Computer-Science Education as a Cultural Encounter: 
A Socio-cultural Framework for Articulating Learning Difficulties 

Yifat Ben-David Kolikant
School of Education
Hebrew University of Jerusalem, Jerusalem
Israel 91905
Tel: 972-2-5882056
yifatbdk@mscc.huji.ac.il

ABSTRACT
We present a framework for articulating learning difficulties in computer science (CS) based on the socio-cultural theoretical idea that learning means entering a culture. In school, teachers and students participate in two cultures simultaneously: (1) school and (2) the domain studied. CS students are members of a third culture, computer users and thus, in CS lessons, three cultural viewpoints are employed simultaneously, which might recast students' understanding of CS with unauthentic CS elements. The power of this framework was demonstrated in a three-phase investigation into difficulties regarding correctness using questionnaires. The first two phases revealed how both school and the user's culture nurture students' misconceptions of correctness, which contribute to students' inadequate work habits. The third phase exposed teachers' dual viewpoint on a programming activity, which students can misinterpret as agreement with their (mis)understanding of the concept.

Categories and Subject Descriptors
K.3.2 [Computers and Education]: Computer and Information Science Education – computer science education.

General Terms
Human Factors

Keywords
Sociocultural theories, correctness, learning difficulties

1. INTRODUCTION
Currently in CS education research, especially regarding learning difficulties, the dominant perspective regarding learning and knowledge is cognitive, as demonstrated, for example, in two recent reviews of the state of the art of research in CS education [20, 22]. Most of these studies concern knowledge of expert and novice programmers, with emphasis on novice difficulties [23]. In addition, there has been emphasis on determining the knowledge needed to bring about effective strategies and in identifying the resulting abilities that emerge [19]. Recently, however, interest has been growing regarding the potential of the socio-cultural perspective on learning to advance research in CS education [1, 9, 13, 14, 15]. This theoretical framework encourages a shift in the foci of CS education research projects to elucidating the processes students undergo as they study, in the hope that this will provide additional explanations of their behavior.

Here I demonstrate the usefulness of such an approach to explore students' learning difficulties, focusing on the concept of correctness. The investigation included three phases. The two first phases focused on revealing students' difficulties, focusing on the concept of correctness. The investigation included three phases. The two first phases focused on revealing students' difficulties and were published elsewhere. The third focused on the multiple viewpoint teachers employ, and its results will be presented and discussed.

2. THE SOCIO-CULTURAL THEORETICAL FRAMEWORK
According to the socio-cultural framework, learning is a process of enculturation into a culture or a community of practice by newcomers' participation in peripheral yet legitimate, genuine activities of the target culture. The newcomers gradually achieve a holistic viewpoint of professional practice and increase their participation, ultimately becoming full-fledged participants [16, 25]. Viewpoint is defined by Schoenfeld as valuing the processes professionals carry out and having the predilection to apply them [21]. There is a reciprocal relationship between learning and cultural viewpoint. One's cultural viewpoint determines what one classifies as important and what as unimportant in any situation. Yet, through continuous learning, namely, participation in another culture, viewpoints change or evolve [25]. For example, it is hard to convey the importance of efficiency and documentation in the context of short and simple programs, whereas as one's programming experience evolves, appreciation of these concepts and the activities they encompass increases.

Conceptual knowledge can be viewed metaphorically as tools. By using a tool, one develops a rich understanding of it and its use, as well as the environment in which it is used [8]. Thus, a learner would judge best the productivity of a tool by using it in its natural context. Moreover, situations in which tools are used, which includes elements that are not aligned with the authentic use of the tool, might affect students' "rich understanding" of the
tools, consequently causing the students to use the tools inadequately.

3. COMPUTER SCIENCE EDUCATION AS CULTURAL ENCOUNTERS
Examination of school activities through a socio-cultural "lens" suggests that in any domain, the teacher and the student are active in two cultures simultaneously, or metaphorically, each of them wears two hats simultaneously. The first hat is the hat of the target community of practice. The teacher, like the entire instructional setting, represents the community of the studied practice, and the student is a newcomer to that community of practice [26]. However, school is a cultural environment by itself. Brousseau [7] defines school activity as creating and playing didactical games, designed by the teachers according to the knowledge they want to devote, with consideration of the students they teach. The game is played by both the teacher and the students, each side taking the other side into account. Learning is achieved when the student develops strategies to win the game. Hence, the other hat that the teacher and students share is the hat of participants in a school milieu.

The uniqueness of computer science education. CS students wear a third hat, that of a computer user or local developer. Their viewpoint regarding the CS world—what constitutes a good problem, an accountable approach to a solution, and a satisfactory solution of the problem—were shaped without much interaction with the other two "professional" CS cultures, the CS academia and industry [2, 4]. A user is defined by Turkle as someone who is "involved in the machine in a hands-on way, but is not interested in the technology except as it enables an application" [24, p.32]. Often the borderline between users and programmers is not clear-cut [12, 17]. In fact, it is all a matter of the context. For example, a software engineer may use a software package, such as an image editor, to prepare a presentation in a manner consistent with the user's perspective.

CS education as a cultural encounter. Thus, CS education can be depicted as an encounter of two computer-literate cultures hosted by a third culture, school [2, 4]. The curriculum is oriented toward the academic community, whose understanding of the computer world involves the abstraction, solution, and proof of algorithmic problems, whereas many students are veterans in the world of computers; this shapes their understanding of and interests in this world. Furthermore, as old-timers in this world, they might delegitimize the CS curriculum as relevant to their CS life, which ultimately leads to a culture clash.

Within this framework, students' tendency to maintain work habits deemed inadequate by CS professionals (e.g. [10, 18]) can be explained by the productivity of these work habits in similar past situations of computer use. The situations are similar because in both situations the students' goal is to make programs work, i.e., to solve problems.

4. CULTURAL ENCOUNTER AS A FRAMEWORK FOR IDENTIFYING CONCEPTUAL DIFFICULTIES
The metaphor of cultural encounters can also aid in designing an investigation into students' conceptual difficulties in CS, as demonstrated here. Our focus was on the concept of correctness and related practices. The underlying theoretical idea is that misconceptions occur when multiple viewpoints on CS are mixed in one's understanding of the discipline.

4.1 The Influence of the User Culture
As a first step, we explored misconceptions that originate in contradictions between the user's viewpoint and the programmer's viewpoint [3]. Therefore, in the experiment the students were given three programs that a user would judge as 'working', because they produced the expected output; yet a programmer would judge these programs as incorrect because they produced unexpected output as well for part of the input space. This judgment is in agreement with the dichotomous definition of correct programs, as defined in the textbook of [11], according to which correct programs are only those that produce the expected output (all the expected output and nothing but it) for all the legal input space. Programs that have a different relationship between the input and output are considered incorrect [23].

The students were asked to decide about the program's correctness. A large proportion of high-school and college students decided that the programs were relatively correct (up to 13% and 38% of the high-school students and college students, respectively) and a noteworthy number of them (up to 16% and 30% of the high-school students and college students, respectively) were indecisive regarding whether the programs were correct or not. Analysis of their explanations revealed that these decisions were influenced not only by the school viewpoint but also by the user viewpoint, for example "The program (assignment 2) is not perfect but it works and that's what counts."

The students' conceptual "tool" (i.e., their conception of relative correctness) was consistent with students' work habits. Specifically, it was found that the students' norm was to tolerate the existence of errors in their programs because they were satisfied with programs that 'worked in general' or for 'many input examples' [4, 6].

4.2 The Influence of School Culture
School also influences students' conceptual understanding, as we found in the second step of our investigation [5]. We asked 159 high-school students from six high schools to judge the correctness of six programs. We also interviewed seven students about their decision.

This time we gave the students programs that a user would judge as 'not working' because they definitely did not produce the expected output. Yet many students judged the erroneous programs as partially correct, although they detected some errors in them. During the interviews, the students explained that they found a 'grain of correctness' in the programs, that is, something in the code that was written correctly. In fact, they referred to program correctness as the sum of the correctness of the code constituents and thus, only on rare occasions were programs judged incorrect because it meant that no single operation was written correctly.

This misconception was probably not due to students' experience as users. User experience is usually associated with students' tendency to make decisions based solely on interaction at the interface level with the program. In contrast, the students' decisions regarding correctness were based on their inspection of
the code. However, the students' decisions were also not governed by the definitions found in textbooks. If they were, students would have promptly decided that the program is incorrect once they had detected an error.

Instead, the students thought in terms of grades. They linked two similar situations, since in both situations the program's quality is judged. Yet, the situations are different. Traditional grading is usually done by dividing the program into segments: each is responsible for a sub-goal of the program and each segment is graded independently. Thus, the program's grade is the (weighted) sum of the points given to each component of the code. In contrast, decisions regarding correctness should be made using a global viewpoint of the program and in an absolute manner. Thus, students mistakenly adopted the local viewpoint used in grading situations also in situations where they had to decide about correctness.

In the rest of this paper, the consecutive steps of this project are described. We sought to empirically ensure that teachers do not directly teach this misconception to their students; however, there are elements in their voice resulting from their use of a "school-member" perspective, which might contribute to students' confusion. The methods by which we pursued these questions and the results are presented in the next sections.

5. METHODS
We gave 34 CS teachers the same three programs originally used to examine the influence of the user's perspective on students [2]. This time, however, they were used to explore the influence of the teachers' "school hat" on their decisions and discourse.

The questionnaire. The questionnaire included three assignments. In each assignment we described an incorrect program and described the output from which one can learn that the program is incorrect. In all the assignments, the mistake was that unexpected output was displayed in addition to the expected output. For each program, we gave three statements; the participants had to respond to each of them whether they agreed, disagreed, or otherwise, and we provided room for comments.

The statements were as follows: (a) the program is correct, (b) the program is incorrect, and (c) the program is correct if the output does not distract from getting the required information (for assignment 1 we used a different version: the program is correct for most cases).

In assignment 1 we gave the following simple if-then program code:

```
if X < 20 then println("low")
if X < 60 then println("medium")
else println("high");
```

We also specified the goals, which were to display different output according to the three categories of the value of input X (those smaller than 20, those equal or larger than 60, and those between). The program was incorrect because for one group of inputs it produced the expected value, yet owing to incorrect nesting of the if-sentence, it also produced one more unexpected output. The testing results of input examples taken from this group and other groups were also provided.

Assignment 2 was phrased as follows: You developed a very complicated program that should display hundreds of outputs. The program displayed all the output you expected to get, but also in the end displayed one output item that does not suit the program's requirements.

Finally, assignment 3 was as follows: At your request, you developed a program that produces information about your family. When you supplied your family data and asked for the names of all your cousins, the program displayed the names of all your cousins, but in addition, in the end you got the name of one of your uncles.

Data analysis. The responses were ranked as follows: All programs given were incorrect. Therefore, the ideal response was to disagree with the statement 'the program is correct' and to agree that the program is incorrect. In addition, since there is no such concept "relatively correct" in the curriculum, they should have disagreed with this statement. This response was given the highest score of 4.

The next best response (rank 3) was given to those who understood that the program is incorrect, yet also agreed that it is relatively correct. A worse response (and thus ranked 2) would be to consider the program to be correct. Finally, rank 1 was given to indecisive responses, that is, responses where "otherwise," was checked, as well as responses in which one either agreed or disagreed that a program was both correct and incorrect.

In addition, the following qualitative analysis of the teachers' explanations of their decisions was conducted. The teachers' explanations were categorized according to the "hats" evident in these utterances, either that of school, users, or programmers. A programmer-driven utterance would be that the definition of correctness is not fulfilled and thus the program is incorrect. User-driven comments would highlight the fact that the expected output appeared on the screen and thus the program is either correct or somewhat correct. School-driven comments were those in which the closeness of the solution to the desired one is evaluated. An example of a school-driven comment is: "Not exactly. You need to add exactly one more word," by which one can hear a teacher addressing an imaginary student whose program was reviewed. This phase of analysis was carried out by two researchers separately and was followed up by a meeting to resolve inconsistencies.

6. FINDINGS
The distribution of the ranks of teacher's responses (n=34) is presented in Table 1. The table also contains the distribution of the ranks of the responses of two groups of students to the same questionnaires obtained in the first step of this research project, described above [3]. One group is of college students (n=35) and one is of high-school students (n=24). Then, a qualitative analysis of the teachers' explanations is presented.

6.1 Analysis of Teachers' Decisions
The decisions of the majority of the teachers for all programs were in agreement with the correctness definition in the textbook. Specifically, 79%, 67%, and 67% of the teachers' responses to programs 1, 2, and 3, respectively, were ranked 4. Therefore, I concluded that most teachers wore the programmers' hat when making the decisions.

A one-way ANOVA test revealed a significant difference among the three groups (F(2,92)= 4.991, p<0.01). A post-hoc Scheffe test
revealed that the teachers' performed significantly different from the college students. The responses of the high-school students were not statistically different from those of the other two groups. This difference implies that the programmers' viewpoint dominated the teachers' decision regarding program 1 more than it dominated the students' decisions. However, the groups' responses to programs 2 and 3 were not statistically different. A one-way ANOVA test revealed a significant difference among the three groups in their responses to program 2, but the Scheffe test found no significant difference among the groups.

Table 1: the groups' performance

<table>
<thead>
<tr>
<th></th>
<th>Program 1</th>
<th>Program 2</th>
<th>Program 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>M</strong></td>
<td><strong>SD</strong></td>
<td><strong>M</strong></td>
<td><strong>SD</strong></td>
</tr>
<tr>
<td>Teachers</td>
<td>3.618</td>
<td>0.888</td>
<td>3.324</td>
</tr>
<tr>
<td>Highschool</td>
<td>3.208</td>
<td>0.833</td>
<td>3.250</td>
</tr>
<tr>
<td>College</td>
<td>2.886</td>
<td>1.105</td>
<td>2.657</td>
</tr>
<tr>
<td>F(2,92)</td>
<td>4.991**</td>
<td>3.289*</td>
<td>2.163</td>
</tr>
</tbody>
</table>

*p<0.05, **p<0.01

The responses to the three programs were compared within each group as well. The group of high-school students responded significantly different to programs 2 and 3, although the only difference between these programs was their "cover story". The college group showed a similar tendency but was not statistically significant (p=.079). In contrast, the teachers judged these programs in a similar manner. In fact, in both programs, 67% of the teachers' responses were ranked 4 (i.e., decided that the program is incorrect), and 15% were ranked 3, and the rest, 18%, were ranked 1 or 2. No other statistical differences within the groups were found.

Interestingly, the cover story of program 3, which concerned the students' own family, led them to judge the program's correctness in more meticulous standards. This implies that there are subjective factors that influence students' tolerance to incorrect I/O, namely to their standards of correctness. Fortunately, this factor does not influence teachers' decisions.

6.2 Qualitative Analysis of Teachers' Discourse

Qualitative analysis of teachers' reasoning revealed that the teachers' agreement and disagreement with the concept of relative correctness were driven by two different, yet not contradictory rationales. Teachers who disagreed with the idea of relative correctness referred to the definition of correctness, emphasizing its dichotomous nature if correct, and pointed out that the program does not fulfill the definition, for example: "a correct program must always work! For every valid input, it should fulfill its requirements!" and "program correctness is black or white, correct or incorrect." This rationale reflects a programmer's viewpoint.

Teachers, who to a certain extent, agreed with the idea of relative correctness—namely, teachers who checked "otherwise" for either one of the statements 'the program is incorrect' and the 'program is correct,' or agreed with the statement that the program was relatively correct—provided a different rationale. They referred to the small effort it would take to fix the program. Moreover, the utterances seem to be derived from a conversation with a student in a situation where the teacher explains to the student how to fix the program. In other words, these explanations were driven from a school viewpoint. Note, for instance, that the discourse is formulated in terms of instructions. Examples:

1. for program 1: "not exactly. You need to add exactly one more word," "the program is almost correct. It misses just one thing;"
2. for program 2: "it must be a little bug that needs to be fixed," "we need to check why the extra number was printed", "you need to define the rule more accurately";
3. for program 3: "maybe it is correct and maybe not. We need to know what the source of the error is. If it is a minor mistake because of sloppiness, then the program is correct, globally speaking"

The user 'hat' was not prominent in the teachers' discourse. Even in the last excerpt above, where the teacher said: "the program is correct, globally speaking", this phrase came after the teacher encouraged the student to find the "minor mistake because of sloppiness," and therefore it is reasonable that it was the school viewpoint that dominated the teachers' thoughts.

The two rationales are not contradictory, but rather complement each other. The first rationale views the program as a final product and thus the program is judged as incorrect, whereas the second rationale views the program as a "work in progress", and thus focuses on indicating the degree of closeness to the final product, which is a relative property.

7. DISCUSSION

The results of the quantitative analysis of the decisions about the correctness of the programs showed that in two out of the three programs, teachers' and students' were not significantly different. More specifically, a considerable proportion of the teachers appeared to agree with the concept of relative correctness. In previous work, we found that students' use of this concept was dominated by a user viewpoint on CS. In contrast, qualitative analysis of the teachers' responses reveals that the teachers applied a school viewpoint in a correctness-related programming activity.

In school life, these alternate viewpoints—elements in schooling situations that are not genuine CS activity—occur frequently. Thus, the three viewpoints constantly interact with each other and hence students' knowledge represents an outcome of this cultural blur, which in turn, might cause the students to adopt a distorted viewpoint regarding CS.

For example, those teachers who decided to agree with the relative concept of correctness were driven by employing a school viewpoint. They possibly assume that including the statement that the program is still incorrect in their interaction with students whose program is discussed might discourage students, whereas speaking in relative terms might encourage the students to keep working and would provide a better indication of their closeness to reaching the correct program. This is probably why the teachers felt it was unfair to consider the program as incorrect.

However, teachers should take into account that students do not automatically distinguish between school-oriented activities, such as grading, and CS-oriented activities, such as deciding on a program's correctness, but rather, it is possible and probable that both elements of the activities they participate in would transform their understanding. Furthermore, CS students bring their user
experience to their CS studies and thus they might interpret the teachers' discourse as a tacit agreement with their concept of relative correctness, which is influenced from a user viewpoint.

Moreover, teachers' multiple hats might be interpreted by students as indecision. For example, consider a situation when students take several CS courses in parallel and encounter both the programmer-viewpoint and the school-viewpoint-driven discourses. They might interpret it as teachers with different standards for correctness, which in turn, might be interpreted as a sense of legitimacy to their user-viewpoint in the field.

Further work is required for designing teaching materials and curricula sensitive to these cultural viewpoints. A possible thread is implied by the fact that students judge program 3 more meticulously than program 2, although the only difference between the two was the cover story. Further work is also required to explore other possible interactions among the three viewpoints regarding correctness and other topics in CS.

8. CONCLUSIONS
In this work we demonstrated the usefulness of a socio-cultural-oriented framework that views CS education as a cultural encounter for articulating students' learning difficulties regarding correctness. However, we believe that this framework is also useful for identifying other learning difficulties within the domain.

9. ACKNOWLEDGEMENT
I thank M. Ben-Ari and S. Pollack for their insightful comments.

10. REFERENCES