Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance

AMAN YADAV, Michigan State University, USA

MARC BERGES, Friedrich-Alexander University Erlangen-Nürnberg, Germany

Computer science education efforts are expanding across the globe to equip students with the necessary computing skills for today's digital world. However, preparing students to become literate in computing activities requires the training of tens of thousands of teachers in computer science. The discrepancy between student needs and teacher preparation in computer science has raised questions of quality teachers, particularly for teachers who do not possess adequate content or pedagogical knowledge to teach computer science efficiently. To address this issue, we designed an instrument to measure knowledge needed to teach computer science (i.e., computer science pedagogical content knowledge). Results exhibited that our instrument measured aspects of teachers' computer science pedagogical content knowledge; however, teachers' prior background in teaching did not influence their performance. We discuss implications for future research and practice.

CCS Concepts: • **Social and professional topics** → **K-12 education**;

Additional Key Words and Phrases: Pedagogical content knowledge

ACM Reference format:

Aman Yadav and Marc Berges. 2019. Computer Science Pedagogical Content Knowledge: Characterizing Teacher Performance. *Trans. Comput. Educ.* 19, 3, Article 29 (May 2019), 24 pages.

<https://doi.org/10.1145/3303770>

1 INTRODUCTION

Current efforts to expand computer science (CS) education in the K-12 curriculum highlight the need for students to understand computer science concepts and practices, as they live in a world shaped by computing [2, 16, 51, 61]. A recent report highlighted the importance of an understanding of CS skills for STEM (science, technology, engineering, and math) disciplines, given that approximately half of jobs by 2018 would require some computing knowledge [12]. Across the globe, including the United States of America, Mexico, Estonia, and United Kingdom, there have been concerted efforts to introduce computer science education in primary and secondary schools under the umbrella of computational thinking and coding as the new literacy [62]. This commitment to broaden computer science education [55, 56] requires training tens of thousands of teachers in

This work was funded by the National Science Foundation (NSF) under grant 1502462. The opinions expressed are those of the authors and do not necessarily reflect those of the NSF.

Authors' addresses: A. Yadav, Michigan State University, College of Education, 620 Farm Lane, East Lansing, MI, 48824; email: ayadav@msu.edu; M. Berges, Friedrich-Alexander University Erlangen-Nürnberg, Department of Computer Science, Martensstr. 3, Erlangen, 91058, Germany; email: marc.berges@fau.de.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2019 Association for Computing Machinery.

1946-6226/2019/05-ART29 \$15.00

<https://doi.org/10.1145/3303770>

computer science [19]. Training these teachers is mainly being done through professional development programs, which range from intensive summer training (such as, Code.org in the United States) to year-long support networks (such as Computing at School in the United States).

In spite of the current efforts and enthusiasm around computer science education, many schools do not offer computer science classes due to difficulties in hiring computer science teachers [21]. Barriers to offering computer science include difficulties in training teachers from other subject areas [13], inadequacy of computer science teaching certifications [17], and the lack of computer science preparation programs [18]. As a result, teachers with limited computer science training, or teachers with only CS industry experience are assigned to teach computer science courses [20]. This means that some of these teachers possess extensive pedagogical knowledge due to their teaching experience or deep content knowledge due to their CS experience, which leads them to not effectively integrate both pedagogical knowledge and content knowledge. However, Shulman [52] argued that mere knowledge of either just the content or just the pedagogy was worthless in the transition to becoming a teacher. He suggested that it is important to pay attention to both the knowledge of subject matter as well as the ways of teaching that subject matter, which he called pedagogical content knowledge.

These issues raise important questions about what skills teachers need to effectively teach computer science: How do teachers acquire knowledge, including the knowledge of subject matter, pedagogy, and pedagogical content knowledge in computer science? How can we evaluate the efficacy of professional development programs and make sure teachers have the knowledge to use effective instructional strategies to improve student outcomes in computer science? How can we ensure that students have access to quality education in computing? To address these questions, we need to better understand computer science teachers' knowledge of computer science concepts and how to teach them (CS pedagogical content knowledge). However, there are few measures that assess this knowledge to teach computer science. This article presents an effort to design and validate a measure of teachers' pedagogical content knowledge for teaching high school computer science. The following sections provide an overview of prior work on pedagogical content knowledge in other disciplines, such as mathematics as well as nascent work in computer science.

1.1 Pedagogical Content Knowledge

Over the past several years there has been considerable attention devoted to teacher quality and its impact on student achievement. A number of organizations, such as ETS, have been examining teacher learning through standardized tests like the Praxis exam, which is then used by states and professional licensing organizations for the teacher certification process. Another way to measure teacher content knowledge is using proxy measures, such as number of college courses taken or content-specific tests [44]. Most of these mechanisms assess focus on knowledge of the subject matter (i.e., content knowledge), which is the "amount and organization of knowledge per se in the mind of the teacher ... [who] must not only be capable of defining for students the accepted truths in a domain. They must also be able to explain why a particular proposition is deemed warranted, why it is worth knowing, and how it relates to other propositions, both within the discipline and without, both in theory and in practice" [52, p.9].

In the past, teachers' knowledge of subject matter and pedagogical knowledge had been the primary focus in teacher education [52]. Pedagogical knowledge was focused on general methods, including pre-instructional strategies, concrete examples, formative testing, questions, curriculum creation, and assessment design [15]. Alone, pedagogical knowledge and content knowledge are not sufficient for teaching purposes [52]. It has become clear that teachers need to be able to relate their pedagogical knowledge to their subject matter within a specific school context and with a specific student population [15]. The teaching process, thus, requires teachers to transform

content knowledge for the purpose of explaining concepts to others, which involves a flexible process of finding multiple representations of information (e.g., analogies, metaphors, examples, problems, activities) and adapting material to students' needs [10, 23]. Shulman extended this notion of knowledge specific to teaching by introducing the idea of pedagogical content knowledge. He suggested that pedagogical content knowledge "goes beyond the knowledge of subject matter per se to the dimension of subject matter knowledge for teaching ... [and includes] the ways of representing and formulating the subject that make it comprehensible to others" [52, p. 9]. He further argued that pedagogical content knowledge also includes teachers having an understanding of conceptions and preconceptions that their students bring to learning of specific topics.

Shulman [52, 53] and others [1, 11, 22] argued that teaching expertise should be described and evaluated in terms of pedagogical content knowledge [44]. As a result, measuring teachers' pedagogical content knowledge has received considerable attention from researchers in teacher education [25, 50] and has proved to be a challenging task [25, 44]. Various instruments have been developed that range from using questionnaires, interviews, observations, and/or student artifacts, either as unique forms of assessment or as ways to triangulate data [7, 25, 32, 44, 57].

Over time, Shulman's work on PCK has been adapted to specific disciplines, such as reading [44], mathematics [26, 33], science [43], and technology [36]. Mathematics education has done considerable work to assess teachers' pedagogical content knowledge (math knowledge to teach) [26, 27]. For example, Hill, Schilling, and Ball [27] developed a measure for assessing knowledge teachers need to teach elementary mathematics. Based on their analyses, the authors found that mathematical knowledge required to teach elementary mathematics is partly domain-specific and not simply related to general factors, such as teaching ability or overall intelligence. They suggested that this knowledge to teach mathematics consists of both general knowledge of content and also knowledge in more specific domains (e.g., knowledge of students and content in number and operations). Research has also been done to assess content knowledge teachers need to teach elementary reading, including "knowledge of text, language, and reading process and how this knowledge is used in practice" [44, p.33]. Similar to Hill et al. [27] findings, the authors found that content knowledge is made up of multiple unique domains (e.g., comprehension, word analysis) rather than a general area of knowledge or reading ability.

1.2 Pedagogical Content Knowledge in CS

In science, PCK is what distinguishes a biologist, whose knowledge is based on a research perspective of science, and a science teacher, whose knowledge is based on a teaching perspective to help students understand concepts [15, 52]. In computer science more specifically, research on PCK is still sparse. Hubwieser et al. [28] attempted a conceptualization of a literature-based CS PCK model. Bolstered by interview data of German educators, the authors found two dimensions of CS teaching: (i) fields of pedagogical operation (i.e., phases in the process of teaching and learning in the classroom), and (ii) aspects of teaching and learning (i.e., other pedagogical categories). The first dimension was composed of elements such as planning and design of learning situations, reacting to students' demands during teaching processes, or evaluating teaching processes. The second dimension included aspects of teaching such as subject content, standards, lesson objectives, and other subject-specific teaching concepts [28]. However, while the model offered a conceptual understanding of PCK in the field of CS teaching, little is known about how these elements manifest in practical teaching situations. Measuring PCK is therefore a logical next step in evaluating teachers' knowledge.

While the literature on CS PCK measures is limited, there has been some prior work on measuring pedagogical content knowledge of computer science topics, such as programming. A German research group developed a competence model for teaching computer science based on

interviews with CS teachers and qualitative analysis of teacher education curriculum [5, 34]. One of the components of the competence model was teachers' pedagogical content knowledge, which consisted of two dimensions: *typical fields of pedagogical operations* (FPO) and *aspects of teaching and learning* (ATL). The first dimension, FPO, represented three categories within the process of teaching and learning in the classroom, which included planning and design of learning situations (FPO 1), reacting on students' demands during instruction (FPO 2), and evaluation of teaching process (FPO 3). The second dimension, ATL, included 15 categories that related to subject matter/curriculum-related issues, teaching methods and use of media, learner-related issues, teacher-related issues, and issues of the educational setting. The current analysis focuses on one of the FPO categories, "reacting on student's demands during the instruction processes," and two ATL categories, "subject specific teaching concepts" (i.e., programming) and "student cognition." Researchers have also argued that conceptualization of pedagogical content knowledge includes teachers' ability to recognize and address student thinking [24, 41]. Thus, it is important that measurement of CS-PCK requires teachers to respond to students' alternate conceptions, specifically addressing how they would assist the student(s).

While there is limited work on measuring how teachers respond to students' ideas in CS, there has been considerable work on student misconceptions related to programming. Most recently, Qian and Lehman (2017) conducted a literature review of student misconceptions related to programming and found that student misconceptions are related to difficulties in their syntactic knowledge, conceptual knowledge, and strategic knowledge [45]. Clancy (2004), however, cataloged a number of programming-related misconceptions, which arise from over- and under-generalization of knowledge transferred from other areas such as English, mathematics, or previous programming experiences [14]. Clancy argued that addressing these misconceptions should focus on learners' efforts to link different knowledge and encouraging knowledge integration. While we know about typical student misconceptions and factors contributing to those misconceptions, there is little understanding of teachers' knowledge on how to address those misconceptions and difficulties. This study built upon this work related to student misconceptions to measure knowledge teachers need to address them.

For instance, in their qualitative exploratory analysis of teaching vignettes to measure teacher knowledge using Scratch programming, Yadav, Berges, Sands, and Good [59] assessed 21 CS teachers' responses to teaching vignettes about students' understanding of programming constructs. To assess their PCK, an open-ended questionnaire asked them 25 teaching vignettes related to a classroom situation that CS teachers would likely encounter. Each vignette prompted participants to describe how they would address the issue. The researchers suggested that a quantitative analysis of the codes found during the qualitative content analysis of the data could lead to validating a measurement tool for CS teachers' PCK. Shulman [52] had previously argued for case-based scenarios to measure pedagogical content knowledge; at the same time, teaching vignettes have been shown to be effective tools to assess teachers' understanding of instructional strategies and professional competencies [8, 9, 29].

2 THE PRESENT STUDY

Given current efforts in expanding computer science education in K–12 and the consequent need to train teachers to teach computer science [17, 51], it is essential to examine CS teachers' pedagogical content knowledge to identify areas of improvement to develop effective teacher training in CS. In this study, we designed and administered a survey to assess CS teachers' pedagogical content knowledge using teaching vignettes to respond to the following research questions:

Table 1. Demographics of the 80 Teachers

Category	n
Gender	
Male	52
Female	26
Did not answer	2
CS teaching experience	
Low (1 year)	19
Medium (2–3 years)	42
High (at least 4 years)	19
General Teaching Experience	
Novice (≤ 5 years)	14
Intermediate (6–10 years)	21
Experienced (> 10 years)	45

- How do computer science teachers respond to an instrument designed to characterize their pedagogical content knowledge?
- How does prior experience in teaching influence teachers' computer-science pedagogical content knowledge?

3 METHOD

3.1 Participants

Eighty computer science teachers from the United States participated in the study. There were 52 male teachers and 26 female teachers (2 teachers did not answer). The sample included 75 white teachers, 2 African-American teachers, 2 Asian teachers, and 1 Native American. The majority of the teachers had been teaching computer science for the last 2–3 years, while 19 teachers had computer science 1 year or less and 19 teachers had taught the subject at least 4 years. In terms of their general teaching experience, 45 teachers had taught more than 10 years, and 35 teachers had taught less than 10 years. All teachers were teaching the same Advanced Placement Computer Science Principles course, which used Scratch and Python as the two main programming languages. Given that we wanted to study K–12 teachers' computer science pedagogical content knowledge, the participants were ideal as a target population, as they were teaching an introductory high school computer science course. Table 1 shows the demographics of the teachers.

3.2 Developing PCK Items

Our development of items to measure teachers' pedagogical content knowledge was grounded in the prior work done in assessing teachers' knowledge to teach mathematics [27]. We also drew from the work of Saeli et al. [49], who schematized and defined PCK within the context of programming using four questions: "what are the reasons to teach programming; what are the concepts we need to teach programming; what are the most common difficulties / misconceptions students encounter while learning to program; and how to teach this topic." Based on these two strands of work, we developed teaching vignettes that focused on learning difficulties students encounter when learning to program and asked teachers to address them. Computer Science at the K–12 level in the United States is primarily taught as a stand-alone subject at the secondary level and heavily focuses on programming. While the Advanced Placement CS Principles (AP-CSP) course takes a broader view of computer science, an examination of popular AP-CSP courses reveals that

Table 2. Computer Science Content Area for PCK Items

Content Area	Knowledge of Students and Content
Computational Arithmetic (Order of Operation, Casting)	X
Boolean Logic	X
Programming Topics (Variables, Conditionals, Loops, Functions, Data Structures)	X

programming still remains integrated in the course. For example, Beauty and Joy of Computing uses the Snap visual programming environment; Project Lead the Way uses Python as the programming language; and Code’s version utilizes JavaScript. Given that programming remains part of most high school introductory CS courses, our teaching vignettes were designed to investigate how teachers would respond to student programming misconceptions.

In addition to developing items based on the theoretical perspectives from prior work on student misconceptions, we also examined a popular AP-CSP curriculum and drew upon experiences of K–12 teachers to inform the development of items. Our items fell under three main constructs: computational arithmetic (order of operation and casting), Boolean logic, and programming topics (variables, conditionals, loops, functions, and data structures). We included computational arithmetic and Boolean logic given their importance in programming, as students need to understand “mathematical expressions and the precedence of operators so they can correctly code, test, and debug programs” [4]. In addition, our items covered fundamental programming topics, which form the basis when learning to program. Given that our focus was to measure teachers’ pedagogical content knowledge, our vignettes included items that measured combined knowledge of students and content. Table 2 provides an overview of the content areas covered by our instrument.

Teaching vignettes are an effective way to get rich descriptions of how teachers respond to address classroom situations and also have been used to measure their knowledge to teach subject areas (i.e., their pedagogical content knowledge) [9, 26]. To develop the vignettes, we drew upon Clancy’s [14] work on student misconceptions about programming constructs. We used Clancy’s view of programming misconceptions as ones that “arise from inappropriate transfer—overgeneralizing or undergeneralizing knowledge from other areas” (p. 87). For example, he suggested that one common source of overgeneralization is algebraic notation, such as the confusion that happens when novice learners use the assignment statement in a program, which was used in PCK 1.

Specifically, we wanted to examine how teachers would respond to students’ thinking related to programming, which served as a measure for teachers’ content knowledge to teach programming. After identifying common student misconceptions teachers likely encounter in the classroom, we worked with practicing teachers to further narrow down and identify most common programming-related misconceptions. Through an iterative process, we developed 25 teaching vignettes to measure how teachers would address student thinking around programming. The three levels in Bigg’s taxonomy [6] suggest that in order for teachers to be able to see how a student thinks and understands concepts as well as address incorrect ideas they hold (i.e., pedagogical content knowledge), teachers themselves should be able to understand those principles. Each vignette presented a computer science classroom scenario that teachers are likely to encounter when working with the students, and the participants were asked to provide an open-ended response on how they would address the problem in their own classroom. We administered the vignettes to 21 computer science teachers in the United States of America. These 21 teachers had similar backgrounds to the target population, as they taught the same high school computer science course as the teachers in this study and also had received the same professional development.

The response to the open-ended questionnaire generated a rich qualitative data set, which was analyzed using content analysis. For that purpose, we combined each participant's responses in one document and used Mayring's [35] step-by-step model of summarizing content analysis. Step 1 of the process involved "determination of the units of analysis," which was the smallest text passage containing relevant content, e.g., a word, phrase, or paragraph. During the next step, all the information in the response not critical to the research question was eliminated and we began paraphrasing of the content-bearing text passages. During Step 3, the "envisaged level of abstraction" was determined and the paraphrases were generalized. Steps 4 to 6 involved transitioning of the generalized paraphrases into a category system. Following this process, we divided each participant's response into text segments with each representing one thought of the participant. Afterwards, the text was coded deductively and categorized by paraphrasing and abstracting. As described in Reference [59], the analysis resulted in four overarching questions:

- How would the teacher address the problem?
- How would the teacher help a student understand a computer science concept?
- How does the teacher identify students' problems?
- What student reasoning does the teacher assume?

These problems were then mapped to the questions of the vignettes. All generalized, abstracted, and paraphrased answers were taken as closed-ended items. After this step, two vignettes were removed because they only contained two or less items, ending up with 23 closed-ended vignettes containing 117 items. For each item, we asked whether the participants would act as described on a Likert scale, reaching either from "definitely will" (1) to "definitely will not" (4) with "probably will" (2) and "probably will not" (3) as steps in between; or "strongly agree" (1) to "strongly disagree" (4) with "agree" (2) and "disagree" (3) in between, depending on the type of question. The following example shows how the items were developed from open to closed format. The stem of the vignette remained the same in both formats. The presented open-ended answers illustrate just some of example responses. All the items for the vignettes were constructed solely based on actual teacher responses to open-ended vignettes.

VIGNETTE:

PCK1: Phil has worked on the following Scratch example and when you ask him about the value "b," he suggested that b would be 20.

How would you help Phil understand this code segment?

SAMPLE ANSWERS:

- "I would explain code is executed line by line and ask which code affected b."
- "the program goes through the code sequentially so when b is set it is set to 10 before a is set to 20 so the output for b would be 10. if another block would be put in after the set a to 20 block that set b to a again then it would read 20."

- *“I would ask Phil to go through the code being read, as if he was an ”computer” running the program”*

CATEGORIES:

- explain what happened in the code (help to understand)
- get the student in the role of a computer (help to understand)

ITEMS:

I would...

PCK1.1: ...explain to Phil what happens in the code.

PCK1.2: ...tell Phil what he has to change in the code to meet his expectations.

PCK1.3: ...ask Phil to read through the code, as if he was a computer running the program.

PCK1.4: ...ask Phil questions until he recognizes his error.

PCK1.5: ...explain to Phil how and when variables change.

PCK1.6: ...ask Phil about his intentions when writing the code.

3.3 Procedure

Two-hundred-and-one computer science teachers were invited to complete the pedagogical content knowledge measure and 93 teachers completed the survey (a 46% response rate, which is good for online surveys [40]). The teachers all taught the same Computer Science Principles course that used Scratch and Python as the programming language. In the final analysis, we included only complete responses to all the items on the survey in the final analysis, which led to 80 teachers. Teachers were invited via email with an embedded link to complete the survey. The teachers were made aware that the survey was designed to measure knowledge teachers need to teach computer science and the results would help inform design of professional development. The teachers were compensated with \$50 Amazon gift cards for their time.

3.4 Data Preparation

For calculating a simple Rasch model, a transformation of Likert scale to dichotomous scale is necessary. In computer science education it is difficult to rate in “right or wrong” categories. So, we calculated the mean and median values for each item for rating the items on the basis of consensus. So, the participants’ answers are rated with 1 if they agree with the majority. Otherwise, the answers are rated with 0. A comparison of both matrices showed that both procedures (mean and median) result equally. Additionally, the proposed approach addresses the fact that whether an answer is right or wrong strongly depends on the investigated population (i.e., CS teachers in our study).

3.5 Data Analysis

For the optimization of the CS-PCK instrument, we used the model fit statistic ($S - X^2$) of Orlando & Thissen [42], also discussed by Reference [30]. The final itemset was modeled with a one parametric Rasch model [46]. This had the opportunity to use the row and column sums as a sufficient statistic. Nevertheless, the data has to fulfill some criteria so the Rasch model is applicable. First, the items must be homogeneous and unidimensional, which means that the items measure only one latent construct. Second, they have to be locally stochastically independent, which means that the difficulty of an item is only dependent of a person’s ability. Last, the items have to fulfill specific objectivity, which means that the order of the items concerning their difficulty does not change with changing abilities.

For the data analysis, we decided to calculate a dichotomous item-response model, applying the following algorithm for model fitting:

- (1) Calculate the 2pl model
- (2) Select one item with itemfit $< .05$ by chance
- (3) Exclude that item
- (4) If only two or less items of a vignette are left, then exclude all items of the vignette
- (5) Repeat with 1 until there are no changes in the itemset
- (6) Exclude all items with a negative discrimination and start from 1 if there are changes in the itemset

The algorithm above is run multiple times to reduce the influence of chance resulted by the Monte-Carlo method of the non-parametric tests. After that, the resulting models were examined to compare the items that were excluded from the model. For the final model, only those items were included that were not excluded in more than a half of the algorithm runs. Afterwards, the final model was improved by eliminating items that had a very small discrimination ($\alpha < .5$). The Rasch model was improved by eliminating items that had a bad model fit ($p < .05$). Starting with the 117 items in 23 vignettes, the first step (elimination due to Rasch constraint reasons) eliminated 33 items. Additionally, 8 items were eliminated due to model misfit (negative slope). The remaining items were investigated regarding their discrimination in the model. Another 7 items were eliminated in this step because of their low discrimination value. Finally, 20 items had to be removed from the item set, because the vignettes contained only 2 or less items. In the end, this process led to a final set of 49 items within 13 vignettes each with more than 2 items (PCK1-13).

4 RESULTS

The results of our study can be divided into two different aspects. First, we calculated a suitable model for the data we gathered. Second, we calculated correlations of personal information to the ability scores achieved by the participants in the test. In the beginning, we started with an itemset of 117 items gathered from the open-ended survey. Those items were put into the above described algorithm to enhance the model fit. For the model fit, first a general goodness-of-fit test described in Reference [3] was calculated. As we only have a small number of participants in comparison to the number of items, we use a bootstrap approach for the test. The test results in a p-value of .08, which indicates that the model fits the data.

An overview of the model can be seen in Figure 1, which showed all the item characteristic curves of the test. We did not label the single curves, as there are too many in the plot. Nevertheless, the figure shows the position of the curves. Resulting from the recalculation from Likert to dichotomous scale based on the median or mean of the given answers, the curves all have a difficulty below 0. So, a statement on the lacking of difficult or easy items is not possible. Nevertheless, as the scale of the Rasch model is dependent on the included items and the used algorithm, a comparison of two models is impossible anyways. The position of the items within the model is not affected, and so the Rasch model and its characteristics can still be used.

As mentioned in Section 3.5, in the Rasch model the row sum is a sufficient statistic for the participants' ability. Table 3 shows the distribution of the row sums (RS) for all participants. The smallest value was 17, the highest 49.

To provide a better overview of the person's abilities, we summarized the row sums in five groups. The following Table 4 shows the groups with the needed percentage of the maximum score, the corresponding score range, and the percentage of participants in a particular group. The distribution over the groups is provided for each experience level.

Fig. 1. Joint item characteristic curves of all included items.

Table 3. Distribution of the Row Sums (RS) for All Participants Reaching from 17 to 49

RS	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
#	0	1	0	0	0	0	1	2	1	1	1	0	2	1	0	4	2
RS	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49
#	3	0	8	4	4	7	3	1	3	5	4	4	6	2	2	4	4

Table 4. Categorization of Score Ranges and the Relative Frequencies for All Participants and Split Up on Computer Science (CS) Teaching Experience and Teaching Experience in General

Group	Score Range	Percentage of Participants	Percentage with overall of CS experience (low/med/high)	Percentage of overall teaching experiences (novice/intermediate/experienced)
A (<100%)	45–49	23%	26%/24%/16%	14%/29%/22%
B (<90%)	40–44	22%	21%/19%/26%	14%/29%/20%
C (<80%)	35–39	33%	42%/36%/16%	50%/37%/24%
D (<70%)	30–34	11%	0%/12%/21%	14%/0%/16%
E (<60%)	0–29	11%	11%/9%/21%	8%/5%/18%

The overall mean of all row scores is 38.15, which corresponds to group C. Besides the questions related to pedagogical content knowledge, we gathered several personal data, for example gender and teaching experience. To find out whether the participants differed based on their background, we compared the mean values for the subgroups built on different criteria. For that reason, we applied a Welch two-sample t-test.

We first examined whether teachers differed in their pedagogical content knowledge based on their general teaching experience. For the grouping on general teaching experience level, we grouped teachers with more than 10 years of experience as experienced ($n=45$) and those with less than 10 years as intermediate (6–10years) ($n=21$) or novices (less than 6 years) ($n=14$). The results suggested that overall there was no significant difference between the three groups on their pedagogical content knowledge. But, there were significant differences on some individual items. For a better readability, the itemstems are left out and can be seen in Appendix A. First, significant differences between the group of novices and the group of intermediate occur only in two items. Here, as expected, the more-experienced teachers have higher mean values.

PCK1.1 I would explain to Phil what happens in the code. ($p<.05$)

PCK13.2 I would ask the student questions until he recognizes the error. ($p<.05$)

Between the novices and the expert group, again, only a few items differ significantly. Surprisingly, the group of novices scores much better on average than the expert group.

PCK4.2 I would let Ruby change the code and explain the different results. ($p<.01$)

PCK8.4 I would explain the underlying mathematical content to the student. ($p<.05$)

PCK12.4 I would ask Tim to write down the values of the variables after each step. ($p<.05$)

Last, several differing items can be found between the groups of intermediates and experts. Again, the intermediates score much better on average than the prospected experts group.

PCK1.1 I would explain to Phil what happens in the code. ($p<.01$)

PCK4.1 I would ask Ruby of her understanding of the “and” expression. ($p<.01$)

PCK4.2 I would let Ruby change the code and explain the different results. ($p<.01$)

PCK4.3 I would ask Ruby to execute her code with different input values and describe the output. ($p<.01$)

PCK5.7 I would point Reggi to the error by asking him questions. ($p<.05$)

PCK8.5 I would tell the student to run the code of each statement to see the outcomes. ($p<.05$)

PCK11.3 I would let the student execute the code. ($p<.01$)

In addition to the overall experience, we also asked the participants about their experience in teaching computer science. Here, we created three groups of teachers: teachers with only 1 year of CS teaching experience (low) ($n=19$), teachers with only two to three years of CS teaching experience (mid) ($n=42$), and teachers with at least 4 years of CS teaching experience (high) ($n=19$). The results suggested that there was not an overall significant difference between the groups; however, there were significant differences on some individual items. Interestingly, all significant differences exhibited that the group with lower CS teaching experience performed better than teachers with higher CS teaching experience. Specifically, teachers with low CS experience compared to the group with mid experience differed on the following items:

PCK2.5: I would ask Emily questions until she recognizes the error. ($p<.05$)

PCK2.8: I would let Emily run the code and see what happens. ($p<.05$)

PCK10.2: I would ask Phil to evaluate only parts of the expressions. ($p<.01$)

PCK11.3: I would let the student execute the code. ($p<.05$)

The teachers with low CS experience differed in almost the same items when compared to the teachers with high CS experience:

PCK2.5: I would ask Emily questions until she recognizes the error. ($p < .05$)

PCK8.5: I would tell the student to run the code of each statement to see the outcomes. ($p < .05$)

PCK10.2: I would ask Phil to evaluate only parts of the expressions. ($p < .05$)

PCK11.3: I would let the student execute the code. ($p < .01$)

Finally, teachers with the mid CS experience differed significantly from the teachers with high CS experience on the following items:

PCK1.6: I would explain to Phil how and when variables change. ($p < .05$)

PCK3.6: I would ask Tim to run the code see what happens. ($p < .05$)

PCK4.3: I would ask Ruby to execute her code with different input values and describe the output. ($p < .05$)

PCK11.3: I would let the student execute the code. ($p < .05$)

5 DISCUSSION

The goal of this study was to design and administer a measure to characterize teachers' computer science pedagogical content knowledge performance. Given there is limited research on knowledge teachers need to teach computer science and there are no validated measures of CS-PCK, results from this study are a first step to measure knowledge needed to teach computer science. Drawing upon prior work in measuring pedagogical content knowledge [27, 59], we utilized teaching vignettes to measure teachers' CS-PCK. Results from the Rasch model suggested that teachers in our sample were able to answer the PCK items with relative ease, which suggests that our questions are measuring aspects of knowledge teachers draw upon. Furthermore, there were no significant differences between teachers with varied levels of CS teaching experience and those with general teaching experience. Using group consensus to view clusters of teacher responses as correct/incorrect showed that responses from teachers with less experience were more likely to be similar to responses from teachers with more experience. This suggests that the PCK measure used in this study assessed components of knowledge teachers need to teach computer science, as most teachers in our study were able to respond to the items. Given that our PCK items were based on prior empirical work with CS teachers, it has important implications for the use of these measures to measure learning for pre-service teachers graduating from CS education programs as well as evaluate the effectiveness of CS teacher professional development. Given the small number of teachers typically involved in professional development programs, computer science education researchers would need to also develop their models when using our measure.

Although we did not find overall significant differences between groups based on gender or teaching experience, we found some items that characterize teachers' performance within these groups. For teachers with low or high general and computer science teaching experience, there were some items where they differed significantly. It is interesting to note that for all these items, teachers with less experience would help students by letting them execute the code. A possible explanation for this could be that teachers with low teaching experience believe that giving students the opportunity to execute the code is a good strategy for debugging the code, while the experienced teachers understand that this strategy could be time-consuming when working with a whole class of students. Prior work on software development has suggested that executing the code could be a good strategy for debugging [38]; however, given that teachers find managing a class of 20–25 students is a challenge [60], working individually with students as they execute the code is not expedient.

In our study, we used teaching vignettes on misconceptions students face when learning to program and how the teacher in the scenario helped address those misconceptions. Our results suggest that teachers in our study were generally able to attend to misconceptions and the vignette teacher's response. Prior work in measuring pedagogical content knowledge has suggested that it includes experiential dimensions [47]. Rovegno found that while PCK is often discussed as knowledge that teachers need, its significance lies in teachers using it in the teaching moment, what she suggested was "pedagogical content knowing in action" (p. 79). While we attempted to measure this "PCK in action," our items might not have captured it all; thus, future works need to also include dimensions of teachers using PCK in the moment. It is also possible that our initial coding of open-ended questionnaire responses to generate close-ended items might have experienced an expert blind spot where coders' CS content knowledge might have influenced the categorizing of responses. Nathan and colleagues discussed how knowledge in a content area could lead to notions of learning that are in conflict with students' actual development process [39]. While our coding process included coders with background in teaching CS, it is possible that our own assumptions about students' understanding of programming concepts influenced the coding. Future research should carefully consider the influence of well-developed content knowledge of researchers as well as how coders might influence how teachers would respond to student programming mis(conceptions).

Developing measures that accurately measure knowledge CS teachers draw upon in their classrooms is a daunting task. Computer science educators and researchers interested in studying CS-PCK could build upon our items and explore other dimensions that need to be added. However, future work on developing items for measuring CS-PCK needs to consider potential pitfalls. As Rowan and colleagues suggested, items and scenarios used to measure PCK need to be clear and provide sufficient information to allow respondents to make informed responses [48]. The authors also stated that "too easy" or "too hard" items can also be problematic in measuring the full range of teachers' abilities and levels. Items in our study might have been too easy, as the majority of teachers were able to answer them correctly. To evaluate the effectiveness of professional development programs for beginning computer science teachers, future PCK measures would need to be sensitive enough to measure changes in teacher knowledge.

The presented approach to characterize the pedagogical content knowledge is based on the assumption that "right" or "wrong" answers in this domain are difficult to define and therefore uses an approach based on consensus within the investigated population. This approach addresses the difficulties in finding appropriate candidates for the investigation, especially finding proper experts. However, this type of analysis based on group consensus could be limited when there are few expert teachers in the sample and "average-level" responses might get prioritized over "expert-level" responses. Future work should further investigate how "expert" CS teachers respond to the PCK items and whether their answers differ from the group consensus. Specifically, researchers could ask expert CS teachers to respond to the PCK items using think-aloud interviews. This approach could help examine whether the consensus-based approach is appropriate when administering the instrument to a large sample of teachers. The think-aloud interviews would also provide a lens into what aspects of CS teaching and learning "expert" teachers focus on.

Overall, we did not find any significant differences between teachers based on their prior experience in teaching, in general as well as CS teaching experience. However, there were some items where teachers with less experience performed better than teachers with more experience: Some research has suggested that beginning teachers score higher on pedagogical content knowledge than experienced teachers [31]. Kleickmann et al. found that development of PCK happens during and immediately after teachers' education experience and then plateaus for

experienced teachers given limited learning opportunities for them. Hence, it is possible that teachers with less teaching experience have higher PCK, as they participate in more opportunities to learn during their induction years [58].

In this study, we utilized Scratch and Python vignettes with close-ended responses, which can potentially exclude additional answers and overlook individual teaching experiences or strategies. Similarly, they can introduce correct answers to participants who might not have thought of it on their own [26]. At the same time, just using open-ended items defeats the purpose of a survey, and it takes more time and skill to measure teachers’ pedagogical content knowledge. As such, future work in this area should use a multi-pronged assessment, such as interviews and classroom observations, to triangulate the data [37]. Shulman himself argued in Reference [54] that multiple measures should be used to examine teachers’ PCK. Future work could adapt our instrument in a programming-neutral format to examine CS-PCK. Finally, our instrument is limited to measuring teachers’ pedagogical content knowledge as related to programming constructs, which is relevant to the current K–12 environment; however, as K–12 computer science expands beyond programming, future research needs to develop measures that include other CS constructs.

APPENDIX
A INSTRUMENT

PCK1:

Phil has worked on the following Scratch example and when you ask him about the value “b,” he suggested that “b” would be 20.

How would you help Phil understand this code segment?		I would...			
		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK1.1:	...explain to Phil what happens in the code.	☐	☐	☐	☐
PCK1.5:	...ask Phil questions until he recognizes his error.	☐	☐	☐	☐
PCK1.6:	...explain to Phil how and when variables change.	☐	☐	☐	☐
PCK1.7:	...ask Phil about his intentions when writing the code.	☐	☐	☐	☐

PCK2:

Emily is working on a Scratch project to have a countdown from 10. Using trial and error, she's worked on the following program when I come to look it over.

How can you help Emily understand why her program does not work?

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK2.3:	...have Emily be the computer and write down her responses.	☐	☐	☐	☐
PCK2.5:	...ask Emily questions until she recognizes the error.	☐	☐	☐	☐
PCK2.8:	...let Emily run the code and see what happens.	☐	☐	☐	☐
PCK3:					

Tim examines each of the following pieces of code and responds that the first ends with x being equal to 5, and the second with x being equal to 16.

How can you help Tim to understand his misconceptions?

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK3.1:	...ask Tim questions until he recognizes the error.	☐	☐	☐	☐
PCK3.3:	...choose another form of representation to explain the conditional statement.	☐	☐	☐	☐
PCK3.4:	...explain to Tim how the conditional statement works.	☐	☐	☐	☐
PCK3.6:	...ask Tim to explain the code to me.	☐	☐	☐	☐

PCK4:

In class, the following code segment is shown on the screen. Ruby raises her hand and suggests that x will be 5 when the code segment completes.

```
when clicked
  set x to 4
  set y to 3
  if x < 10 and y > 3
    change x by 1
  else
    change y by 1
```

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK4.1:	...ask Ruby of her understanding of the “and” expression.	☐	☐	☐	☐
PCK4.2:	...let Ruby change the code and explain the different results.	☐	☐	☐	☐
PCK4.3:	...ask Ruby to execute her code with different input values and describe the output.	☐	☐	☐	☐
PCK4.4:	...let Ruby step through the code and explain each line.	☐	☐	☐	☐

PCK5:

Reggie is using the following piece of code and keeps getting an unexpected result. He feels that this code should change the value of y and not the value of x.

```
when clicked
  set x to 4
  set y to 4
  if x < 10 or y > 4 then
    change x by 1
  else
    change y by 1
```

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK5.1:	...let Reggie evaluate the code with pen and paper.	☐	☐	☐	☐
PCK5.2:	...let Reggie explain each line of code.	☐	☐	☐	☐
PCK5.5:	...let Reggie write down all values in each step.	☐	☐	☐	☐
PCK5.6:	...ask Reggie to read the code aloud.	☐	☐	☐	☐
PCK5.7:	...point Reggie to the error by asking him questions.	☐	☐	☐	☐

PCK6:

Cassandra responds to a question about the following code segment stating that the final value of *y* will be 44. The code returns 120.

```
when clicked
  set x to 4
  set y to 0
  repeat 5
    repeat 6
      set y to y + x
```

How would you help Cassandra to understand her misconception with this code?

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK6.1:	...ask Cassandra questions to let her see her error.	☐	☐	☐	☐
PCK6.4:	...let Cassandra read through the code as if she was a computer.	☐	☐	☐	☐
PCK6.5:	...provide another example that addresses the misconception.	☐	☐	☐	☐
PCK6.8:	...let Cassandra change the values of the loop and compare the different results.	☐	☐	☐	☐

PCK7:

Determine the values stored in the following variables, or state that the code segment will not compile and the reason for the error.

$a = (3 - 2)/(8/4)$
 $b = 7 + 2/3$
 $c = 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9 + 1.0/9$

A student responds with the following to the question above:

$a = 0.5$
 $b = 3$
 $c = 1$

How would you address her misconceptions?

I would...		Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK7.1:	...let the student evaluate only parts of each code segment.	☐	☐	☐	☐
PCK7.3:	...explain the underlying CS-content to the student.	☐	☐	☐	☐
PCK7.4:	...remind the student of the order of operations learning in math.	☐	☐	☐	☐
PCK7.5:	...present similar but easier examples to the student.	☐	☐	☐	☐

PCK8:

Below is a question that a student is working on: Which of the following expressions does not evaluate to 0.4?

- A. float (4 / 10)
- B. int (4.5) / float (10)
- C. float (4) / float (10)
- D. 4.0 / 10

A student responds to the question above with the choice “D.”
How would you address his misconception?

	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK8.1:	...ask the student to evaluate the code by hand.	☐	☐	☐	☐
PCK8.2:	...ask the student what happens in each expression.	☐	☐	☐	☐
PCK8.4:	...explain the underlying mathematical content to the student.	☐	☐	☐	☐
PCK8.5:	...tell the student to run the code of each statement to see the outcomes.	☐	☐	☐	☐

PCK9:

Below is a question that a student is working on: Which of the following statements evaluates to 3?

- I. $x = 7 * 4 \% 5$
- II. $y = 20 \% 8 * 3 / 4$
- III. $z = 8 + 7 \% (2 * 4) / 3$

- A. I only
- B. I and II
- C. I and III
- D. I, II, and III

A student responds with the choice “A.”
How would you address her misconception?

	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK9.1:	...ask the student to evaluate the code by hand.	☐	☐	☐	☐
PCK9.2:	...ask the student to evaluate each operation step-by-step.	☐	☐	☐	☐
PCK9.3:	...ask the student to run each statement in Python.	☐	☐	☐	☐
PCK9.4:	...explain to the student how the order of operations work.	☐	☐	☐	☐

PCK10:

Below is a question that a student is working on: Which of the following expressions evaluates to True when x and y are Boolean variables with different values?

- I. $(x \text{ or } y) \text{ and not } (x \text{ and } y)$
- II. $(x \text{ or } y) \text{ and } (\text{not } x \text{ or not } y)$
- III. $(x \text{ and not } y) \text{ or } (\text{not } x \text{ and } y)$

- A. I only
- B. I and II
- C. II and III
- D. I, II, and III

Phil responds with the choice “C” but admits later that he wasn’t sure how to approach this question.

How would you work with Phil to address his needs?

	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK10.2:	...ask Phil to evaluate only parts of the expressions.	☐	☐	☐	☐
PCK10.3:	...explain the content of Boolean logic to Phil.	☐	☐	☐	☐
PCK10.4:	...ask Phil to set real values to the variables and evaluate the expressions.	☐	☐	☐	☐
PCK10.5:	...ask Phil to write down the values of each part of the expressions and evaluate the complete expression afterwards.	☐	☐	☐	☐

PCK11:

Below is a question that a student is working on: How many times will the innermost statement execute in the following code segment?

```
for x in range (0,7):
    for y in range (3,10):
        print (x + y)
```

- A. 16
- B. 49
- C. 56
- D. 64
- E. 70

A student responds with the choice “D.”

How would you address his misconception?

	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK11.1:	...let the student change the code and compare the different outputs.	☐	☐	☐	☐

PCK11.2: ...let the student evaluate only parts of the code. ☐ ☐ ☐ ☐

PCK11.3: ...let the student execute the code. ☐ ☐ ☐ ☐

PCK12:

Consider the following two codes that Tim has been working on and believes that both should produce the same output:

Code 1:

```
def printCountNTimes(n):
    count = 0
    while (count < n):
        print ('The count is: ', count )
        count = count + 1
printCountNTimes (8)
```

Code 2:

```
def printCountNTimes(n):
    count = 0
    while (count < n):
        print ('The count is: ', count )
        count = count + 1
printCountNTimes (4)
printCountNTimes (4)
```

How would you help Tim understand the difference between the two codes?

	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK12.1:	...let Tim evaluate only the initializing of the count variable.	☐	☐	☐	☐
PCK12.2:	...present the code twice. One time for each call of printCountNTimes(4).	☐	☐	☐	☐
PCK12.4:	...ask Tim to write down the values of the variables after each step.	☐	☐	☐	☐

PCK13:

Below is a question that a student is working on: Which of the following best describes the value of sum2 output by doSomething?

```
def doSomething(myList):
    sum1 = 0
    sum2 = 0
    for num in myList:
        if num > 0 or (num % 2) != 0:
            sum1 += num
        elif (num % 2) == 0:
            sum2 += num
    print(sum1)
    print(sum2)
```


	A. The sum of all negative values in myList				
	B. The sum of all negative even values in myList				
	C. The sum of all even values in myList				
	D. The sum of all negative values and positive even values in myList				
	A student responds with the choice "C."				
	How would you address his misconception?				
	I would...	Definitely Will	Probably Will	Probably Will Not	Definitely Will Not
PCK13.1:	...let the student walk through the code line-by-line	☐	☐	☐	☐
PCK13.2:	...ask the student questions until he recognizes the error.	☐	☐	☐	☐
PCK13.3:	...explain the underlying content to the student	☐	☐	☐	☐

ACKNOWLEDGMENT

We would like to thank all the teachers who participated in this study.

REFERENCES

[1] Patricia Teague Ashton. 1990. Editorial. *J. Teach. Ed.* 41, 3 (1990), 2. DOI: <https://doi.org/10.1177/002248719004100301>

[2] Valerie Barr and Chris Stephenson. 2011. Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community? *ACM Inroads* 2, 1 (2011), 48–54. DOI: <https://doi.org/10.1145/1929887.1929905>

[3] David J. Bartholomew. 2008. *Analysis of Multivariate Social Science Data* (2nd ed.). CRC Press, Boca Raton, FL.

[4] Theresa Beaubouef. 2002. Why computer science students need math. *SIGCSE Bulletin* 34, 4 (2002), 57–59.

[5] Elena Bender, Peter Hubwieser, Niclas Schaper, Melanie Margaritis, Marc Berges, Laura Ohrndorf, Johannes Magenheimer, and Sigrid Schubert. 2015. Towards a competency model for teaching computer science. *Peabody J. Ed.* 90, 4 (2015), 519–532. DOI: <https://doi.org/10.1080/0161956X.2015.1068082>

[6] John Biggs. 1999. What the student does: Teaching for enhanced learning. *Higher Ed. Res.Dev.* 18, 1 (1999), 57–75. DOI: <https://doi.org/10.1080/0729436990180105>

[7] Janina A. Bindernagel and Ingo Eilks. 2009. Evaluating roadmaps to portray and develop chemistry teachers’ PCK about curricular structures concerning sub-microscopic models. *Chem. Ed. Res. Pract.* 10, 2 (2009), 77–85. DOI: <https://doi.org/10.1039/B908245J>

[8] Andreas Borowski, J. Carlson, Hans Fischer, Ineke Henze, Julie Gess-Newsome, and Sophie Kirschner. 2012. Different models and methods to measure teachers’ pedagogical content knowledge. In *E-Book Proceedings of the ESERA 2011 Conference: Science Learning and Citizenship*, C. Bruguière, A. Tiberghien, and P. Clément (Eds.). Lyon, France.

[9] Dorothee Brovelli, Katrin Bölsterli, Markus Rehm, and Markus Wilhelm. 2014. Using vignette testing to measure student science teachers’ professional competencies. *Amer. J. Ed. Res.* 2, 7 (2014), 555–558. DOI: <https://doi.org/10.12691/education-2-7-20>.

[10] Margret Buchmann. 2006. The flight away from content in teacher education and teaching. *J. Curr. Studies* 14, 1 (2006), 61–68. DOI: <https://doi.org/10.1080/0022027820140106>

[11] William S. Carlsen. 1987. Why do you ask? The effects of science teacher subject-matter knowledge on teacher questioning and classroom discourse. In *Proceedings of the Meeting of the American Educational Research Association*.

[12] Anthony P. Carnevale, Nicole Smith, and Michelle Melton. 2014. *STEM*. Georgetown University - Center on Education and the Workforce. Retrieved from <https://georgetown.app.box.com/s/cyrrqbjiyrjy64uw91f6>.

[13] J. Century, M. Lach, H. King, S. Rand, C. Heppner, B. Franke, and J. Westrick. 2013. Building an operating system for computer science. University of Chicago’s Center for Elementary Mathematics and Science Education. Retrieved from <http://outlier.uchicago.edu/computerscience/OS4CS/>.

[14] M. Clancy. 2004. Misconceptions and attitudes that interfere with learning to program. In *Computer Science Education Research*, S. Fincher and M. Petre (Eds.). Taylor & Francis, London, UK, 85–100.

[15] Kathryn F. Cochran, R. A. King, and J. A. DeRuiter. 1991. Pedagogical content knowledge: A tentative model for teacher preparation. In *Proceedings of the Meeting of the American Educational Research Association*.

- [16] CSTA. 2011. Computational thinking: Leadership toolkit. Computer Science Teachers Association. Retrieved from <http://www.csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershipToolkit-SP-vF.pdf>.
- [17] CSTA. 2013. Bugs in the system: Computer science teacher certification in the U.S. Computer Science Teachers Association. Retrieved from http://csta.acm.org/ComputerScienceTeacherCertification/sub/CSTA_BugsInTheSystem.pdf.
- [18] CSTA. 2015. National secondary school computer science survey. Google. Retrieved from http://www.csta.acm.org/Research/sub/Projects/ResearchFiles/CSTA_NATIONAL_SECONDARY_SCHOOL_CS_SURVEY_2015.pdf.
- [19] Jan Cuny. 2012. Transforming high school computing: A call to action. *ACM Inroads* 3, 2 (2012), 32–36. DOI: <https://doi.org/10.1145/2189835.2189848>
- [20] Judith Gal-Ezer and Chris Stephenson. 2010. Computer science teacher preparation is critical. *ACM Inroads* 1, 1 (2010), 61–66. DOI: <https://doi.org/10.1145/1721933.1721953>
- [21] Google. 2015. Searching for computer science: Access and barriers in U.S. K–12 education. Retrieved from https://services.google.com/fh/files/misc/searching-for-computer-science_report.pdf.
- [22] Pamela L. Grossman, Suzzane M. Wilson, and Lee S. Shulman. 2005. Teachers of substance: Subject matter knowledge for teaching. *Prof., Rev. Curric. Form. Prof.* 9, 2 (2005), 1–25. <https://recyt.fecyt.es/index.php/profesorado/article/download/42833/24723>.
- [23] Sigrun Gudmundsdottir. 1987. Pedagogical content knowledge: Teachers’ ways of knowing. In *Proceedings of the Meeting of the American Educational Research Association*.
- [24] Orit Hazzan, Tami Lapidot, and Noa Ragonis. 2014. *Guide to Teaching Computer Science: An Activity-based Approach*. Springer, New York.
- [25] Heather C. Hill, Deborah Loewenberg Ball, Merrie Blunk, Imani Masters Goffney, and Brian Rowan. 2007. Validating the ecological assumption: The relationship of measure scores to classroom teaching and student learning. *Meas.: Interdisc. Res. Persp.* 5, 2–3 (2007), 107–118. DOI: <https://doi.org/10.1080/15366360701487138>
- [26] Heather C. Hill, Deborah Loewenberg Ball, and Stephen G. Schilling. 2008. Unpacking pedagogical content knowledge: Conceptualizing and measuring teachers’ topic-specific knowledge of students. *J. Res. Math. Ed.* 39, 4 (2008), 372–400.
- [27] Heather C. Hill, Stephen G. Schilling, and Deborah Loewenberg Ball. 2004. Developing measures of teachers’ mathematics knowledge for teaching. *Elem. School J.* 105, 1 (2004), 11–30. DOI: <https://doi.org/10.1086/428763>
- [28] Peter Hubwieser, Johannes Magenheimer, Andreas Mühling, and Alexander Ruf. 2013. Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the 9th International ACM Conference on International Computing Education Research*. ACM Press, New York, NY, 1–8. DOI: <https://doi.org/10.1145/2493394.2493395>
- [29] Carolyn Jeffries and Dale W. Maeder. 2005. Using vignettes to build and assess teacher understanding of instructional strategies. *Profess. Ed.* 27 (2005), 17–28. <http://files.eric.ed.gov/fulltext/EJ728478.pdf>.
- [30] Taehoon Kang and Troy T. Chen. 2011. Performance of the generalized S-X2 item fit index for the graded response model. *Asia Pacific Education Review* 12, 1 (2011), 89–96. DOI: <https://doi.org/10.1007/s12564-010-9082-4>
- [31] Thilo Kleickmann, Dirk Richter, Mareike Kunter, Jürgen Elsner, Michael Besser, Stefan Krauss, and Jürgen Baumert. 2013. Teachers’ content knowledge and pedagogical content knowledge. *J. Teach. Ed.* 64, 1 (2013), 90–106. DOI: <https://doi.org/10.1177/0022487112460398>
- [32] Stefan Krauss, Martin Brunner, Mareike Kunter, Jürgen Baumert, Werner Blum, Michael Neubrand, and Alexander Jordan. 2008. Pedagogical content knowledge and content knowledge of secondary mathematics teachers. *J. Ed. Psych.* 100, 3 (2008), 716–725. DOI: <https://doi.org/10.1037/0022-0663.100.3.716>
- [33] John K. Lannin, Matthew Webb, Kathryn Chval, Fran Arbaugh, Sarah Hicks, Cynthia Taylor, and Rebecca Bruton. 2013. The development of beginning mathematics teacher pedagogical content knowledge. *J. Math. Teach. Ed.* 16, 6 (2013), 403–426. DOI: <https://doi.org/10.1007/s10857-013-9244-5>.
- [34] Melanie Margaritis, Johannes Magenheimer, Peter Hubwieser, Marc Berges, Laura Ohrndorf, and Sigrid Schubert. 2015. Development of a competency model for computer science teachers at secondary school level. In *Proceedings of the IEEE Global Engineering Education Conference*. IEEE Press, Los Alamitos, CA, 211–220. DOI: <https://doi.org/10.1109/EDUCON.2015.7095973>
- [35] Philipp Mayring. 2014. *Qualitative Content Analysis: Theoretical Foundation, Basic Procedures and Software Solution*. Klagenfurt. <http://nbn-resolving.de/urn:nbn:de:0168-ssaoar-395173>.
- [36] Punya Mishra and Matthew J. Koehler. 2006. Technological pedagogical content knowledge: A framework for teacher knowledge. *Teach. Colle. Rec.* 108, 6 (2006), 1017–1054.
- [37] Ann Morrison and Kathleen Luttenegger. 2015. Measuring pedagogical content knowledge using multiple points of data. *Qual. Rep.* 20, 6 (2015), 804–816. <http://nsuworks.nova.edu/tqr/vol20/iss6/6>.
- [38] Laurie Murphy, Gary Lewandowski, Renée McCauley, Beth Simon, Lynda Thomas, and Carol Zander. 2008. Debugging. In *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*. ACM Press, New York. DOI: <https://doi.org/10.1145/1352135.1352191>

- [39] Mitchell Nathan, Kenneth Koedinger, and Martha Alibali. 2001. Expert blind spot: When content knowledge eclipses pedagogical content knowledge. In *Proceedings of the 3rd International Conference on Cognitive Science*. USTC Press, Beijing, China.
- [40] Duncan D. Nulty. 2008. The adequacy of response rates to online and paper surveys: What can be done? *Assess. Eval. High. Ed.* 33, 3 (2008), 301–314. DOI: <https://doi.org/10.1080/02602930701293231>
- [41] Laura Ohrndorf and Sigrid Schubert. 2014. Students' cognition: Outcomes from an initial study with student teachers. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE'14)*. ACM, New York, NY, 112–115. DOI: <https://doi.org/10.1145/2670757.2670758>.
- [42] Maria Orlando and David Thissen. 2000. Likelihood-based item-fit indices for dichotomous item response theory models. *Appl. Psych. Meas.* 24, 1 (2000), 50–64. DOI: <https://doi.org/10.1177/01466216000241003>.
- [43] Soonhye Park, Jeong-Yoon Jang, Ying-Chih Chen, and Jinhong Jung. 2011. Is pedagogical content knowledge (PCK) necessary for reformed science teaching? Evidence from an empirical study. *Res. Sci. Ed.* 41, 2 (2011), 245–260. DOI: <https://doi.org/10.1007/s11165-009-9163-8>.
- [44] Geoffrey Phelps and Stephen Schilling. 2004. Developing measures of content knowledge for teaching reading. *Elem. School J.* 105, 1 (2004), 31–48. DOI: <https://doi.org/10.1086/428764>.
- [45] Yizhou Qian and James Lehman. 2017. Students & misconceptions and other difficulties in introductory programming: A literature review. *ACM Trans. Comput. Educ.* 18, 1, Article 1 (Oct. 2017), 24 pages. DOI: <https://doi.org/10.1145/3077618>.
- [46] G. Rasch. 1980. *Probabilistic Models for Some Intelligence and Attainment Tests*. University of Chicago Press, Chicago.
- [47] Inez C. Rovegno. 1992. Learning to teach in a field-based methods course: The development of pedagogical content knowledge. *Teaching. Teach. Ed.* 8, 1 (1992), 69–82. DOI: [https://doi.org/10.1016/0742-051X\(92\)90041-Z](https://doi.org/10.1016/0742-051X(92)90041-Z).
- [48] Brian Rowan, Steven G. Schilling, Deborah L. Ball, and Robert Miller. 2001. *Measuring Teachers' Pedagogical Content Knowledge in Surveys: An Exploratory Study*. Consortium for Policy Research in Education. University of Michigan, Ann Arbor.
- [49] Mara Saeli, Jacob Perrenet, Wim M.G. Jochems, and Bert Zwaneveld. 2012. Programming: Teachers and pedagogical content knowledge in the Netherlands. *Inform. Ed.* 11, 1 (2012), 81–114.
- [50] Stephan Schmelzing, Jan H. van Driel, Melanie Jüttner, Stefanie Brandenbusch, Angela Sandmann, and Birgit J. Neuhaus. 2013. Development, evaluation, and validation of a paper-and-pencil test for measuring two components of biology teachers' pedagogical content knowledge concerning the "cardiovascular system." *Int. J. Sci. Math. Ed.* 11, 6 (2013), 1369–1390. DOI: <https://doi.org/10.1007/s10763-012-9384-6>.
- [51] Cynthia C. Selby. 2015. Relationships: Computational thinking, pedagogy of programming, and Bloom's taxonomy. In *Proceedings of the 10th Workshop in Primary and Secondary Computing Education (WiPSCE'15)*, Judith Gal-Ezer, Sue Sentance, and Jan Vahrenhold (Eds.). The Association for Computing Machinery, New York, New York, 80–87. DOI: <https://doi.org/10.1145/2818314.2818315>.
- [52] Lee S. Shulman. 1986. Those who understand: Knowledge growth in teaching. *Ed. Res.* 15, 2 (1986), 4–14.
- [53] Lee S. Shulman. 1987. Knowledge and teaching: Foundations of the new reform. *Harvard Ed. Rev.* Feb. (1987), 1–22.
- [54] Lee S. Shulman. 1988. A union of insufficiencies: Strategies for teacher assessment in a period of educational reform. *Educational Leadership* 46, 3 (1988), 36–41.
- [55] D. Strickland. 2014. L. A. Unified Announces Larger Focus on Computer Science for K-12. <http://home.lausd.net/apps/news/article/407400>
- [56] K. Taylor and C. C. Miller. 2015, September 15. De Blasio to announce 10-year deadline to offer computer science to all students. *The New York Times* (2015, September 15). <http://www.nytimes.com/2015/09/16/nyregion/de-blasio-to-announce-10-year-deadline-to-offer-computer-science-to-all-students.html>.
- [57] Jan H. van Driel, Nico Verloop, and Wobbe de Vos. 1998. Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching* 35, 6 (1998), 673–695. DOI: [https://doi.org/10.1002/\(SICI\)1098-2736\(199808\)35:6<673::AID-TEA5>3.0.CO;2-J](https://doi.org/10.1002/(SICI)1098-2736(199808)35:6<673::AID-TEA5>3.0.CO;2-J)
- [58] Jian Wang, Sandra J. Odell, and Sharon A. Schwille. 2008. Effects of teacher induction on beginning teachers' teaching: A critical review of the literature. *Journal of Teacher Education* 59, 2 (2008), 132–152. DOI: <https://doi.org/10.1177/0022487107314002> arXiv:<https://doi.org/10.1177/0022487107314002>
- [59] Aman Yadav, Marc Berges, Phil Sands, and Jon Good. 2016. Measuring computer science pedagogical content knowledge: An exploratory analysis of teaching vignettes to measure teacher knowledge. In *Proceedings of the 11th Workshop in Primary and Secondary Computing Education*, Jan Vahrenhold and Erik Barendsen (Eds.). ACM Press, New York, New York, USA, 92–95. DOI: <https://doi.org/10.1145/2978249.2978264>
- [60] Aman Yadav, Sarah Gretter, Susanne Hambrusch, and Phillips Sands. 2016. Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education* 26, 4 (2016), 235–254. DOI: <https://doi.org/10.1080/08993408.2016.1257418>

- [61] Aman Yadav, Hai Hong, and Chris Stephenson. 2016. Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends* 60, 6 (2016), 565–568. DOI: <https://doi.org/10.1007/s11528-016-0087-7>
- [62] Aman Yadav, Phil Sands, Jon Good, and Alex Lishinki. 2018. Computer science and computational thinking in the curriculum: Research and practice. In *Handbook of Information Technology in Primary and Secondary Education*, Joke Voogt and Gerald Knezek (Eds.). Springer. DOI: https://doi.org/10.1007/978-3-319-53803-7_6-1

Received February 2018; revised November 2018; accepted January 2019