International Society of the Learning Sciences

Discovery of Similarities across Debugging Tasks in Relations within and between Virtual and Physical Objects

ChanMin Kim, Emre Dinç, Eunseo Lee, Afaf Baabdullah, Anna Y. Zhang, Brian R. Belland
cmk604@psu.edu, ezd287@psu.edu, eul374@psu.edu, afb5304@psu.edu, ybz5148@psu.edu, brb288@psu.edu
The Pennsylvania State University

Abstract: Analogical reasoning is critical in debugging. Still, the literature is unclear on how non-CS major novice programming learners draw inferences from previous debugging tasks when attempting to understand the present debugging task. We addressed the research question *How do novice programming learners discover relational commonalities across debugging tasks in which block-code contains structural relations within itself and with a corresponding robot?* We used a theory-informed coding scheme to analyze interviews and screencasting data. We found that (a) noticing similarities in *function* between the target task and the source task(s) guided learners to discover *relational commonalities within block code* between the source task(s) and the target task, (b) functional analogy was not necessary for discovery of every relational commonality between the target task and the source task in block code, and (c) noticing and using relational commonalities did not always lead to successful debugging.

Introduction

Strategies to facilitate analogical mapping between a source context and a target context have been studied in many domains. In programming learning contexts, strategies tend to involve analogies connecting unfamiliar programming concepts and rules to familiar ones (Muller & Haberman, 2008). But there has been less attention paid to how non-CS majors draw inferences from their previous debugging task when attempting to understand the present debugging task. Doing so requires that multiple relations be highlighted to make analogical comparisons within debugging. That is, comparisons are not just between one structure (of a source task) to another structure (of a target task) (e.g., leaves : a tree :: petals : a flower) but also for one set of structures from multiple relations within a source debugging task to another set of structures from multiple relations within a target debugging task. In this study, debugging tasks contain both virtual and physical objects and their relations. For example, comparing *block code* in a source debugging task to *block code* in a target debugging task is part of analogical reasoning, but so is comparing *relations between block code and the robot* in a source debugging task to *relations between block code and the robot* in a target debugging task. As shown in the figures below, there are common relational structures between virtual objects, Figure 1 (the source) and Figure 2 (the target). The relation between the repeat # times block A and the movement blocks B is that B is repeated # times indicated in A. There are 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 (the robot moves # times depending on the parameter in the repeat function), which also creates parallel connectivity between the source debugging task and the target debugging task.

Figure 1
Source Task (Tracing a Square Once)

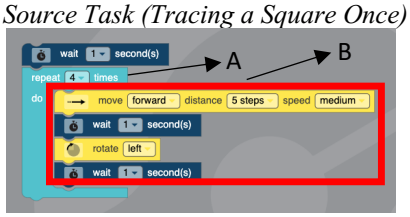
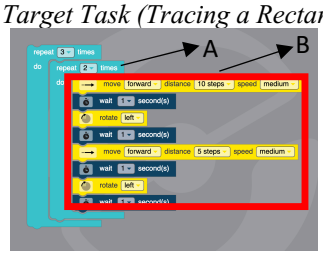


Figure 2
Target Task (Tracing a Rectangle 3 Times)



Research shows that people can recognize relational commonalities even with "conflicting object matches" between two tasks (Gentner & Smith, 2013, p. 4). For example, when given one picture showing a tow truck towing a car and the other picture showing a car towing a boat, people are highly likely to choose the boat in the second picture as a corresponding object to the car in the first picture due to the common structural relation between towing and being towed (Gentner & Smith, 2013). Block code in the present study was not an object as familiar as daily objects like vehicles to our participants and also contained symbolic, visual, and material forms. Thus, our research question was: *How do novice programming learners discover relational commonalities across debugging tasks in which block-code contains structural relations within itself and with a corresponding robot?*

Conceptual framework

Our conceptual framework was grounded in the analogical reasoning (e.g., Gentner & Smith, 2013) and computer programming (e.g., Clement, Kurland, Mawby, & Pea, 1986) literatures but also design studies that investigated analogical reasoning during design (e.g., Ahmed & Christensen, 2009; Chai, Cen, Ruan, Yang, & Li, 2015; Cheong, Hallihan, & Shu, 2014; Gero & Kannengiesser, 2004). Our framework includes three perspective angles. Analogical reasoning *processes* include the encoding, inferring, mapping, applying, and verifying phases (Gentner & Smith, 2013; Sternberg & Rifkin, 1979). Analogical reasoning *foci* include visual, functional, behavioral, and structural analogies (Chai et al., 2015; Gero & Kannengiesser, 2004). Analogical reasoning *forms* include analogical comparisons in virtual objects, physical objects, and relation between virtual and physical objects. Our framework defines small, large, and mixed analogical reasoning distances between the source and target tasks, meaning a small distance having greater superficial similarities (Ahmed & Christensen, 2009; Muller, 2005).

Methods

Participants and context

Participants were 19 undergraduates who engaged with a robotics and play unit in a required three-credit play-based activities course in an early childhood and elementary education program of a large public university in the northeastern United States. The unit was for 90 minutes per week for eight weeks. All but two participants had completed field experience at the preschool, kindergarten, and/or lower elementary grades. All but one participant indicated no to little programming knowledge prior to the unit. Both members of the pair reported in this proposal - Judith and Anne - were female and juniors. All names have been changed.

Debugging tasks

Participants were invited to debug a series of buggy code segments given along with descriptions that explained expected behaviors of robots in an early childhood's play and learning context. In total, nine debugging tasks with increasing difficulties were given, five of which were done in class as paired debugging tasks and four of which were done as homework individually. Three practice tasks were given between debugging tasks. In this proposal, we present one debugging task called "Cleaning the Playroom" as the target task (Figure 4), and its similarities and dissimilarities with one of the source tasks called "Color Game" (see Figures 3 and 4).

Figure 3
Code for the Color Game Task

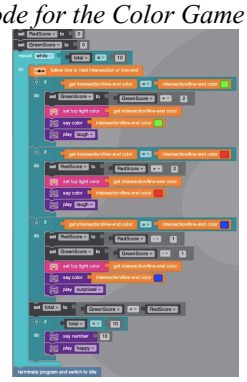
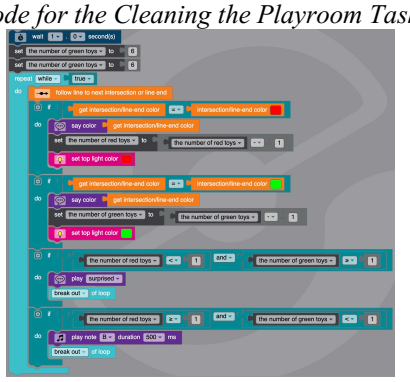


Figure 4
Code for the Cleaning the Playroom Task



Data collection

Participants' computer screens were recorded during debugging. Their reflections and final code were collected during the unit and interviews were conducted after the unit ended. The total length of all screen-recordings was about 33 hours. Anne and Judith's screen-recording was about 3.5 hours. They were also video-recorded.

Data analysis

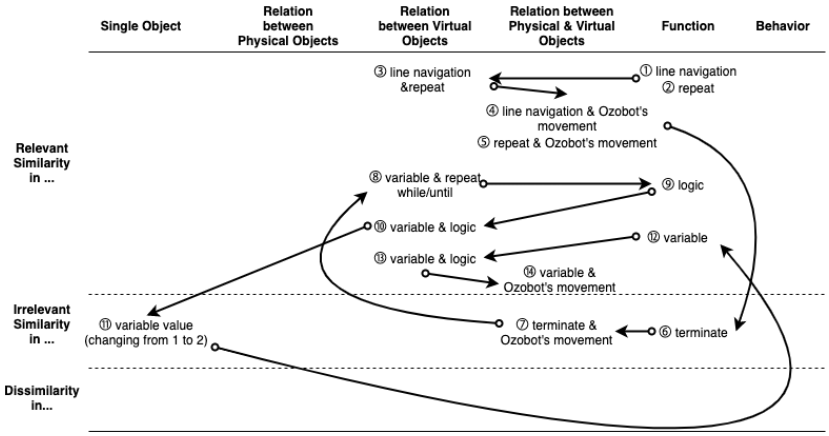
We developed a coding scheme based on our conceptual framework grounded in the analogical reasoning literature (Chai et al., 2015; Clement et al., 1986; Gentner & Smith, 2013; Gero & Kannengiesser, 2004, 2014; Gero & McNeill, 1998; Ruppert, 2013; Sternberg, 1977; Sternberg & Rifkin,1979). The coding scheme included encoding and inferring, mapping, applying, and verifying as high-level nodes. All nodes had sub-nodes for small distance, large distance, and mixed distance. *Mapping* sub-nodes were used to code visual, functional, behavioral, and structural analogies. Applying sub-nodes were used to code correct and incorrect uses of relevant or irrelevant similarities and dissimilarities. We first pilot-tested the coding scheme by applying it to

data from two participants' debugging tasks (Octagon and Hexagon). After coding independently, we met to reach consensus and address discrepancies in coding. Then, we independently coded two other participants' data from debugging the Energy versus Obstacle task. We met again to reach consensus and address discrepancies in coding. Then, we did another round of independent coding for data from Anne and Judith's debugging the Energy versus Obstacle task. Upon completion of the third pilot coding task, we again met to discuss coding, and decided to add mixed distance to the coding scheme. Then, we recoded using the revised coding scheme independently and reached consensus. Finally, we coded Anne and Judith's data from their Cleaning the Playroom debugging in NVivo. The average interrater reliability ICC score was 0.849.

Findings and Discussions

Figure 5

Summary of Anne and Judith's Analogical Comparisons during Debugging



Noticing similarities in *function* between the target task and the source task(s) guided learners to discover *relational commonalities within block code* between the source task(s) and the target task. For example, analogical comparisons that Anne and Judith went through from ① to ② and ③ in Figure 5 depict that they did not begin seeing ③ the *relational commonality* between the target task (the Cleaning the Playroom task) and the source task (the Color Game task) in the structural relation between the line navigation and the repeat blocks until they noticed ① ② the functional commonalities of line navigation and repeat blocks between the target and source tasks. The role of such a functional analogy, in this case, is productive considering that it eventually led to the discovery of ④ ⑤ *relational commonalities* between the block code and the robot in the target and source tasks. 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. Due to symbolic, visual, and material forms used in debugging tasks and 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. It seems natural to use tangible analogies ① ② to visualize analogies that are not immediately visible ③. Table 1 shows the pair's discourse and actions during analogical comparisons.

Functional analogy was not necessary for discovery of every relational commonality between the target and the source tasks in the structure within block code. As shown in ⑧, the participants were able to map the relation between the repeat while and the variable blocks in the target task to that in the source task. Noticing this relational commonality led to attention to ⑨ functional commonalities of logic blocks in the target and source tasks. While functional analogies were used again to understand the logic blocks and their structures in ⑨, as in ①, this could be attributed to less familiarity with logic blocks compared to more familiarity with repeat blocks built through analogical comparison from ① to ⑤.

Noticing and using relational commonalities did not always lead to successful debugging. For example, when Anne and Judith noticed ⑩ the relational commonality between the target and the source tasks in the structural relation between the variable and the logic blocks, they then fixated on ⑪ analogical comparison of the numeric value in the math block between the target and source tasks. This analogy was unrelated to bugs in the target task, thereby leading to studying ⑫ the function of variable blocks.

Implications for research and practice

The present study contributes to advancing the analogical reasoning literature by adding empirical findings of novice programmers' learning processes that involve tasks with complex relational commonalities. It also enriches research on debugging through the lens of analogical reasoning. Its practical contribution is to CS education in that understanding of non-CS majors' reasoning during debugging can be used in broadening participation in CS.

Table 1

Part of Anne and Judith's Debugging Episode

Debugging scene	Source task	Target task	Analogical comparisons
Judith: Why do you think the Ozobot did what it did? Anne: Because this [follow line to next intersection or line end] block wasn't here [in the repeat block] like this if statement (*She pointed out the if/do condition for red intersection to indicate the correct place for the follow line to next intersection or line end block)* ① ②	Line navigation block was within repeat block before logic block, which made the robot move continuously.	Line navigation block was out of the repeat block in the buggy code.	They focused on the location of line navigation block in the target task: outside of repeat block, not inside repeat block. The pair noticed similarities in function of line navigation and repeat blocks between the target and the source tasks ① ②, which made the robot move continuously.
Judith: What is a rule? Anne: but like a movement [line navigation block] before the if statement ③. I don't know.	Line navigation block was within repeat.	Line navigation block should be within the repeat block to make line navigation block work repeatedly between intersections.	They began noticing similarity in relation between line navigation block and repeat block between the target and source tasks ③.
Judith: All right. What if we put a follow line or line navigation block underneath of a repeat block ③ in order for it to follow a grid [on the map]? ④ ⑤ *(She indicated the rule by highlighting the correct place of line navigation block within repeat and its relation with the robot's movement)*	The robot moved on the map because line navigation block was within the repeat block.	The robot did not move on the map because line navigation block was outside of repeat block.	They developed a rule indicating the relationship between line navigation block and repeat block ③. They then noticed similarity in relation between line navigation block within repeat block and the robot's movement on the map in the target and source tasks ④ ⑤.

References

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). doi:10.1115/1.3184693

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.

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). Dordrecht: Springer Netherlands. doi:10.1007/978-94-017-9112-0_2

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.

Gentner, D., & Smith, L. A. (2013). *Analogical learning and reasoning*. Oxford University Press.

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

Muller, O. (2005). Pattern oriented instruction and the enhancement of analogical reasoning. In *Proceedings of the 2005 international workshop on Computing education research - ICER '05* (pp. 57–67). Seattle, WA, USA: ACM Press. doi:10.1145/1089786.1089792

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

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