Moving Between Abstraction Levels by Linking Recursion and Induction

L. Marizza A. Bailey
Arizona State University

Dov Zazkis
Arizona State University

Alison Mirin
University of Arizona

The relationship between mathematical induction (MI) and recursion compels us to ask how we could leverage recursive functions to bolster students' understanding of MI. We describe task-based interviews that utilized concurrent interactions with MI tasks and recursive functions that mirrored those induction tasks via a character-based user-interface. To gain insights into how students' conceptions of MI and recursion co-evolved as they interacted with these tasks, it was necessary to accommodate these multiple concurrent contexts by extending Hazzan's (1999) Reducing Abstraction framework. Our extended framework, called the Navigating Abstraction framework, documents ascending, descending, and transferring abstraction levels across contexts. Viewing the data through this lens allowed us to illustrate the way in which these three mechanisms together play a crucial role in students joint development of their understandings of both recursion and induction.

Keywords: Mathematical Induction, Recursion, Proof, Reducing Abstraction

In this paper, we study how students cope with the abstraction required to understand both mathematical induction (MI) and recursion by leveraging the link between them. Accordingly, we designed and implemented an instructional sequence of parallel recursion-MI tasks. In order to determine the abstraction level at which students understood recursion and MI, it was necessary to extend the Reducing Abstraction framework (Hazzan, 1999, 2003). Our extended framework, called the *Navigating Abstraction* framework, provided a lens for examining how students' understanding of recursion and induction co-evolved as they ascended and descended levels of abstraction as well as transferred levels of abstraction across contexts. We use the term *recursive program* to describe a computer program whose primary purpose is educational, but whose primary mechanism is recursion. Our study addresses the two research questions:

1. How does interaction with a recursive program (which mirrors an MI task) affect students' conception of MI?
2. How does concurrent interaction with both a recursive program and MI task affect student's understanding of the relationship between them?

Literature and Framework

Mathematical Induction involves a logical statement which is applied as a technique for proving statements involving countably infinite sets. In fact, taken separately, the abstract nature of proofs (Davis et al., 2009; Healy & Hoyles, 2000; Knuth et al., 2009), complex logical statements (Durand-Guerrier, 2003; Roh, 2010; Stylianides et al., 2004), and infinity (Davis et al., 2009; Wijeratne & Zazkis, 2015; Dubinsky et al., 2005) have all been shown to entail conceptual hurdles, resulting in a substantial body of literature on MI highlighting obstacles to its conceptualization (Baker, 1996; Brown, 2008; Dubinsky, 1989; Harel & Sowder, 1998). Research on students' understanding of MI (Brown, 2008; Davis et al., 2009; Harel, 2002) suggests that the inductive step may pose an obstacle to understanding MI. Furthermore, findings from these studies indicate that understanding the inductive step of MI requires a transition from perceiving MI as a process relating finitely many cases to interpreting MI as an infinite iterative process (Brown, 2008). Additionally, the statement of MI is a complex conditional statement

regarding the truth value of a predicate function defined on the positive integers, and studies have found that conditional statements are often misinterpreted (Hoyles & Küchemann, 2002; Stylianides et al., 2004).

Given that the Reducing Abstraction framework (Hazzan, 1999, 2003) has been effectively used in both computer science education research (Hazzan & Hadar, 2005; Rich & Yadav, 2020; Sakhnini & Hazzan, 2008) and in mathematics education research (Hazzan, 1999; Hazzan & Zazkis, 2003; Raychaudhuri, 2014), we felt it was well suited for our Computer Science/Mathematics context.

Hazzan's (1999, 2003) framework is built on a variety of interpretations of abstraction that collectively form the basis of the framework. Each interpretation can be applied to examine students' understanding of concepts in pure mathematics, applied mathematics, computation, and data structures. We summarize the characterizations of each of her interpretations of abstraction below:

1. *Abstraction level as the quality of the relationship (QR) between the object of thought and the thinking person* (Hazzan, 1999) refers to the interpretation that abstraction is not an inherent property of a concept, but rather the relationship between the person thinking about the concept and the concept itself. For example, when students are confronted with a problem and are "fumbling in the dark without any mental object to hang onto" (Hazzan, 2003, p. 107), they may subconsciously reduce the level of abstraction by turning to a familiar concept, experience, or object.
2. *Abstraction level as reflection of the process-object duality (POD)* (Hazzan, 1999) is built on earlier work on the process object distinction (Dubinsky, 1991; Sfard, 1991*)*. This construct refers to the interpretation that conceiving a mathematical concept as a process is at a lower level of abstraction than examining it as a mathematical object. When a concept is understood as an object, one can examine the relationships between two concepts. However, if it is perceived as a process, the concept is expressed using canonical procedures (Hazzan, 2003).
3. *Abstraction level as the degree of complexity (DC) of the mathematical concept (Hazzan, 1999)* refers to the interpretation that the more complex an idea is, the more abstract it is. One may reduce the level of abstraction of an object by reducing its complexity. For example, a student may replace a set with an element of the set.

To demonstrate the utility of Hazzan's framework, we view the following results from existing MI literature through the interpretations the framework provides. (1) We could interpret Harel and Sowder's (1998) findings that students viewed MI as generalizing a conclusion by computing a few cases as reducing the abstraction level as the quality of the relationship (QR) because they were choosing to rely on more familiar computations of concrete examples rather than the unfamiliar concept of MI. (2) We can interpret Brown's (2008) observation that students had ignored some aspects of an MI task and "reduced the task to an algebraic" (p.13) process as reducing abstraction as a reflection of process-object duality (POD). Finally, (3) interpreting the results of Baker (1996) through the lens of reducing abstraction, we can conclude that by omitting the base case, students removed a conjunction from the hypothesis of MI. This rendered the complex statement comprising MI, which is of the form " if [P and (Q implies R)] then S" to a simpler statement "If [Q implies R], then S", thereby reducing the abstraction level of MI as the degree of complexity (DC). The above three examples demonstrate established results from MI literature can be viewed through the lens that the reducing abstraction framework provides.

During our analyses of students working on parallel MI-recursion tasks, it became clear that (a) students sometimes engage with recursion and MI at different levels of abstraction and (b) as students used parallel contexts, their movement between levels of abstraction was not limited to reducing levels of abstraction. These observations catalyzed our expansion of the framework. We call this expanded framework the *Navigating Abstraction* framework. The expanded framework utilizes constructs not included in Hazzans (1999) framework. More specifically, we say that one *ascends levels of abstraction* when they move from a lower to a higher level of abstraction and *descends levels of abstraction* when they move from a higher to a lower level of abstraction. Additionally, we identify instances when a student navigates through different aspects of the task but remains on the same level of abstraction. When such a phenomenon occurs, we say that the student has *transferred abstraction levels* from one aspect to another. Therefore, our framework provides a lens through which we can make sense of the interplay between a student's movement between levels of abstraction and between the MI and recursion aspects of our tasks. This allows us to document how interacting with the tasks in this study facilitates the development of students' understandings of MI and recursion and the connection between them.

The framework for analyzing the level of abstraction as QR with respect to each aspect of a task is shown in Figure 1. There are analogous figures for navigating abstraction via the other interpretations in the framework, but they are too large to fit within the margins of this paper.

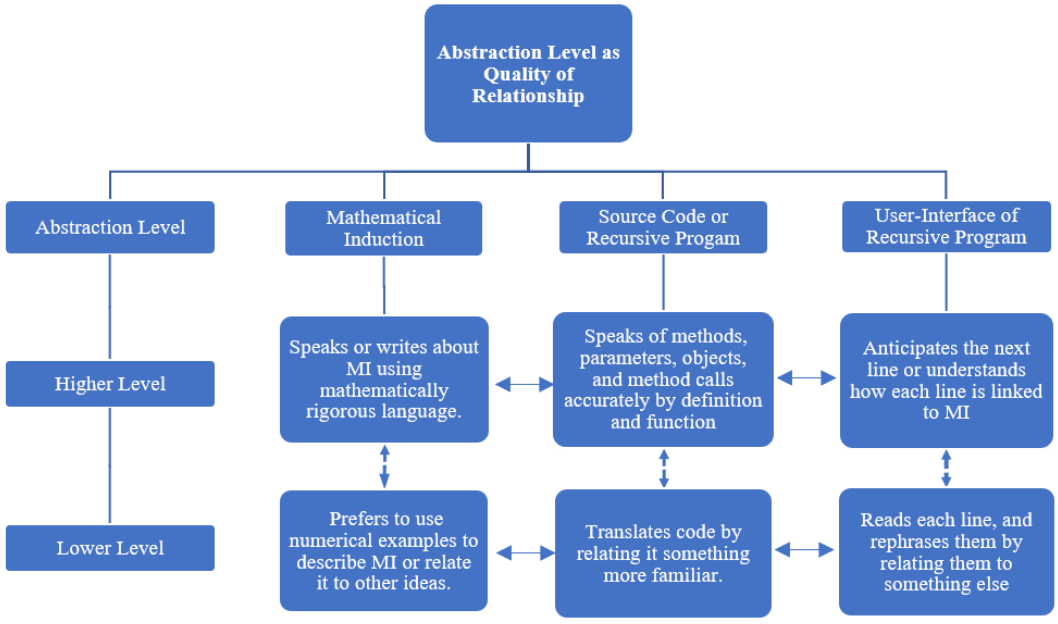

Figure 1. Framework for analyzing abstraction levels as QR with respect to the three aspects of each task.

Methods

Mario, Harry, and Sarah were senior math majors at a public university who took part in a 90-minute, semi-structured, task-based interview study, based on the exploratory teaching experiment methodology (Steffe & Thompson, 2000). The first author of this manuscript functioned as the teacher/researcher and performed the interviews.

The interviews consisted of tasks where MI was presented along with a user-interface and the source code for the corresponding recursive program, to which students could attend at their own pace. Students were prompted to think aloud while they worked, especially if the direction of

their gaze changed from screen to paper, or vice-versa. While we found that the *Navigating Abstraction* framework was productive for accounting for our entire data set, due to space constraints, we limit ourselves to discussing Mario's work on Task 1 (The sum of the first n integers is $\frac{n(n+1)}{2}$) and Task 4 (Every positive integer can be written as a sum of powers of 2).

Results

To identify movement between abstraction levels, we focused on the utterances and inscriptions that indicated the participants were directing attention to mathematical induction (MI), the source code (RS), or the output of the recursive program (RP). We focus on two episodes that illustrate how Mario both transferred and navigated between abstraction levels. These examples simultaneously illustrate our *Navigating Abstraction* framework and provide insights into how a student might leverage the connections between MI and recursion to strengthen their understanding of both.

Mario Ascends Abstraction Levels

When Mario worked on Task 1 before accessing its corresponding recursive program, he only spoke about the computation required to validate the equation on which induction was to be performed. Mario's work, which can be seen in Figure 2, does not include explanations or justifications. Later in this manuscript, we will see that Mario understood MI the full logical complexity of its statement, but there was no indication that he was evaluating the validity or truth of any statement; therefore, he did not understand that MI involved a predicate. As seen in Figure 2, Mario considered the entire hypothesis of MI from his written work, but never states the conclusion of MI to finish the proof. We interpret this interaction as working at a lower level of abstraction as POD while working at a higher level of abstraction as DC.

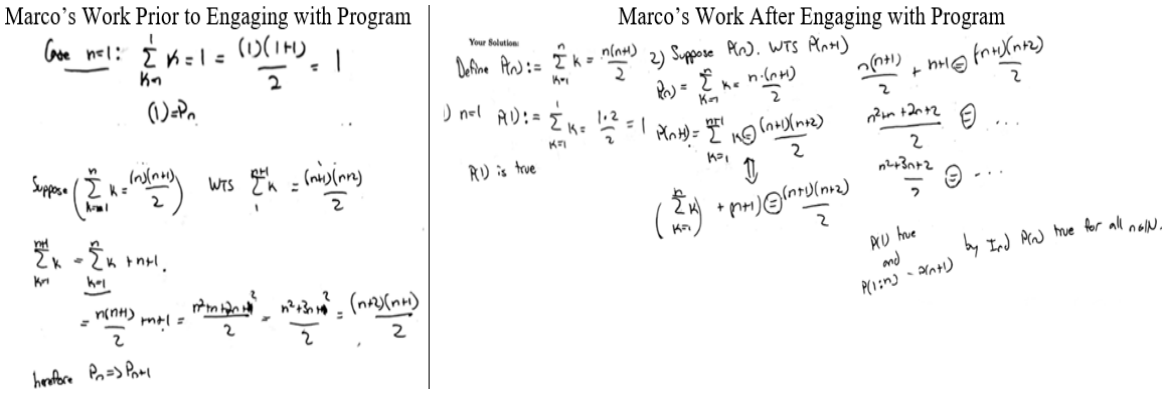

Figure 2. Marco's work on Task 1 before (left) and after (right) working with the recursive program.

The excerpt below highlights how he first makes sense of the MI task:

Mario: What I want to show is that the sum from 1 to $n + 1$ is n times $n(n+1)/2$. If I can show that this (the sum from 1 to n is $n(n+1)/2$) implies this (the sum from 1 to $n+1$ is $(n+1)(n+2)/2$). I have the sum that looks like the thing that I'm assuming I know what it is, and if I manipulate it, I should get the thing that I want.

While engaging with the RP aspect of Task 1, Mario first interpreted $P(n)$ not as a predicate, but as the partial sum from 1 to n, and then corrected himself: "Oh. It (the program) defines the statement ($P(n)$) to mean that those two things (left and right side of the equation) are equal. That's not the same thing." Mario expressed a shift in his thinking from $P(n)$ as an algebraic

expression in n, to an algebraic statement in n which is either true or false. In other words, Mario was starting to conceptualize $P(n)$ as a predicate which indicates that he ascended levels of abstraction as POD with respect to the RP aspect of Task 1.

Mario chose to re-attempt the MI aspect of Task 1 and defined $P(n)$ as a function of n at the onset of the task, shown in Figure 2. After verifying the base case, he wrote "$P(1)$ is true", indicating that he now thought of $P(n)$ as a Boolean-valued function. Mario had transferred his perception of $P(n)$ as a predicate from the RP aspect of the task to the MI aspect of the task.

Furthermore, as seen at the bottom right of Figure 2, Mario finished the proof by writing the entire statement of MI, which was not included in the program's output or source code. Now that Mario is considering the entire conditional statement when working with the MI aspect of Task 1, we can deduce that he has ascended levels of abstraction as DC.

Mario navigates between abstraction levels

When attending to Task 4, Mario asked a question that he had not asked during any of the other tasks.

Mario: What's my $P(n)$ statement? My first thought is maybe I don't want to use induction on n, but maybe I do. I guess I'll try induction on n and see if that's the right thing. So what's my $P(n)$ statement? n is equal to a sum (pause) is equal to the sum over i equals something to something else.

Although Mario had not hesitated to define the predicate during the other tasks, his work in Figure 3 exhibits his indecision. In the previous tasks, MI was applied to a single-variable equation, but the equation generated by the statement in Task 4 had three unknowns. Mario's question demonstrates an unfamiliarity with the type of MI statement in Task 4. His discomfort with this new type of problem may explain his decision to work out some examples, as shown in Figure 3. Computing examples requires substituting variables with numerical values, consequently decreasing the number of variables. Thus, we could interpret Mario's decision to compute examples as working at a lower level of abstraction as QR with respect to the MI aspect of Task 4.

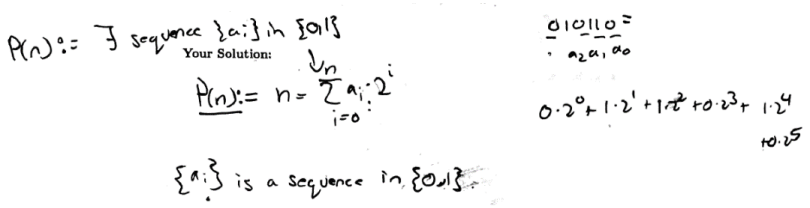

Figure 3. Mario's work prior to interacting with the recursive program.

To see more examples, Mario decided to use the recursive program for Task 4. By delegating the calculation of examples to the RP aspect of Task 4, Mario transferred the level of abstraction as QR from the MI aspect to the RP aspect of Task 4.

Mario: Okay, so we start, I guess an interesting case might be 15 because that's when we overflow the ones into a bunch of zeros.

Mario fluidly transitioned between the MI task and the recursive program's user-interface and source code. After each interaction with the program, Mario progressed further with the MI task. The excerpt below exemplifies the connections Mario was making between the MI aspect of Task 4 and the recursive function, Power2, located in the source code.

Mario: PowerTwo returns the string, which is yeah, the, the sum of powers of two that equal the number that you give it. So, how does it know to stop when n is greater or equal to

0?.. Oh, I have an idea. So, it recursively divides your, the integer by two. Okay and so I can do that too.

Interviewer: So, what gave you that idea?

Mario: Yeah, the PowerTwo. As long as we have a positive integer. Okay, it recursively divides by two and looks at it whether it's zero or one. So, I guess that is how you get the binary decomposition of a number. You keep dividing by two and you see if the thing you get is even or odd. So yeah, and this method uses that recursive loop to construct the sequence.

After connecting Power2 to a proof strategy, Mario resumed writing his proof. Note that Mario interpreted the code in Power2 in terms of its function as a program, indicating that he was working at a higher level of abstraction as QR with respect to RS aspect of Task 4.

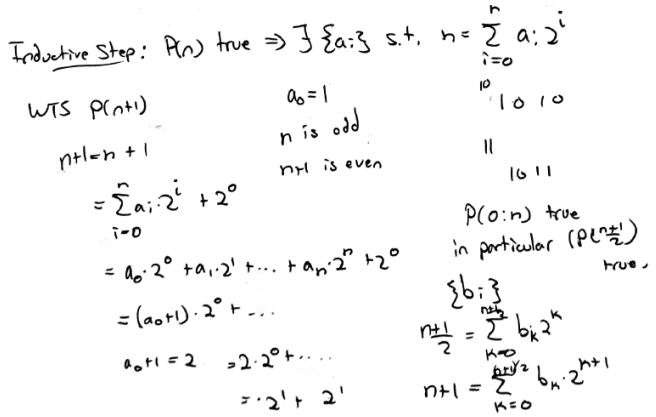

Figure 4. Mario's Proof of the Inductive Step of Task 4.

In his proof, shown in Figure 4, Mario divides $n+1$ by 2, then uses strong induction to represent $(n+1)/2$ as a sum of powers of 2. The sophistication of using two cases along with strong induction indicates an increase of rigor in his proof, suggesting that Mario transferred the higher level of abstraction as QR from the RS aspect to the MI aspect of Task 4. Hence, Mario ascended levels of abstraction as QR with respect to the MI aspect of Task 4.

During the final segment of the interview, the interviewer asked Mario how he conceived of mathematical induction. His response indicates that completing the tasks allowed him to make connections between recursion and induction that shifted his perception of MI.

Mario: Something I noticed. Instead of starting with *n* and going to *n* +1, you are decomposing *n* +1 into *n*…which reminds me a lot of recursion. Then induction is still *P*(*n*) implies *P*(*n*+1), but the way to do it is decompose *P*(*n*+1) into *P*(*n*) and that reminds me a lot of recursion.

To Mario, making the link between induction and recursion provided him with an additional strategy when approaching MI tasks. Specifically, he explains "if you know how to do an easy problem and you know how to decompose a hard problem into an easy problem, then you know how to do the hard problem."

Summary

During Task 1, Mario ascended abstraction as POD from the MI aspect of the task to the RP aspect of the task and then transferred the higher abstraction level from the RP aspect of Task 1

to the MI aspect of Task 1. Additionally, as shown in Figure 5, after interacting with the program, Mario ascended abstraction levels as DC with respect to the MI aspect of the task.

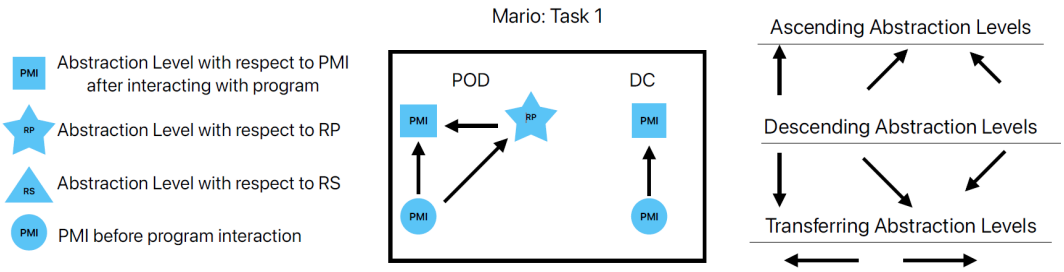

Figure 5. The Navigating Abstraction Framework applied to Mario's work on Task 1

During Task 4, Mario found it necessary to reduce the abstraction level as QR before ascending abstraction levels with respect to the MI aspect of the task. As shown in Figure 6, Mario transferred the lower level of abstraction as QR with respect to MI by interacting with the user-interface of the program and leveraged the source code to ascend levels of abstraction.

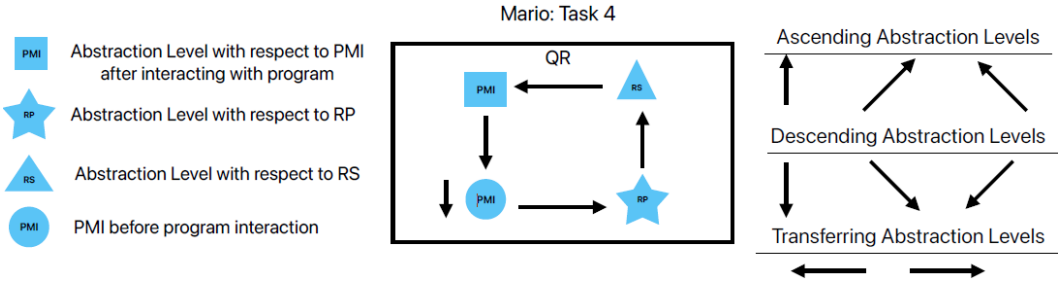

Figure 5. The Navigating Abstraction Framework applied to Mario's work on Task 4..

Discussion

Modifying the reducing abstraction framework to highlight the link between a proof by mathematical induction and a recursive program allowed us to recognize instances in which students transferred an abstraction level from their mathematical proof-work to their interaction with the recursive programs. Whereas Hazzan's (1999) Reducing Abstraction framework was a useful lens in determining when students reduced the abstraction level to cope with a concept, it did not provide the tools to account for how students moved *between* levels of abstraction or contexts. By extending the Reducing Abstraction framework to our *Navigating Abstraction* framework, we were able to recognize and examine how students navigated between abstraction levels and transferred abstraction levels to better understand both recursion and MI.

This framework provided a lens to interpret how students leveraged the back-and-forth navigation between induction/recursion and lower/higher levels of abstraction to ascend levels of abstraction, thereby allowing us to gain insights into how students understanding of induction and recursion co-evolved. This, in turn, helps demonstrate both the educational potential of using recursive programs to reinforce students' understanding of induction as well as the potential for using induction to enforce their understanding of recursion. Hence, the *Navigating Abstraction* framework is itself a research contribution in that it characterizes the movement between abstraction levels while simultaneously providing a bridge between mathematical and computational levels of abstraction.

References

Baker, J. D. (1996). *Students' Difficulties with Proof by Mathematical Induction.*

Brown, S. (2008). *Exploring Epistemological Obstacles to the Development of Mathematics Induction*.

Davis, M., Grassl, R., Hauk, S., Mendoza-Spencer, B., & Yestness, N. (2009). *Learning proof by mathematical induction*.

Dubinsky, E. (1989). Teaching mathematical induction II. *Journal of Mathematical Behavior*, *8*(3), 285–304.

Dubinsky, E. (1991). Reflective abstraction in advanced mathematical thinking. In D. Tall (Ed.), *Advanced mathematical thinking* (pp. 95–123). Kluwer.

Dubinsky, E., Weller, K., Mcdonald, M. A., & Brown, A. (2005). Some Historical Issues and Paradoxes Regarding the Concept of Infinity: An Apos-Based Analysis: Part 1. *Educational Studies in Mathematics*, *58*(3), 335–359. https://doi.org/10.1007/s10649-005-2531-z

Durand-Guerrier, V. (2003). Which notion of implication is the right one? From logical considerations to a didactic perspective. *Educational Studies in Mathematics*, *53*(1), 5–34. https://doi.org/10.1023/A:1024661004375

Harel, G. (2002). *The Development of Mathematical Induction as a Proof Scheme: A Model for DNR-Based*. 47.

Harel, G., & Sowder, L. (1998). Students' proof schemes: Results from exploratory studies. In A. Schoenfeld, J. Kaput, & E. Dubinsky (Eds.), *Research on collegiate*.

Hawthorne, C., & Rasmussen, C. (2015). A framework for characterizing students' thinking about logical statements and truth tables. *International Journal of Mathematical Education in Science and Technology*, *46*(3), 337–353.

Hazzan, O. (1999). Reducing Abstraction Level when Learning Abstract Algebra Concepts. *Education Studies in Mathematics*, 71–90.

Hazzan, O. (2003). How Students Attempt to Reduce Abstraction in the Learning of Mathematics and in the Learning of Computer Science. *Computer Science Education*. https://doi.org/10.1076/csed.13.2.95.14202

Hazzan, O., & Hadar, I. (2005). Reducing Abstraction when Learning Graph Theory. *Journal of Computers in Mathematics and Science Teaching*, *24*(3), 255–272.

Hazzan, O., & Zazkis, R. (2003). Reducing Abstraction: The Case of Elementary Mathematics. In *International Group for the Psychology of Mathematics Education* (Vol. 3, pp. 39–46). International Group for the Psychology of Mathematics Education. https://eric.ed.gov/?id=ED501007

Healy, L., & Hoyles, C. (2000). A Study of Proof Conceptions in Algebra. *Journal for Research in Mathematics Education*, *31*(4), 396–428. https://doi.org/10.2307/749651

Hoyles, C., & Küchemann, D. (2002). Students' understandings of logical implication. *Educational Studies in Mathematics*, *51*, 193–223.

Knuth, E. J., Choppin, J. M., & Bieda, K. N. (2009). Proof: Examples and Beyond. *Mathematics Teaching in the Middle School*, *15*(4), 206–211. https://doi.org/10.5951/MTMS.15.4.0206

Raychaudhuri, D. (2014). Adaptation and extension of the framework of reducing abstraction in the case of differential equations. *International Journal of Mathematical Education in Science and Technology*, *45*(1), 35–57. https://doi.org/10.1080/0020739X.2013.790503

Rich, K. M., & Yadav, A. (2020). Applying Levels of Abstraction to Mathematics Word Problems. *TechTrends*, *64*(3), 395–403. https://doi.org/10.1007/s11528-020-00479-3

Roh, K. H. (2010). An empirical study of students' understanding of a logical structure in the definition of limit via the epsilon-strip activity. *Educational Studies in Mathematics*, 263–279.

Sakhnini, V., & Hazzan, O. (2008). Reducing Abstraction in High School Computer Science Education: The Case of Definition, Implementation, and Use of Abstract Data Types. *Journal on Educational Resources in Computing*, *8*(2), 5:1-5:13. https://doi.org/10.1145/1362787.1362789

Sfard, A. (1991). On the dual nature of mathematical conceptions: Reflections on processes and objects as different sides of the same coin. *Educational Studies in Mathematics*, *22*(1), 1–36.

Steffe, L. P., & Thompson, P. W. (2000). Teaching experiment methodology: Underlying principles and essential elements. *Handbook of Research Design in Mathematics and Science Education*, 267–306.

Stylianides, A. J., Stylianides, G. J., & Philippou, G. N. (2004). Undergraduate students' understanding of the contraposition equivalence rule in symbolic and verbal contexts. *Educational Studies in Mathematics*, *55*(1), 133–162. https://doi.org/10.1023/B:EDUC.0000017671.47700.0b

Wijeratne, C., & Zazkis, R. (2015). On Painter's Paradox: Contextual and Mathematical Approaches to Infinity. *International Journal of Research in Undergraduate Mathematics Education*, *1*(2), 163–186. https://doi.org/10.1007/s40753-015-0004-z