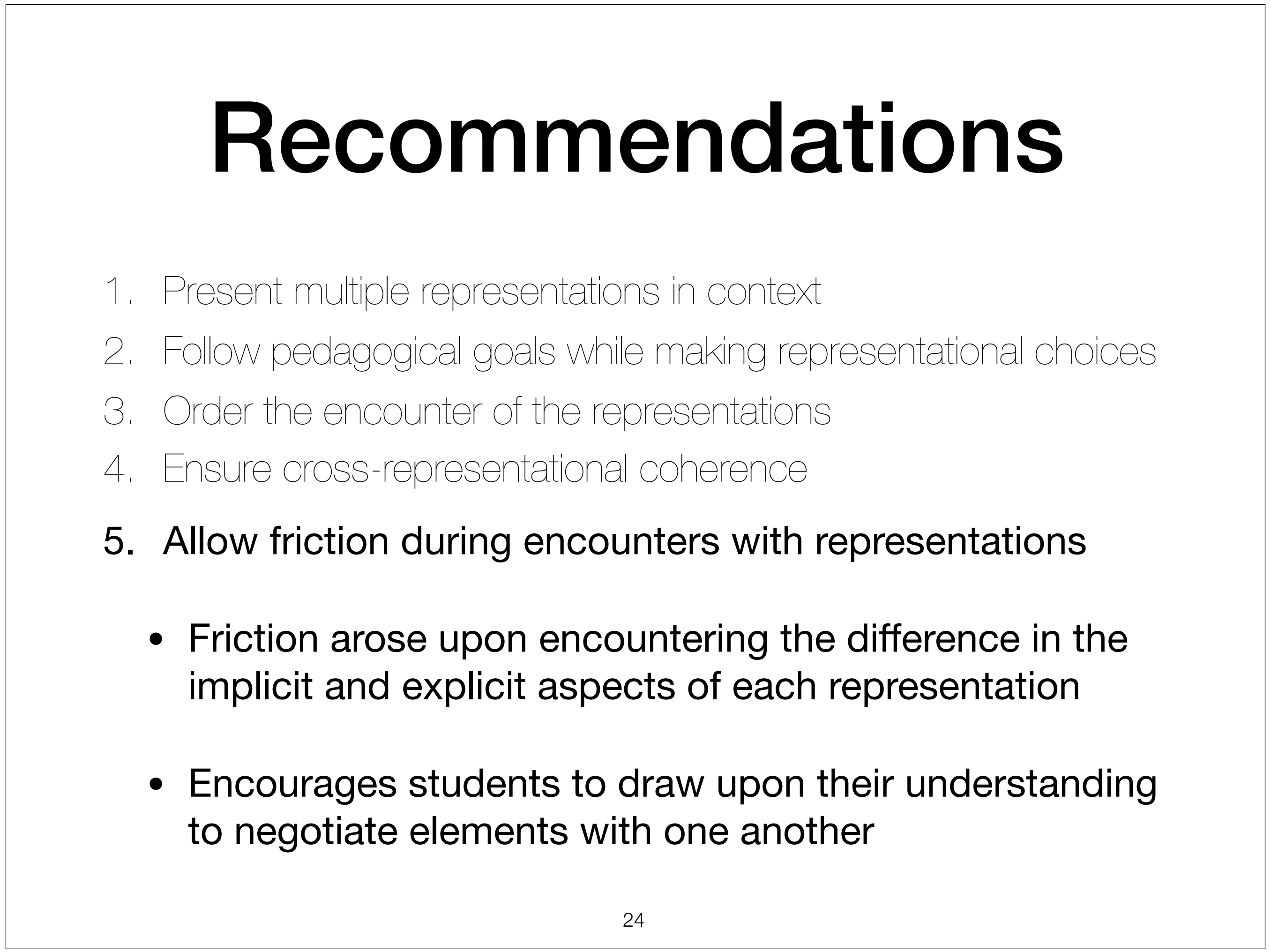
Third, we believe that it is critical to order the students' encounter with the different representations.
Beginning from representations that they were familiar with allowed them to draw upon their understanding when they encounter newer representations.
The need for ordering representations has been argued in prior learning science literature particularly in the brilliant work by Shaaron Ainsworth.



Recommendations

1. Present multiple representations in context
2. Follow pedagogical goals while making representational choices
3. Order the encounter of the representations
4. Ensure cross-representational coherence
    - Coherency in representations facilitate transition
    - Helps in seeing the similarities and differences
5. Allow friction during encounters with representations

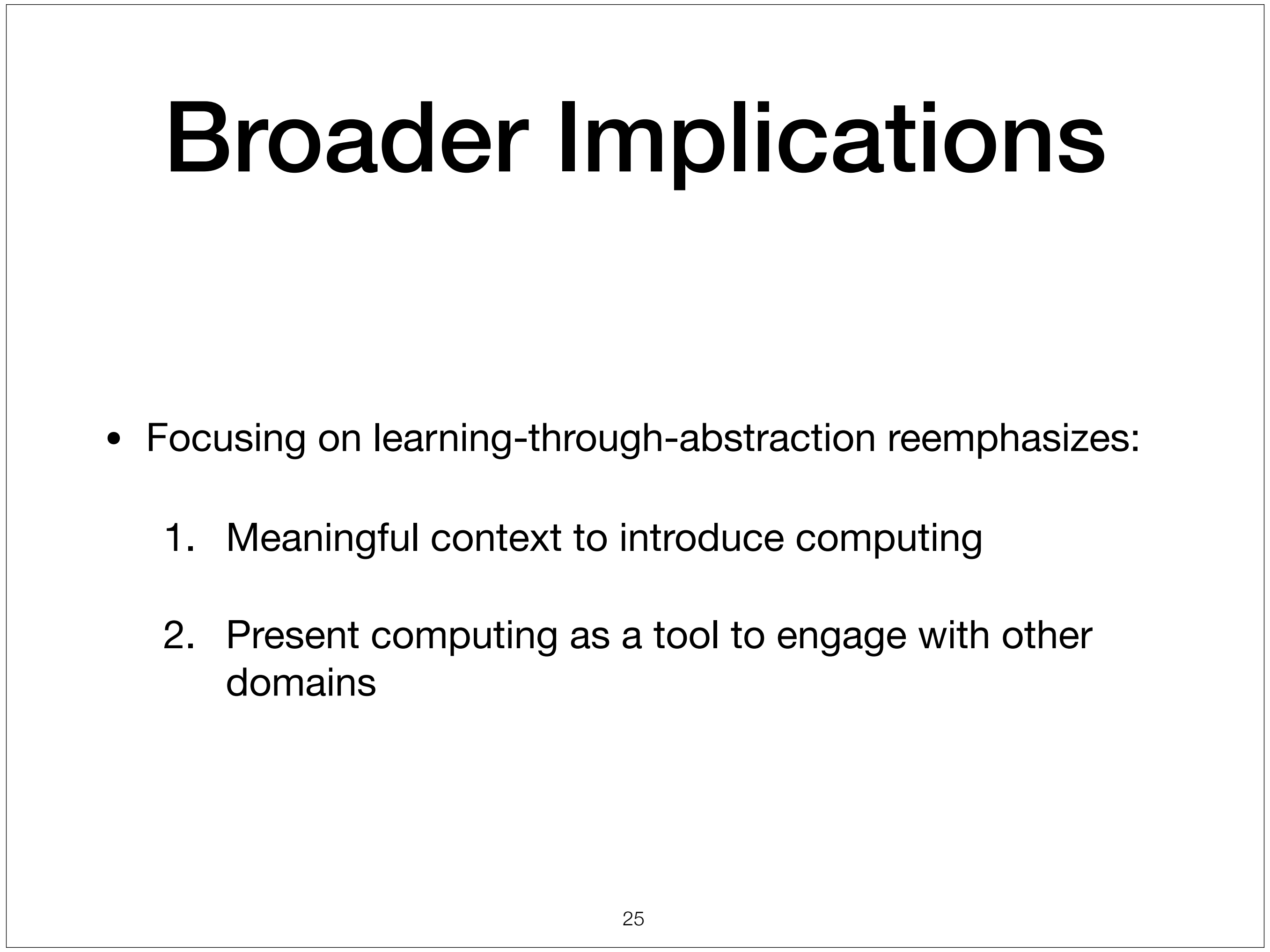
Related to the ordering, we believe that ensuring that the representations were coherent helped students to transition between representations.
The students were able to see the similarity and notice the differences that arose in a new representation.
For example, students were able to connect the overlaid representation of the glucose molecule to the plant-glucose-level in the code.



Recommendations

1. Present multiple representations in context
2. Follow pedagogical goals while making representational choices
3. Order the encounter of the representations
4. Ensure cross-representational coherence
5. Allow friction during encounters with representations
    - Friction arose upon encountering the difference in the implicit and explicit aspects of each representation
    - Encourages students to draw upon their understanding to negotiate elements with one another

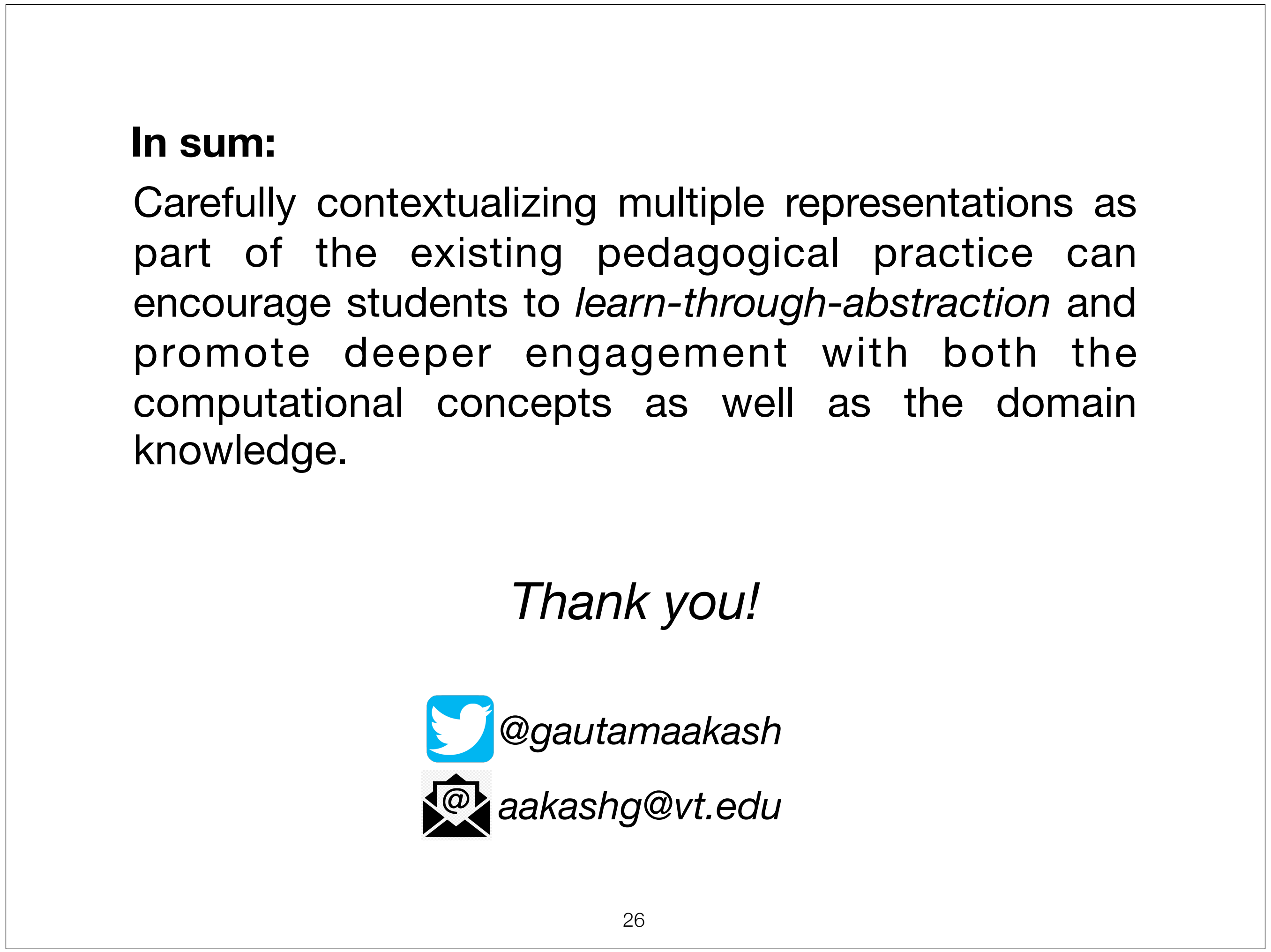
Finally, we argue that there is a need to allow for friction to occur in the students' discussions when they encounter different representations.
This was most prominently seen when the students encountered things that were implicit in the equation being presented explicitly in the code.
Building on the importance of students' struggle in learning concepts, such as Piaget's assimilation and accommodation, we believe that the friction encouraged student to draw upon their understanding of the phenomenon from other representations and negotiate it with the elements in the new one they encountered.



Broader Implications

- Focusing on learning-through-abstraction reemphasizes:
    1. Meaningful context to introduce computing
    2. Present computing as a tool to engage with other domains

We argue that focusing on learning-through-abstraction promotes deeper engagement with both the critical idea within computing as well as the domain knowledge.
More broadly, the focus on learning-through-abstraction reemphasizes the need for students to be introduced to computing through meaningful context.
Further, it emphasizes the value of computing as a tool, one that promotes deep engagement with the ideas in other domains.



In sum:

Carefully contextualizing multiple representations as part of the existing pedagogical practice can encourage students to *learn-through-abstraction* and promote deeper engagement with both the computational concepts as well as the domain knowledge.

Thank you!

@gautamaakash

aakashg@vt.edu

We are sad that the couldn’t be there to answer questions or take this discussion further but please feel free to send us an email to continue the conversation.

Thank you!