TechTrends
https://doi.org/10.1007/s11528-020-00479-3

ORIGINAL PAPER

Applying Levels of Abstraction to Mathematics Word Problems

Kathryn M. Rich[1] · Aman Yadav[2]

Abstract
In many discussions of the ways in which abstraction is applied in computer science (CS), researchers and advocates of CS education argue that CS students should be taught to consciously and explicitly move among levels of abstraction (Armoni *Journal of Computers in Mathematics and Science Teaching*, *32*(3), 265–284, 2013; Kramer *Communications of the ACM*, *50*(4), 37–42, 2007; Wing *Communications of the ACM*, *49*(3), 33–35, 2006). In this paper, we describe one way that attention to levels of abstraction could also support learning in mathematics. Specifically, we propose a framework for using abstraction in elementary mathematics based on Armoni's (2013) framework for teaching computational abstraction. We propose that such a framework could address an enduring challenge in mathematics for helping elementary students solve word problems with attention to context. In a discussion of implications, we propose that future research using the framework for instruction and teacher education could also explore ways that attention to levels of abstraction in elementary school mathematics may support later learning of mathematics and computer science.

Keywords Mathematics education · Computer science education · Abstraction · Problem solving · Elementary education

Introduction

The *CS For All* movement (White House 2016) has spurred a great deal of interest in bringing computer science education to all students in K-12 (Grover and Pea 2013; Perez 2018; Yadav et al. 2017). When it comes to elementary and middle school, many of the current initiatives have focused on developing frameworks (e.g., CSK-12.org 2016), curriculum materials (e.g., code.org 2016), and instructional techniques (Armoni 2013) for stand-alone teaching of computer science (CS) to young students. Little attention has been given to how these new educational ideas might support learning beyond CS. In this paper, we explore how one innovation developed in support of CS education could be adapted to support learning of another subject, mathematics. Specifically, we argue that application of a particular CS idea, levels of abstraction, has potential to help elementary educators address a difficult area of instruction in mathematics: solving word problems.

We begin by summarizing prior research on abstraction and word problems. Next, we discuss how moves to a lower level of abstraction (Hazzan 2003) can be used as a lens that reveals connections between the bodies of research on abstraction in computer science and word problems in mathematics. Finally, we propose an adaptation of Armoni's (2013) framework for teaching abstraction and describe how its use might help students to more productively approach solving word problems.

Background

In this section, we review two bodies of research we aim to bring together in this paper. First, we describe computer science students' tendency to move from higher to lower levels of abstraction when problem solving. Second, we describe elementary mathematics students' tendencies to solve word problems without consideration of context.

Levels of Abstraction in Computer Science Education

Although specific definitions are varied and many, computational *abstraction* is generally understood to refer to the act or

✉ Kathryn M. Rich
richkat3@msu.edu

Aman Yadav
ayadav@msu.edu

[1] Michigan State University, 620 Farm Ln, East Lansing, MI 48824, USA

[2] Michigan State University, 620 Farm Ln, East Lansing, MI 48824, USA

Published online: 10 February 2020

process of ignoring irrelevant details for the purpose of focusing on the essential aspects of a problem or situation. Abstraction is both a process and an entity — a process of extracting essential features and the resulting simplified representation or entity (Muller and Haberman 2008). In ongoing discussions what quality computer science education should entail, *abstraction* has been suggested as one of the main pillars (Wing 2006). Indeed, Aho and Ullman (1995) described computer science as a "science of abstraction – creating the right model for thinking about a problem and devising the appropriate mechanizable techniques to solve it" (p. 1, as cited in Muller and Haberman 2008, p. 188). Thus, in computer science, abstraction is "a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems" (College Board 2017, p. 9).

Within computer science, representations are typically not discussed in simple terms of whether they are abstract or not. Rather, representations exist along a continuum of abstraction and are often described as being at a higher or lower level of abstraction relative to each other (Hillis 1998). Representations at high levels of abstraction show broad pictures of a phenomenon, whereas representations at low levels of abstraction show a smaller piece of the phenomenon in more detail. For example, Hillis (1998) points out that a function call within a program is at a higher level of abstraction than the lines of code in the function's definition. The function call is a representation showing *what* a function does. The placement of function calls within code require relating the function to the broader problem the program is addressing, so placing the calls requires a programmer to work at a higher level of abstraction. By contrast, the more detailed information about *how* the function works is in the function definition. Coding the definition requires thinking in detail about how the function will accomplish its task, but does not require thinking about how the use of the function will contribute to solving the broader problem. Thus, programmers coding function definitions, as opposed to adding function calls, are working at lower levels of abstraction.

In her seminal paper discussing computational thinking, Wing (2006) stated that thinking like a computer scientist "requires thinking at multiple levels of abstraction" (p. 35). The focus in computer science, therefore, is not on creating or using abstractions at a particular level, but rather on being able to move among different levels of abstraction (Wing 2006) and choose the level (or the "right model") that is most useful for the particular stage of problem solving (Aho and Ullman 1995, as cited in Muller and Haberman 2008). Although abstraction is understood, colloquially, as a process of moving from a concrete representation to a more abstract representation, in computer science the creation of high-level abstract representations is not necessarily the end goal. Fluid and easy movement among levels — whether these are moves from higher to lower levels of abstraction or vice versa — and consideration of the appropriate level of abstraction are the ultimate goals. Skillful use of abstraction entails the use of the appropriate amount of detail, which varies by context (Kramer 2007).

Hazzan (2003) showed that mathematics and computer science students have a tendency to move from higher to lower levels of abstraction while problem solving. She analyzed undergraduate students' problem-solving strategies and noted three consistent patterns. First, students tended to think in terms of processes—the specifics of what a solution would require the student to *do*—rather than in terms of concepts—the general ideas of *what* needs to be done without specific consideration of how to accomplish it. For example, one undergraduate computer science student thought of an array in terms of how to implement it in a computer's memory, rather than in terms of what an array might be used to accomplish. This is parallel to the difference between a placing a function call and writing a function definition described by Hillis (1998). Thinking about processes moves students to a lower level of abstraction because the focus shifts to implementation details—similar to writing a function definition where one has to focus on how the function is implemented—rather than purposes for using the array—which would be similar to using a function call in the main program where the programmer does not have to worry about *how* a function does what it does, only *what* it does. Processes are generally understood in more detail than concepts because they are learned first (Sfard 1991). The inclusion of more detail is indicative of a lower level of abstraction.

Second, Hazzan (2003) found that students often reformulated problems in familiar terms. For example, one undergraduate mathematics student thought of a solution of a differential equation in terms of a solution to a non-differential (e.g., linear, quadratic) equation. Familiar problems are generally considered to be at a lower level of abstraction than unfamiliar problems, as the former are more highly connected to an individual's other knowledge about the problem context or related solutions and thus more detailed (Wilensky 1991). In particular, familiar problems are often connected to prior knowledge about the processes used to solve them—knowledge that allows students to work at a lower level of abstraction, as just described.

Lastly, Hazzan (2003) found that students tended to think about one particular element of a set, rather than the whole set, when solving problems. For example, when asked about linked lists, one undergraduate computer science student spoke specifically about one-dimensional linked lists. While thinking about a single element, students can focus on the details of that element and therefore stay at a low level of abstraction. However, focusing on a single element may mean losing sight of the essential

features of the set and why that particular element fits into the set—views provided at a high level of abstraction. These examples, and how they are related along a continuum of levels of abstraction, are shown in Fig. 1.

Although moves to a lower level of abstraction are not necessarily errors — indeed, moving to a lower level of abstraction can often be a productive problem solving technique (Hazzan 2003) — this consistent pattern of moving to lower levels becomes problematic when students do not shift back to higher levels of abstraction at appropriate times. Thinking about one element of a set can be instructive when solving a problem about the entirety of the set, for example, but eventually one must shift back to thinking about the whole set. In reflecting on this study and others in her body of work, Hazzan (2008) noted that the moves to lower levels of abstraction were often not conscious on the part of the students. She argued that students may be able to more productively move among levels of abstraction if computer science instructors explicitly articulated the ways they use abstraction as they solve problems. Finding ways to help students consciously and purposefully move among levels of abstraction is thus a current problem of practice in computer science education.

In the next section, we summarize a body of work on elementary mathematics students' difficulties solving word problems, emphasizing the evidence showing that students' lack of attention to context is a key issue. In the subsequent section, we reinterpret the word problem research through the lens of students moving among levels of abstraction to argue that a key instructional difficulty—namely, supporting students in attending to problem contexts—might be addressed by drawing students' attention to the ways they move among levels of abstraction as they solve word problems.

Word Problems and Lack of Attention to Context

Since the results of the 1982 National Assessment of Educational Progress (NAEP) exhibited disappointing and startling results in relation to mathematics problem solving (Carpenter et al. 1983), there has been a good deal of research on elementary and middle schoolers' solving of word problems. Mirroring two previous sets of NAEP results, the 1982 results showed a discrepancy between students' performance on routine and nonroutine word problems. *Routine problems* can be solved via rote application of an algorithm, while *nonroutine problems* require consideration of contextual factors in the problem to obtain a correct answer. For example, consider the following problem: "An army bus holds 36 soldiers. If 1128 soldiers are being bussed to their training site, how many buses are needed?" (NAEP, as cited in Carpenter et al. 1983, p. 656). This problem is considered nonroutine, as rote use of a division algorithm produces the answer 31 remainder 12, or 31⅓ — an inadequate answer to the number of buses required. On the 1982 NAEP, 70% of 13-year-old students performed the correct division calculation, but only 23% used the result to report the correct answer of 32 buses. Twenty-nine percent gave the exact quotient as the answer, neglecting to consider that a fractional number of buses does not make sense, and 18% ignored the remainder and reported 31 as the answer, neglecting to account for the 12 soldiers who would not have a seat (Carpenter et al. 1983). These results suggested that students' poor performance on the NAEP was not due to lack of arithmetic skills, but rather due to students' tendency to ignore contextual factors when solving word problems.

A seminal study by Verschaffel et al. (1994) expanded on these results to explore word problems involving all four arithmetic operations with younger participants. The authors

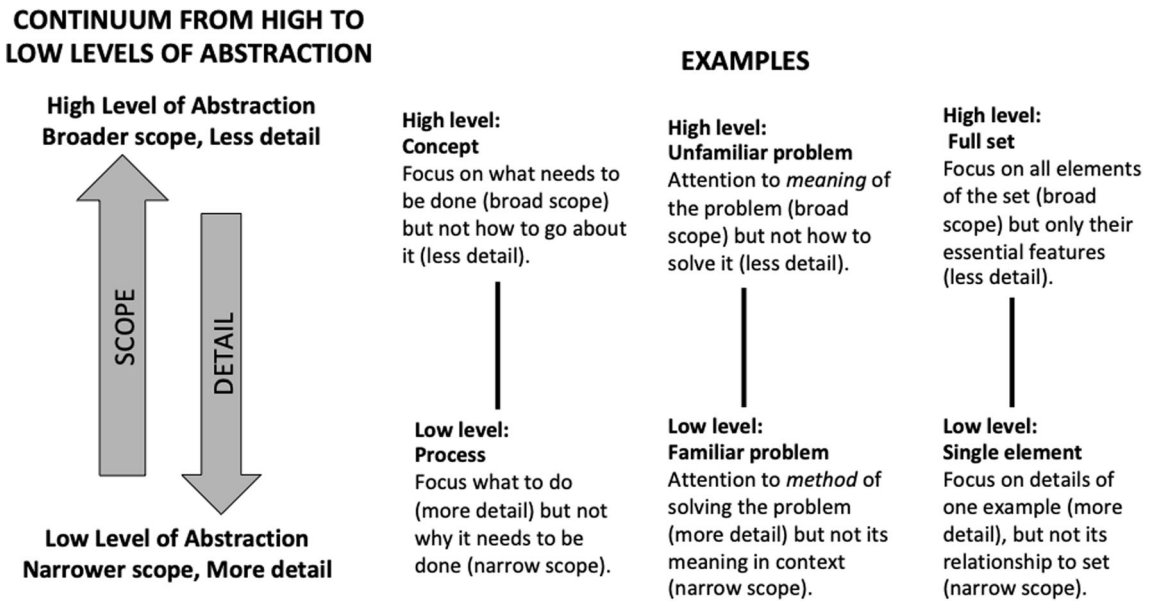

Fig. 1 Examples of high versus low levels of abstraction from Hazzan (2003)

administered 20 word problems to 75 fifth graders. Ten of the problems were standard (S-problems), solvable via straightforward application of an algorithm. The other ten problems were "problematic" (P-problems) in that the result produced by application of an algorithm would be incorrect or insufficient as a solution. For example, an S-problem and its paired P-problem are as follows:

> S-problem: "A shop-keeper has two containers for apples. The first container contains 60 apples and the other 90 apples. He puts all apples into a new, bigger container. How many apples are there in that new container?" (Verschaffel et al. 1994, p. 276).
> P-problem: "What will be the temperature of water in a container if you pour 1 [liter] of water at 80° and 1 [liter] of water of 40° into it?" (Verschaffel et al. 1994, p. 276).

The P-problem above is nonroutine, as the problem is about combining two things, but adding the two salient numbers in the problem (80 and 40) will not result in the correct answer.

Over 80% of the S-problems were solved correctly; by contrast, only 17% of the answers given to the P-problems were classified as realistic by the researchers. This is particularly striking given that the researchers marked answers as realistic when they were consistent with what the students' written work revealed about their beliefs about the problem context. For example, for the P-problem above one student gave an answer of 80°, explaining that when liquids of different temperature are combined they take on the temperature of the hottest liquid. This answer was marked as realistic, despite being incorrect, because it reflected use of the students' (admittedly erroneous) real-world knowledge. Even with this definition of what counts as a realistic answer, only 17% of students gave answers classified as realistic to the above P-problem. These results, again, are consistent with the hypothesis that students' poor performance on word problems is better explained by lack of attention to context than by lack of mathematical ability.

Many studies have produced similar patterns of results over the last 25 years of mathematics education research (e.g., Reusser and Stebler 1997; Palm 2008; Weyns et al. 2017). As such, various strategies have been implemented in efforts to help students be more attentive to word-problem contexts throughout their problem-solving processes. For example, Palm (2008) investigated the impact of the level of authenticity of problems on students' consideration of real-world knowledge. He found that more authentic tasks — defined as modified versions of typical word problems that "better simulate some of the aspects of real life" (p. 42) — led to more realistic answers from students.

Other researchers have focused on changing the norms of the classroom environment to focus on consideration of context and sense-making. For example, Verschaffel and De Corte (1997) described a teaching experiment focused on bringing a mathematical modeling perspective to word problems. Students in the experimental group worked on nonroutine problems through processes of cooperative learning, and the teacher focused on establishing norms that encouraged collective sense-making and multiple interpretations of answers. The study led to moderate improvements in students' consideration of real-world knowledge when problem solving. Thus, a focus on more challenging problems and processes of sense-making shows promise in addressing students' tendency to ignore real-world information during problem solving. More frameworks and strategies for implementing such instruction would be welcome in the mathematics education field. In the sections that follow, we argue that using computational abstraction ideas to frame word-problem instruction is a promising strategy.

Connecting Word Problems and Computational Abstraction

In this section, we point out similarities between elementary students' performance on word problems (Silver et al. 1993; Verschaffel et al. 1994) and undergraduate students' tendency to shift to lower levels of abstraction (Hazzan 2003). We then discuss how a recently proposed framework for teaching computational abstraction (Armoni 2013) might be used in elementary mathematics classes to serve the dual purpose of improving instruction on word problems and giving students early experience with consciously and deliberately moving among levels of abstraction.

Using Levels of Abstraction to Interpret Word Problem Research

One way of interpreting students' lack of attention to context in solving word problems is in terms of moving to lower levels of abstraction, similar to the patterns Hazzan identified in older students' problem-solving strategies (Hazzan 2003; Hazzan and Zazkis 2005). Specifically, when elementary students solve word problems, they tend to reformulate them from contextualized situations to problems involving only decontextualized arithmetic. Figure 2 connects these two ways of thinking about word problems to levels of abstraction. The line of research summarized in the previous section suggests that when elementary school students solve word problems, they move from the contextualized problem (a higher level of abstraction) to decontextualized arithmetic (a lower level of abstraction), and do not return to the higher level to interpret the result of the arithmetic. Ergo, it seems that explicitly teaching the solving of word problems via attention to multiple levels of abstraction — in particular, attention to shifting back

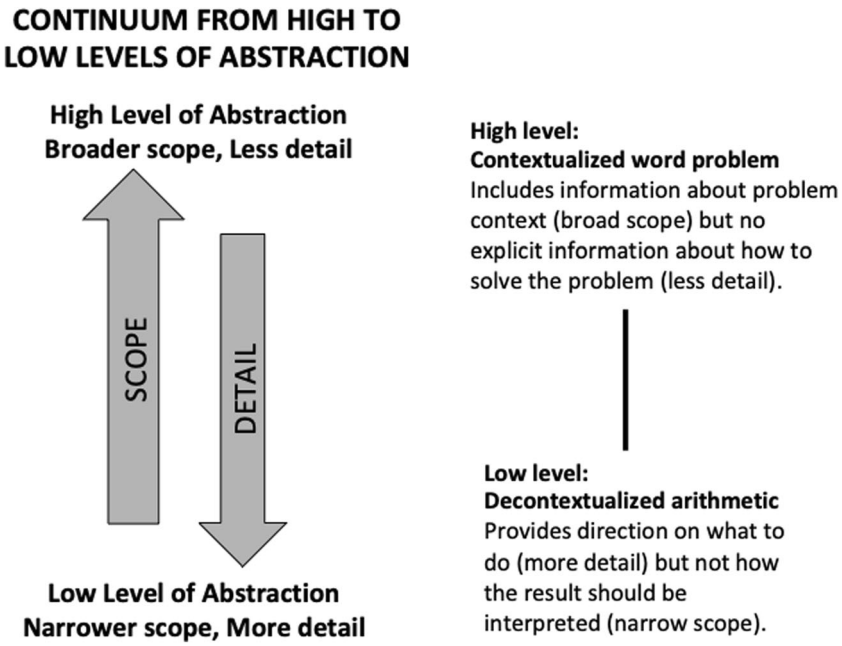

Fig. 2 Connecting word problems to levels of abstraction

to a higher level of abstraction after completing the arithmetic — could improve students' consideration of context and therefore their performance on word problems.

In the next section, we discuss a framework for teaching abstraction taken from the computer science education literature. After its description, we describe its application to the solving of word problems.

A Framework for Teaching Abstraction

Armoni and colleagues recently developed and used a framework for explicitly teaching abstraction in computer science (Armoni 2013; Statter and Armoni 2016; Statter and Armoni 2017). The framework is based on a hierarchy of levels of abstraction first proposed by Perrenet et al. (2005). The levels of the hierarchy, presented from the highest level of abstraction to the lowest, are as follows:

- The *problem* level, at which the problem is considered as its own entity with attributes such as solvability and complexity.
- The *object* level, at which a solution to the problem — in the form of an algorithm — is considered as its own entity, not connected to any particular programming language.
- The *program* level, at which the problem solution is considered as an algorithm written in a particular programming language.
- The *execution* level, at which the problem solution is considered as a particular run of a program on a particular machine.

Given that prior work suggested that students tend to operate at the object and program levels without attending to the problem and execution levels, Armoni (2013) argued that students should be taught how to consciously consider and move among all four levels. Specifically, Armoni suggested that her aim was to help students to develop four related skills:

(1) differentiate between levels of abstraction;
(2) consciously and freely move between levels of abstraction;
(3) decide the level of abstraction at which it is most comfortable and appropriate to work during each phase of problem solving; and
(4) use successive refinements of abstraction.

Armoni (2013) highlighted eight guidelines for how her framework could be applied in instructional contexts. Four of the guidelines are general and apply to discussion of all four levels of the framework. They include:

- *Be persistent and precise* about consistently and clearly distinguishing among levels.
- *Use language cues* as an aid in distinguishing between levels. For example, restrict references to specific programming constructs to the Program and Execution levels.
- *Move from high levels of abstraction to lower levels,* rather than starting at low levels and then attempting to move up.
- *Be explicit and reflective*, prompting students to look back at their own processes and consider the levels of abstraction used.

The three of the four remaining guidelines relate to techniques for differentiating between particular levels of abstraction: distinguishing the Problem level from the rest,

distinguishing the Object level and the Program level, and distinguishing Program level and the Execution level. The final guideline suggests using refinements of the Object level to illustrate the varying amount of detail that might be appropriate for discussing an algorithm. For example, at some stages it might be appropriate to discuss the generic idea of a sorting algorithm, and at other times it might be appropriate to refer to the name of a particular algorithm, such as BubbleSort.

Statter and Armoni (2016, 2017) examined how the framework could be used to teach abstraction in an introductory computer science course for 7th graders. The authors compared the performance on course exams between students taught using Armoni's (2013) framework and students in a control group who took the same course (but were taught without the framework). Results exhibited that students in the treatment group were more likely to include verbal descriptions of their exam solutions, along with their code, on a course exam than were their peers in the control group. The authors considered verbal descriptions to be evidence of working at the Object level, because it showed that students were able to describe their algorithms without reference to specific code elements. Providing code as part of the solution, by contrast, was considered evidence of working at the Program level. Thus, the authors argued that the students in the treatment group were better able to use multiple levels of abstraction in problem solving, as evidenced by their use of description at the Object level and code at the Program level in their solutions. Moreover, students in the treatment group were also more likely than those in the control groups to provide *only* verbal descriptions of the solution for problems on which a verbal description was sufficient. Statter and Armoni (2016) argued that this was evidence that the students in the treatment group were better able to judge which level of abstraction was most appropriate for particular problems. Students in the control group always included code in their solutions and thus always did some work at the Problem level. Students in the treatment group, by contrast, recognized that sometimes working only at the Object level was sufficient. They did not consciously or unconsciously move to lower levels of abstraction as the students in Hazzan's (2003) analysis did, suggesting the framework successfully steered students away from doing so.

Adapting an Abstraction Framework to Teach Students to Solve Word Problems

The preliminary evidence of the success of Armoni's (2013) framework for helping 7th-grade students develop abstraction skills in computer science (Statter and Armoni 2016, 2017) suggests its potential for giving students an earlier start at developing abstraction skills, possibly at the elementary level. Moreover, Armoni's attention to differentiation between all four levels of abstraction suggests the framework might be helpful for improving student performance in solving word problems that require attention to the context when choosing methods and interpreting the results of solution methods, such as the P-items used by Verschaffel et al. (1994). In this section, we propose an adapted form of Armoni's (2013) abstraction framework that could be applied in the context of word problems in elementary classrooms.

First, we consider what the four levels of Armoni's (2013) framework might look like in the context of elementary mathematics. We transformed the framework from computer-science-specific ideas like programming languages to the solving of word problems (see Table 1). The Problem level is largely the same, except that the problem, in the elementary mathematics case, is the text of a word problem. Hence, in our framework this is called the Problem Comprehension level. The Object level, in the case of computer science, refers to consideration of an algorithm as a black box — considering what the algorithm accomplishes without attention to how it does so. As such, the analogue in the solving of word problems would be to consider what kind of answer is needed, or what the solution might look like, without yet considering what mathematics might be used to obtain that solution. We call this the Solution Visualization level. The Program level, in the case of computer science, refers to attending to the details of how to implement the algorithm in a specific language. In the context of word problems, we argue that the Program level analogue would be attention to specific mathematical procedures that students can carry out to obtain the desired solution. We call this the Solution Planning level. Finally, at the Execution level, running the program would be analogous to carrying out the mathematical procedures to solve the word problem. Hence, we call this the Solution Enactment level.

We see potential benefits of using our proposed framework for students' learning in mathematics. Our framework bears a strong resemblance to other instructional approaches proposed by mathematics education researchers for improving students' performance on word problems. For example, Greer's (1997) model of students' ideal solution strategies for solving word problems is shown at the top of Fig. 3. Greer argued that, ideally, students would begin by reading the problem text. Next they create a *situation model,* or a representation that reflects the meaning of the problem and what can be done to solve it. Once the problem is well understood, students create a mathematical model (often, an expression or equation) that reflects what mathematics can be used to find a solution. Students compute using the mathematical model to get a solution, which is then interpreted in the context of the situation model.

The four stages outlined by Greer (1997) map onto the four levels of our proposed framework, as shown at the

Table 1 Armoni's (2013) framework for abstraction alongside proposed new framework for use with word problems

Armoni's level	Armoni's description for computer science	New framework level	Use with word problems
Problem level	The problem is considered as its own entity with attributes such as solvability and complexity.	Problem Comprehension level	The given text of a word problem is examined with attention to comprehension, without discussion of solution methods.
Object level	A solution to the problem — in the form of an algorithm — is considered as its own entity, not connected to any particular programming language.	Solution Visualization level	The problem is interpreted to determine the qualities of the solution and what form it might take, without reference to any specific mathematical procedures.
Program level	The problem solution is considered as an algorithm written in a particular programming language.	Solution Planning level	Particular mathematical methods, procedures, and models are chosen that can be used to produce the desired solution, without carrying them out.
Execution level	The problem solution is considered as a particular run of a program on a particular machine.	Solution Enactment level	The chosen mathematical methods are carried out to obtain a result.

bottom of Fig. 3. Furthermore, Greer (1997) emphasized the importance of students returning to their situation models after finding tentative solutions, as shown by the curved arrow in Fig. 3. This return to the situation model reflects a move to a higher level of abstraction, which is a key missing piece in students' typical solution strategies in both mathematics and computer science (Hazzan 2003; Verschaffel et al. 1994). Greer (1997) argued that students' typical word problem strategies tend to skip over the creation of a situation model altogether, which corresponds to lack of attention to the Solution Visualization level. Applying our framework to the solving of word problems in elementary classrooms has the potential to remedy this lack of attention to situation models as well as provide needed experience in returning to a higher level of abstraction.

To illustrate this point, we give an example of how our proposed framework could be applied to solving one of the P-problems from Verschaffel et al. (1994): "What will be the temperature of water in a container if you pour 1 [liter] of water at 80° and 1 [liter] of water of 40° into it?" (p. 276). Teachers and students would start at the Problem Comprehension level, and then make clear and explicit shifts to the other levels, moving freely among levels as needed. Table 2 details how this process might play out in a classroom.

Conclusion and Future Directions

In this paper, we have proposed a framework describing how levels of abstraction could be applied and discussed in elementary mathematics classrooms to help students include more consideration of context in their problem-solving processes. Admittedly, other models of instruction have been suggested that could accomplish the same goal (notably, Verschaffel and De Corte's (1997) framework for teaching from a modeling perspective, which inspired Greer's (1997) model that forms the basis of Fig. 3). However, our proposed framework has the additional advantage demonstrating a close connection between general processes of abstraction, useful in computer science, and problem solving in mathematics activities common in elementary school. We believe articulating this close connection has implications for future research on student learning and teacher education.

First, further research on this issue is merited to better understand the potential power of the instructional strategy of making abstraction explicit for supporting mathematics learning. Specifically, future research should examine how the proposed framework could influence students' problem-solving within word problems. Research could also examine how the use of the proposed framework affects students' problem-solving

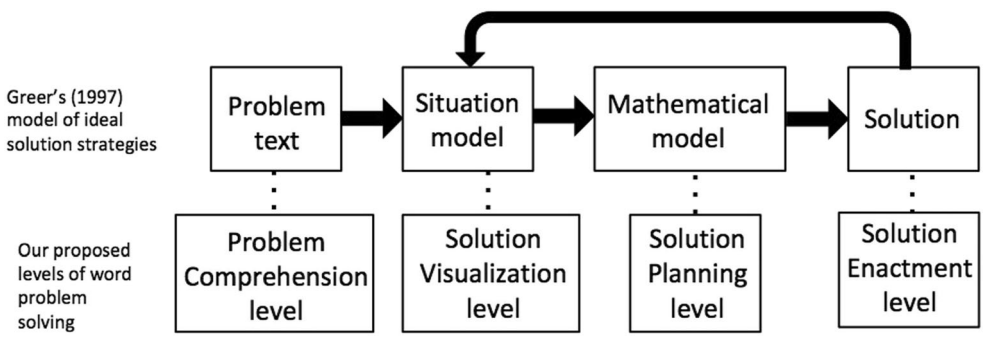

Fig. 3 Greer's (1997) model of ideal word problem solution strategies mapped onto our proposed framework

Table 2 Applying our adapted version Armoni's (2013) framework to solving Verschaffel et al.'s (1994) temperature P-problem

Framework levels	Greer solution stage	Teacher and student actions
Problem Comprehension level	Problem text	Carefully read the problem. Discuss anything that students find interesting or confusing about the problem statement. Clearly identify what the problem is asking, without discussing what the answer might be. In this example, the problem is asking for the temperature of the water after the two liters have been combined.
Solution Visualization level	Situation model	Transition from talking about the problem to talking about what a potential solution might look like, without discussing what mathematics might be used to solve the problem. Discuss what students know about what happens when water samples with different temperatures are combined. Would both samples get warmer? Colder? Would they change in different ways, or the same way? The discussion should lead to the idea that when two samples of the same size are combined, their temperatures will even out to a shared temperature halfway between the original temperatures.
Solution Planning level	Mathematical model	Transition to discussion of mathematics. Discuss how mathematics can be used to determine a number halfway between 80 and 40. Possible mathematical models include an expression reflecting the calculation of an average (i.e., (80 + 40)/2), a number line with 80 and 40 marked from which the point halfway between can be identified, and so on. Do not yet discuss the specifics of how the chosen mathematics should be carried out.
Solution Enactment level	Solution	Transition to execution of the mathematics. Carry out the chosen mathematics to obtain a numerical answer. For example, add 80 and 40 and divide the result by 2, or locate 80 and 40 on a number line and then locate the number halfway in between.
Solution Visualization level	Situation model	Consider the numerical answer in comparison to the expected characteristics determined in the situation model. Is our temperature between 80° and 40°? Is it halfway between? Does the solution we obtained make sense?

processes in other areas of mathematics. A recent analysis of fourth graders' work on a contextualized problem involving interpretation of a bar graph showed that students make more errors as they shift among levels of abstraction than they do when executing routine mathematics such as counting and addition (Rich et al. 2019b). Use of our proposed framework to guide classroom instruction, with minor modifications to suit the context of the problem, may have the potential to ameliorate these difficulties. Intervention studies, where student strategies are analyzed before and after teachers use the framework for instruction, could address these questions.

Second, the close connection between mathematics problem solving and computational abstraction raise questions about how explicit attention to abstraction, through the use of the framework in elementary mathematics, may support concurrent or later student learning of computer science. Prior work suggests that elementary teachers readily make connections between CT ideas and their teaching in multiple subjects, including mathematics, science, social studies, and literacy (Rich et al. 2019a). In our work with elementary school teachers, we have observed them incorporating computer-science-inspired problem-solving heuristics, such as decomposing problems and debugging solutions, into multiple subjects including mathematics, science, social studies, and literacy. Classroom observations suggest that students applied these heuristics in multiple contexts. Given the suggested generalizability of these heuristics, we believe that exploration of the ways students may apply their understanding of levels of abstraction to computational problems later in their education, after being introduced to the ideas in elementary mathematics, would be productive. That is, we suggest that longitudinal studies may support or refute the hypothesis that exposure to levels of abstraction in mathematics may build readiness for thinking about levels of abstraction in computer science.

Third, we believe our framework may have productive uses in teacher education. Further research could examine how providing opportunities for teachers to examine student work through the lens of levels of abstraction might allow them insight into students' thinking processes, and how these insights impact their instructional techniques. Think-aloud studies with teachers could provide insight into how they make sense of levels of abstraction as applied to elementary school topics, and how they believe consideration of the levels is helpful in students' problem solving or assessments of student work.

Lastly, collaborating with teachers to implement and study how the framework could be made suitable for elementary school is important given that Armoni's (2013) original framework was designed for middle schoolers. Adapting it for use

with younger students, who are at a different developmental level, may take more than simply shifting the context to word problems or other elementary mathematics topics. Studying how teachers, who bring their own experiences and knowledge about students to bear when using instructional tools, apply the framework, and how students respond to such instruction, could provide insight into how to adapt the framework to be accessible and helpful to younger students.

Acknowledgements This work was supported by the National Science Foundation under Grant number 1738677. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Compliance with Ethical Standards

Potential Conflicts of Interest The authors have no conflict of interest to report.

Research Involving Human Participants All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Informed Consent Informed consent was obtained from all individual participants included in the study.

References

Aho, A. V., & Ullman, J. D. (1995). *Foundations of computer science*. New York: W. H. Freeman and Company.

Armoni, M. (2013). On teaching abstraction in computer science to novices. *Journal of Computers in Mathematics and Science Teaching, 32*(3), 265–284.

Carpenter, T. P., Lindquist, M. M., Matthews, W., & Silver, E. A. (1983). Results of the third NAEP mathematics assessment: Secondary school. *The Mathematics Teacher, 76*(9), 652–659.

Code.org. (2016). Computer Science Fundamentals for elementary school. Retrieved from https://code.org/educate/curriculum/elementary-school

College Board. (2017). AP Computer Science Principles course and exam description. Retrieved from https://apcentral.collegeboard.org/courses/ap-computerscience-principles/course

CSK-12.org. (2016). K-12 Computer Science Framework. Retrieved from https://k12cs.org/

Greer, B. (1997). Modelling reality in mathematics classrooms: The case of word problems. *Learning and Instruction, 7*(4), 293–307.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: a review of the state of the field. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051.

Hazzan, O. (2003). How students attempt to reduce abstraction in the learning of mathematics and in the learning of computer science. *Computer Science Education, 13*(2), 95–122.

Hazzan, O. (2008). Reflections on teaching abstraction and other soft ideas. *ACM SIGCSE Bulletin, 40*(2), 40–43.

Hazzan, O., & Zazkis, R. (2005). Reducing abstraction: the case of school mathematics. *Educational Studies in Mathematics, 58*(1), 101–119.

Hillis, W. D. (1998). *The pattern on the stone: The simple ideas that make computers work*. New York: Basic Books.

Kramer, J. (2007). Is abstraction the key to computing? *Communications of the ACM, 50*(4), 37–42.

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

Palm, T. (2008). Impact of authenticity on sense making in word problem solving. *Educational Studies in Mathematics, 67*(1), 37–58.

Perez, A. (2018). A framework for computational thinking dispositions in mathematics education. *Journal for Research in Mathematics Education, 49*(4), 424–461.

Perrenet, J., Groote, J. F., & Kaasenbrood, E. (2005). Exploring students' understanding of the concept of algorithm: Levels of abstraction. In *Annual joint conference integrating technology into computer science education* (pp. 64–68). New York: ACM.

Reusser, K., & Stebler, R. (1997). Every word problem has a solution – the social rationality of mathematical modeling in schools. *Learning and Instruction, 7*(4), 309–327.

Rich, K. M., Yadav, A., & Schwarz, C. V. (2019a). Computational thinking, mathematics, and science: elementary teachers' perspectives on integration. *Journal of Technology and Teacher Education, 27*(2), 165–205.

Rich, K. M., Yadav, A., & Zhu, M. (2019b). Levels of abstraction in students' mathematics strategies: what can applying computer science ideas about abstraction bring to elementary mathematics ? *Journal of Computers in Mathematics and Science Teaching, 38*(3), 267–298.

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*, 1–36.

Silver, E. A., Shapiro, L. J., & Deutsch, A. (1993). Sense making and the solution of division problems involving remainders: an examination of middle school students' solution processes and their interpretations of solutions. *Journal for Research in Mathematics Education, 24*(2), 117–135.

Statter, D., & Armoni, M. (2016). Teaching abstract thinking in introduction to computer science for 7th graders. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education - WiPSCE '16* (pp. 80–83).

Statter, D., & Armoni, M. (2017). Learning abstraction in computer science: A gender perspective. In E. Barendsen & P. Hubwieser (Eds.), *Proceedings of the 12th workshop on primary and secondary computing education* (pp. 5–14). New York: ACM.

Verschaffel, L., & De Corte, E. (1997). Teaching realistic mathematical modeling in the elementary school: a teaching experiment with fifth graders. *Journal for Research in Mathematics Education, 28*(5), 577–601.

Verschaffel, L., De Corte, E., & Lasure, S. (1994). Realistic considerations in mathematical modeling of school arithmetic word problems. *Learning and Instruction, 4*(4), 273–294.

Weyns, A., Van Dooren, W., Dewolf, T., & Verschaffel, L. (2017). The effect of emphasising the realistic modelling complexity in the text or picture on pupils' realistic solutions of P-items. *Educational Psychology, 37*(10), 1173–1185.

White House. (2016). Computer science for all [Blog post dated January 30, 2016]. Retrieved from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Wilensky, U. (1991). Abstract meditations on the concrete and concrete implications for mathematical education. In I. Harel & S. Papert (Eds.), *Constructionism: Research reports and essays, 1985–1990* (pp. 193–203). Boston: Epistemology & Learning Research Group.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591.