Exploring Middle School Students' Understanding of Algorithms Using Standards-aligned Formative Assessments: Teacher and Researcher Perspectives

Satabdi Basu, Daisy Rutstein, Carol Tate, Arif Rachmatullah, Hui Yang, and Christopher Ortiz
{satabdi.basu, daisy.rutstein, carol.tate, arif.rachmatullah, hui.yang, christopher.ortiz}@sri.com
SRI International

Abstract: 'Algorithms' is a core CS concept included in the K-12 CS standards, yet student challenges with understanding different aspects of algorithms are still not well documented, especially for younger students. This paper describes an approach to decompose the broad middle-school 'algorithms' standard into finer grained learning targets, develop formative assessment tasks aligned with the learning targets, and use the tasks to explore student understanding of, and challenges with, the various aspects of the standard. We present a number of student challenges revealed by our analysis of student responses to a set of standards-aligned formative assessment tasks and discuss how teachers and researchers interpreted student responses differently, even when using the same rubrics. Our study underscores the importance of carefully designed standards-aligned formative assessment tasks for monitoring student progress and demonstrates the need for teacher content knowledge to effectively use formative assessments during CS instruction.

1. Introduction

The demand for computer science (CS) learning opportunities in K-12 is rapidly increasing as it becomes clear to policy makers, educators, and parents that an understanding of computing is essential to success in a technology and automation-rich society. One of the most fundamental CS concepts is the algorithm, an ordered set of precise and clear instructions to solve a problem or generate a desired output. An algorithm is often a first step towards planning the logical flow of a computer program. Algorithmic thinking encompasses knowledge and skills specific not only to CS and programming but also to general problem-solving and computational thinking (ISTE, 2016).

The Computer Science Teachers Association (CSTA) K-12 CS Standards identify 'Algorithms and Programming' as a key CS concept across all grade bands and 'Algorithms' as one of its five sub-concepts (CSTA, 2017). Most state CS standards in the U.S. also include algorithms as a core CS concept (Guo & Ottenbreit-Leftwich, 2020). Algorithms are often introduced in K-12 through simple activities such as students writing instructions for making a peanut butter and jelly sandwich and teachers enacting the instructions. These lessons are appropriate for communicating the idea that computers will execute instructions as written (rather than as intended) but they do not touch upon other important ideas emphasized in the 'algorithms' standards such as representation, interpretation, comparison, testing and debugging of algorithms. In our review of several existing K-12 CS curricula, we have found that many of them do not include the full scope of concepts and practices covered by 'algorithms' standards. CS education research currently provides little guidance on how to unpack the broad 'algorithms' standards, making it difficult for educators to understand the full scope of the standards and the range of skills that comprise proficiency.

A thorough understanding of the algorithms standards is a necessary, though not sufficient, component of teachers' ability to develop and effectively use formative assessments on algorithms (Basu et al., 2022). In addition to content knowledge, a deep understanding of how K-12 students think about algorithms would allow teachers to identify and address student challenges and advance student understanding. Formative assessment tasks that intentionally target individual aspects of algorithms standards can reveal useful information about student understanding and specific challenges on each of those aspects (Basu et al., 2022). The ability to use such standards-aligned formative assessments to measure and support student progress on CS standards is articulated as part of the CS teacher standards (CSTA, 2020). However, there is currently limited literature on K-12 students' conceptualization of, and challenges with, the concept of 'algorithms', and many CS teachers report feeling underprepared to use formative assessments to monitor student learning (Gordon & Heck, 2019).

In this paper, we unpack a broad CSTA middle school 'algorithms' standard, 2-AP-10, "Use flowcharts and/or pseudocode to address complex problems as algorithms" into finer grained learning targets. We then discuss an approach to developing aligned formative assessment tasks (and rubrics) and describe how we used a set of tasks to explore student understanding of, and challenges with, the fine-grained learning targets underlying the standard. Finally, we compare our evaluation of student responses with how middle school CS teachers evaluated the same student responses using the same rubrics and how teachers' own understanding of algorithms informed their evaluation of student responses. By looking at both students' responses and teachers' interpretations, we are able to infer student understanding and challenges as well as explore how teacher knowledge mediates the impact of formative assessment. We are guided by the following research questions:

RQ1: How can formative assessment tasks be developed to examine student understanding on various aspects of the middle school algorithms standard?

RQ2: What can we infer about middle school students' understanding of and challenges with the concept of algorithms from students' responses to standards-aligned formative assessment tasks?

RQ3: How do teachers' interpretations of middle school students' understanding of algorithms compare to researchers' interpretations, when using the same assessment tasks and rubrics?

2. Theoretical perspectives

2.1 Relevant related work

2.1.1 Characterizing and assessing students' understanding of algorithms.

Recent research in CS education includes efforts to assess K-12 students' algorithmic thinking skills and have been situated within a computational thinking (CT) framework (e.g., Basu et al., 2021). These research studies have noted middle school students' challenges with devising algorithms to solve real-world problems (e.g., Wong & Jiang, 2018) and comparing algorithms when the comparison is based on multiple criteria (Basu et al., 2021).

However, there is still limited research on several aspects of the 'algorithms' concept. For example, the middle school standard for algorithms includes the practices of representing algorithms as flowcharts and pseudocode, testing algorithms with a wide variety of inputs, predicting algorithm behavior, and debugging algorithms, many of which are currently under-investigated. Assessments for algorithmic thinking are typically part of broader CT assessments and hence do not cover all aspects of the 'algorithms' concept. Some research on high school students' understanding of algorithms has revealed challenges with using flowcharts to create, call, and manage different sub-algorithms (Rahimi et al., 2018), as well as misconceptions related to the efficiency of algorithms, thinking that fewer lines of code and fewer variables characterize algorithm efficiency (Gal-Ezer & Zur, 2004).

2.1.2 Teachers' content knowledge and pedagogy related to algorithms.

While information about how teachers conceptualize algorithms and perceive their own knowledge of algorithms is limited, there have been some studies that explore teachers' understanding of algorithmic thinking in the context of CT. Rich and colleagues found that elementary school teachers who are new to the field of CT associated the term 'algorithmic thinking' with the mathematical term *algorithm* that specifies steps for performing traditional arithmetic operations, and conceptualized algorithmic thinking as "following steps", and "discovering and explaining strategies" (Rich et al., 2019, p. 179). Research has also shown that appropriate professional development (PD) opportunities can expand teachers' knowledge of algorithmic thinking. For example, Yadav and colleagues (2018) analyzed and compared teacher responses before and after a year-long CT PD and found that teachers initially made generic comments about algorithms but were later able to identify and discuss specific characteristics of algorithms such as efficiency, abstraction, and generalization.

Given the fundamental role of algorithms in CS education, researchers have sought to understand how teachers teach algorithms. A common instructional task is having students construct an algorithm in the form of a flowchart or pseudocode to solve a given problem. A study by Vivian and Falkner (2019) found that teachers with high confidence in teaching a digital CS curriculum frequently used algorithmic language, made more connections to learning objectives on algorithms and programming, and were more likely to engage students in algorithm development and manipulation activities before programming. In our current study, several middle school teachers mentioned that their teaching of algorithms was limited to facilitating activities such as instructions for creating peanut butter and jelly sandwiches, because the CS curricula they used did not include specific lessons on algorithms. Teachers remarked that they formatively assessed their students' understanding of algorithms based on completion of project-based activities rather than the knowledge and skills students demonstrated on individual activities.

2.2 Evidence-centered design (ECD)

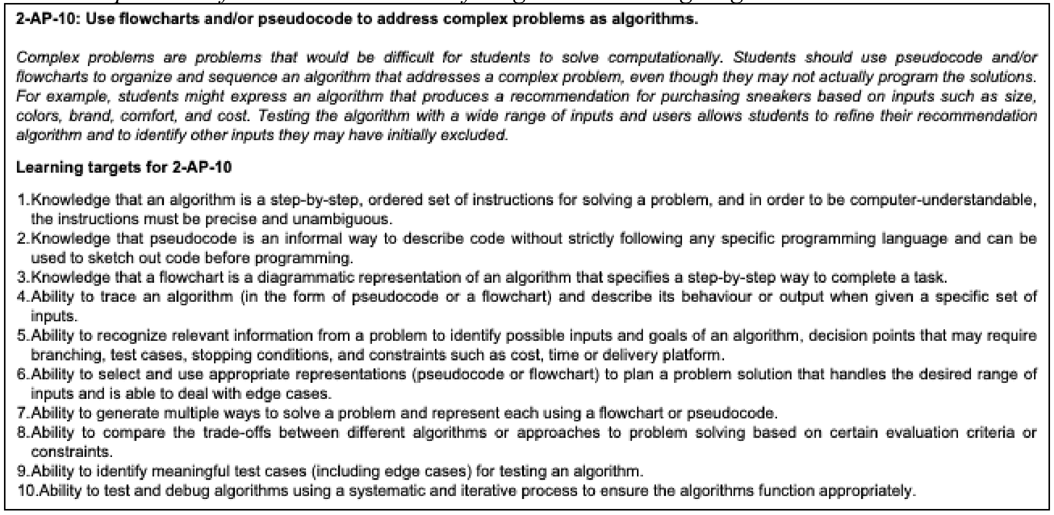
We employed ECD (Mislevy & Riconscente, 2006), a principled assessment design approach, to analyze and unpack the middle school 'algorithms' standard and develop aligned formative assessment tasks. ECD helps 1) define what to measure, 2) identify the evidence needed to measure these goals, and 3) design tasks to produce this desired evidence. The ECD process starts with analyzing the target domain (outlining the scope of the middle-school algorithms standard) and decomposing the standard into a set of finer-grained knowledge, skills, and abilities that students should possess. The ECD process results in a set of assessment tasks that elicit desired evidence on different aspects of the standard as well as aligned rubrics.

3. Methods and Data Sources

3.1 Designing standards-aligned formative assessments

We employed the ECD approach to define the scope of the 2-AP-10 standard by clarifying the knowledge and skills expected of this standard relative to elementary and high school 'algorithms' standards. We found that this middle-school standard transitions students from interpreting algorithms in upper-elementary grades to creating algorithms that solve complex problems and testing algorithms using a variety of inputs. The standard focuses on fluency with representations such as flowcharts and pseudocode that represent the steps to solve a problem pictorially or using plain language description. It includes the ability to identify relevant information from a problem description (e.g., inputs, goals and decision points) and translate it into an algorithm, as well as the practices of testing and debugging algorithms. Analyzing the scope of the 2-AP-10 standard helped us to decompose the broad standard into a set of ten fine-grained learning targets (LTs) on which we could individually examine student understanding (see Figure 1). Specific LTs enable teachers and curriculum developers to design targeted instruction and assessment opportunities to check student understanding and help students overcome specific learning gaps (Basu et al., 2022).

Figure 1. *The CSTA Middle School 'Algorithms' standard and its description, followed by out decomposition of the standard into ten fine-grained learning targets.*



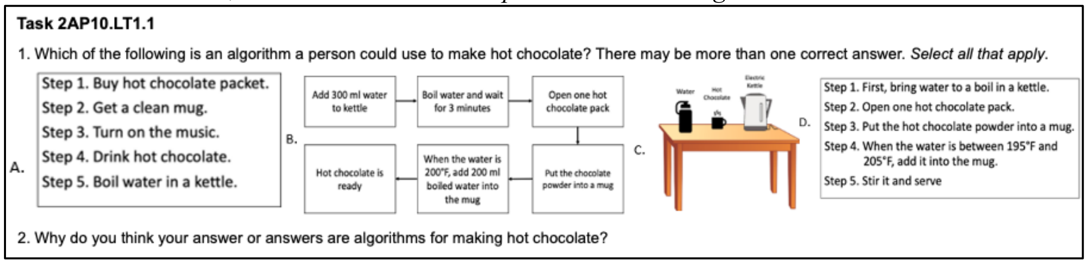
2-AP-10: Use flowcharts and/or pseudocode to address complex problems as algorithms.

Complex problems are problems that would be difficult for students to solve computationally. Students should use pseudocode and/or flowcharts to organize and sequence an algorithm that addresses a complex problem, even though they may not actually program the solutions. For example, students might express an algorithm that produces a recommendation for purchasing sneakers based on inputs such as size, colors, brand, comfort, and cost. Testing the algorithm with a wide range of inputs and users allows students to refine their recommendation algorithm and to identify other inputs they may have initially excluded.

Learning targets for 2-AP-10

1. Knowledge that an algorithm is a step-by-step, ordered set of instructions for solving a problem, and in order to be computer-understandable, the instructions must be precise and unambiguous.
2. Knowledge that pseudocode is an informal way to describe code without strictly following any specific programming language and can be used to sketch out code before programming.
3. Knowledge that a flowchart is a diagrammatic representation of an algorithm that specifies a step-by-step way to complete a task.
4. Ability to trace an algorithm (in the form of pseudocode or a flowchart) and describe its behaviour or output when given a specific set of inputs.
5. Ability to recognize relevant information from a problem to identify possible inputs and goals of an algorithm, decision points that may require branching, test cases, stopping conditions, and constraints such as cost, time or delivery platform.
6. Ability to select and use appropriate representations (pseudocode or flowchart) to plan a problem solution that handles the desired range of inputs and is able to deal with edge cases.
7. Ability to generate multiple ways to solve a problem and represent each using a flowchart or pseudocode.
8. Ability to compare the trade-offs between different algorithms or approaches to problem solving based on certain evaluation criteria or constraints.
9. Ability to identify meaningful test cases (including edge cases) for testing an algorithm.
10. Ability to test and debug algorithms using a systematic and iterative process to ensure the algorithms function appropriately.

Figure 2. *An Example 2-AP-10 Assessment Task Aligned with LT 1 "Knowledge that an algorithm is a step-by-step, ordered set of instructions for solving a problem, and in order to be computer-understandable, the instructions must be precise and unambiguous."*



Task 2AP10.LT1.1

1. Which of the following is an algorithm a person could use to make hot chocolate? There may be more than one correct answer. *Select all that apply.*

A.
Step 1. Buy hot chocolate packet.
Step 2. Get a clean mug.
Step 3. Turn on the music.
Step 4. Drink hot chocolate.
Step 5. Boil water in a kettle.

B.
Add 300 ml water to kettle → Boil water and wait for 3 minutes → Open one hot chocolate pack → Put the chocolate powder into a mug → When the water is 200°F, add 200 ml boiled water into the mug → Hot chocolate is ready

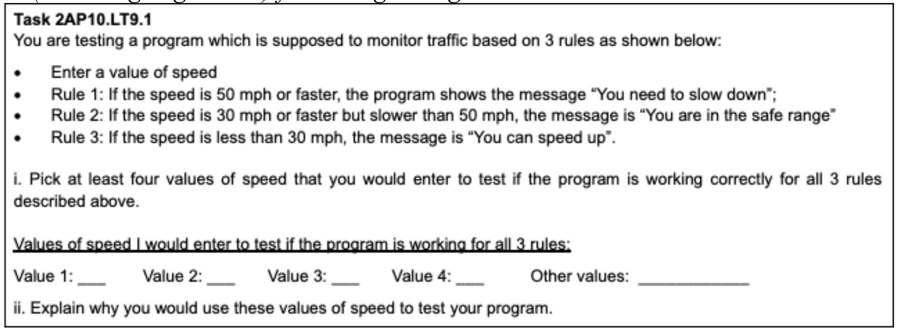
C.

D.
Step 1. First, bring water to a boil in a kettle.
Step 2. Open one hot chocolate pack.
Step 3. Put the hot chocolate powder into a mug.
Step 4. When the water is between 195°F and 205°F, add it into the mug.
Step 5. Stir it and serve

2. Why do you think your answer or answers are algorithms for making hot chocolate?

Next, we followed the ECD framework to develop a set of formative assessment tasks, each focused on eliciting evidence of student understanding on one LT (see Figures 2, 3 for examples). Task formats included multiple choice questions, short-answer prompts, and open-ended explanations. Tasks were short enough to be practical for in-class use to diagnose student understanding of the LTs. Each sub-task was accompanied by one of two rubric types (see Table 1). A RC rubric grouped student responses into Response Categories (RCs) such that each RC was indicative of a certain level of student understanding (a student can be in only one RC). An IC rubric listed a set of Indicated Challenges (ICs) that could be inferred from student responses (a student can have multiple ICs).

Figure 3. *An Example 2-AP-10 Assessment Task Aligned with LT 9 'Ability to identify meaningful test cases (including edge cases) for testing an algorithm'.*



Task 2AP10.LT9.1
You are testing a program which is supposed to monitor traffic based on 3 rules as shown below:

- Enter a value of speed
- Rule 1: If the speed is 50 mph or faster, the program shows the message "You need to slow down";
- Rule 2: If the speed is 30 mph or faster but slower than 50 mph, the message is "You are in the safe range"
- Rule 3: If the speed is less than 30 mph, the message is "You can speed up".

i. Pick at least four values of speed that you would enter to test if the program is working correctly for all 3 rules described above.

Values of speed I would enter to test if the program is working for all 3 rules:

Value 1: ___ Value 2: ___ Value 3: ___ Value 4: ___ Other values: ____________

ii. Explain why you would use these values of speed to test your program.

Table 1. *Rubric for analyzing student responses to Task 2AP10.LT1.1 shown in Figure 2*

Rubric for Part 1			Rubric for Part 2		
Student Response	**Possible inference about student understanding**	**Response Category**	**Student Response**	**Possible inference about student understanding**	**Indicated Challenge**
A only	Challenges with order and final goal: Student does not realize that the steps of an algorithm need to be in order/sequential and the last step should be the goal which is making hot chocolate.	RC1	Student does NOT indicate that their selected options contain instructions that are "clear" or "specific".	Student may have difficulty recognizing that an algorithm needs to include a set of clear or specific instructions.	IC1
C only	Challenge with representation: Student does not realize that an algorithm is a process or ordered set of steps to get to a goal state; a static diagram depicting objects is not an algorithm.	RC2	Student does NOT indicate that their selected options contain instructions that are ordered in a logical way.	Student may have difficulty recognizing that an algorithm includes instructions that are ordered in a logical sequence.	IC2
B and D	**Student understands what defines an algorithm and can recognize an algorithm in various forms.**	**RC3**	Student does NOT mention that the algorithm will fulfil the goal of making hot chocolate.	Student may have difficulty recognizing that the instructions in an algorithm need to fulfil the goal of the algorithm.	IC3
B only or D only	Student understands what defines an algorithm but can only recognize an algorithm in certain forms (i.e., flowchart/diagram process or procedure list).	RC4	Students states that the algorithm "makes sense" or that is how they would make hot chocolate.	Student may not recognize that their explanation should include characteristics of an algorithm instead of their personal opinion.	IC4
Any other response	Student may not understand what an algorithm is.	RC5	Student only restates the algorithm without explaining why it is an algorithm.	Student has difficulty understanding the concept of algorithms and/or articulating it.	IC5
			Student states something that is incorrect or not relevant	Student has difficulty understanding the concept of algorithms and/or articulating it.	IC6

3.2 Small-scale classroom study

We collaborated with eight middle school CS teachers from an urban school district in the Midwestern U.S. who used code.org's CS Discoveries (CSD) curriculum for their teaching. All teachers had participated in CSD PD offered by code.org prior to participating in our study. The teachers varied widely in terms of overall teaching experience (1-21 years), but they had all taught CS for less than three years. We provided teachers with professional learning in the form of educative resources on unpacking the 2AP10 standard and standards-aligned formative assessments for algorithms. Teachers participated in an hour-long, online, synchronous PD session on algorithms where we introduced them to the 2-AP-10 standard, corresponding state standards, our unpacking of the standard, and the aligned formative assessment tasks and rubrics that we had developed. Throughout the PD session, the teachers were engaged in short activities that provided us with important information about their understanding of and familiarity with the 'algorithms' standard. For example, teachers had to identify the LTs with which a given task was aligned. They were also asked to indicate which LTs they were already addressing during instruction and which they planned to address in the future. Teachers were divided into two groups where each group looked at two examples of student responses to an algorithm task (see Figure 3) and discussed how to evaluate the responses using the provided rubrics or by modifying the given rubrics. Teacher responses and discussions were recorded for future analyses. Teachers took a CS pedagogical content knowledge (PCK) survey before participating in the PD and at the end of the classroom study. We designed the survey to measure teachers' attitudes towards CS, knowledge of algorithms and programming concepts, and ability to interpret student work. For this paper, we analyzed teachers' pre-survey responses to two tasks that asked teachers to interpret and compare students' algorithms and predict possible student challenges.

Here we report on student data from formative assessments administered by two out of the eight CS teachers – Dina and Remi (pseudonyms) – who consented to sharing student work and their evaluation of student work. Dina and Remi administered a subset of the formative assessment tasks to their students, evaluated student responses, and shared both student responses and their evaluations with our research team. We collected student data for 36 students across grades 7 and 8. Most students had taken at least one other CS class in the previous school year, though classes were disrupted by school closures and remote instruction during the global pandemic. At the time of the study, students in both Dina and Remi's classes had completed CSD units 1 and 2 (Problem Solving, Web Development) and were working on CSD Unit 3 (Interactive Animations and Games). We gathered data on student responses to six algorithm tasks aligned with LTs 1, 6, 8, and 9, with each task yielding 15 to 36 student responses. Using the same task-specific rubrics we shared with teachers, we coded student responses to the tasks. For each open-ended prompt, at least two researchers coded student responses into RC or IC categories. Researchers met regularly to discuss discrepancies in their coding until they reached consensus. Memos were recorded throughout the discussion and used to refine the rubrics. We then compared our coding to that of the teachers. We conducted descriptive and thematic analyses (Saldaña, 2016) of our coding of student responses and the comparison findings to explore students' understanding of algorithms and the similarities and differences in how teachers and researchers coded students' responses.

4. Findings

4.1 Students' understanding of algorithms

We summarize our findings about middle school students' understanding of algorithms in terms of the LTs targeted by the tasks we analyzed. These findings have implications for CS instruction, curriculum design, and PD design.

LT1: Understanding what an algorithm is. Analysis of student responses to two tasks aligned with this LT revealed that most students understood what an algorithm is. For example, for task 2AP10.LT1.1 shown in Figure 2, 21 of 35 (60%) students who completed the task were able to identify both options B and D as algorithms for making hot chocolate, while 10 (29%) students identified only one of options B and D. However, only nine of the 31 students who selected options B and/or D could justify their selection(s) to any degree. Explanations often mentioned ordered steps, clear instructions, or achieving the goal of making hot chocolate, but rarely included all aspects (see Table 2).

LT6: Selecting flowcharts representing problem solutions. Students responded to one task asking them to select a flowchart that appropriately represents a delivery robot's actions. Ten of 15 (67%) students who worked on this task could represent the given text-based problem solution for the robot as a flowchart. Among the remaining five students, three struggled with using a decision box and selected a flowchart where the decision box did not have an arrow labeled 'NO' flowing out of it. Two students had difficulty understanding that the steps of an algorithm should appear in the same order regardless of the algorithm representation, whether text-based or pictorial as in a flowchart.

LT8: Interpreting and comparing algorithms. Students responded to two tasks involving comparison of algorithms. The first task involved comparing three simple algorithms for navigating a robot based on time and cost criteria. Most students (23 of 34 or 68%) correctly identified the algorithms that would reach the goal, the fastest algorithm and the cheapest algorithm. However, students found it challenging to compare algorithms based on multiple criteria. Only 37% students could compare the algorithms correctly when considering both speed and cost criteria simultaneously. This finding is similar to that observed by Basu and colleagues (2021) with students in grades

4-6 in Hong Kong. The second task involved comparing two relatively complex algorithms that included user input and compound conditionals. Only 20 of 36 (56%) students were able to correctly interpret the individual algorithms. Several students seemed to have difficulty following the logic of conditional statements in the algorithms. Twenty-two (61%) students were able to compare the algorithms and decide which one to use for solving a given problem. However, few of these students could justify their selection. Seven students gave a fully correct explanation, 2 students gave a partially correct explanation, and 13 students were not able to give any suitable explanation for their selection.

LT9: Selecting meaningful test cases to test algorithms. Student responses to Task 2AP10.LT9.1 (Figure 3) revealed that only 5 of 20 (25%) students could identify appropriate test cases (i.e., numbers in different ranges that would test all three program rules). Several (40%) students identified test cases that only tested one of three program rules, while others tested two of the rules. Only four of 20 (20%) students could justify their selection of test cases by describing how they were trying to test all three rules included in the program (see Table 3 for sample responses).

Table 2. *Analysis of sample student responses to Task 2AP10.LT1.1 (Figure 2) using the rubric shown in Table 1.*

Selection for part 1	Response to part 2	Possible inferences about student understanding
B, D *(RC3)*	They're both algorithms for making hot chocolate because they're in the correct order and they give the instructions in a stepwise, efficient order that can be easily read and followed through. *(no IC)*	Student can recognize algorithms and understands the characteristics that define an algorithm.
B, D *(RC3)*	I selected both of those answers because they're explaining the instructions clearly. *(IC2, IC3)*	Student can recognize algorithms but may not understand all the characteristics that define an algorithm.
D *(RC4)*	I think my answer is an algorithm for making hot chocolate because this would be how I make my hot chocolate. *(IC1, IC2, IC3, IC4)*	Student can only recognize algorithms written as a list of steps, may not be aware of characteristics of an algorithm, and is using their experiences to explain their choices.

Table 3. *Sample student responses to Task 2AP10.LT9.1 shown in Figure 3*

Value 1	Value 2	Value 3	Value 4	Other values	Explanation for choice of values to test	Possible inferences about student understanding
55 mph	50 mph	49 mph	30 mph	29 mph, 20 mph	I would use these speeds to see if the program gives each of the three messages only when it is supposed to.	Student is able to identify meaningful test cases to test all three rules (both values in the range and at the boundaries of the rules).
40 mph	30 mph	5 mph	0 mph	-	I would use these values of speed to make sure the program works as it should.	Student is able to identify some meaningful test cases but may not recognize that testing an algorithm requires testing all of its rules or conditions.
49 mph	48 mph	47 mph	46 mph	45 mph, 44 mph	I would use these speeds to see if I need to be exactly on 50 mph to slow down.	Student struggles to identify a full range of meaningful test cases and has challenges recognizing that they need to test the algorithm under different scenarios to make sure it works for all three rules.

4.2 Teacher interpretation of student work

Next, we compare our evaluation of student responses with teachers' interpretations of the same responses. Teachers used the same set of rubrics as the researchers, and though they were encouraged to modify the rubrics as needed, neither Dina nor Remi changed the rubric for any of the tasks. Teachers were consistent with researchers when evaluating the multiple-choice tasks, which allowed no room for subjectivity in terms of which rubric category each response corresponded to. For the open-ended tasks, such as selecting meaningful test cases and explaining selection of algorithms, there were several differences in teacher and researcher interpretations of the same student responses. For example, for Task 2AP10.LT1.1 Part 2 (Figure 2) where students explain why they think their selection is an algorithm, teacher and researcher categorization of student responses matched for only 25% of the responses. While researchers considered certain explanations to be inadequate evidence of student understanding of what constitutes an algorithm (e.g., "I chose these because you end up with the right result", "I think my answers are algorithms because they make sense"), teachers did not select any ICs and interpreted those responses to be indicative of a complete understanding of the 'algorithms' concept. When evaluating student explanations that compared two relatively complex algorithms (aligned to LT8), agreement between researchers and teachers was fairly high (agreement on 30 of 36 or 83% responses). In instances where there were disagreements, there was no clear pattern. Teachers appeared to focus only on student explanations and not on students' choice of algorithm when applying the rubric, leading to differences in overall interpretation of student understanding.

For Task 2AP10.LT9.1 (Figure 3), there was a 43% agreement between teacher and researcher evaluations of student responses to part i, where students identified a range of test cases, and a 67% agreement on part ii, where students explained their choice of test cases. Teachers consistently assigned student responses to a category that was indicative of greater student understanding. Though the rubric explicitly mentioned various acceptable values or value ranges for students to use as test cases so that they could test all the rules in the program, teachers could not identify student challenges when students picked distinct values that tested only one or two of the three rules (e.g., "45, 39, 25, 47") or when students picked non-numeric values (e.g., "45, 40, safe range, 40"). For the explanations, disagreements in evaluating student responses seemed to stem from teachers' preference for articulate, logical, and well written explanations, rather than explanations that articulated the need to test all three rules of the program.

Teachers' knowledge of algorithms. While some of these disagreements between teachers and researchers may have been caused by lack of clarity in the rubrics, our analysis of teacher engagement during the PD and teachers' PCK survey responses suggest that some of the disagreement may have also stemmed from teachers' lack of knowledge of the full scope of the 'algorithms' standard. During the PD, when asked about which LTs teachers already use in their instruction, most teachers identified LT1 and LT5, but no teacher selected LT9, indicating that testing algorithms and identifying test cases were not things they associated with 'algorithms.' Additionally, during a PD group activity where teachers practiced applying rubrics to evaluate example student responses to Task 2AP10.LT9.1, we found that teachers struggled to identify negative numbers as legitimate test cases. Also, consistent with what we found when teachers evaluated their students' work, teachers evaluated correct explanations to be inadequate during PD just because they were short and did not provide enough details about the program's behavior. Further, analysis of teachers' pre-PD PCK survey responses revealed teachers' challenges with interpreting and comparing algorithms. These algorithms included nested conditionals or nested loops and were more complex than those featured in students' formative assessment tasks. Most teachers, including Dina and Remi, had difficulty interpreting some of the individual algorithms which then translated to difficulty comparing the algorithms.

5. Discussion, Limitations and Conclusion

In response to the current demand for quality CS learning opportunities and useful CS pedagogy in K-12, we offer an approach to unpack a broad CS standard into fine-grained LTs, develop formative assessments aligned with the LTs, and use the assessments to reveal student understanding and challenges related to the standard. Both CS educators and curriculum developers can benefit from this approach by gaining a more complete understanding of CS standards and known student challenges. This approach allows teachers to focus on conceptual understanding of CS standards and formative assessment practices, which they can then leverage in any curriculum to improve CS instruction.

In this paper, we chose a fundamental CS concept, 'algorithms,' and demonstrated our approach by decomposing a middle school CS standard on algorithms into ten discrete LTs. We employed an ECD approach to design formative assessments that capture information about student understanding and challenges for each LT (RQ1). The assessments we developed allowed us to identify common student challenges related to aspects of the 'algorithms' concept, such as comparing, and testing algorithms (RQ2). This work aligns with and adds to a growing literature on middle school students' understanding of algorithms. These are, however, preliminary findings, and we acknowledge that generalizations from this study are limited by our small sample size. Another limitation of our study is that the requirements of our formative assessment tasks did not always align well with typical middle school CS classroom expectations. For example, many of our tasks required open-ended responses and/or explanations of selected responses – something many students were not accustomed to doing in CS class. In many CS classrooms, teachers do not always expect detailed written responses from students and when they do, they often provide students with clear rubrics. Hence, it is unclear whether some student responses to our formative assessment tasks are truly indicative of a lack of understanding or merely reflect students' understanding of classroom expectations.

This paper also adds to the limited literature on middle school teachers' understanding of algorithms and describes how teachers' understanding of CS standards may influence their interpretation of student work and their ability to identify student challenges. While our teacher sample was limited, all teachers in our sample agreed that they did not associate the concept of 'algorithms' with testing algorithms systematically and identifying test cases. Several teachers remarked that they did not introduce their students to algorithmic representations such as flowcharts because it was not covered in the middle school CS curricula they used. Upon examining the types of student responses for which teachers and researchers differed in their interpretation of student understanding of algorithms, despite using the same set of rubrics (RQ3), we concluded that some inconsistencies could be reduced by increasing the clarity of our rubrics, while others needed to be addressed through additional teacher training on the concept of 'algorithms.'

Our findings indicate that middle school CS teachers may benefit from additional PD on the ‘algorithms’ standard, in particular the concepts of algorithm representation, comparison, and testing and debugging. Merely designing principled formative assessments and rubrics that can elicit evidence of student understanding may not be sufficient to support teachers; teachers also need deep content knowledge to ensure they can use the formative assessments effectively. While it may seem unsurprising that teachers who lack content knowledge are not able to support their students adequately, empirical data to showcase these connections between teacher knowledge and student learning is important in a nascent field like CS education research, especially when many CS teachers claim to learn along with their students and be self-taught. We hope that with a more complete understanding of the ‘algorithms’ standard, teachers can move beyond just using the peanut butter and jelly sandwich activity to employing a range of activities and assessments that promote a more complete understanding of algorithms.

This work is part of a larger project to support middle school CS teachers with their understanding of five different ‘algorithms and programming’ standards and their ability to use formative assessment tools aligned with these standards. Next steps on this project will involve similar research using different standards and examining whether the findings in this paper hold with a larger sample of students and teachers.

References

Basu, S., Rutstein, D., Tate, C., Rachmatullah, A., & Yang, H. (2022, February). Standards-Aligned Instructional Supports to Promote Computer Science Teachers' Pedagogical Content Knowledge. In *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education V. 1* (pp. 404-410).

Basu, S., Rutstein, D. W., Xu, Y., Wang, H., & Shear, L. (2021). A principled approach to designing computational thinking concepts and practices assessments for upper elementary grades. *Computer Science Education*, *31*(2), 169-198.

Computer Science Teachers Association (2017). *CSTA K-12 computer science standards*. Revised 2017. Retrieved from http://www.csteachers.org/standards

Gal-Ezer, J., & Zur, E. (2004). The efficiency of algorithms—misconceptions. *Computers & Education*, *42*(3), 215-226.

Gordon, E. M., & Heck, D. J. (2019). 2018 NSSME+: Status of High School Computer Science.

Guo, M., & Ottenbreit-Leftwich, A. (2020, October). Exploring the K-12 computer science curriculum standards in the US. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (pp. 1-6).

International Society for Technology in Education (ISTE). (2016). ISTE Standards: Students. https://www.iste.org/standards/iste-standards-for-students

Mislevy, R.J., & Riconscente, M.M. (2006). Evidence-centered assessment design. In S. M. Downing & T. M. Haladyna (Eds.), *Handbook of test development* (pp. 61-90). Mahwah, NJ: Erlbaum.

Rahimi, E., Barendsen, E., & Henze, I. (2018, October). An instructional model to link designing and conceptual understanding in secondary computer science education. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education* (pp. 1-4).

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

Saldaña, J. (2021). The coding manual for qualitative researchers. The coding manual for qualitative researchers, 1-440.

Vivian, R., & Falkner, K. (2019, July). Identifying teachers' Technological Pedagogical Content Knowledge for computer science in the primary years. In Proceedings of the 2019 ACM conference on international computing education research (pp. 147-155).

Wong, G. K., & Jiang, S. (2018, December). Computational thinking education for children: Algorithmic thinking and debugging. In *2018 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 328-334). IEEE.

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. Computer Science Education, 28(4), 371-400

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. DRL-2010591. 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.