Proof by Incomplete Enumeration and Other Logical Misconceptions

Geoffrey L. Herman
University of Illinois at Urbana-Champaign
1308 W. Main St.
Urbana, IL 61801
glherman@illinois.edu

Lisa Kaczmarczyk
University of California-San Diego
9500 Gilman Dr.
La Jolla, CA 92093
lisak@ucsd.edu

Michael C. Loui
University of Illinois at Urbana-Champaign
1308 W. Main St.
Urbana, IL 61801
loui@illinois.edu

Craig Zilles
University of Illinois at Urbana-Champaign
201 N. Goodwin Ave.
Urbana, IL 61801
zilles@illinois.edu

ABSTRACT

The ability to reason with formal logic is a foundational skill for computer scientists and computer engineers that scaffolds the abilities to design, debug, and optimize. By interviewing students about their understanding of propositional logic and their ability to translate from English specifications to Boolean expressions, we characterized common misconceptions and novice problem-solving processes of students who had recently completed a digital logic design class. We present these results and discuss their implications for instruction and the development of pedagogical assessment tools known as concept inventories.

Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education — Computer science education; F4.1 [**Mathematical Logic and Formal Languages**]: Mathematical logic; B.6 [**Logic Design**]

General Terms

Human Factors, Languages, Verification

Keywords

discrete math, digital logic, formal logic, misconceptions, concept inventory

1. INTRODUCTION

Practicing computer scientists and engineers require the ability to reason rigorously. When they design software and hardware systems, propositional logic and Boolean algebra (hereafter "Boolean logic") play a fundamental role in writing specifications, validating designs, testing rigorously, and optimizing safely. These skills are typically taught early in computer science and computer engineering curricula (typically as part of discrete mathematics or digital logic design classes) and serve as foundations for many of the classes that follow.

These topics, however, are challenging for many students. This paper focuses on one particular skill related to propositional and Boolean logic — the ability to translate natural language (e.g., English) specifications to Boolean expressions — that was identified by experts in teaching discrete math and digital logic design classes as a topic that is both important and difficult for students to learn [12].

The goal of this paper is to understand the ways in which students fail to translate accurately from English to Boolean expressions, in terms of both conceptual misunderstandings and failures in the problem solving processes used by the students.

This work is part of an ongoing project to develop concept inventories (CIs), multiple-choice tests that reliably measure student conceptual understanding; knowledge of student misconceptions is necessary for writing compelling incorrect answers (*distractors*) for a CI. Based on the impact of the Force Concept Inventory (FCI) in physics education [10, 13], we believe that well-designed CIs can facilitate the development of better pedagogical approaches (by providing a mechanism for rigorous comparison) as well as for driving their adoption, a view shared by others in computer science education [1, 6, 16].

We believe, however, our results can also be used directly by instructors of discrete mathematics and digital logic design classes to inform interventions that address these misconceptions and inferior problem solving approaches.

We undertook this research using a traditional qualitative approach, the motivation for which we describe in Section 2. We transcribed and had the four authors independently code a collection of interviews in which students were asked to verbalize their thought process while solving Boolean word problems; our methodology is described in Section 3. From

ICER'08, September 6–7, 2008, Sydney, Australia.

our coding, we identified nine themes relating to the mistakes that students made and the processes that the students used that led to these mistakes, described in Section 4. In Section 5, we discuss how these results relate to misconceptions identified in other disciplines before concluding, in Section 6, with our ideas for how these findings should be used to modify classroom instruction.

2. BACKGROUND

It has previously been shown that the average college student does not reason with formal logic well. For example, Cheng and Holyoak [4] found that fewer than 10% of college students reason correctly about the conditional logic statement "if A then B." Most of these students mistranslated the conditional to be A AND B, instead of the correct formal logic statement of B OR NOT A [4]. While their study shows that average college students struggle with formal logic, it does not show whether formal instruction in propositional logic, such as that found in discrete mathematics and digital logic classes, dispels these misconceptions.

Physics education researchers have found that identifying misconceptions alone is not enough to direct instructional reform. In addition, instructors must know why students (commonly referred to as physics novices) make the mistakes they do. Previous research has shown that physics novices think about physics in fundamentally different ways from physics experts [2]. One difference is that physics novices focus on surface features of problems rather than the underlying concepts and try to recall as quickly as possible any formula that seems to match the surface features of the problem they have encountered [5]. For example, physics novices focus on the physical objects in a physics problem (e.g., inclined planes, blocks, balls), and then they try to recall any formulas that match the variables and objects of the problem. However, physics experts focus on the principles that can be used (e.g., conservation of energy, work) and then use these principles to guide their search for strategies and formulas [2].

In our investigations of student misconceptions in Boolean logic, we seek to describe both the what and the why of their misconceptions. Because there have been few previous studies of these misconceptions, we took a grounded theory approach. Grounded theory is a qualitative research method that can be used when little is known about what people do and how they think in a given context. In grounded theory, no theory or hypothesis should be formed prior to the collection of data [11]. Instead, theories should emerge from the use of open-ended data collection, and the analysis of this data should inform future data collection [17]. One such data collection method is asking subjects to think aloud about what they are doing within a familiar context [9, 18]. The data should then be analyzed with rigorous coding schemes that protect against bias and the premature formation of conclusions. Rigor in coding schemes is enhanced by requiring that multiple researchers independently code the data; they then discuss their encodings and emergent themes and retain only those codes and themes where unanimous agreement exists [17].

3. METHODOLOGY

Subjects in this study were interviewed for one hour about their understanding of a wide range of concepts in digital logic design. Due to time constraints, each participant was interviewed on only a portion of the selected concepts. The interview questions resembled problems that the subjects may have encountered previously in a digital logic class. Subjects were paid for their participation.

3.1 Subjects

In Spring 2008, seven undergraduate volunteers from the University of Illinois at Urbana-Champaign were interviewed about translating English specifications to Boolean expressions. Two women and five men were interviewed; two were international students. All volunteers were traditional age (18-22) undergraduates majoring in computer science, electrical engineering, or computer engineering who had completed a three credit digital logic design class with a simulation lab in the Fall 2007 semester and had earned grades of B or C. Students who had earned B and C grades were selected because their understanding was likely to be less complete, and they were more likely to have misconceptions than students who had earned A grades. Our pilot interviews confirmed these expectations, as the interviews with students who had earned grades of A had yielded fewer mistakes or misconceptions. All subjects had also completed a discrete mathematics class in a previous semester. Due to their relative lack of experience, these subjects were considered to be domain content novices. In addition, two graduate teaching assistants who have taught a digital logic class for several semesters were interviewed. The teaching assistants were considered to be domain content experts.

3.2 Interview Process

Interviews were conducted in a "think-aloud" format: subjects were instructed to vocalize their thoughts as they solved problems and responded to questions [9]. Prior to the interview, subjects were briefed on the study's goal of understanding how they think through various topics in digital logic design. They were told to not expect feedback about whether their answers were correct during the interviews, but to expect being frequently asked to expand on what they were doing [9]. All interviews were recorded using a document camera (which recorded what the subject wrote) and microphone. The audio tracks of the interview recordings were transcribed verbatim, the subjects' gestures were included in the transcript, and every piece of paper the subject wrote on was scanned electronically.

3.3 Interview Questions

As part of the interview, all subjects were asked the three questions about Boolean word problems in Figure 1. Questions 2 and 3 were sometimes asked in the opposite order to reduce the impact that answering one question may have had on answering the other question. Question 1 was designed to probe the students' conceptual understanding of if-then, while Questions 2 and 3 were designed to simulate questions the students may have encountered in their digital logic design class. A list of acceptable answers to these questions is in Figure 2. Additional clarifying questions were asked in response to what subjects said.

3.4 Data Analysis

The interviews were analyzed using the following steps of grounded theory and qualitative data analysis as described by Kvale [15], Strauss and Corbin [17], and Miles and Hu-

Question (1) Explain the meaning of *if A then B* in Boolean logic. Give an example that demonstrates your understanding.

Question (2) A campus sandwich shop has the following rules for making a good sandwich:
(1) A sandwich must have at least one type of meat,
(2) A sandwich must have roast beef or ham, but not both,
(3) If a sandwich has turkey then it must also have cheese.
Write a Boolean expression for the allowed combinations of sandwich ingredients using the following variables:
c = cheese
h = ham
r = roast beef
t = turkey

Question (3) A recipe for apple pie has the following instructions:
(1) Do not use both allspice and nutmeg simultaneously; and
(2) Use nutmeg if and only if you use cinnamon.

Write a Boolean expression for the allowed combinations of spices for the apple pie, using the following variables:
a = 1 if allspice is used
c = 1 if cinnamon is used
n = 1 if nutmeg is used

Figure 1: List of interview questions.

Question (1) At minimum the subject should be able to correctly translate *if A then B* to the Boolean expression $\overline{A} + B$. Identifying the conditional as implication was also expected.

An example that a student might give to demonstrate a deeper understanding of *if A then B* would be, "If a greeting card is blue (A), then it must be a birthday card (B)." This statement is falsified only when the card is blue, but is not a birthday card $(A\overline{B} = \overline{(\overline{A} + B)})$. A blue birthday card (AB) would be acceptable; a red birthday card $(\overline{A}B)$ would be acceptable; and a white wedding card $(\overline{A}\,\overline{B})$ would be acceptable.

Questions (2) & (3) approach: It was expected that the subjects would solve the written Boolean word problems by first translating each rule of the problem statement into a Boolean expression independently. They would then compose the separate expressions into a single expression by ANDing the rules together.

Question (2) solution

rule 1 = $r + h + t$; rule 2 = $r\overline{h} + \overline{r}h = r\ XOR\ h$; rule 3 = $\overline{t} + c$;

Final rule = $(r + h + t) \cdot (r\overline{h} + \overline{r}h) \cdot (\overline{t} + c)$ = Simplified rule = $(r\overline{h} + \overline{r}h) \cdot (\overline{t} + c)$

Although not required, this rule can be simplified by recognizing that rule 2 subsumes rule 1.

Question (3) solution

rule 1 = $\overline{(an)} = a\ NAND\ n$; rule 2 = $(nc + \overline{n}\,\overline{c}) = n\ XNOR\ c$;

Final rule = $\overline{(an)} \cdot (nc + \overline{n}\,\overline{c})$

Figure 2: List of acceptable answers.

Action Prefixes	Prefix Definition (Codes with this prefix pertain to…)
Behavior	Behaviors of interest that are not used to derive a solution
Process	Strategies and processes that subjects use to derive a solution
Conception Prefixes	
Code	The relationship between programming and Boolean logic
Composition	Conceptions used to create a single expression from multiple expressions
Correct	Correct conceptions
iff	Conceptions related to if-and-only-if
if-then	Conceptions related to if-then
Mistranslation	Conceptions that lead to mistranslations between English logic and Boolean logic
Translation	Conceptions that lead to correct translations between English logic and Boolean logic

Table 1: Code Prefixes

berman [14]. The four authors analyzed the data: the interviewer (Herman), a former instructor of a digital logic class (Loui), a colleague with content knowledge in Boolean logic (Zilles), and a researcher with extensive experience in qualitative research methods (Kaczmarczyk).

Step (1) - To avoid bias, all seven interviews were analyzed. While the interviews included questions on other topics, only the subjects' responses to the Boolean word problems were analyzed, because this study focused on novice Boolean logic misconceptions.

Step (2) - All researchers analyzed the interviews independently without a predetermined coding scheme, as prescribed by grounded theory [17]. Principles of grounded theory were used to uncover the subjects' misconceptions, because it allows the misconceptions to emerge from the data without an *a priori* theoretical framework that would influence the observations. Eschewing an initial coding scheme also allows for fuller descriptions of what the subjects did correctly or incorrectly for each statement.

Step (3) - The four researchers met and discussed every annotation and observation that they had made. To ensure the accuracy and completeness of our coding, a unanimous decision was needed for an annotation to be included for coding or rejected from coding. If a unanimous decision was reached, then it was counted as an agreement; otherwise it was counted as a disagreement. Preliminary code names and definitions were created for every accepted annotation.

Step (4) - After all interviews were discussed, the preliminary code names and definitions were refined by two researchers(Herman and Kaczmarczyk) to facilitate the identification of thematic patterns. The refined list of codes and definitions was given to all four researchers to identify the thematic elements of the codes independently. All researchers then met again to discuss the thematic elements that they had noted. A unanimous decision about the presence of a theme was needed for it to be included in the final list of themes.

3.5 Codes

Through the process described in Section 3.4, 58 codes emerged; a complete list of the codes can be found at the end of this paper as Figures 4 and 5. The codes are divided into two primary categories — Actions and Conceptions — and annotated with a collection of nine prefixes (shown in Table 3.1). The prefixes are used to help categorize the codes and identify themes. Some codes have multiple prefixes (e.g., Composition - Correct - AND (C9)) because both prefixes accurately described the code. One code has no prefix (Improper distribution (C34)), because it did not fit into any of the more populous categories of code.

An inter-rater reliability of 95% was calculated as follows:

$$R = A_n/(A_n + D_n), \tag{1}$$

where R is inter-rater reliability, A_n is the total number of agreements, and D_n is the total number of disagreements.

The conception codes are of most direct interest to this research, because they indicate the misconceptions that can be used to create the concept inventory. These codes also help to gauge the relative difficulty of different concepts. While the Behavior and Process codes will not be used to create the concept inventory directly, these codes offer additional insights about why the subjects had these misconceptions. Therefore, they provide guidance towards instructional interventions to help students overcome their misconceptions. These codes also demonstrate the expertise level of the subjects.

4. THEMES

The data analysis revealed five themes about student misconceptions and four themes about their actions. The misconception themes were (1) a tendency to reduce harder concepts to easier concepts, (2) if-then translation misconceptions, (3) confusion about the meaning of a false antecedent, (4) omission of complemented variables, and (5) misconceptions stemming from interference from exposure to a concept in multiple contexts. The action themes centered on novice behaviors and processes such as (1) reliance on recall over reasoning, (2) proof by incomplete enumeration, (3) non-systematic approaches to problem solving, and (4) a lack of metacognition.

4.1 Reduction to Easier Concepts

The coding process revealed that there are two classes of Boolean operators. One class of operators, the easy class, includes OR, AND, and XOR. Subjects could correctly perform activities involving these operators in almost all instances. The other class of operators includes NAND, if-then, and if-and-only-if. Subjects showed incomplete understanding of the latter operators, and subjects tended to incorrectly reduce these harder operators to the easier operators. To facilitate discussion on these operators a set of truth tables for these concepts is given in Figure 3.

4.1.1 OR, AND, and XOR - easy operators

The data showed that OR, AND, and XOR are easier operators, because there are a large number of codes that deal

inputs		*easy operators*					
A	**B**	NAND	XOR	OR	AND	if A then B	if-and-only-if
0	**0**	1	0	0	0	1	1
0	**1**	1	1	1	0	1	0
1	**0**	1	1	1	0	0	0
1	**1**	0	0	1	1	1	1

Figure 3: Truth tables of the NAND, XOR, OR, AND, If-then, and if-and-only-if concepts. The two loops show the similarities between the hard operators and the easy operators.

with the subjects' ability to correctly translate and manipulate expressions involving OR (C18, C19, C52), AND (C9, C13, C51, C53), and XOR (C20, C58). Notably, all subjects demonstrated more than one of these correct conceptions. This finding is in contrast to the paucity of codes related to incorrect handling of these operators. Some subjects made minor mistakes with these operators, but these mistakes were often isolated to a single subject and may have been caused by one of the other themes described later (Sections 4.5 and 4.8). For example, only one subject mistranslated "A or B, but not both," as OR. This mistranslation might be explained by the interference theme, because in colloquial English "or" often means exclusive-or. All other subjects were able to translate the previous statement correctly and were also able to express the XOR operation using the standard Boolean operations of AND, OR, and NOT when asked.

The following excerpt shows how a subject correctly translates and explains Question 2 Rule 2 (see Figure 1) with a full case enumeration:

> STUDENT 7: There is also statement 2 which says that a sandwich must have roast beef or ham, but not both at the same time. So, that would be r **XOR** h. And ... that really sums up that whole statement right there.
>
> INTERVIEWER: Can you expand your symbol for XOR?
>
> STUDENT 7: Yeah, basically that would mean that r **AND** h is false [writes **!(**r **AND** h**)**] and also that it would be **NOT(NOT** r **AND NOT** h**)**. If that makes any sense, because that would mean that r is zero and h is zero and that does not fulfill the **XOR** condition, and neither does having both of them be 1. So, basically [**XOR** is] **(NOT** r **AND** h**) OR (**r **AND NOT** h**)**.

Furthermore, no subject mistranslated the phrase "at least one" from Question 2 Rule 1 (see Figure 1), and many demonstrated a deeper understanding of why OR is the correct translation.

> STUDENT 7: [Rule] (1) is basically "must have one type of meat," so h **OR** r **OR** t.
>
> INTERVIEWER: And how did you decide that?
>
> STUDENT 7: Well, **OR** is fulfilled anytime any variable included in this expression has a value of 1, so the only way that this wouldn't... it only needs one type of meat, so any of these could be so that would fulfill that requirement.

4.1.2 Not both (NAND) reduction

Several subjects incorrectly reduced the NAND operator to XOR (C36). The phrase "do not use both allspice and nutmeg simultaneously" from Question 3 Rule 1 was mistranslated by more than half of the subjects to be allspice XOR nutmeg.

> STUDENT 7: ... so do not use both [allspice] and [nutmeg] simultaneously means that you use **XOR**, because **XOR** means that only one can happen. So, [writes a **XOR** n]. So, for this part it is just a **XOR** n.

It is possible that subjects mistranslate "not both" because they see the phrase "not both" in NAND and XOR specifications and are overeager to match the "not both" phrase with the XOR. Because XOR and NAND have the same values for three cases ($\langle a, n\rangle = \langle 1, 1\rangle$, $\langle 1, 0\rangle$, and $\langle 0, 1\rangle$) the overaggressive pattern matching is reinforced as subjects match these cases to the easy operator, XOR (This pattern match can be seen with the left loop of Figure 3). The following subject checks only three cases before concluding that he has verified his use of XOR, (the explication of cases is added in italics for clarity).

> INTERVIEWER: So how'd you come up with $\bar{a}c$ **OR** $a\bar{c}$ for "do not use both?"
>
> STUDENT 2: Well, when we do not have allspice, I mean it says do not use allspice and nutmeg simultaneously ($\langle a, c\rangle = \langle 1, 1\rangle$), right?
>
> INTERVIEWER: Okay.
>
> STUDENT 2: So if allspice is not being used, we can use cinnamon ($\langle a, c\rangle = \langle 0, 1\rangle$). And if allspice is used, then we cannot use cinnamon ($\langle a, c\rangle = \langle 1, 0\rangle$).

One student seemed to even struggle with what "not both" meant.

> STUDENT 1: "Do not use both allspice and nutmeg simultaneously" would be the same thing as saying "use allspice if and only if you are using nutmeg, which is the **XOR**, so $\bar{a}c$ OR $\bar{c}a$.

4.1.3 if-and-only-if (XNOR) reduction

No subject was able to correctly translate the biconditional statement "A if and only if B." Most subjects reduced the biconditional statement into only one of its constituent single-direction conditional statements "If A then B" or "If B then A." This loss of one direction was also typically accompanied by a further reduction of "A if and only if B" to be "A AND B". STUDENT 1 misinterprets "n if and only if c" to be "if n then c" (italics are added for emphasis).

> STUDENT 1: If-and-only-if would be, if ... if I use cinnamon ... no ... *if-and-only-if means, if I use nutmeg, then there has to be cinnamon*, but there if doesn't have to mean that .... If A is nutmeg then there will be cinnamon, *but if there is cinnamon, then that does not necessarily mean there is nutmeg.*

Students 3, 6 and 7 mistranslated the biconditional as AND, and explained their choice by reducing "n if only if c" to "if c then n."

> STUDENT 6: I'm trying to remember something about if-and-only-if .... I'll write c first and see what happens .... *I guess c* ***AND*** *n]* .... I think it should be something like this. I know it might be wrong, but there is something related to if-and-only-if that I couldn't remember ... it's something related to using **AND**, because *the only way for nutmeg to occur is if you use cinnamon first .... Because if I look to this thing here [expression] if n is 0 [the expression] will be 0.* The only way for this [rule 2] to happen is for n to be 1 and c to be 1 which means that you use both.

> STUDENT 3: So you only want cinnamon to be used .... I mean nutmeg to be used if you have cinnamon, so I think if you take c AND n, the only way for that to be equivalent to 1 is if we use them ... so ... yeah .... Let's see ... you only want to use nutmeg if you use cinnamon. So *if I use cinnamon, so cinnamon would be 1, then the use of nutmeg would determine the value of this expression. And then if you don't use cinnamon, then it would automatically be 0, for the whole thing.*

> STUDENT 7: I interpret [if-and-only-if] as *cinnamon has to be used in order for nutmeg to be used, but not the other way around.* So, ... I guess it's probably [*writes n* ***AND*** *c*]

4.1.4 *if-then reduction*

Four subjects mistranslated implication as "A AND B," which causes implication to lose its directionality (C28).

> STUDENT 5: If you have turkey, then you must also have cheese [write $+tc$] so it's **turkey AND cheese**

> STUDENT 2: [writes 3:] Okay, well if it has turkey then it also has cheese. [writes tc]

It is possible that the subjects translated "if-and-only-if" and "if-then" to both be AND, because the subjects had first reduced "if-and-only-if" to "if-then" and then, based on this misconception, they further reduced "if-then" to AND.

4.2 If-then Misconceptions

Although all but one subject recognized the conditional statement, "if A then B" to be implication, no subject was able to correctly translate or interpret the statement. Most students appeared to hold multiple misconceptions about implication. These misconceptions stemmed from faulty recall, reduction to easy operators, incomplete case analysis, and struggles with understanding the relationship between the antecedent (A) and the consequent (B). Two subjects attempted to remember the Boolean expression for implication and incorrectly recalled the expression $\bar{B} + A$ (C29) instead of $\bar{A} + B$.

> STUDENT 4: if A implies B then you get A **OR NOT** B, so I can't remember if that it's exactly what it is, but it's something like A **OR NOT** B.

As mentioned above, subjects had misconceptions resulting from reduction. Two of these subjects who initially mistranslated implication to be "A AND B," later used incomplete case analysis to derive the expression "$AB + \bar{A}B$" after realizing that B could be true by itself without violating implication (C27); both students failed to include the $\bar{A}\bar{B}$ case. (Cases are added for clarity.)

> STUDENT 3: And then [rule] 3 ... I guess would just be like, turkey implies cheese, so let's see ... **turkey AND cheese** ($\langle t, c\rangle = \langle 1, 1\rangle$) because **OR... NOT turkey AND cheese** ($\langle t, c\rangle = \langle 0, 1\rangle$)? I think, because this would be such true, if it has turkey and cheese, but it doesn't say anywhere that cheese cannot be by itself. So this can also be true. [writes $tc + \bar{t}c$].

Some subjects also had misconceptions resulting from faulty understandings of what the false antecedent implied.

4.3 False Antecedent Confusion

All subjects demonstrated confusion about the relationship between the antecedent and the consequent in a conditional statement. The most common misconception was that the antecedent is a prerequisite for the consequent to be true (C6) or more specifically that the truth of the antecedent causes the consequent to be true (C4) (emphasis added in italics).

> STUDENT 7: I would say, well, given two statements that are each either true or false you could arbitrarily call one A and the other B and *the only reason why B would be true would be if A is true first. So, if A then B.*

> STUDENT 7: Well because t ... *the implication arrow says that t is a prerequisite for c and so if t is not true, then I mean the whole thing does not work.*

> STUDENT 5: If A then B, I would say *it's like a cause and effect type of relationship* where if whatever A is is true then that means that B *is automatically true.*

A potentially related misconception is the belief that if the antecedent is false, then the conditional is false (C30).

> STUDENT 7 : I'm just trying to remember what it was. Although I guess I could do a quick truth table. [draws truth table] So ... *whenever t is not true then that means this [$t \rightarrow c$] is also not true.*

> STUDENT 3: You only want to use nutmeg if you use cinnamon. So if I use cinnamon, so cinnamon would be 1, then the use of nutmeg would determine the value of this expression. And then *if you don't use cinnamon, then it would automatically be 0, for the whole thing.*

The cause of this misconception likely stems from the ambiguity of the language used during instruction to describe the concept of implication.

4.4 Omission of Complemented Variables

Subjects demonstrated difficulties in using complemented variables. When enumerating cases to evaluate, subjects frequently failed to evaluate the case where all variables were false. For example, only one subject evaluated the $\langle A, B\rangle = \langle 0, 0\rangle$ case for the if-then constructions, and one subject evaluated the $\langle a, n\rangle = \langle 0, 0\rangle$ case for the "not both" construction. If more subjects had evaluated these $\langle 0, 0\rangle$ cases, they may have found that their Boolean expressions did not match the English specifications.

Subjects also had difficulties including complemented variables in their expressions when encountering the English specifications "by itself" and "without" (e.g., a pie with allspice and cinnamon, but *without* nutmeg is written as "ac" instead of "$ac\bar{n}$"). Subjects translated "allspice by itself" as just a (e.g., $f(a, c, n) = a$) rather than a ANDed with the complements of all other ingredients (e.g., $f(a, c, n) = a\bar{c}\bar{n}$). Similar mistakes were made for the phrase "without." In the following example, the subject incorrectly interprets cinnamon by itself as c instead of $c\bar{n}\bar{a}$ and allspice by itself as a instead of $a\bar{c}\bar{n}$.

> STUDENT 5: You can use cinnamon by itself without the nutmeg, because that doesn't break rule (2) [writes $+c$] ... or you could just use allspice by itself [writes $+a$].

Another subject failed to include any complemented variables in her expression despite describing these complemented variables with her English specifications.

> STUDENT 3: so that you can use **cinnamon AND allspice** ... **OR cinnamon AND nutmeg** ... **OR cinnamon** [writes $(c) \vee (c \wedge n) \vee (c \wedge a)$] ... because you can't use allspice and nutmeg at the same time, and you can't use nutmeg without using cinnamon, so I think you're already limited to these 3 options.

4.5 Ambiguity and Interference

Most subjects had previously encountered logical constructions in multiple contexts where each context assigned a different concept to these constructions. These different contexts can cause the constructions to be ambiguous. Other concepts are introduced in multiple contexts with different notations causing interference. The subjects exhibited misconceptions resulting from the ambiguity or interference of their different exposures to the constructions.

The most ambiguous construction for the subjects was "if-then." On several occasions subjects specifically mentioned that they knew that there is a difference between how "if-then" is used in programming or colloquial English and how "if-then" is used in Boolean logic. Despite this knowledge many subjects were unable to articulate the difference between the contexts (C7).

> INTERVIEWER: How would you describe the phrase if A then B in Boolean logic?
>
> STUDENT 4: In Boolean logic or in plain English?
>
> INTERVIEWER: Imagine that you are teaching them.
>
> STUDENT 4: Uh ... If A then B would mean that if the expression after the "if," like if you had "if X=1," then some other expression such as X++ or increment X then if X=1 then you would do the statement after that, saying "okay that is true." That's more of a programming statement than it is the Boolean logic approach. The [the digital logic design class] approach would be something more like what it means to have "if-then" is some sort of implication, where if you have one then you have the other.
>
> INTERVIEWER: Okay. So you mentioned this isn't C [the programming language]. So what's the difference between a C "if-then" and a Boolean "if-then?"
>
> STUDENT 3: In C, if you say **if** something, **then** you will uh, do some sort of further work, if the first condition is true you do some stuff. Then you've got other stuff. **If** and **then** are usually ... they're related, but they're not ... reliant on each other. Like, they are in Boolean ... logic, I guess.

Interference between the different notations used for the same concept was seen in the way that subjects used multiple symbols to indicate the complement of a variable. Many subjects used programming symbols such as '!' to indicate complementation rather than the more commonly used Boolean symbols such as the over bar or apostrophe (C2).

> STUDENT 3: I guess you just put each part together with an **AND** because you want each part of it to be true on its own, and then .... *I'm using lots of different symbols for **NOT**.*

4.6 Recall Versus Reasoning

Subjects in the study exhibited novice behaviors through their reliance on memorization, recall, and manipulation of equations, rather than reliance on reasoning [2]. The code Process - Rote Recall (C50) was the most frequent code and was used three times as often as any other code, except Process - Incomplete enumeration (C44). The subjects' significant reliance on recall also led to instances of faulty recall (C42). Perhaps most striking about this behavior was that some subjects readily admitted that they relied only upon recall to succeed in the class.

While working through Question 2, one subject said,

> STUDENT 6: ...I also think how I learned this classes [sic] are wrong. Like the way I approach this classes [sic] are like I just memorize all the concepts and stuff before I go to class and I just go in and can do all the tests. Yeah, that's my lowest grade so far, like a B+ and B ...so ...and for all the classes I forgot everything.

Other subjects relied heavily on recalling symbols and equations. The following example, concerns translating "if T then I" to a Boolean expression. While the right arrow is a standard symbol for implication in propositional logic, it doesn't satisfy the question of expressing if-then as a Boolean function.

> INTERVIEWER: So, if this was a function of like $f(T, I)$. What would that equal?
>
> STUDENT 7: it's been a while, but [writes $f(T, I)$]... I don't quite remember what the notation for that was honestly...
>
> INTERVIEWER: Let's try it in a different context then maybe ....
>
> STUDENT 7: Oh! [writes $T \rightarrow I$]

Similarly another subject incorrectly recalls a Boolean expression for "if A then B."

> STUDENT 4: ... it's something like ... if A implies B then you get A **OR NOT** B, so I can't remember if that is exactly what it is, but it's something like A **OR NOT** B.

4.7 Proof by Incomplete Enumeration

The enumeration of cases to prove the correctness of a logical expression (proof by exhaustion) is a foundational law within Boolean logic, yet subjects often felt they had proved equivalence after enumerating only one or two cases (C44 was the second most common code). What we came to refer to as "proof by incomplete enumeration" resulted in two types of errors for the subjects: reduction errors and faulty error correction.

As mentioned before, subjects tended to reduce hard concepts to easier concepts. In Figure 3, it can be seen that XOR and NAND are equivalent for three cases and that AND, "if-then," and, "if-and-only-if" are equivalent for two cases. Subjects frequently enumerated only the cases where the hard concept and the easy concept were equivalent, and failed to enumerate the cases where the two concepts were not equivalent. Examples of faulty proofs can be seen in Section 4.1.2. Another subject checked only one test case ($\langle t, c\rangle$=$\langle 1, 0\rangle$) for Question 2 Rule 3 (see Figure 1) before deciding his recalled expression was correct.

> INTERVIEWER: How would you interpret [rule 3] by itself?
>
> STUDENT 6 : I would just start with turkey ... okay I think it is t **AND** c.
>
> INTERVIEWER: And why do you think that?
>
> STUDENT 6: Because if it is 1 which means you have **turkey**, and you have 0 **cheese** ($\langle t, c\rangle$= $\langle 1, 0\rangle$) this statement is 0 which is wrong, and we want this statement to be 1 which means that we want both t **AND** c.

Other subjects enumerated some cases which disproved their original expression, but fell short of the complete enumeration and therefore still had an incorrect expression. The following subject enumerated only two cases of Question 2 Rule 3 ($\langle t, c\rangle$=$\langle 1, 1\rangle$ and $\langle 0, 1\rangle$) to derive her expression, but failed to consider the case where neither turkey nor cheese ($\langle t, c\rangle$=$\langle 0, 0\rangle$) were used, which was also permitted by the rule.

> STUDENT 3: And then [rule] 3 .... I guess would just be like, turkey implies cheese, so let's see ... **turkey AND cheese** ($\langle t, c\rangle = \langle 1, 1\rangle$) because ... **OR** ... **NOT turkey AND cheese** ($\langle t, c\rangle = \langle 0, 1\rangle$)? I think, because this would be such true, if it has **turkey AND cheese**, but it doesn't say anywhere that **cheese** cannot be by itself. So this can also be true. [writes $tc + \bar{t}c$].

4.8 Cowboy Composition and Non-systematic Approaches

The imperfect use of proof by exhaustion suggests that subjects are non-systematic in their approaches to problem solving. The theme of non-systematic approaches was supported by other codes as well, including Process - Reasoning disconnect (C49), Process - Nonsystematic analysis (C45), Composition - Cowboy (C11). The term "cowboy composition" was coined when subjects "shot from the hip" to simplify the composition of two complex rules in their heads, rather than write down the complete compound expression and then use Boolean simplification.

Student 5 used cowboy composition to answer Question 3. In this example, the subject ignored the individual rules she had derived earlier and derived a single incorrect expression by "stringing" different cases together.

> STUDENT 5: I guess *I can just start out by stringing all the possibilities together.* So you can use nutmeg and cinnamon [writes nc] or you can use cinnamon by itself without the nutmeg, because that doesn't break rule (2) [writes $+c$]. "Do not use both allspice and nutmeg together," so that means you could use cinnamon and nutmeg or cinnamon and allspice rather [writes $+ca$] or you could just use allspice by itself [writes $+a$].

4.9 Lack of Metacognition

Finally, subjects exhibited novice behaviors by how they rarely used metacognition (C1) to monitor their work [2]. During all of the interviews on Boolean word problems, only four instances emerged where the subjects monitored the accuracy of their work or strategies. In particular, after subjects had derived Boolean expressions, they rarely returned to the original English expressions to check the accuracy of their results (C49).

5. DISCUSSION

The results of our research show that digital logic students have some ability to reason using formal logic, yet still struggle to use important underlying concepts such as proof by exhaustion. For example, many students demonstrated considerable aptitude with some of the "easy" translation tasks and were able to recognize immediately that Rule 2 (XOR) of Question 2 fully subsumes the conditions described in Rule 1 (OR). Despite their facility with these simple rules, these same students could not translate harder statements like if-then and tended to rely on ad hoc reasoning schemes rather than the foundational principles of formal logic. The finding that students struggle to use underlying concepts is consistent with the findings of physics educators discussed in Section 2.

Novice physics students tend to focus on surface features of a problem, and they attempt to recall relevant equations. Similarly, novice digital logic students seem to focus on the few easy cases that they can perceive directly from the initial word problem statement, and they recall the Boolean expression that best matches these surface cases. Not surprisingly, the first expression that they often recall is the simplest expression they can remember rather than the correct expression.

Novice behavior can be clearly seen in how our subjects analyzed the expressions "if A then B" and "not both A and B." The statement "if A then B" provides two readily perceived cases to evaluate, $\langle A, B\rangle = \langle 1, 1\rangle$ and $\langle 1, 0\rangle$. The statement is clear that in the $\langle 1, 1\rangle$ case the expression is true and that in the $\langle 1, 0\rangle$ case the expression is false, but it doesn't explicitly offer information about the $\langle 0, 0\rangle$ and $\langle 0, 1\rangle$ cases. We hypothesize that the subjects in our study aggressively pattern matched the two explicit cases to the simplest expression they could recall – AND (see right loop of Figure 3). Furthermore, it seemed that many subjects considered the $\langle 0, 0\rangle$ and $\langle 0, 1\rangle$ cases to be unimportant because many subjects failed to address these cases at all in their spoken reasoning.

A similar explanation can be provided for the mistranslation of "not both A and B" to "A XOR B." The case $\langle A, B\rangle = \langle 1, 1\rangle$ is false is the only case explicitly described by the statement "not both." If a subject used aggressive pattern matching, this case analysis maps to only one easy concept – XOR (see left loop of Figure 3). When asked to explain their choice of XOR in this situation, these subjects used partial case analysis of the explicit XOR cases ($\langle A, B\rangle = \langle 1, 1\rangle$, $\langle 0, 1\rangle$, and $\langle 1, 0\rangle$). Unfortunately, checking these cases only confirmed their previous overaggressive pattern match. It is also important to note that in both cases the complemented case $\langle A, B\rangle = \langle 0, 0\rangle$ was omitted by the subjects.

Students could have avoided these mistakes by using appropriate translation techniques. To check the correctness of their translation of an English statement into a Boolean expression, students should apply the principle of exhaustive enumeration. Students were reluctant to check exhaustively, however: they failed to use truth tables and Karnaugh maps, which facilitate systematic checking of all cases, particularly in cases when students needed them most (i.e., when they were struggling). Exhaustive enumeration protects against overaggressive pattern matching and the omission of the complemented case. Further research should investigate why students were so reluctant to rely on truth tables except in the simplest cases.

We also found that student misconceptions arise when they encounter a term that has different meanings in different contexts. This finding is consistent with the physics education research as well [7]. As an example from physics, the physical terms *force* and *work* are similar to the colloquial meanings of the English words force and work, yet are different in very significant ways (i.e., in mechanics you do no *work* by jumping up and down in one spot even though you have worked very hard while jumping). This problem of conflicting definitions of terms is similar to the problem we observed of ambiguous terms and concepts such as "if-then" and "by itself."

In programming and often in English, the "if-then" construction is used to create causal relationships: if a condition is true, then do this action (or this action will occur) and if the condition is false, then the action will not occur. The confusion between the two uses of if-then became particularly clear when the students said things like "A happens first," "A causes B," or "A is a pre-requisite for B" to describe the logical if-then. When students provided their own examples of a logical if-then statement, they often took the form of if "condition A is true", then "do this action" or "this action will occur." From this conception of if-then, the case of the false antecedent makes little logical sense. The logical statement can be "expressed" as "if A is false, then B cannot occur." Students interpreted that either "B cannot occur" is not expressible in Boolean logic (C5, C32) or that "B cannot occur" should evaluate to false (C30).

The ambiguity of the term "by itself" arises from the interference between an an action-oriented interpretation and a logic interpretation. For example, if the student is thinking visually, "cheese by itself" is the variable c with nothing else connected to it (e.g., $f(c, h, r, t) = c$). If the student is thinking about making a "cheese by itself" sandwich, it is very odd to think about grabbing cheese and the complements of the other ingredients from the refrigerator to place them into the sandwich. The interpretation of "cheese by itself" as $f(c, h, r, t) = c\bar{h}\bar{r}\bar{t}$ does not accord well with common, everyday experience.

Because misconceptions about Boolean operators seem common, we recommend that in the classroom, concepts such as "if-then," "if-and-only-if," "by itself" and "not both" (NAND) be explained carefully using well-crafted, concrete examples that students can relate to intuitively. Instruction should take into account the causes for the ambiguity of these concepts, and students should be warned where confusion might arise [3]. We also recommend that instructors emphasize proper translation and composition techniques by modeling the correct techniques and evaluating student performance with these techniques on examinations and homework [8].

Our results may not be generalizable to all computer science students because all interviewed students were traditional age engineering students from a single institution. Because we found similar misconceptions for students who had taken digital logic classes in two different departments, however, we believe that the results have a degree of generalizability. Another limitation of this research was that many of the students (especially the international students) were inarticulate and vague when answering questions. More themes and misconceptions might have been found, but sometimes the student's poor command of spoken English obscured the student's reasoning.

6. CONCLUSION

The results of this study demonstrate that students who passed digital logic classes with grades of B and C are unable to solve basic conceptual problems even shortly after completing a digital logic class. This weakness has important implications for the development of our concept inventory. As we develop the concept inventory, we will include questions that probe students' understanding of the concepts of "if-then," "if-and-only-if," "by itself," and "NAND." In addition, we will investigate if there are other conceptual difficulties that lead to overaggressive pattern matching or that require students to use complete case analysis to derive the correct solution.

Future student interviews will try to complete our understanding of the misconceptions found so far and attempt to find out how robust and common these misconceptions are. If students are explicitly asked to write the truth tables for the phrases "not both" or "if A then B," will they still translate "not both" as XOR or "if A then B" as A AND B or will they realize that their original translations were incorrect? If some misconceptions can be corrected by explicit enumeration of cases and others cannot be corrected, we can further classify the types of misconceptions by their robustness. In addition, these interviews will investigate whether students struggle to translate other English specifications such as "unless," "exactly one of," or "for all" and whether students can recognize a correct translation even if they cannot generate the correct translation.

We believe that instruction in propositional logic should emphasize modeling and assessing proper translation and composition techniques. This emphasis must also be coupled with instruction that specifically addresses the misconceptions found in our research. To better inform our decisions of what to model in the classroom, we will interview more propositional logic experts so we can examine how they translate English specifications to Boolean expressions. Once developed, assessments such as the concept inventory can also be used to ensure that new pedagogies actually address student misconceptions. We plan to use the concept inventory to rigorously compare the effectiveness of different teaching methods.

In addition to these instructional recommendations, we believe that emphasis on proper logical thinking and complete enumeration of cases will help students in other computer science learning goals such as learning to debug programs and circuits. If students struggle to properly check that all cases satisfy the English specification they were given in logic contexts, how can we expect them to think logically through what test cases are relevant to debugging a program? The ability to translate English specifications into Boolean expressions with rigorous, systematic methods will provide them with valuable analytical thinking skills that can empower students for future learning in computer science and engineering.

7. ACKNOWLEDGMENTS

We thank Jonathan Markowski for transcribing the interviews. This work was supported by the National Science Foundation under grants DUE-0618589, DUE-0618598, and CAREER CCR-03047260. The opinions, findings, and conclusions do not necessarily reflect the views of the National Science Foundation or the authors' institutions.

8. REFERENCES

[1] M. Ben-Ari. The Concorde doesn't fly anymore. In *the Technical Symposium on Computer Science Education (SIGCSE), keynote address*, Feb 2005.

[2] J. D. Bransford, A. L. Brown, and R. R. Cocking. *How People Learn: Brain, Mind, Experience, and School*. National Academy Press, Washington, DC, 1999.

[3] S. Carey. Sources of conceptual change. In E. K. Scholnick, K. Nelson, S. Gelman, and P. Miller, editors, *Conceptual development: Piaget's legacy*, pages 293–326. Erlbaum, Manwah, NJ, 1999.

[4] P. W. Cheng and K. J. Holyoak. Pragmatic reasoning schemas. *Cognitive Psychology*, 17:391–416, 1985.

[5] M. T. H. Chi, P. J. Feltovich, and R. Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 52:121–152, 1981.

[6] J. Clement. A call for action (research): Applying science education research to computer science instruction. *Computer Science Education*, 14(4):343–364, 2004.

[7] A. A. diSessa, N. M. Gillespie, and J. B. Esterly. Coherence vs. fragmentation in the development of the concept of force. *Cognitive Science*, 28 (6):843–900, 2004.

[8] W. J. Dufresne, R. J. Dufresne, and J. P. Mestre. Using qualitative problem-solving strategies to highlight the role of conceptual knowledge in solving problems. *Am. J. Physics*, 64:1495–1503, 1996.

[9] K. A. Ericsson and H. A. Simon. *Protocol Analysis: Verbal Reports as Data*. MIT Press, Cambridge, MA, 1984.

[10] D. Evans et al. Progress on concept inventory assessment tools. In *the Thirty-Third ASEE/IEEE Frontiers in Education*, pages T4G–1–T4G–8, Nov 2003.

[11] B. Glaser and A. Strauss. *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Aldine, Chicago, IL, 1967.

[12] K. Goldman, P. Gross, C. Heeren, G. Herman, L. Kaczmarczyk, M. C. Loui, and C. Zilles. Identifying important and difficult concepts in introductory computing courses using a delphi process. In *Proceedings of the 39th annual ACM technical symposium on Computer Science Education*, pages 256–260. ACM SIGCSE, March 2008.

[13] R. Hake. Interactive-engagement vs traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. *Am. J. Physics*, 66:64–74, 1998.

[14] M. Huberman and M. B. Miles. *Qualitative Data Analysis: A Sourcebook of New Methods*. Sage Publications, Thousand Oaks, CA, 1984.

[15] S. Kvale. *Interviews: An Introduction to Qualitative Research Inquiry*. Sage, CA, 1996.

[16] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz. A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. In *ITiCSE-Working Group Reports (WGR)*, pages 125–180, 2001.

[17] A. Strauss and J. Corbin. *Basics of Qualitative Research*. Sage, Thousand Oaks, CA, 1998.

[18] R. S. Weiss. *Learning from Strangers: The Art and Method of Qualitative Interview Studies*. The Free Press, New York, NY, 1994.

Code #	Code name	Code Definition
C1	Behavior – Metacognition	Subject shows evidence of monitoring what they are doing and checking their work.
C2	Code - Generic overload	Subject has been exposed to a topic in multiple contexts and is demonstrating interference between the contexts
C3	Code - if-then – action	Subject's example of B in an implication is an action and not a variable
C4	Code - if-then – causal	Subject falsely believes that A being true causes B to be true in if-then statements
C5	Code - if-then – impossible	Subject falsely believes that it is impossible to express if-then in Boolean algebra
C6	Code - if-then - pre-requisite	Subject falsely believes that A being true is a prequisite for B being true in if-then statements(sequencing)
C7	Code - if-then overload	Subject recognizes that the if-then construction has different meanings in different contexts for example Boolean vs. Programming and Boolean vs. English
C8	Code - Pseudocode equivalence	Subject falsely believes that any pseudocode can be expressed with Boolean algebra
C9	Composition - Correct – AND	Subject correctly composes multiple rules together with an AND
C10	Composition - Correct - Subsuming rules	Subject correctly recognizes that a second rule or multiple rules fully subsume a previous rule
C11	Composition – Cowboy	Subject falsely combines rules by using ad hoc composition schemes instead of a systematic approach
C12	Composition - OR	Subject incorrectly composes multiple rules together with an OR
C13	Correct - AND analysis	Subject demonstrates a correct intuitive analysis of what AND means
C14	Correct - Boolean Proof	Subject correctly knows that Boolean algebra can be used to prove equivalence between two expressions
C15	Correct - Converse not true	Subject correctly recognizes that the converse of a statement is not necessarily true
C16	Correct - Formal DeMorgans	Subject uses a correct formal definition of DeMorgan's Law, but doesn't demonstrate a deeper level of understanding
C17	Correct - NAND analysis	Subject demonstrates a correct intuitive analysis of what NAND means
C18	Correct - OR analysis	Subject demonstrates a correct intuitive analysis of what OR means
C19	Correct - OR overload	Subject correctly recognizes that the word OR construction has different meanings in different contexts
C20	Correct - Translation - XOR decomposition	Subject correctly decomposes A XOR B to A'B + AB'
C21	Correct - Truth table interpretation	Subject is able to correctly evaluate a truth table into a Boolean expression
C22	Disagreement	Coders have come to a disagreement
C23	iff - Mistranslation - AND	Subject mistranslates A if and only if B as AB
C24	iff - Mistranslation - XOR	Subject mistranslates A if and only if B as XOR
C25	iff - Mistranslation - unidirectional	Subject mistranslates A if and only if B as implication in one direction
C26	iff violation	Subject fails to recognize that a statement explicitly violates the condition of A if and only if B. (C does not violate C if and only if N)
C27	if-then - Mistranslation - A'B	Subject mistranslates if A then B as AB + A'B
C28	if-then - Mistranslation - AND	Subject mistranslates if A then B as AB
C29	if-then - Mistranslation - B'	Subject mistranslates if A then B as A + B'
C30	if-then - false antecedent	Subject falsely believes that a false antecedent in an if-then statement makes the statement false
C31	if-then – Correct - implication	Subject correctly recognizes if-then as implication
C32	if-then - meaningless	Subject falsely believes that a false antecedent in an if-then statement makes the statement meaningless (programming confusion?)
C33	if-then reversal	Subject falsely analyzes the if A then B statement directionality as being from B to A instead of from A to B
C34	Improper distribution	Subject is unable to perform the distribution operation correctly

Figure 4: List of Codes and Definitions.

C31	if-then – Correct - implication	Subject correctly recognizes if-then as implication
C32	if-then - meaningless	Subject falsely believes that a false antecedent in an if-then statement makes the statement meaningless (programming confusion?)
C33	if-then reversal	Subject falsely analyzes the if A then B statement directionality as being from B to A instead of from A to B
C34	Improper distribution	Subject is unable to perform the distribution operation correctly
C35	Mistranslation - False as nonexistent	Subject incorrectly interprets a variable of value false as not being part of the expression at all (e.g., "A by itself" translated as f(A,B,C) = A instead of = AB'C')
C36	Mistranslation - Not both	Subject mistranslates the phrase "not both" as XOR instead of NAND
C37	Mistranslation - Not both - iff	Subject falsely reinterprets the phrase "not both" as "if and only if"
C38	Mistranslation - XOR	Subject mistranslates A OR B but not both as OR instead of XOR
C39	Process - Boolean to English	Subject tries to interpret a Boolean expression into an English statement
C40	Process - Complete enumeration	Subject analyzes their answer with a full enumeration of cases
C41	Process - Expression Evaluation	Subject can evaluate an expression correctly given a set of input values
C42	Process - Faulty recall	Subject recalls a formula incorrectly
C43	Process - Ignores simplification	Subject fails to see that an expression can be quickly and easily simplified
C44	Process - Incomplete enumeration	Subject only enumerates an incomplete set of test cases and then decides that the incomplete set of test cases proves that their expression is correct
C45	Process - Nonsystematic analysis	Subject reasons to an expression using a nonsystematic listing of cases
C46	Process - Past experience	Subject recognizes that one problem can be solved in the same manner as a previous problem
C47	Process - Reasoning	Subject uses step-by-step reasoning to derive the answer
C48	Process - Reasoning connection	Subject can reason correctly about a Boolean expression and an English expression, and recognizes when the two expressions are not equivalent.
C49	Process - Reasoning disconnect	Subject can reason correctly about Boolean expressions but does not connect it with the English specification
C50	Process - Rote recall	Subject tries to recall information rather than reason through the problem
C51	Translation - And	Subject recognizes that “and” in English is translated as AND in Boolean
C52	Translation - At least one	Subject correctly translates "at least one" as OR
C53	Translation - but	Subject correctly translates the word "but" as AND
C54	Translation - by itself	Subjects correctly translates the phrase "A by itself" as f(A,B) = AB'
C55	Translation - NAND abstraction	Subject correctly recognizes the minterms of NAND and abstracts them from the expression to the term NAND
C56	Translation - nor	Subject correctly translates the phrase "neither A nor B" as NOR
C57	Translation - Not both	Subject correctly translates "not both" as NAND
C58	Translation - XOR	Subject correctly translates A OR B but not both as XOR

Figure 5: List of Codes and Definitions (continued).