Revisiting Analogical Reasoning in Computing Education: Use of Similarities between Robot Programming Tasks in Debugging

ChanMin Kim
Emre Dinç
Eunseo Lee
Afaf Baabdullah
Anna Y. Zhang
Brian R. Belland

This is an author-created version of the manuscript published as:
Kim, C., Dinç, E., Lee, E., Baabdullah, A., Zhang, A. Y., & Belland, B. R. (2023). Revisiting analogical reasoning in computing education: Use of similarities between robot programming tasks in debugging. *Journal of Educational Computing Research*, *61*(5), 1036–1063. https://doi.org/10.1177/07356331221142912

Abstract

Analogical reasoning is considered to be a critical cognitive skill in programming. However, it has been rarely studied in a block-based programming context, especially involving both virtual and physical objects. In this multi-case study, we examined how novice programming learners majoring in early childhood education used analogical reasoning while debugging block code to make a robot perform properly. Screen recordings and scaffolding entries, reflections, and block code were analyzed. The cross-case analysis suggested multimodal objects enabled the novice programming learners to identify and use structural relations. The use of a robot eased the verification process by enabling them to test their analogies immediately after the analogy application. Noticing similar functional analogies led to noticing similarities in the relation between block code as well as between block code and the robot, guiding to locate bugs. Implications and directions for future educational computing research are discussed.

Keywords: computing education, robot programming and debugging, analogical reasoning, novice programming learners, block-based coding

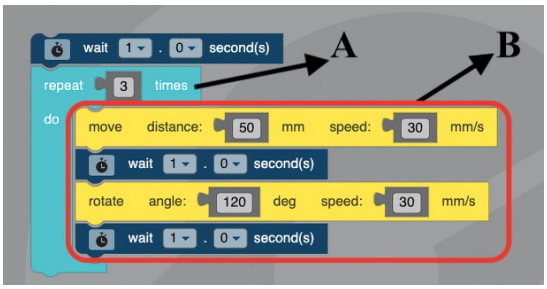
Revisiting Analogical Reasoning in Computing Education:

Use of Similarities between Robot Programming Tasks in Debugging

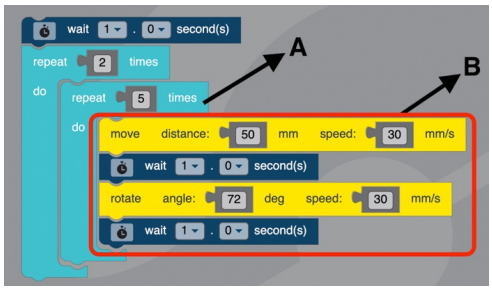
Analogical reasoning is critical in programming as it is in other design and problem-solving contexts. Analogical reasoning is defined as a way of thinking in which similarities between different tasks are perceived and used (Gentner & Smith, 2013; Holyoak et al., 1984; Holyoak & Thagard, 1995). Computing research has attended to analogical reasoning for decades. For example, work published in the *Journal of Educational Computing Research* studied analogical reasoning as a way of understanding learners' ability to program (Clement et al., 1986) and used analogical reasoning in programming instruction (Grandgenett & Thompson, 1991). More recent computing research engaged learners in analogical reasoning to improve programming learning (e.g., Aggarwal et al., 2019; Cao et al., 2016; Ichinco et al., 2017). Nonetheless, analogical reasoning in *block-based* programming has rarely been studied (c.f., Ichinco et al., 2017) while the role of visual analogy could be critical in analogical reasoning in visual programming environments. There is also little knowledge of analogical reasoning in the context of programming learning that involves *both virtual and physical* objects. Analogical mapping could be more complicated with multimodal objects than unimodal objects because relational structures matter not only within virtual objects but also between virtual and physical objects. For instance, when programming block-based code (virtual object) to control a robot (physical object), finding similarities in relational structures within block code between the previous task and current task is part of analogical reasoning but so is in relational structures between the block code and the robot. As shown in Figure 1, there are common relational structures between the virtual objects. That is, the relation between the repeat # times block A and the movement blocks B is that B is repeated # times indicated in A. There are also relational structures also between the block code (the virtual object) and the robot (the physical object) in that the robot is moved by the block code.

Figure 1

A Sample Source and Target Tasks for Relational Structures between Virtual Objects



Source Task (Tracing a triangle)



Target Task (Tracing a pentagon twice)

Given these gaps in analogical reasoning research, we examined analogical reasoning during robot programming using block code in the present study. We studied the analogical reasoning processes of non-CS novice programming learners. The knowledge produced from the present study is applicable to understanding broader learner populations, including female students with no initial interest in majoring in CS who could be invited to computing careers. We addressed the following research question in the present study: *How do novice programming learners use similarities between robot programming tasks in debugging?*

Literature Review

Debugging is the process by which programmers identify and resolve bugs. While research on debugging has proliferated, it has largely been on debugging text-based languages (e.g., Katz & Anderson, 1987; McCauley et al., 2008; Spinellis, 2018). One thrust of this research extends the research on analogical reasoning to the computer science domain. Namely, it describes a process by which programmers faced with buggy code create a representation of the current state of the program, and then search through their prior knowledge of similar problems to find and adapt a solution key (i.e., analogue) to the problem at hand (Burstein, 1983). This is akin to the idea of near and far transfer, where people use what they learned previously in a similar or different situation (Barnett & Ceci,

2002; Nokes-Malach & Mestre, 2013). In so doing, programmers create a hypothesis that applying the analogue will resolve the bug and lead to effective execution of the code. They then implement the solution to test the hypothesis. This idea fits with the computer science literature, in which a hypothesis-driven approach to debugging is often considered the gold standard (Araki et al., 1991; McCauley et al., 2008).

When it is near, or low road transfer, users simply have to remember and apply a solution from a very similar problem (Salomon & Perkins, 1989). But success in this endeavor relies on the creation of an accurate mental model of the current state of the buggy program (Burstein, 1988; Gentner & Stevens, 2014; Nersessian, 2008). Such a model needs to take into consideration not only surface features but also structural elements and how they interact (Jonassen, 2011). Creating models of problems requires robust disciplinary knowledge, which many novices lack (Magana et al., 2020). Ideally, a mental model represents causal mechanisms by which different elements in the model change (Nersessian, 2008). Such representation of causal mechanisms can help users understand what is happening and why in the program, which in turn can help to pinpoint the bug that is causing the program to malfunction. Far too often, beginning programmers rely on surface features to create mental models of programs, which can cause the wrong analogues to be applied, or for potentially relevant analogues to be inaccessible as inert knowledge (Perkins & Martin, 1986).

That most debugging literature examines how people debug text-based languages is an important limitation. Block-based coding languages involve some of the same computer science processes (e.g., recursion, looping, variable specification) as text languages, but the former were developed for use among student populations with lower programming motivation and programming knowledge. As such, it is unwise to assume that learners using block-based languages will debug in the same manner as learners using text-based languages.

First, a hypothetico-deductive approach to analogical reasoning requires that the reasoner have vast stores of relevant prior knowledge, which many novice block-based programmers do not have. Furthermore, the way that coders create mental models of programs may differ between those using block-based coding and those using text-based coding languages, which can result from limited prior knowledge of coders using block-based coding (Robins et al., 2003).

Relevant alterative perspectives on transfer include the actor-oriented view (Lobato, 2003, 2006) and preparation for future learning (Bransford & Schwartz, 1999). The actor-oriented view holds that researchers should examine a transfer situation from the reasoners' perspective to see what they notice and how they use what they notice to solve the problem (Lobato, 2003, 2006). According to the preparation for future learning perspective, one should judge the success or lack thereof in the use of an analogue according to the extent to which the reasoner is better prepared to learn new material based on the experience applying the learned material to the new situation (Bransford & Schwartz, 1999).

Conceptual Framework

To examine what novice programming learners notice and how they use what they notice to solve the problem of buggy code, we constructed a conceptual framework using literature on analogical reasoning in computer programming (Aggarwal et al., 2019; Basawapatna, 2016; Cao et al., 2016; Clement et al., 1986; Clements, 1987; Grandgenett & Thompson, 1991; Ichinco et al., 2017; Jang, 1992; Kurland et al., 1986; Muller & Haberman, 2008), in design (Ahmed & Christensen, 2009; Chai et al., 2015; Cheong et al., 2014), and in general (Gentner & Smith, 2013; Holyoak et al., 1984; Holyoak & Thagard, 1995; Sternberg & Rifkin, 1979). Such a multidisciplinary conceptual framework was necessary since there was no single line of literature readily applicable to the unique context of the present study in

which analogical reasoning involved multimodal objects (virtual and physical objects), design (for/through debugging), and block-based programming.

The conceptual framework was organized through three perspective angles: analogical reasoning processes, foci, and forms. Table S1 in the supplementary documents summarizes the literature used in constructing the perspective angles (the supplementary tables are available in the online version of the journal; readers of the print version can access the online supplement in the online version of the journal or request a copy from the first author). First, analogical reasoning processes in the framework refer to phases of structuring (encoding and inferring), mapping, applying, and verifying. Analogies share a basic set of processes that include retrieval (i.e., access to prior analogous cases), mapping (i.e., structural alignment between source and target tasks), and evaluation (i.e., judging quality and accuracy of the analogy and inference) (Gentner & Smith, 2013). Elaborated processes noted in Sternberg and Rifkin (1979) are comprised of the phases of structuring, mapping, applying, and verifying. Encoding, inferring, mapping and applying as phases in a framework enables more verbally expressed analogies (Grandgenett & Thompson, 1991). Execution of a program by testing the applied code, which is verifying it, is a significant part of analogical reasoning while programming (Gentner & Smith, 2013; Ichinco et al., 2017).

In the first phase, structuring, reasoners analyze the components of the source and target tasks and infer the structure of the target task by comparison to structures with the source task. Identifying analogs between domains is not an easy task (Holyoak et al., 1984) because recognizing different analogous solutions depends on domain knowledge of the reasoner (Cheong et al., 2014). In the mapping phase, reasoners map relationships between source and target task elements by identifying similarities and differences between them. Albeit similar to mapping described in Gentner and Smith (2013), dissimilarities are part of mapping in Sternberg and Rifkin (1979). The framework of the present study covers both

similarities and dissimilarities because the knowledge of how novice programming learners discern dissimilarities between the source and target tasks could reveal the (ir)relevance or (in)accuracy of their analogical reasoning. In the applying phase, reasoners use identified similarities or dissimilarities to produce the desired output in the target task. Correct or incorrect use of analogies can occur as a result of actively applying the identified and mapped analogies (Aggarwal et al., 2019). In the last phase, verifying, reasoners evaluate and justify the analogy used through understanding the output of the target task.

Second, analogical reasoning *foci* in the framework refer to visual, functional, behavioral, and structural analogies. The present study involved design considering that analogical reasoning was needed not only in understanding the current task in relation to the prior task but also in programming, more specifically in debugging (Clement, 1987). The participants were asked to redesign the given buggy code to work as expected. Thus, we integrated the design literature also using analogical reasoning into our framework to examine analogical reasoning with foci not only on visual and structural analogies but also on functional and behavioral. According to Gero and colleagues (Gero & Kannengiesser, 2004; Gero & Mc Neill, 1998), reasoners navigate through a design problem by focusing on function, behavior, and structure of an object used in design. They solve design problems based on understanding of the purpose of the object (i.e., function), the actions or processes of the object (i.e., behavior), and relations between the elements of the object (i.e., structure). Analogical reasoning also occurs with visual cues such as pictures and words, and these visual analogies are often used by designers to solve design problems (Chai et al., 2015). Similarly, to navigate programming tasks in the present study and find out solutions for debugging, analogical reasoning had to have a particular focus on the visual appearance, the function, the behavior, and the structure of objects (e.g., block-code, robots).

Last, while relational structure matches are usually more important than property (e.g., visual appearance) matches (Gentner & Smith, 2013), our framework includes all these foci because it is to examine which ones are (un)used/useful in analogical reasoning that uniquely involves multimodal objects in complex ways. Thus, combined with these analogical reasoning foci, the conceptual framework also specifies *forms* of the objects on which the analogical reasoning is centered. That is, the framework specifies virtual objects, physical objects, and the relation between virtual and physical objects within the structural analogies focus. For example, physical objects are often considered in design studies (e.g., Chai et al., 2015). Also, relations exist between objects and among problems, and these relations enrich the analogical reasoning process (Muller & Haberman, 2008). When mapping between the current and previous programming tasks focuses on similarities in structures, the framework notes that structures within *virtual* objects (e.g., block-code), structures within *physical* objects (e.g., robots), and structures within the *relation between virtual and physical objects* (e.g., block-code controlling the robot) are compared.

Method

Research Design

To gain an in-depth understating of how novice programming learners use similarities between robot programming tasks in debugging, we employed a multi-case study method (Creswell, 2013). Multi-case study design was chosen to study diversity or typicality of the phenomenon (Bogdan & Biklen, 1992), analogical reasoning in this study.

Participants and Context

This study was part of a larger research project for early childhood education majors' learning to program and debug. Two cases were selected based on variation (e.g., race, programming experiences) in a class of nineteen undergraduates who participated in a robotics and play unit. Participants learned coding for children's play and playful learning

using Ozobot and Ozoblockly. Except for the last two of eight 80-minute classes, they engaged in programming mainly as debugging tasks. They were invited to debug nine sets of erroneous block code with increasing difficulties during the unit. No participant had prior robot programming experience. All but one participant indicated no to little programming knowledge prior to the unit. One participant who reported intermediate programming knowledge was Judith. She and her partner, Anne, are one of the two cases in the study. The other case was the pair of Arianna and Kimberly. These two cases were selected as representative cases of the current study to best understand the studied phenomenon (Creswell, 2013) of analogical reasoning. All were female and juniors. Anne was Asian and the rest were Caucasian. All names in this paper are pseudonyms.

Data Collection

The data were collected while participants engaged in the debugging task called "Cleaning the Playroom" as the target task and a task called "Color Game" as the source task (see Table S2 in the supplementary documents for the similarities and dissimilarities between the source and target tasks). Participants' computer screens were recorded during debugging and while responding to scaffolding prompts. The length of screen-recordings for the Cleaning the Playroom task was 61 minutes for Anne and Judith and 91 minutes for Ariana and Kimberly. Their reflections on the challenges encountered during debugging and final code were collected during the unit and artifact-based interviews were conducted after the unit ended. The total length of these interviews was about 1.5 hours.

Data Analysis

We developed a coding scheme based on the conceptual framework of the study grounded in the analogical reasoning literature (Chai et al., 2015; Clement et al., 1986; Gentner & Smith, 2013; Gero & Kannengiesser, 2004, 2014; Gero & McNeill, 1998; Ruppert, 2013; Sternberg, 1977; Sternberg & Rifkin,1979). The highest-level nodes of the

coding scheme were the four components of analogical reasoning; that is, structuring (encoding/inferring), mapping, applying, and verifying. Example subnodes for each component included 'refer to the source explicitly' (encoding/inferring), 'identify similarity based on the relationship between physical and virtual objects' (mapping), 'use similar analogy correctly-lead to successful debugging' (applying), and 'test code and confirm of meeting programming main goal and subgoals' (verifying) (see Table S3 in the supplementary documents for more examples of the coding scheme). We went through three rounds of pilot coding to revise and refine the coding scheme and to address discrepancies among coders. Then, we coded Anne and Judith's data from their Cleaning the Playroom debugging in NVivo. The average interrater reliability ICC score was 0.849. Next, we coded data from Ariana and Kimberly's Cleaning the Playroom debugging task (average ICC = 0.732). Then we aggregated preliminary findings into sense-making tables based on the coded data. We engaged in multiple rounds of discussions on the coded data and sense-making tables before finalizing our findings.

Findings and Discussion

Overall Flow of Noticing and Using Similarities between Debugging Tasks

First, noticing similarities in *relational commonalities between block code and the robot* guided to the discovery of *functional* analogy. For example, Ariana and Kimberly noticed the *function* of the line navigation block in tasks when they saw the *relational commonality* between line navigation and the robot behaviors (see ① to ② in Figure 2). This in turn led to the discovery of ③ *relational commonalities* within block code (the line navigation and loop blocks) between the target and source tasks. Noticing relational structures ① between virtual and tangible objects (the line navigation block and the robot), mediated by the function ② of the noticed virtual object (the line navigation block), led to

noticing relational structures ③ with other objects (the loop block). A similar flow was found in Anne and Judith's debugging where noticing similarities in the function of elements in a virtual object guided them to the discovery of relational commonalities within block code. Specifically, Anne and Judith went through from ① to ② and ③ in Figure 3, which depicts that they did not begin seeing ③ the *relational commonality* in the structural relation between the line navigation block and the repeat block until they noticed ① and ② the functional commonalities of the line navigation and repeat blocks between tasks.

Figure 2

Flow of Ariana and Kimberly's Analogical Reasoning Foci and Forms

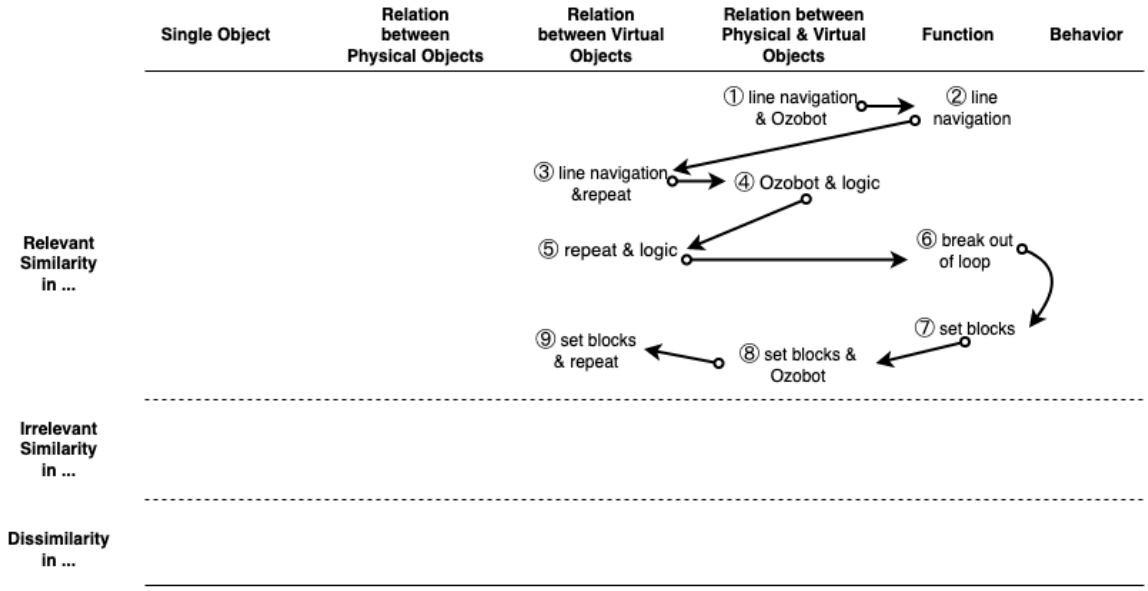
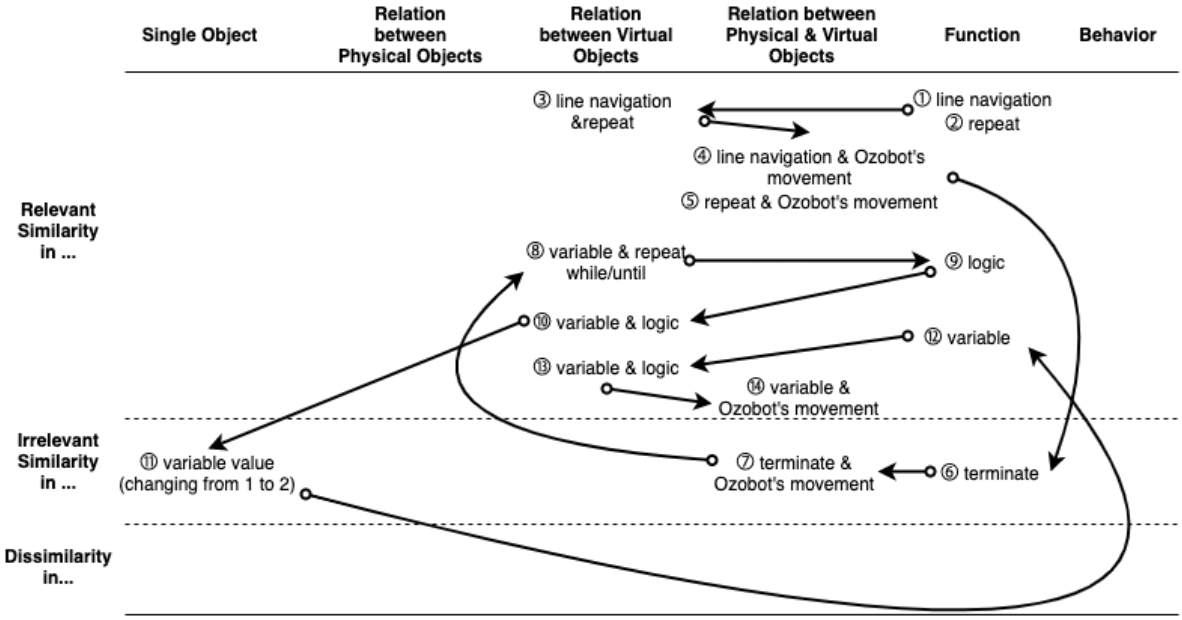


Figure 3

Flow of Anne and Judith's Analogical Reasoning Foci and Forms



Both groups also discovered *similarities in relational commonalities between the block code and the robot* after noticing similarities in relational commonalities between the block code in target and source tasks. For example, the relation between the line navigation and the repeat blocks ③ guided Anne and Judith to discover the relation between the line navigation block and the robot's movement ④ as well as the repeat block and the robot's movement ⑤ in Figure 3. Ariana and Kimberly first noticed the relation between line navigation and repeat blocks ③ and then noticed the relation between the logic block and the robot's movement ④ in Figure 2.

The opposite trend was also observed, where noticing similarities in *relational commonalities between the block code and the robot* guided them to the discovery of *relational commonalities between the block codes* between the target and source tasks. For example, Ariana and Kimberly mapped the *relational commonality* between the logic block and the robot ④ in the target and source tasks (see Figure 2). This mapping led them to notice the *relation* between the logic and loop blocks ⑤. After noticing relational structure ④ between block code and the robot (the robot and logic), as in ①, mediated by relational

commonalities ⑤ between block code (logic and loop blocks), Ariana and Kimberly noticed the *function* ⑥ of the break out of loop. The flow starting from relational commonality related to the logic block was mediated with a more familiar repeat block and ended with the function of a special repeat block (i.e., break out of loop) within the logic. This flow can be attributed to less familiarity with logic blocks compared to more familiarity with the repeat block throughout their analogical comparison process in the target task until ⑥. The flow of relational commonalities between visual objects indicated a high level of systematicity in mapping. That is, a relation between visual objects triggered another relation between visual objects while reasoning analogically in debugging.

Noticing either *relevant similarities* (Ariana and Kimberly) or *irrelevant similarities* (Anne and Judith) *in relational commonalities between block code and the robot* guided the two pairs to discover *relevant similarities in relational commonalities between block codes* in target and source tasks. Ariana and Kimberly identified the relevant similarity in relational commonality between the missing logic block and the robot's movement and then discovered the relational commonality between repeat and missing logic blocks. On the other hand, Anne and Judith noticed an irrelevant similarity in relational commonality between the terminate block and the robot's movement. The irrelevant similarity guided the pair to discover relevant similarity in relational commonality between repeat (while/until) and variable blocks. This result is important because even analogies based on irrelevant similarity in relational commonality led these novice programmers, Anne and Judith, to notice analogies based on relevant similarity in relational commonality while debugging.

There were also cases where noticing the *function of block code* led to the *relational commonalities between block code and the robot* between target and source tasks. For example, Ariana and Kimberly in Figure 2 noticed the *function* of set blocks ⑦ between target and source tasks. Set blocks were needed to be set to a specific number in the target

task. The noticed function of set blocks guided them to *structural relation* between set blocks and the robot ⑧. The part of the analogical reasoning flow starting from the function of block code was helpful for identifying a bug in the target task. Similarly, Anne and Judith noticed the function of block code (i.e., line navigation and repeat blocks), which led to the discovery of ④, ⑤ *relational commonalities* between the block code and the robot (see Figure 3). In both cases, functional analogy led to noticing the relational commonalities between block code and the robot, which in turn helped to identify the bugs in the target task. Such a productive role of functional analogy is contradictory to findings in analogical reasoning literature in which functional comparisons were found to limit reasoners' capacity to identify relevant analogy (Cheong et al., 2014). The multimodal forms of objects in the present study may have impacted this finding. That is, due to symbolic, visual, and material forms used in debugging tasks but also relational structures embedded within and between the multiple forms, understanding function of objects was critical to understanding objects and structures within them and associated with other objects.

However, noticing and using relational commonalities did not always lead to successful debugging. In Figure 2, after noticing the structural relation between set blocks and the robot's movement ⑧, Ariana pointed out moving set blocks into the repeat block. This was a *relational commonality between block code* ⑨. They applied this analogy and ended up with unsuccessful debugging. The directed *structural relation between block codes* ⑨ (i.e., set and repeat blocks) by *the function* ⑦ of a block code (i.e., set blocks) led Ariana and Kimberly to unsuccessful debugging. Similarly, when Anne and Judith in Figure 3 noticed ⑩ the relational commonality between the target task and the source task in the structural relation between the variable block and the logic block, they then fixated on ⑪ analogical comparison of the numeric value in the math block between tasks. ⑪ Judith

suggested changing the numeric value within a math block from 1 to 2 in the Cleaning the Playroom debugging task. In the Color Game task, math blocks had the number = 2 in logic (if/do) blocks for intersection colors red and green. This was an unrelated single object between the two tasks to debug the Cleaning the Playroom. Anne and Judith did not apply this single object as discussed earlier, meaning they did not change the value in the math blocks in the Cleaning the Playroom. This analogy was unrelated to bugs in the target task, thereby leading to studying ⑫ the function of variable blocks. Anne and Judith wanted to add the missing logic block after noticing that they need to use variables to create the condition for the missing logic. ⑬ The function of variables to create the missing condition led to the relation between variable blocks and logic blocks. Anne duplicated the existing logic block to create the missing condition with variables blocks within compare and AND/OR operations. Anne and Judith incorrectly created the condition. ⑭ Then, Anne noticed the structural relation between variables for red and green toys (virtual objects) and the robot's movement (physical object) accordingly.

Cross-Case Analysis Themes

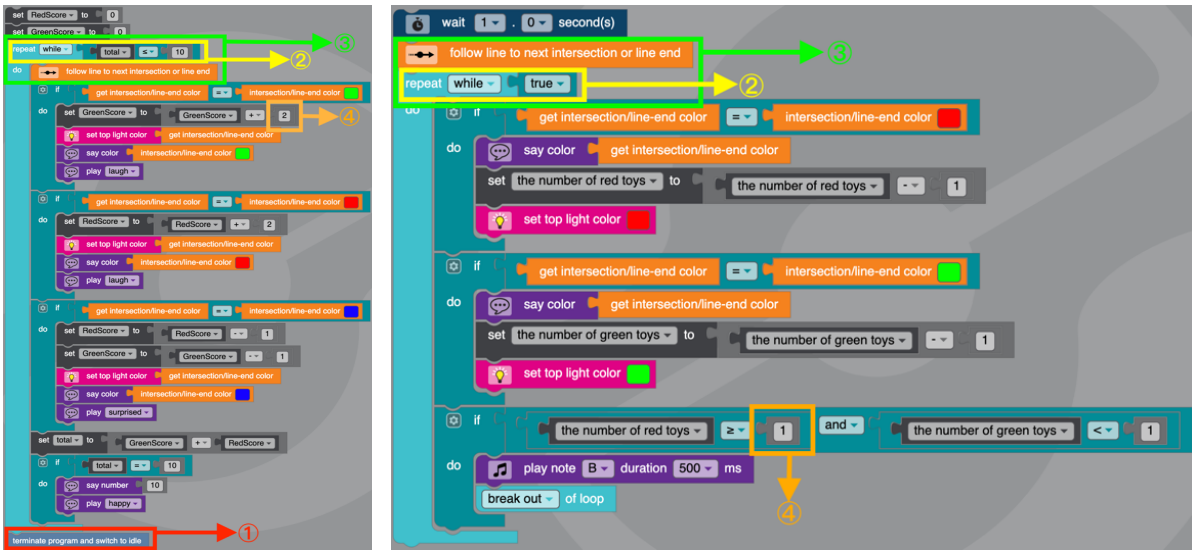
Analogical reasoning began with attending to relational similarities rather than surface similarities

According to prior research, novices' similarity-based retrieval (i.e., analogical retrieval) is usually driven by *surface* similarities whereas experts are more likely to retrieve similarities in relational structure because their encoding process includes more relational knowledge (Gentner et al., 1993, 2013; Holyoak & Koh, 1987; Ross, 1987). On the contrary, in the present study, analogical retrievals were often driven by *structural* similarities. Structural similarities were relational commonalities between the virtual object (i.e., terminate block in code) and physical object (i.e., the robot's movement) as well as between elements within the virtual object (e.g., a relation between the repeat and logic blocks). For

example, when the target task, Cleaning the Playroom, was provided, both pairs referred to the source of their analogy as they identified similarities with the source task. As Judith tried to stop the robot in the target task, she referred to the source task regarding the terminate block (① in Figure 4) by saying “Because it [the robot] doesn't stop, so we have to terminate it [the program] at the end like we did with the last one.” Anne recalled the repeat while/until and compare logic blocks in the source task (② in Figure 4) as she identified the *relational similarity* between the repeat block and the compare logic block next to it, by mentioning “I think it's like one of those times where you do like repeat until the number of red toys is like equal six.”

Figure 4

The Code for Source Task and the Buggy Code for Target Task with Identified Analogies



Source Task
(The Color Game)

Target Task
(The Cleaning the Playroom - buggy code)

Ariana and Kimberly also went through a similar encoding and inferring process. As they attempted to make the robot move continuously on the map in the target task, Ariana referred to the source task regarding the location of the line navigation block (③ in Figure 4) by saying “I'm not sure where to put [the line navigation block]. How does it look, how did

we do it in the last one? I think it's just, this [the line navigation block] needs to go up here." When given a new task, both pairs reminded themselves of analogous experiences from the past debugging task and retrieved similar attributes in terms of code structure and robot movement.

Using virtual and physical objects together seemed to have promoted the pairs' attention to the relational structure since they needed to compare the structure of the code to that of the robot behaviors to debug. While it has been argued that text-based programming inherently facilitates learning of analogical reasoning (Grandgenett & Thompson, 1991; Jang, 1992), there was no discussion on whether and how block-based programming can promote analogical reasoning based on structural similarities. Visually discernable structures in block code and tangible outputs (robot behaviors) may have made structural similarities prominent. Nonetheless, the pairs noticed no *dissimilarities* between the source and target tasks. The pairs may have paid more attention to similarities than dissimilarities in the early phase of the analogical reasoning process considering that similarities between the source and target tasks are used in reducing the perceived complexity of the target task (Ahmed & Christensen, 2009) and acquiring complete knowledge of the elements in an object and their relations (Gentner et al., 1993; Holyoak & Koh, 1987; Novick, 1988).

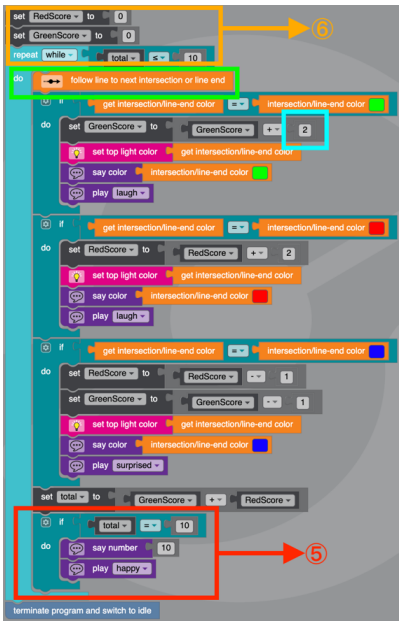
Analogical mapping was often centered around functional similarities between debugging tasks

The pairs demonstrated high structural consistency within each analogical mapping in which one element in the source task was matched with at most one element in the target task, not with multiple elements (Gentner & Smith, 2013). As illustrated in Figure 5 below, this also involved parallel connectivity. These findings align with the literature showing that people tend to keep structural consistency in the mapping (e.g., Spellman & Holyoak, 1992). The pairs' mapping also showed high systematicity (Clement & Gentner, 1991) in which

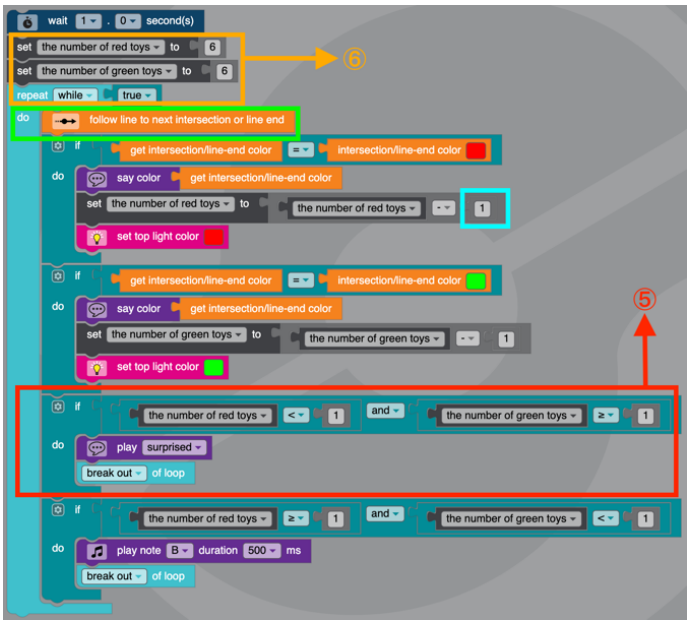
commonalities in the structure that they found also entailed a deeply connected system of relations. For example, the same causal patterns between the code (virtual object) and the robot's movement (physical object) were applied in both debugging tasks.

Figure 5

Parallel Connectivity between the Code for Source Task and the Correct Code for Target Task and Identified Analogies



Source Task
(Color Game)



Target Task
(Cleaning the Playroom - correct code)

For example, both pairs identified similarities associated with the function of the line navigation block from the source task, Color Game, and based on its relationship with other surrounding blocks (i.e., structural similarity in the virtual object) as well as the relationship with the robot behaviors (i.e., structural similarity in the virtual and physical object). The pairs recognized how the repeat block functioned in relation to the line navigation block and the robot behaviors, as shown in the following discourse.

> Ariana: (06:18) Oh, it [the robot] follows the line to the next intersection or line end *(She noticed the relation between the robot and line navigation block)*. So, it [the robot] would stop at that intersection *(She noticed the function of the line navigation block, allowing the robot*

to move on the map until the next intersection). So, it's the first part [of the code where] we made a mistake. So, I can choose, read all of these.

Kimberly: (06:38) Should it be this follow line [block]?

Ariana: (06:53) I'm not sure where to put [the line navigation block]. How does it look, how did we do it in the last one [task]? I think it's just… This [line navigation block] needs to go up here *(She noticed the relation between the line navigation and repeat blocks, requiring the line navigation block to be within the repeat block. See ③ in Figure 4)*.

Kimberly: (07:01) Which one? This? *(She pointed out the line navigation block located out of the repeat block)*

Ariana: (07:03) Yeah. I think it [the line navigation block] needs to get moved from the top into the repeat block, I think. *(She moved the line navigation block from out of the repeat block into the repeat block)* There we go. *(They ran the robot and confirmed that the robot moved continuously on the map)*

Both pairs also noticed that the code lacked a chunk of blocks making the robot get surprised and stop when no more red toys were left. They determined that a chunk of logic blocks was needed. As shown below, their reasoning was based on the similarity in the function of the logic block, the relational similarity between the logic and loop blocks, and the logic block and robot movement used in both the source task, Color Game, and the current target task.

Ariana: (13:10) So, when Ozobot has no more green toys left, but there's no more red toys left for some reason [the Ozobot] got surprised and stopped. *(She read the code and the task description)* So it [the robot] didn't get surprised and stop 'cause it [a chunk of logic blocks] is not in there. *(She noticed the function of the logic block that caused the robot to get surprised and stopped in the loop block)*

Kimberly: (13:20) Yeah. So.

Ariana: (13:22) We need to go to that if [logic] one. So I can just put... *(She checked the logic block category and added if/do block to the code)* I guess it doesn't really matter. It's, because as long as it, as long as it [a chunk of logic block] is in the loop. *(She noticed the relational commonality between the logic and loop blocks and decided to put the if/do block within the loop block. See ⑤ in Figure 5)* Let's put it [logic block] there, and then if, I guess do the same thing. We'll have to make, or wait, no.

Kimberly: (14:07) Oh that's not. *(The pair started building blocks for the missing chunk of the logic blocks. They added a math block but decided that it is not the correct block here)*

Ariana: (14:07) The number, if the number. This way has two seconds. If the number of green toys is greater than one, and then we have to add

on like that other part. *(The pair checked the variable blocks in the variable category to add the variables into the newly added if/do logic block)*

Ariana: (14:34) Oh it [compare logic block with 'the number of green toys' variable block] might have to go into that big thing [AND/OR operation block]. Might have to put it on there [in the logic block]. *(She noticed that the variable block needs to be in the compare logic block within the newly added if/do logic block. See ⑤ in Figure 5)*

Ariana: (14:56) The number of red toys is less than one. Just don't move the positioning. Do it, it [the robot] will get surprised and stop.

Kimberly: (15:31) Do we need this? Like this. *(The pair continued to build the newly added if/do logic block with 'the number of red toys' variable block and other blocks required to be in it)*

Ariana: (15:46) Let's just try putting in get surprised and stops. So, it [the robot] can make a surprise noise and stop. I don't know if there's like a stop. And add a break out of loop [block]. *(She highlighted the relation between the code and the robot)*

The pairs also identified similarities regarding the variable blocks between the Color Game task and the Cleaning the Playroom task. Specifically, they recognized how the variable block functioned in relation to the other blocks (e.g., the repeat until block and the if-do logic block) and the robot behaviors as shown in the discourse below.

Judith: (08:27) Or above here? *(She noticed the missing sound block after reading the task description and added the play happy sound block between two logic blocks)* We need it to be like in a variable. *(She noticed that there is a missing logic block consisting of a variable block to place the sound block. See ⑤ in Figure 5)*

Anne: (08:36) Exactly. That's why I thought here let's duplicate this [the existing if/do logic block consisting of play note sound block], so I thought it'd be like this [with the added missing logic], right? I hate that it does this. *(She commented on her relocating a block that caused also moving other connected blocks)* And then it'd be like this because I'm putting these two *(She changed the variables in the logic block including the play happy sound block).* 'Cause it's one of [if/do logic blocks] saying *(She read the code of the logic blocks)* like if there are more green toys than red toys, then it [the robot] plays the note. And this one [the other if/do logic block] is saying if there's more red toys and green toys, then it [the robot] would be happy or no, it [the code of the if/do logic blocks] should be the other way around. Because if there are green toys left. *(She noticed that the selection of variables for different colors needs to change in the logic block based on the robot's behaviors)* Why is it [putting correct variables into the logic blocks] so hard?

Anne: (11:30)	No, it's like six toys or something. So the number of green toys.. repeat while [block] something, like [the number of] green toys [variable] is like six or something. This is annoying me.
Anne: (12:56)	I think it's like one of those times where you do repeat until the number of red toys is like equal six. *(She noticed the relational commonality between the repeat while/until and variable blocks. She reminded Judith of the previous task by saying "one of those times". See ② in Figure 4)*
Judith:	(13:10) Change while, change until… *(She changed the repeat while block to repeat until block)*

Among the similarities the pairs identified, some were unrelated to fixing the bugs in the current target task. For example, Judith recognized a similarity in the use of math blocks. This noticing was based on her mapping of a single object (i.e., the numeric value in the math block) in the code between tasks. The high level of structural consistency without considering systematicity led this pair to mapping on an irrelevant single object between tasks. The mapping without considering the relation of the numeric value with other blocks or with the robot behaviors led to proposing a change in a block without a bug as shown below.

Judith: (07:10)	Less than 1... What if we make it less than or equal to …? Oh, wouldn't really work. What if you make less than or equal to 2 instead of 1? *(She noticed the similarity in the value in the math block and offered to change the value. See ④ in Figure 4)*
Anne: (07:20)	This one? *(She pointed out the value 1 in the if/do logic block for 'the number of green toys' variable)*
Judith: (07:21)	Yeah, 'cause they're supposed to be out of or no, red. Some red toys are supposed to be to, yeah, maybe make that, too. *(She offered to change the value 1 in the if/do logic block for 'the number of red toys' variable)* I don't know. It's probably not gonna work. It's probably a waste of time.
Anne: (07:32)	You wanna try it?
Judith: (07:39)	No, I don't really care. I'm pretty certain it [changing value 1 to 2 in the math blocks] won't work. So just spitting out some options.

Judith pinpointed the similarity in the terminate block based on its function and relationship with other blocks and the robot behaviors as shown in the following debugging segment. However, the similarity was irrelevant to the bug in the target task. Still, Judith's mapping based on an irrelevant similarity indicated that certain selection criteria, such as debugging goal, was considered for generating inferences.

Anne: (11:13)	Wait, there's something wrong with this code, 'cause it [the robot] won't stop, but it [the robot] should stop eventually. We need to put something.
Judith: (11:30)	Terminate at the end? *(She noticed the similarity in the function of the terminate block in the Cleaning the Playroom with that in the Color Game task. See ① in Figure 4)*
Anne: (11:30)	No, it's like six toys or something. So, the number of green toys… repeat while something, like green toys is like six or something. *(She wanted to use 'the number of green toys' variable in the code, and then she did not put the variable in the code)*
Judith: (12:18)	I don't know what to do. Um, it [the robot] doesn't stop. Because it [the robot] doesn't stop, so we have to terminate it [the program] at the end like we did with the last one [task]. (*She highlighted the robot's behavior and reminded Anne of the previous task, Color Game. See ① in Figure 4)*
Anne: (12:30)	We have to set it [the variable] to like... *(After checking the previous task, she wanted to set the variable to a number next to the repeat while block)*
Judith: (12:31)	Should we send it [the terminate block] to the end?
Anne: (12:35)	Okay. But it [the code] is like repeats, and indefinitely that's the thing. Like it [the code] doesn't ever say like repeat until blah, blah, you know
Judith: (12:54)	What if we still get the bottom here? *(Judith offered to put the terminate block at the end of the code)* I don't know [if] he [the robot] [is] supposed to do this.

The pairs' mapping also involved analogical inference in that the information they used in their mapping was selective. The pairs made analogical inferences based on certain selection criteria to navigate and complete the most accurate and useful mapping. According to Gentner and Smith (2013), the most important selection criterion (i.e., constraint) for candidate inferences is the systematicity and structural consistency as mentioned above. Holyoak and Thagard (1989) took a more pragmatic perspective and argued that purpose/goal relevance is also an important criterion, and all these different kinds of constraints interact to determine the optimal set of correspondences between the source and target tasks. The finding of the present study is aligned with Holyoak and Thagard's (1989) perspective given that the pairs' mapping was driven by their goal of debugging. Despite abundant structural and relational similarities between tasks, the pairs did not make one-to-one correspondence on elements they saw as irrelevant to their goal. Their mapping was centered around certain

blocks of the code that were only relevant to their debugging goals (i.e., the line navigation block, logic block, set variable block) even when their view was inaccurate. This seems to have resulted from analogical inferencing processes during their mapping based on such criteria as structural consistency, systematicity, and debugging purpose. Still, these inferences had to be tested in debugging.

The pairs used all possible analogies in debugging

Both pairs applied *all* analogical inferences that they had mapped and were relevant to debugging in the current code. This finding is in contrast to the literature showing that reasoners struggle applying analogs and often apply relatively adaptable analogs (Keane, 1996; Novick & Holyoak, 1991). Regardless of debugging outcomes, all *relevant* similarities from analogical mapping were applied (i.e., analogies regarding the line navigation, logic, and variable blocks) in the pairs' debugging attempts. For example, Ariana and Kimberly used similarities with the line navigation block (i.e., relocating the line navigation block from outside to inside of the loop block) as well as the loop and logic blocks correctly (i.e., adding the if/do logic block into the loop block), which were correct attempts leading to successful debugging. They used similar analogies in debugging for the robot to continuously follow lines on the map and get surprised and stop when no more red toys were left.

(Discourse on using the line navigation block)
Ariana: (07:03) Yeah. I think it [the line navigation block] needs to get moved from the top into the repeat block, I think. There we go. *(The pair had used the relevant similarity in relational commonality between the line navigation and repeat blocks. See ③ in Figure 4)*

(Discourse on using the loop and logic blocks)
Ariana: (13:10) So, when Ozobot has no more green toys left, but there's no more red toys left for some reason for green toys get surprised and stopped so it [the robot] didn't get surprised and stops 'cause it [a chunk of logic block] is not in there.
Kimberly: (13:20) Yeah. So.
Ariana: (13:22) We need to go to that if [logic] one. So, like that yeah, I can just put... I guess it doesn't really matter. It's, because as long as it, as long as it [a chunk of logic block] is in the loop. Let's put it [logic

> block] there, and then if, I guess do the same thing. we'll have to make, or wait, no. *(The pair applied the relevant similarity in relational commonality between the repeat and logic blocks. See ⑤ in Figure 5)*

Similarly, Anne and Judith used the similarity with the line navigation block and the loop block from the source task correctly, which led them to fix one of the bugs and made the robot continuously follow the line on the map.

> Judith: (07:43) What is your solution to the problem? We will stick the line navigation block underneath the repeat block so that it [the robot] will follow the whole grid [on the map] and not stop after one block [grid on the map]. *(The pair used the relevant similarity in relational commonality between the line navigation and repeat blocks. See ③ in Figure 4)*

Ariana and Kimberly attempted to use similarity with the set variable and loop blocks. Although they added set variable blocks first to the code with only the number of green toys variable at the top, the pair incorrectly relocated the set blocks into the loop block. After revisiting the source task and reviewing the blocks used in the task, the pair relocated the set block out of the loop block in the target task, which was a correct attempt to debug the code.

> Ariana: (26:23) Does it [the set blocks with value 6] have to go into the repeat thing [block]? (*The pair had noticed the relevant similarity in relational commonality between set variable and repeat blocks. See ⑥ in Figure 5. She relocated the set blocks into the loop block)*
> Kimberly: (27:18) Okay. *(She put the set blocks out of the repeat block again)* Let's see. Green toys are 6 and 6 *(She read the set variable blocks)*

Anne and Judith also tried to use the similarity with the variable block and the logic block from the source task but were not able to apply it correctly to the target task, which led to unsuccessful debugging. Anne and Judith's attempt using analogy (i.e., adding the missing logic block with other required blocks in it, such as math, compare logic, AND/OR operation, play surprised, and break out of loop blocks) was close to fixing the bug in the target task. However, their attempt did not lead to fixing the bug due to their incomplete understanding

of the given problem. This finding is aligned with that of the previous studies in which the successful application of analogy for the desired outcome depended on an accurate understanding of the given problem (e.g., Ahmed & Christensen, 2009).

The adaptability afforded by using block code and robots seems to have facilitated the pairs' experiments with using all analogies. That is, these tools made analogic inferences more adaptable to the target task, which allowed pairs to notice and apply relevant similarities in blocks and/or the robot as needed. By clicking, dragging, and dropping in the block code, they were able to apply the noticed analogy from the source task to the target task. According to previous studies, even when reasoners notice correspondence between two tasks, they often struggle to apply them to the target task and usually use analogical inferences that are relatively adaptable to the target task (Keane, 1996; Novick & Holyoak, 1991). The findings of the present study suggest the role of block-based robot programming in facilitating analogical reasoning.

The pairs evaluated their applied analogies

The pairs evaluated their analogy application. This finding is unique in that it is usually hard to immediately identify whether the analogical inference is true unless the outcome of the analogy application is immediately testable (Gentner & Smith, 2013). In the present study, the outcome was readily observable in the robot behaviors. For example, after applying the similarity in the line navigation block to the target task, both pairs went over the process of verifying their analogy by testing the revised code and confirming if the robot performed desired behaviors in relation to the line navigation block. As shown in the following example from Anne and Judith's excerpt, they verified the goal of making the robot move continuously on lines on the map.

> Anne: (08:56) It's okay. We're trying our best. *(They loaded the revised code to the robot and ran it on the map after they changed the location of the line navigation block from outside to inside of the repeat block)*

Anne: (10:05) Success, right? *(She verified the goal of making the robot move continuously on the map)*

The pairs also revised the code again when they saw that the analogy application did not lead to the desired robot behavior. Both pairs revised the code after applying the analogy related to missing logic blocks. Anne and Judith applied the analogy between the logic and variable blocks and Ariana and Kimberly applied the analogy between the loop and logic blocks. However, Anne and Judith tested their applied analogy and observed that the robot did not move properly, but they did not refine their applied analogies for the terminate, repeat until, and variable blocks. Ariana and Kimberly tested their applied analogy for the missing logic and set blocks, and revised their code. Ariana and Kimberly continued to work on their code after applying and testing the identified analogy to debug.

Ariana: (18:29) *(They added a chunk of logic block for making the robot get surprised and stop, loaded the revised code to the robot and ran it on the map)* I don't know why it [the robot] is not picking up what that said. (*She highlighted the robot not behaving as coded in the variable blocks in the logic block)*

Kimberly: (18:48) Yeah.

Ariana: (18:49) I don't know if that should be equal. *(They changed the mathematical operation from > 1 to = 0 for the number of green toys in the compare logic block)*

Anne and Judith refined the code based on their evaluation of the analogy. For example, they attached the compare logic block to the repeat while block. After testing the revised code, they changed the mathematical symbol in the compare logic block. They also tested the newly added logic block and deleted it after seeing that the robot did not perform properly.

Anne: (00:03) So let's try this bad boy [the robot]. Oh. *(They applied the similarity in relation between the variable and repeat while/until blocks. They added compare logic blocks with variables =< 6 instead of the true block connected to the repeat while block. They tested the revised code)*

Judith: (00:55) Is that the B noise? *(She commented on the sound coming from the robot coded to play B note)* Oh, that's not good. Because of the terminate [block]? (*She pointed out the terminate block at the end*

	causing the robot not to move after the first intersection on the map. She deleted the terminate block)
Anne: (01:02)	I think there's something wrong up here. *She pointed out the compare logic blocks connected to the repeat while block)* Green toys equals six? (She changed =< 6 to = 6 in the compare logic blocks) Oh my god. I'll calibrate again.

Evaluation of analogy application involves a process where reasoners check if their mapping and inferences are true and thus lead to the desired outcome. It is usually hard to immediately identify whether the inference is true unless the outcome of the analogy application is immediately testable and confirmed in an observable way (Gentner & Smith, 2013). Immediate verification through robot behaviors does not mean that the verification process always led to successful debugging. Although Ariana and Kimberly used the analogy with the set variable and loop blocks correctly, they revised their code and moved the set blocks to the loop block, as described the third theme above. Anne and Judith showed a similar episode in terms of analogies with the variable block and two unrelated similarities in the math block and the terminate block, as described in the second theme above. They fixated on their initial ideas as to those blocks with a lack of verification, which led to unsuccessful debugging. This aligns with literature indicating that fixation has a reverse effect on the evaluation of applied analogies (Cheong et al., 2014). Anne and Judith spent most of their time on analogical mapping and applying rather than verifying their analogy application in comparison to Ariana and Kimberly who spent more on the verification process and refining the code according to the verification outcome.

General Discussion

We examined how novice programming learners used analogical reasoning in debugging that involved both code (virtual object) and robots (physical object). The two pairs exhibited analogical reasoning using structural similarities in contrast to prior research in which novices focused on surface similarities in non-CS design or problem solving contexts

(Ahmed & Christensen, 2009; Cheong et al., 2014; Novick, 1988). This finding may be attributed to the nature of the debugging tasks in the present study. Multimodal objects used in debugging seemed to have made structural relations salient in that debugging inherently asked them to map the relation of the code to the robot behaviors. In turn, such mapping may have led to analogical comparisons of the target task to the source task. The analogical reasoning of the pairs demonstrated structural consistency (Holyoak & Thagard, 1989) and systematicity (Clement & Gertner, 1991). The finding may have resulted also from the analogy applications that were immediately feasible in the code.

Use of multimodal objects during debugging engaged the pairs in the evaluation process of analogical reasoning. Being a tangible object, the robot enabled participants to test immediately after their analogy application. Inference made between source and target tasks can be correct or incorrect to solve the target problem and evaluating the analogy applied in the target task is critical for successful problem solving (Gentner & Smith, 2013). In the present study, both pairs applied the relevant similarity in the relation between the loop and line navigation blocks and verified the analogy application by running the robot on the map, and there was only one case where the evaluation process was not performed after the application of analogy. Although participants were able to engage in more evaluation processes, the evaluation itself was not always accurate enough to verify the appropriateness of the analogical reasoning. Sometimes, they fixated on irrelevant similarities and inaccurately evaluated the analogy application even after testing with the robot.

Another notable finding was that the functional analogies were used effectively along with other foci such as visual or structural analogies in multiple phases of the analogical reasoning process as well as debugging. Noticing similarities in function led to the pairs' discovery of similarities in the relation between block codes as well as between block code and the robot, resulting in finding the bugs in the buggy code. This finding stands in contrast

to the finding in a prior study in which functional analogies prevented identifying relevant analogies (Cheong et al., 2014).

The presence of the robot played a critical role in the overall flow of analogical reasoning foci and forms. Noticing relevant similarities and irrelevant similarities in the relation between the block code and the robot guided the pairs to discover relevant similarities between block code in the target and source tasks. However, noticing a relevant similarity in the relation between the block code and the robot led Anne and Judith to discover an irrelevant similarity in function but Ariana and Kimberly to discover a relevant similarity in function. The flow from relevant or irrelevant relational similarity between the block code and the robot guided to relevant relational similarity but not always to the relevant similarity in function. Since analogy enables reasoners to focus more on relations than objects (Gentner & Smith, 2013), use of the robot with block code enriched the analogical reasoning flow from relational similarity to relational similarity rather than to the objects' functions.

Despite the benefits of block-based coding using multimodal objects for novices' analogical reasoning, the difficulty that the pairs experienced with the role of variables negatively affected their use of related analogies. This finding aligns with a previous finding that successful use of analogy is influenced by relevant domain knowledge (Ahmed & Christensen, 2009). In addition, both pairs did not notice and use any dissimilarity between the source and target tasks, and their analogical reasoning process was based on the similarities. A possible reason is that similar attributes can be perceived without creating additional cognitive load and readily used in reducing the perceived complexity of the target task (Ahmed & Christensen, 2009). Noticing only dissimilarities rarely helps to reduce perceived complexity of the target task, and using dissimilar attributes requires reasoners to integrate new information presented in the target task and to create or modify their schema

accordingly, increasing cognitive load. Reasoners are also required to have concrete knowledge of elements involved in the task and their relations to use the dissimilarities properly in problem solving.

Table S4 in the supplementary highlights the study findings in relation to the existing literature and also implications for computing education research.

Limitations and Suggestions for Future Research

Only two pairs of participants were presented in the paper. However, generalizability was not the purpose of the paper. The qualitative method was the most suitable for our research goal: to uncover novice programming learners' analogical reasoning process in a naturalistic setting. A qualitative approach is suitable when a straight-forward description of the phenomenon is desired (Lambert & Lambert, 2012). Such an approach can also generate new insights and generate conceptual frameworks, theories, and hypotheses (Kramer, 1985). Our approach was designed to study participants in their natural state to the extent possible within the context of the research area (Lambert & Lambert, 2012). Moreover, descriptive research should be viewed as an initial stage leading to development of new knowledge (Kramer, 1985). As the study of analogical reasoning in block-based programming is still in its infancy, this article sets the foundation for more future research on analogical reasoning in block-based programming.

Further research needs to examine strategies to help novice programming learners leverage analogical reasoning in block-based programming. For example, providing relevant retrieval cues could facilitate the phase of structuring (Mozzer & Justi, 2012). Offering scaffolding tools to help them evaluate their analogy application could be studied in future studies. The ultimate goal is to create a safe learning environment that encourages students to generate and explain their analogical reasoning processes.

Implications for Educational Computing Research

Though this study is not meant to be generalizable, it does indicate that novice programmers have the potential to engage in analogical reasoning focusing on structural similarities between the source and the target task. Doing so has the potential to help novice programmers learn to debug effectively. Some conditions that helped the current participants do so include the multimodal nature of the objects with which they interacted. Specifically, it helped them to notice structural elements in the code and link such to robot behaviors. Next, robot behavior lent a concrete manifestation of the code that allowed participants to link code fragments to robot behavior and to immediately test revisions to the code. These findings call for further research in computing education toward culturally responsive design that enables multimodal analogical reasoning and debugging (see Table S4 in the supplementary for related implications).

The findings of this study may be applied to debugging within other block-based programming platforms, such as Scratch. There are many commonalities among block-based coding languages, but also some important differences such as the inclusion of functions and Boolean operators (Kraleva et al., 2019). Still, the existence of a physical robot in this study compared to other block-based platforms that have virtual agents may be critical to fostering effective analogical reasoning.

This study also points to the potential of the inclusion of robots within early childhood education. If preservice early childhood teachers can program and debug, then they have the potential to teach with robots. Of course, skill is only one part of the equation, and much work needs to examine the dispositions and motivation to teach with robots among preservice, early childhood teachers.

Acknowledgments

This research is supported by grants 1927595 and 1906059 from the National Science Foundation. Any opinions, findings, or conclusions are those of the authors and do not necessarily represent official positions of the National Science Foundation.

References

Aggarwal, A., Gardner-McCune, C., & Touretzky, D. S. (2019). Evaluating the effectiveness of explicit instruction in reducing program reasoning fallacies in elementary level students. *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, 292. https://doi.org/10.1145/3304221.3325588

Ahmed, S., & Christensen, B. T. (2009). An in situ study of analogical reasoning in novice and experienced design engineers. *Journal of Mechanical Design*, *131*(11). https://doi.org/10.1115/1.3184693

Araki, K., Furukawa, Z., & Cheng, J. (1991). A general framework for debugging. *IEEE Software*, *8*(3), Article 3. https://doi.org/10.1109/52.88939

Barnett, S. M., & Ceci, S. J. (2002). When and where do we apply what we learn?: A taxonomy for far transfer. *Psychological Bulletin*, *128*(4), 612–637. https://doi.org/10.1037//0033-2909.128.4.612

Basawapatna, A. (2016). Alexander meets Michotte: A simulation tool based on pattern programming and phenomenology. *Journal of Educational Technology & Society*, *19*(1), 277–291.

Bogdan, R.C., & Biklen, S. K. (1992). *Qualitative research for education: An introduction to theory and methods*. Allyn & Bacon.

Bransford, J. D., & Schwartz, D. L. (1999). Rethinking transfer: A simple proposal with multiple implications. *Review of Research in Education*, *24*, 61–100. https://doi.org/10.2307/1167267

Burstein, M. H. (1983). Concept formation by incremental analogical reasoning and debugging. *Proceedings of the International Machine Learning Workshop*, 19–25.

Burstein, M. H. (1988). Combining Analogies in Mental Models. In D. H. Helman (Ed.), *Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science, and Philosophy* (pp. 179–203). Springer Netherlands. https://doi.org/10.1007/978-94-015-7811-0_9

Cao, Y., Porter, L., & Zingaro, D. (2016). Examining the value of analogies in introductory computing. *Proceedings of the 2016 ACM Conference on International Computing Education Research*, 231–239. https://doi.org/10.1145/2960310.2960313

Chai, C., Cen, F., Ruan, W., Yang, C., & Li, H. (2015). Behavioral analysis of analogical reasoning in design: Differences among designers with different expertise levels. *Design Studies*, *36*, 3–30. https://doi.org/10.1016/j.destud.2014.07.001

Cheong, H., Hallihan, G., & Shu, L. H. (2014). Understanding analogical reasoning in biomimetic design: An inductive approach. In J. S. Gero (Ed.), *Design Computing and Cognition '12* (pp. 21–39). Springer Netherlands. https://doi.org/10.1007/978-94-017-9112-0_2

Clement, C. A., & Gentner, D. (1991). Systematicity as a selection constraint in analogical mapping. *Cognitive Science*, *15*(1), 89–132. https://doi.org/10.1016/0364-0213(91)80014-V

Clement, C. A., Kurland, D. M., Mawby, R., & Pea, R. D. (1986). Analogical reasoning and computer programming. *Journal of Educational Computing Research*, *2*(4), 473–486. https://doi.org/10.2190/DFH5-E0PG-1ML4-M34J

Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. *Journal of Educational Computing Research*, *3*(1), 73–94. https://doi.org/10.2190/RCNV-2HYF-60CM-K7K7

Creswell, J. W. (2013). *Qualitative inquiry research design: Choosing among five appraoches* (3rd ed.). SAGE Publications, Inc.

Gentner, D., Özyürek, A., Gürcanli, Ö., & Goldin-Meadow, S. (2013). Spatial language facilitates spatial cognition: Evidence from children who lack language input. *Cognition*, *127*(3), 318–330. https://doi.org/10.1016/j.cognition.2013.01.003

Gentner, D., Rattermann, M. J., & Forbus, K. D. (1993). The roles of similarity in transfer: Separating retrievability from inferential soundness. *Cognitive Psychology*, *25*(4), 524–575.

Gentner, D., & Smith, L. A. (2013). *Analogical learning and reasoning*. Oxford University Press. https://doi.org/10.1093/oxfordhb/9780195376746.013.0042

Gentner, D., & Stevens, A. L. (Eds.). (2014). *Mental models*. Psychology Press.

Gero, J. S., & Kannengiesser, U. (2004). The situated function–behaviour–structure framework. *Design Studies*, *25*(4), 373–391. https://doi.org/10.1016/j.destud.2003.10.010

Gero, J. S., & Kannengiesser, U. (2014). The function-behaviour-structure ontology of design. In A. Chakrabarti & L. T. M. Blessing (Eds.), *An Anthology of Theories and Models of Design* (pp. 263–283). Springer London. https://doi.org/10.1007/978-1-4471-6338-1_13

Gero, J. S., & Mc Neill, T. (1998). An approach to the analysis of design protocols. *Design Studies*, *19*(1), 21–61. https://doi.org/10.1016/S0142-694X(97)00015-X

Grandgenett, N., & Thompson, A. (1991). Effects of guided programming instruction on the transfer of analogical reasoning. *Journal of Educational Computing Research*, *7*(3), 293–308. https://doi.org/10.2190/CKGG-EKP5-34YW-YKHB

Holyoak, K. J., Junn, E. N., & Billman, D. O. (1984). Development of analogical problem-solving skill. *Child Development*, *55*(6), 2042–2055. https://doi.org/10.2307/1129778

Holyoak, K. J., & Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition*, *15*(4), 332–340. https://doi.org/10.3758/BF03197035

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, *13*, 295–355.

Holyoak, K. J., & Thagard, P. (1995). *Mental leaps: Analogy in creative thought* (pp. xiii, 320). The MIT Press.

Ichinco, M., Harms, K., & Kelleher, C. (2017). Towards understanding successful novice example use in blocks-based programming. *Journal of Visual Languages and Sentient Systems*, *3*(1), 101–118. https://doi.org/10.18293/VLSS2017-012

Jang, Y. (1992, April). *Cognitive transfer of computer programming skills and analogous problem solving*. The American Educational Research Association Annual Meeting, San Francisco, California, USA. https://eric.ed.gov/?id=ED350981

Jonassen, D. H. (2011). *Learning to solve problems: A handbook for designing problem-solving learning environments*. Routledge.

Katz, I. R., & Anderson, J. R. (1987). Debugging: An analysis of bug-location strategies. *Human-Computer Interaction*, *3*(4), 351.

Keane, M. T. (1996). On adaptation in analogy: Tests of pragmatic importance and adaptability in analogical problem solving. *The Quarterly Journal of Experimental Psychology*, *49*(4), 1062–1085.

Kraleva, R., Kralev, V., & Kostadinova, D. (2019). A methodology for the analysis of block-based programming languages appropriate for children. *Journal of Computing Science and Engineering*, *13*(1), 1–10. https://doi.org/10.5626/JCSE.2019.13.1.1

Kramer, R. F. (1985). A overview of descriptive research. *Journal of the Association of Pediatric Oncology Nurses*, *2*(2), 41–45. https://doi.org/10.1177/104345428500200208

Kurland, D. M., Pea, R. D., Clement, C., & Mawby, R. (1986). A study of the development of programming ability and thinking skills in high school students. *Journal of Educational Computing Research*, *2*(4), 429–458. https://doi.org/10.2190/BKML-B1QV-KDN4-8ULH

Lambert, V. A., & Lambert, C. E. (2012). Qualitative descriptive research: An acceptable design. *Pacific Rim International Journal of Nursing Research*, *16*(4), 255–256.

Lobato, J. (2003). How design experiments can inform a rethinking of transfer and vice versa. *Educational Researcher*, *32*(1), 17–20. https://doi.org/10.3102/0013189X032001017

Lobato, J. (2006). Alternative perspectives on the transfer of learning: History, issues, and challenges for future research. *Journal of the Learning Sciences*, *15*, 431–449. https://doi.org/10.1207/s15327809jls1504_1

Magana, A. J., Vieira, C., Fennell, H. W., Roy, A., & Falk, M. L. (2020). Undergraduate engineering students' types and quality of knowledge used in synthetic modeling. *Cognition and Instruction*, *38*(4), 503–537. https://doi.org/10.1080/07370008.2020.1792912

McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: A review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67–92. https://doi.org/10.1080/08993400802114581

Mozzer, N. B., & Justi, R. (2012). Students' Pre- and Post-Teaching Analogical Reasoning When They Draw their Analogies. *International Journal of Science Education*, *34*(3), 429–458. https://doi.org/10.1080/09500693.2011.593202

Muller, O., & Haberman, B. (2008). Supporting abstraction processes in problem solving through pattern-oriented instruction. *Computer Science Education*, *18*(3), 187–212. https://doi.org/10.1080/08993400802332548

Nersessian, N. (2008). *Creating scientific concepts*. MIT Press.

Nokes-Malach, T. J., & Mestre, J. P. (2013). Toward a model of transfer as sense-making. *Educational Psychologist*, *48*, 184–207. https://doi.org/10.1080/00461520.2013.807556

Novick, L. R. (1988). Analogical transfer, problem similarity, and expertise. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *14*(3), 510–520. https://doi.org/10.1037/0278-7393.14.3.510

Novick, L. R., & Holyoak, K. J. (1991). Mathematical problem solving by analogy. *Journal of Experimental Psychology*, *17*(3), 398–415. https://doi.org/10.1037/0278-7393.17.3.398

Perkins, D. N., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. *Papers Presented at the First Workshop on Empirical Studies of Programmers on Empirical Studies of Programmers*, 213–229. http://dl.acm.org/citation.cfm?id=21842.28896

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*(2), 137–172. https://doi.org/10.1076/csed.13.2.137.14200

Ross, B. H. (1987). This is like that: The use of earlier problems and the separation of similarity effects. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *13*(4), 629–639.

Ruppert, M. (2013). Ways of analogical reasoning—Thought processes in an example based learning environment. *Eighth Congress of European Research in Mathematics Education*, 6–10.

Salomon, G., & Perkins, D. N. (1989). Rocky roads to transfer: Rethinking mechanism of a neglected phenomenon. *Educational Psychologist*, *24*, 113–142. https://doi.org/10.1207/s15326985ep2402_1

Spellman, B. A., & Holyoak, K. J. (1992). If Saddam is Hitler then who is George Bush? Analogical mapping between systems of social roles. *Journal of Personality and Social Psychology*, *62*(6), 913–933. https://doi.org/10.1037/0022-3514.62.6.913

Spinellis, D. (2018). Modern debugging: The art of finding a needle in a haystack. *Commun. ACM*, *61*(11), 124–134. https://doi.org/10.1145/3186278

Sternberg, R. J. (1977). Component processes in analogical reasoning. *Psychological Review*, *84*(4), 353–378. https://doi.org/10.1037/0033-295X.84.4.353

Sternberg, R. J., & Rifkin, B. (1979). The development of analogical reasoning processes. *Journal of Experimental Child Psychology*, *27*(2), 195–232. https://doi.org/10.1016/0022-0965(79)90044-4