EVALUATING PEER INSTRUCTION IN FIRST-YEAR UNIVERSITY COMPUTER SCIENCE COURSES

by

Daniel Zingaro

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Curriculum, Teaching and Learning
University of Toronto

© Copyright 2014 by Daniel Zingaro
Abstract

Evaluating Peer Instruction in First-Year University Computer Science Courses

Daniel Zingaro

Doctor of Philosophy

Graduate Department of Curriculum, Teaching and Learning

University of Toronto

2014

Computer Science 1 (CS1) is the first Computer Science (CS) course taken by those interested in post-secondary CS study. The research community has long lamented high failure rates that seem to be common across schools and countries. In addition, much research over the past several decades suggests that students are not learning what we hope for them to learn.

In this thesis, I investigate Peer Instruction (PI) as a new pedagogical approach for teaching CS1. PI is an active learning pedagogy developed by physics education researchers where students discuss conceptual material in groups and respond to questions using clickers. PI has been taken up by CS educators in the past few years.

I begin by identifying the components of PI and how they might be appropriated to a CS (rather than physics) context. I then offer a quasi-experiment of a PI class and a lecture class: a “learning competition” to determine which pedagogical approach wins. I use a variety of outcome measures, in addition to grade, to gain a deeper understanding into the effect that the pedagogical shift has on particular types of students. I find that students in the PI section score 4.4% higher on a common final exam than students in the lecture section. Though this is substantial in terms of grades, the non-significant statistical result means that we cannot infer that similar gains can be expected in other situations. In addition, PI increases students’ self-efficacy (a construct known to be closely linked to performance accomplishment and persistence) and enjoyment. Further, using achievement goal theory, I show that PI increases interest for those students oriented toward interpersonal achievement but not for those oriented toward intrapersonal improvement. Taken together, my findings offer evidence for the utility of PI as a CS1 pedagogy that not only leads to better grades but also to enhanced sociocognitive and socioemotional investment.
Acknowledgements

Because events are intertwined in ways we cannot always see [...] Sometimes small things can make huge differences.

– K. A. Applegate

My thesis advisor, Jim Hewitt, focuses on online learning, CSCL, and educational technology. My thesis work does not. I am grateful to have an advisor who not only supported my computer science education focus, but also actively encouraged the work. I am a teacher now, supervising some undergraduate students, so I have some basis for this: supervising students in one’s own research area is challenging; supervising research students in a completely different research area ... I don’t even know. My sincere appreciation to Jim for gamely making it happen. All of the advice, meetings, paper edits, AERAs, CSSEs, lunches, emails, and research opportunities are very much appreciated. Thanks, Jim!

I was also fortunate to be part of the Pepper research group, where I worked with Jim Hewitt, Clare Brett, and graduate students on various online learning projects. Often I struggled to balance my time between my thesis work and pursuing interesting ideas brought by Jim, Clare, or their graduate students in our weekly meetings. How many times I got pulled into something before a conference deadline! This was a good problem to have, and I thank the team for sharing their ideas and openly collaborating on our common research goals. In particular, Murat Oztok was my partner in crime on many research projects. I have Murat to thank for teaching me how to publish. Thanks, Oz. We did a lot of work and a lot of goofing-off, and I think both are equally important for an enjoyable and memorable grad school experience. I learned a lot from you. Finally, a shoutout to Rebecca Cober for the conferences, lunches, paper edits ... Continued success in your studies!

I thank my thesis committee — Jim Hewitt, Clare Brett, Jim Slotta, Alyssa Wise, and Doug McDougall — for their support and encouragement of my work. Each of my committee members contributed unique insights based on their own research histories, and I thoroughly enjoyed and benefited from their combined efforts.

Leo Porter contributed considerably to my thinking in several of the papers constituting this thesis. More important than that, Leo is a class act and a fantastic researcher with whom I have had the privilege of collaborating on many projects.

Andrew Petersen and Gerhard Trippen were very supportive of my work and offered their time
and CS teaching expertise over many discussions and lunches. Thanks, guys!

Last but not last (there’s one more after this), thanks to Mark Castrodale for signing up for this Ph.D thing too. Mark has played a role (I said “role”, which doesn’t necessarily mean “positive role” . . .) in literally every step of my education, from being goofs in high school to being older, slightly more refined goofs in grad school. Thanks, Mark. No more school, OK?

Last but not least, thanks to my family. Brut, Gats, Mom, Papa, Nort, Spino. In terms of helping me write this thing, you did nothing. But besides that small detail, I am very lucky to have a family like this.

Chapter 2 contains material from “Peer Instruction in Computing: the Role of Reading Quizzes”, by Daniel Zingaro, Leo Porter, and Cynthia Lee. It appears in Proceedings of the 44th Technical Symposium on Computer Science Education (SIGCSE). The material in this chapter is copyright 2013 by the Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that the 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 in print or the first screen in digital media. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or email permissions@acm.org.

Chapter 3 contains material from “Peer Instruction in Computing: The Value of Instructor Intervention”, by Daniel Zingaro and Leo Porter. It appears in the journal Computers & Education. This paper is copyright 2014 Elsevier Ltd. All rights reserved.

Chapter 4 contains material from “Peer Instruction: a Link to the Exam”, by Daniel Zingaro and Leo Porter. It appears in Proceedings of the 19th annual joint conference on Innovation and technology in computer science education (ITiCSE). The material in this chapter is copyright 2014 by the Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that the 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 in print or the first screen in digital media. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires
prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or email permissions@acm.org.

Chapter 5 contains material from “Peer Instruction Contributes to Self-Efficacy in CS1”, by Daniel Zingaro. It appears in Proceedings of the 45th Technical Symposium on Computer Science Education (SIGCSE). The material in this chapter is copyright 2014 by the Association for Computing Machinery, Inc. (ACM). Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that the 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 in print or the first screen in digital media. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Publications Dept., ACM, Inc., fax +1 (212) 869-0481, or email permissions@acm.org.

Chapter 6 contains material from “Examining Interest and Grades in Computer Science 1: A Study of Pedagogy and Achievement Goals”, by Daniel Zingaro. The paper is currently in submission to a peer-reviewed education journal.
## Contents

1 Introduction ................................. 1
   1.1 Purpose .................................. 1
   1.2 Conceptual Framework ..................... 3
      1.2.1 PI and Try-before-Telling .............. 3
      1.2.2 In-class PI and Situated Learning .... 5
   1.3 Motivation ................................ 7
   1.4 Research Questions and Papers ............ 10
      1.4.1 RQ1: Value of PI Components .......... 10
      1.4.2 RQ2: Comparison of PI and Lecture .... 11

2 The Role of Reading Quizzes ................. 12
   2.1 Introduction ................................ 12
   2.2 What is PI? ................................ 13
   2.3 Why PI in Computing? ....................... 14
   2.4 Setting ................................... 16
   2.5 Data Analysis and Results ................. 18
      2.5.1 Completion and Correctness ............ 18
      2.5.2 Confusion Question ...................... 19
      2.5.3 Correlations with Course Performance .. 21
      2.5.4 Correlations with Clicker Scores .... 22
   2.6 Discussion ................................ 23
   2.7 Future Work ............................... 24
   2.8 Conclusion ................................ 24
3 The Value of Instructor Intervention

3.1 Introduction ................................................. 27
3.2 Background and Related Work ................................. 28
  3.2.1 Peer Instruction ......................................... 29
  3.2.2 Isomorphic Questions ................................. 30
3.3 Hypotheses .................................................. 32
3.4 Method ....................................................... 32
  3.4.1 Study Context ......................................... 32
  3.4.2 Question Administration ................................. 33
  3.4.3 Data Analysis ........................................ 35
3.5 Results ...................................................... 36
  3.5.1 Potential Learners in CS1 ................................. 36
  3.5.2 Comparison of Peer and Combined Modes ................. 37
  3.5.3 Comparison of Normalized Change between Modes ............ 39
  3.5.4 Question Difficulty .................................... 40
3.6 Discussion .................................................. 42
  3.6.1 Quality of Instructor Explanation ......................... 43
  3.6.2 Use of Difficult Questions ................................. 44
  3.6.3 NC as Proxy for Total Learning ......................... 45
3.7 Future Work .................................................. 46
3.8 Conclusion .................................................. 46

4 A Link to the Exam ........................................... 48

4.1 Introduction ................................................. 49
4.2 Background and Literature Review ............................. 50
  4.2.1 Active Learning in Computer Science .................... 50
  4.2.2 Peer Instruction ....................................... 51
4.3 Method ....................................................... 52
  4.3.1 Study Context ......................................... 52
  4.3.2 Question Administration ................................. 53
  4.3.3 Final Exam Questions .................................. 54
4.4 Results .......................................................... 56
4.5 Ancillary Analyses ........................................... 58
4.6 Discussion .................................................. 59
4.7 Conclusion .................................................. 60

5 Peer Instruction Contributes to Self-Efficacy 62
5.1 Introduction ................................................ 63
5.2 Background and Literature Review ..................... 64
  5.2.1 Peer Instruction ........................................ 64
  5.2.2 Self-Efficacy as Valued Outcome .................... 65
  5.2.3 Hypotheses ............................................. 66
5.3 Method ..................................................... 67
5.4 Threats to Validity ......................................... 69
5.5 Results .................................................... 70
5.6 Discussion ................................................ 71
5.7 Conclusion ................................................ 73

6 A Study of Pedagogy and Achievement Goals 74
6.1 Introduction ................................................ 75
6.2 Background and Literature Review ..................... 76
  6.2.1 Predicting Student Performance in CS1 .......... 76
  6.2.2 Achievement Goals and Student Interest .......... 78
  6.2.3 Peer Instruction ....................................... 80
  6.2.4 Context as Achievement Goal Moderator .......... 80
  6.2.5 Research Questions .................................. 81
6.3 Method ..................................................... 82
  6.3.1 Study Context .......................................... 82
  6.3.2 Questionnaire Administration ....................... 83
  6.3.3 Data Analysis ......................................... 85
6.4 Results .................................................... 86
  6.4.1 Final Exam Grade ...................................... 86
  6.4.2 Interest ............................................... 86
List of Tables

2.1 Average per-student responses on content-based questions, broken down by response category. .......................... 18

2.2 Significant Spearman correlations between reading quiz codes and final exam grade. 22

2.3 Significant Spearman correlations between reading quiz codes and unsupervised grade. 22

3.1 Linear Mixed-Effect model of Q1, mode, and the interaction, on Q2. ** $p < 0.01$ 38

3.2 Comparison of Q1, Q1_{ad} and Q2 for Peer and Combined. .......................... 42

4.1 $R^2$ values for each statistical model predicting various facets of final exam performance. Across each row, the change in $R^2$ between adjacent columns is statistically significant. 55

4.2 Statistical models predicting overall final exam performance .......................... 57

4.3 Statistical models predicting final exam code-tracing performance .......................... 58

4.4 Statistical models predicting final exam code-writing performance .......................... 58

5.1 Multiple regressions for post-self-efficacy and final exam grade. .......................... 71

6.1 Range, mean, and standard deviation for continuous variables. .......................... 85

6.2 Correlations (below diagonal) and p-values (above diagonal) for continuous variables. 86

6.3 Multiple regressions for exam grade, interest, and enjoyment. $a : b$ is used to denote the interaction between $a$ and $b$. .......................... 87

6.4 Effect on interest for low, medium, and high levels of performance goals. .......................... 88
List of Figures

2.1 The fifth reading quiz. ................................................................. 17
2.2 Average per-student response percentages, broken down by codes on the 786 submitted responses to the “confusion question”. .................................. 20
3.1 The P (Peer) and C (Combined) administration modes used in the study. ....... 34
3.2 Student performance on P (Peer) and C (Combined) questions. From left to right, the groups of bars represent weak, average, strong, and all students. Error bars show the within-group standard errors. .................................................. 38
3.3 Student normalized change on Peer and Combined questions. From left to right, the groups of bars represent weak, average, strong, and all students. ................. 40
3.4 Student performance on Peer and Combined questions. From left to right, the groups of bars represent Q2 performance on difficult, easy, and all questions. Error bars show the within-group standard errors. ............................................. 41
3.5 Two isomorphic questions on complexity. Correct answers in bold. ................. 43
4.1 The P (Peer) and C (Combined) administration modes used in this study. .......... 54
Chapter 1

Introduction

1.1 Purpose

CS1 (Computer Science 1) is the first course taken by CS majors at the post-secondary degree level. Many students who take this course have no programming experience; for these students, it is their initial exposure to a new problem domain and a new way of thinking. Understanding the material taught in CS1 is critical for students who intend to pursue a CS major, and therefore it is important for teachers and researchers to assess whether students are learning the prescribed curriculum.

Unfortunately, several large research projects have demonstrated that student learning in CS1 is often unsatisfactory. For example, Soloway, Bonar, and Ehrlich (1983) asked students three-quarters of the way through CS1 to write a solution to what is now known as the “rainfall problem”. The problem asks students to write a program that reads integers until the number 99999 is entered, and then averages the numbers (not including the 99999). This problem requires students to use a loop and a conditional, which are two core concepts from CS1. Many educators would agree that students who lack an understanding of one or both of these core concepts have not succeeded in CS1. Only 14% of students were able to write an acceptable solution to the rainfall problem. This poor performance is not restricted to code-writing. Lister et al. (2004) argued that students might not be able to write an entire program correctly, but should certainly be able to trace existing code. Unfortunately, even on these relatively simple tracing tasks, only 60% of students demonstrated adequate understanding of how to trace code and follow the values of variables. Furthermore, since the time that these studies were conducted, there has been little progress in improving student
performance. These problems are as pressing as ever, in spite of the focus garnered by CS1 in the CS education community (Guzdial, 2011). At the department or school level, students’ struggles with CS1 lead to high failure rates and low retention (Bennedsen & Caspersen, 2007).

Seeking to increase student engagement, enjoyment, and performance, CS education researchers have begun investigating the use of new pedagogical approaches to teach CS courses. For reasons mentioned earlier, most of this effort has focused on CS1. Some of the enthusiasm for these new approaches is attributable to successes in other disciplines with rich histories of educational research, such as physics (Crouch, Watkins, Fagen, & Mazur, 2007). One such pedagogical innovation successfully used in physics is Peer Instruction (PI).

PI is the focus of this thesis. It involves students pondering, discussing, and responding to conceptual multiple choice questions presented by the instructor. Multiple choice questions are used because they are easily asked and answered using clickers. Clickers are small, wireless keypads that students use to enter their responses and send those responses to the instructor’s base station. The instructor is thus given an accurate, aggregated form of student response patterns for each question.

The time spent on student interaction, group discussion, and argumentation leaves little time for the instructor to introduce new material. In addition, students must understand something about the topics to be covered in order to engage in productive conceptual debate. To that end, students typically read introductory material and respond to a brief reading quiz prior to each class.

In physics, PI has demonstrated large learning gains compared to traditional lectures. In the largest study of “interactive engagement”, learning gains in the interactive condition were double that of the lecture condition (Hake, 1998). In the last four years, PI has been adopted by some CS educators hoping for similar positive effects in CS.

In this thesis, I investigate the value of PI as an approach for teaching CS1. In Section 1.2, I provide a brief conceptual rationale for the study of PI in CS teaching contexts, and in Section 1.3, I present my personal motivation. In Section 1.4, I outline the studies that make up this work. This thesis is organized as a series of accepted or in-submission peer-reviewed papers, with one paper per chapter. In Chapter 7, I offer a synthesis of the papers and conclude with limitations of the work and future directions.
1.2 Conceptual Framework

1.2.1 PI and Try-before-Telling

Which is better: teaching students with up-front instruction, or supporting students in contextualized, inquiry-based learning? This question has resisted answer, partially due to issues of experimental control. Wise and O’Neill (2009) explain that one difficulty with such comparisons is that there are many variables at play, including the amount of guidance, purpose of guidance, and structure of problems, and that these variables cannot be controlled in a way that respects the precepts of each instructional approach. Answering the question, however, may in fact be unnecessary: it seems that there is a place for both inquiry activities and traditional lecture, and that these activities can work synergistically to increase student learning. This is the thesis of Schwartz and Bransford (1998), who argue that there is a time for lecture, and that this time depends on students being prepared to learn from that lecture.

Students can learn a great deal from lecture when they have sufficient background knowledge, particularly when they have an appreciation for the types of problems that they’d like to be able to solve (Schwartz & Bransford, 1998). Students, of course, do not always possess such background knowledge when entering a lecture, and in these cases may learn considerably less. Without prior knowledge, lectures may provide solutions to problems that, to the students in the room, did not exist. New information becomes a collection of isolated facts that serves no immediate purpose. The solution offered by Schwartz and Bransford (1998) relies on what is known about knowledge differentiation. The idea is that analyzing contrasting cases is a powerful means of drawing attention to features of importance in the problem domain. Students can differentiate their knowledge more effectively through active comparison of cases rather than passive reading of or listening to those cases. However, while such comparisons make important differences salient, students may not understand why the differences are important in the first place, or how these distinctions fit into general theories that govern how the domain fits together. So, once students know that these distinctions exist, it is time to tell them the rest of the story. Rather than introducing a collection of facts in lecture and hoping that students will assemble them in coherent ways, students enter lecture already with relevant information and can leverage the lecture to put the information together.

Schwartz and Bransford (1998) offer three experiments to support this position. In all three, students learned concepts related to memory schemas and encoding. One experiment, experiment 3,
is particularly relevant for the upcoming PI discussion. That experiment concerned a comparison among three groups of students:

- Students who summarized a passage, then attended a lecture. (These students are engaging in “telling” by reading, and then more “telling” by attending lecture. Lots of telling.)
- Students who contrasted cases, then contrasted cases again. (These students are engaging in “contrasting” once, and then “contrasting” again. Lots of contrasting, but no telling.)
- Students who contrasted cases, then attended a lecture. (These students are engaging in “contrasting” and then “telling”.)

To measure learning, all students were later presented with a description of a psychological experiment and asked to predict its outcome. The authors found that the third group of students demonstrated the most learning, perhaps because those students first differentiated their knowledge structures through the contrasting and then superimposed a theory on those differentiations through the lecture. In particular, summarizing a text and then attending a lecture was less effective than comparing and contrasting cases and then attending a lecture.

In PI, students complete a reading quiz prior to each lecture. Are these reading quizzes exercises in summarization, exercises in contrasting, or something else? We do not ask students to summarize a text. On occasion, we do ask students to compare and contrast (for example, the difference between a parameter and an argument, the differences between a list and a dictionary, the differences between sorting methods). But, in particular for CS courses, we ask students to write small pieces of code, explain why code does what it does, and predict the results of code prior to running it. If the “time for telling” occurs when students come to class with unanswered questions and unsettling inconsistencies in their knowledge, then I think reading quizzes will set that stage quite well.

The connection I am making between try-before-telling approaches in PI is that, in both cases, students are required to do “more than reading” prior to class. And, in both cases, the purpose of class is to take students’ pre-class learning to resolution. At this point, however, differences between PI and try-before-telling become apparent. Rather than straightaway listening to an expert account of concepts and their relationships, students first grapple with material on their own and in groups. Only then does the instructor do the telling. One way to understand the individual and group discussion portions of PI is as “more trying”. The work of Schwartz and Bransford (1998) makes it clear that more trying without a follow-up lecture is not beneficial, but it does not make predictions
for the effects of additional trying prior to a lecture. As argued next, I suspect that this additional trying will be particularly valuable in CS contexts.

### 1.2.2 In-class PI and Situated Learning

There are many critiques of the tendency of universities to impart decontextualized, formal knowledge (Laurillard, 2002). I will rely here on a situativist’s critique, culminating in my rationale for why student learning in PI may be superior to student learning in lectures.

The situative view is that learning is tightly bound to and a component part of context, not something that can be formalized in a transfer from teacher to student. When the learner is able to leverage context, solving a problem involves a negotiation of problem, solution, and environment (Brown, Collins, & Duguid, 1989). A classic example of environment-supported learning is that of a weight-watcher who was able to measure “three-quarters of two-thirds of a cup of cheese”, using environmental tools rather than abstract academic knowledge. The argument here is that abstract knowledge by definition has no contextual association, so students will not be able to mobilize such knowledge when working in real situations. Students should come to appreciate an abstraction, not by being told the abstraction outright, but by working toward it through examples grounded in experience. Conceptual knowledge is likened to a set of tools, in that possessing a tool does not, in itself, teach one how to use that tool.

That said, Laurillard (2002) makes a compelling case for why situated learning cannot be directly applied to the school context. She acknowledges that situated learning is useful for explaining learning in trade-based apprenticeships, because such apprentices can situate knowledge in the physical world of their craft. But in the educational context (where we have student apprentices), all we can do is situate students in descriptions of experiences of the world, and descriptions do not easily stand-in for the activities on which they are based (Brown et al., 1989). Schools, compared to other apprenticeships, are mediated (second-order) learning environments where students work with representations of experiences (verbal, pictorial, symbolic), not the experiences themselves. Abstractions will not automatically be learned through example, because such examples are of an abstraction that can only be indirectly experienced by the students. And in CS in particular, the importance of abstraction cannot be oversold. Developing software requires abstracting from problem domain to software domain, from natural language to programming language, from complex and ambiguous to structured. If nowhere else, abstraction is something we uniquely have to get right.
How can we then suffuse students’ educational experience with authentic context, and at the same time guide them towards higher and higher levels of abstraction? My argument is that PI can do this, by serving as a conduit between mediated-world-of-teacher and experienced-world-of-student. When engaged in PI, students work in groups, collaborate, problem-solve, come to consensus, share viewpoints, and ultimately work together. This is not far from the skills required in the software industry at large. To be sure, PI iterations last only a few minutes, and represent teamwork only in quality, not quantity. But, importantly, such PI iterations are not themselves mediated by the teacher. Students can experience first-hand what it is like to problem-solve with like-minded peers.

The PI process affords students an opportunity to engage in something we value as computing professionals. It brings students closer to the day-to-day negotiations and discussions that they will experience in their careers. With suitably-designed questions by the teacher and sufficient effort by the students, it seems feasible that situated learning and the benefits of context can be made available to students after all.

In a lecture setting, an experienced student’s knowledge is not part of the knowledge of the lecture community at large. In PI, where expertise is differentially distributed, suitably-formed PI groups may exhibit some of the features of cognitive apprenticeships. Cooperative learning literature espouses the cognitive and social benefits of groups that are heterogeneous on ability (Vermette, 1998). Furthermore, to the extent that learning is socially mediated (Duffy & Cunningham, 1996), students in all groups — containing a knowledgeable member or not — should learn from the dialogue. Meta-analyses of small group learning confirm that student achievement and attitudes are significantly improved by small-group activities (Springer, Stanne, & Donovan, 1999). It seems reasonable to hypothesize that PI environments, which privilege social learning, would distribute the available prior experience more widely, making it available to students without that experience.

In an ethnography of gendered experiences in CS, Margolis and Fisher (2002) relate the importance of contextualized learning for women’s success. They describe the computing-related narratives of many male college students as reflecting an almost in-born, magnetic attraction to computers. These males are attracted to the physical computer for its own sake, in contrast to females who often understand computers as serving larger societal goals. Women’s stories run counter to the idea that CS majors “hack for hacking’s sake”. They are motivated by a desire to help others, an appreciation of the versatility of CS, and encouragement from family and friends. Large, impersonal and competitive lectures disadvantage women who are bolstered by social and academic support. Margolis and
Fisher (2002) offer three suggestions for creating what they call a course in “contextualized computer science”: situating technology in realistic settings, making connections to other disciplines, and using diverse problems and teaching methods. I suggest that PI addresses, at least partially, the first and third of these guidelines. Garvin-Doxas and Barker (2004) further explain that large lecture settings tend toward — but of course do not deterministically generate — defensive social climates. Defensive climates are characterized by communication patterns that lead to defensive behavior (e.g. lacking empathy, intending to control, demonstrating superiority), rather than behavior focused on exploration and learning. Prior experience tends to become equated with intelligence, exemplified by the common practice of students’ attempting to demonstrate advanced but peripheral knowledge in lecture settings. Women who lack such prior experience may therefore feel devalued and lack competence in their choice of major.

In summary, PI may help all students learn, and may additionally help particular groups of students who tend to be disadvantaged in traditional lecture contexts.

1.3 Motivation

In Winter 2010, I was asked to teach APS105, an introduction to computer science for engineers at the University of Toronto. Computer engineers had their own course, so APS105 was the required computing course for students interested in any other program in the school of engineering. My specific course was a remedial offering for students who had failed the course in the previous offering. Two thoughts led my course design. First, I suspected that these students weren’t terribly interested in learning to program; not only had they chosen a non-CS focus, but they had failed the course once before. Second, I suspected that lecturing to these students would not be very effective. Their previous instructor was well-known and his lectures were not effective in helping these particular students pass. Perhaps I could have offered some reasonable lectures, but would these students want to hear that again? Could I offer something different than the previous instructor?

At around the same time, I was introduced to PI at a CS education conference. It struck me that PI might be effective in teaching APS105 because it was decidedly “not lecture”. I didn’t research the benefits of PI in physics, or why it might or might not work in my context — I simply saw an alternative to lecture and took it.

All but two of my students passed (many of them received excellent marks). My course evaluations
spoke of encouraged and grateful students. I had a great time teaching that course. I gave a post-
course survey to my students where I asked some PI questions, and 90-100% of students agreed with
each positive PI statement that I gave (“using clickers helped me learn”, “discussing with peers
helped solidify understanding”, and so on). So, I wrote a paper in 2010 outlining my experience and
why PI was good. Everything was OK until I read a paper by Wieman (2007), who advocated a
scientific approach to education. The thesis of Wieman’s paper is that we should apply the core
components of science to the teaching of science. We should draw conclusions from objective data
rather than individualized anecdotes. We should convey our results in peer-reviewed venues so that
others may build on what works and move on from what doesn’t work. We should evaluate modern
technologies and use them if they contribute to student learning. Writing papers based on individual
experience, I realized, was not science. Certainly papers like my 2010 paper have a role — in fact, it
was a similar paper that got me started using PI — but for widespread change to take hold, more
convincing evidence was necessary.

I sought to apply this scientific mindset in my thesis. Ultimately, I wanted to know whether
PI was “better” than lecture. If I had one class of students taught using PI and another class of
students taught using lecture, and the students were similar, and the instructors were similar, and
other sources of variation were kept to a minimum, would PI win? This question served as my main
motivation for studying PI. However, I soon realized that such a question was premature. Having
adopted PI from physics, CS educators had in fact co-opted a collection of pedagogical elements that
had been shaped over many years to create PI-in-physics. What was PI-in-CS? What did it look like,
and in what ways was it similar or different from PI-in-physics? Before running my “does PI win”
experiment, I wanted to know whether PI was being used in ways that matched the new disciplinary
context of CS.

I began with reading quizzes. These are small quizzes that physics students complete before class
to prime them for in-class discussion with peers. Reading quizzes had not been studied at all in
CS. In fact, among the small contingent of CS education researchers interested in PI, there were
few if any who had adopted reading quizzes. One reason for the slow adoption of reading quizzes
may be found in a disconnect between quizzes and CS skills. As CS educators, we hope that
our students leave CS1 both with skills (reading, writing, and tracing code) and with conceptual
knowledge (programming constructs, design strategies, understanding abstraction). Should reading
quizzes target skills, concepts, or both? I began with a study of reading quizzes where, like physics
educators, I focused on conceptual quiz questions to prepare students for conceptually-focused class meetings.

Moving on, I asked: what did physics PI students do next? After completing their reading quizzes, what happened in class? Certainly they discussed with peers, but this wasn’t all that they did for each question. Following peer interaction, the instructor led their own discussion: a mix of small lectures and brief discussions with students supporting the various question alternatives. I therefore became interested in which parts of the PI process were useful for engendering learning in CS. Some early work in CS had demonstrated that students seem to learn from peer discussion (Simon, Kohanfars, Lee, Tamayo, & Cutts, 2010), but this said nothing of the longevity of this supposed learning or the importance of the instructor in the learning process. I sought to add to both of these gaps through an investigation of learning attributable to instructor impact, and a study of relationships between PI gains and exam scores.

Having completed this preliminary work, I had confidence that I had implemented satisfactorily each PI element. Reading quizzes appeared effective, students were learning from peers and from the instructor, and learning gains were evident on the final exam. I then moved on to study whether “PI wins”. By now, however, my understanding of what it meant for a pedagogical approach to “win” had changed. Initially, I planned to simply compare grades on a shared final exam, and I would judge the effectiveness of PI based on whether PI students outperformed lecture-taught students. However, I had come to appreciate that grades were not the only measure of a pedagogy’s worth. I was influenced by two areas of the psychological literature: self-efficacy theory and goal theory (Bandura, 1977; Harackiewicz, Barron, & Elliot, 1998). Each of these areas suggests that sociocognitive attributes of students are powerful in shaping consequent learning and success, and that success itself is a multifaceted construct composed of related but independent elements. Rather than study exam grades in and of themselves, I also studied surrounding sociocognitive attributes known to influence future effort and impact student decision-making processes. While grades are certainly the most evident and perhaps most important metric for determining the pedagogical approach that “wins”, studying other outcome measures has helped me gain a more fine-grained understanding in support of PI.
1.4 Research Questions and Papers

Toward the overall goal of evaluating learning outcomes associated with PI, I investigated two broad research questions:

- RQ1: What is the value of each PI component?

- RQ2: Does PI lead to better learning outcomes than traditional lecture-style courses?

1.4.1 RQ1: Value of PI Components

In the few years in which PI has been researched as a CS pedagogical style, most inquiry has focused on the in-class portion of PI. This portion involves the repeated cycle of asking a multiple choice question, having students vote individually on that question, providing time for students to discuss the question with peers, and then asking students to re-vote in light of their discussions. This is an important part of PI, but not the only component. My first research question is: what is the value of each of the components of PI?

In the paper “Peer Instruction in Computing: the Role of Reading Quizzes”, I investigate the use of reading quizzes in an introductory CS course. Reading quizzes are a PI best-practice (Crouch et al., 2007), but have not been frequently used in CS. Furthermore, the usefulness of these quizzes has not been examined in CS courses. This paper shows that students take reading quizzes seriously, that the answers students provide are high-quality, and that quiz performance and effort relate to other aspects of the course. This paper was accepted to SIGCSE 2013, a top-tier peer-reviewed CS education conference.

In the paper “Peer Instruction in Computing: The Value of Instructor Intervention”, I offer a controlled experiment into the learning benefits associated with peer discussion and instructor intervention. At the core of the experiment is the use of isomorphic questions: questions where the concept is the same but the particular surface features and context are different. These questions can be used to measure learning and, depending on placement, can disentangle the learning effects of peer interactions and the instructor. This paper was accepted to Computers & Education, a top-tier educational journal.

In the paper “Peer Instruction: a Link to the Exam”, I correlate student performance on in-class PI questions to various parts of a final exam. I show that learning from peer discussion and instructor
intervention persists until the end of the course, and that such relationships exist for different types of final exam questions. This paper was accepted to ITiCSE 2014, a top international CS education conference.

Collectively, these papers demonstrate that each component of PI has value and is connected to student learning.

1.4.2 RQ2: Comparison of PI and Lecture

Having gained confidence in PI as a viable pedagogical approach for CS1, I next investigate whether PI leads to better learning outcomes than a matched control section that uses a traditional lecture-style approach.

In the paper “Peer Instruction Contributes to Self-Efficacy in CS1”, I investigate whether exam grades and measures of self-efficacy differ between a PI section and a lecture section of CS1. The reason for including self-efficacy in the study is because of the strong relationship between self-efficacy and later performance attainment (Ramalingam & Wiedenbeck, 1998). In general, student self-beliefs are strong predictors of learning, and can sometimes be more important than previous domain-specific instruction (House, 1995). This paper was accepted to SIGCSE 2014.

In “Examining Interest and Grades in Computer Science 1: A Study of Pedagogy and Achievement Goals”, I use achievement goal theory to measure student outcomes in PI vs. traditional lecture sections of CS1. I explore whether there are particular subsets of students who benefit from PI, and expand the notion of “success” to include interest and enjoyment in addition to grades.
Chapter 2

The Role of Reading Quizzes

Note: This paper was presented at SIGCSE ’13 (Zingaro, Bailey-Lee, & Porter, 2013). The second and third authors are Leo Porter and Cynthia Bailey-Lee. They helped generate and test the categories used in the coding scheme. The purpose of this paper was to study the utility of reading quizzes, which are a PI best-practice in physics. At the time of this study, reading quizzes were not being used in CS courses. This paper contributes to RQ1 by investigating the form that reading quizzes should take when adopted from physics to CS courses.

Peer Instruction has recently gained interest in computing as an effective active learning pedagogy. The general focus of PI research has been on the in-class portion of PI: multiple choice questions and group discussion. Here, our focus is the reading quizzes completed by students for purposes of class preparation. These quizzes contain content questions but also ask for difficulties or confusion with course material. Consistent with expectations, we demonstrate that providing correct responses to quiz questions positively correlates with other course assessments. Somewhat counter-intuitively, we find that identifying confusions, noting problematic sections, or asking questions about the reading are also correlated with lab grades.

2.1 Introduction

Peer Instruction (PI) is a pedagogical technique developed in physics that has since been used with considerable success in computing. Physics educators realized that standard lectures are ineffective for teaching core concepts and addressing misconceptions (Crouch et al., 2007), and PI has been
shown to remedy such concerns as measured through course-based assessments and standardized concept inventories. The core of PI is the multiple-choice question (MCQ) posed by instructors and answered by students. Lecture meetings hence become sequences of posing questions, answering questions, and discussing the questions in small groups. As detailed below, such activity driven by MCQs has received significant attention from those CS educators interested in PI. However, PI consists of much more than this “MCQ core”. In particular, in order to engage in productive in-class discussions, students are expected to complete reading and a corresponding assessment prior to each class. Typically, such quizzes, like the MCQs they portend, are graded based on participation and not correctness. To what extent do students complete such reading quizzes? Do students tend to provide complete (correct or incorrect) responses or provide only minimal information? Do students acknowledge confusion or difficulty, and how does this relate to in-class or course-based performance? We seek to broaden understanding of PI in CS through such analyses of reading quizzes. In particular, we demonstrate that quiz completion and quiz correctness positively correlate with other areas of course performance. In addition, we demonstrate that the ability or willingness to identify confusions about the reading also positively correlates with other measures of course performance.

2.2 What is PI?

Each PI meeting consists of the following steps. The instructor begins with a mini-lecture that quickly overviews what the students read for the pre-lecture quiz. Then, the instructor poses a ConcepTest (Crouch et al., 2007)—a multiple-choice question designed both to focus attention on and raise awareness of the key course concept from the mini-lecture (Beatty, Gerace, Leonard, & Dufresne, 2006). After individually thinking about and voting on the correct answer (the solo vote), students discuss the question in small groups, reach a consensus, and vote again (the group vote). Students are encouraged to discuss each answer, verbalizing why it is correct or incorrect (Simon et al., 2010).

While student voting patterns can be estimated using flashcards (Lasry, 2008), instructors can more easily obtain and communicate clicker data. Clickers also afford immediate, accurate vote counts to the instructor, as well as compelling graphical displays for the students. Ideally, each response option for a ConcepTest will correspond to a common student misconception, so that the instructor can identify the range of understandings present among the students (Cutts, Kennedy,
Mitchell, & Draper, 2004). Seeing the histogram of results, students come to realize that they are not alone in their confusion (Knight & Wood, 2005).

Following the group vote and displaying of response graphs, the instructor leads a class-wide discussion. Students are asked to support various response options as the class moves toward consensus. Once the discussion is complete, a new PI cycle begins.

2.3 Why PI in Computing?

Since 2010, many investigations of PI in CS have been published. The first wave of papers focused exclusively on the gains exhibited between the solo and group votes of an MCQ (Simon et al., 2010; Zingaro, 2010). To measure such improvements, the normalized gain metric (NG) is frequently used. NG captures the proportion of students who answer incorrectly in the solo vote but correctly in the group vote. For example, if 60% of students answer correctly in the solo vote and 80% answer correctly in the group vote, the NG for the question is 50% (i.e. 50% of “potential learners” demonstrated new understanding). Representative findings include NGs of 41% in CS1 (Simon et al., 2010) and 29% in a CS1 where all students were taking the course for the second time (Zingaro, 2010). These NG increases are particularly promising in light of the fact that peer discussions generally last only a couple of minutes, suggesting that learning occurs in a relatively brief timeframe.

Unfortunately, NG does not discriminate active learning from passive copying. Perhaps NG increases are due partly or mostly to students copying from peers whom they perceive as more knowledgeable. To help determine the extent to which the cited NG measures reflect one or the other of these scenarios, more recent papers have measured learning through the use of isomorphic questions. The methodology was first used in 2009 in the context of a biology course (Smith et al., 2009). A pair of isomorphic questions tests the same concept, but the questions have different cover stories. In CS, this might reflect two questions both testing understanding of variables, but with different values being assigned or variable assignments occurring in different orders. Answered individually, the second question of a pair can give an idea of what students can now do on their own, outside of their PI groups. In both biology and CS (Smith et al., 2009; Porter, Bailey-Lee, Simon, & Zingaro, 2011), isomorphic questions have been used to demonstrate that significant learning takes place during peer discussion.

The present wave of CS-PI research moves beyond per-question gains to an understanding of
Chapter 2. The Role of Reading Quizzes

One recent argument suggests that PI enables a form of cognitive apprenticeship, where the MCQs set by the instructor can help inculcate students into the professional discipline of CS (Cutts, Esper, Fecho, Foster, & Simon, 2012). When instructors use MCQs to lead students toward the types of thinking engaged in by experts, PI can provide opportunities for the deliberate practice and scaffolding that have been lauded by decades of educational research (Ericsson, Krampe, & Tesch-Romer, 1993). Yet, all of this research presupposes, at least implicitly, the requirement for students to be well-prepared to deliberately engage with PI materials. There is little time to “lecture” in a PI class, so preliminary material cannot be taught in the traditional way.

In the physics PI literature, student preparation often comes in the form of reading quizzes (Crouch et al., 2007), and it is likely that student experiences on reading quizzes are important in order for the core of PI to be effective. Authors of seminal physics PI research (Crouch & Mazur, 2001) note that it can be effective to administer short, three-question web-based quizzes prior to each lecture. The first two questions relate directly to the content covered in the required pre-class reading, and the third question asks, “what did you find difficult or confusing about the reading? If nothing was difficult or confusing, tell us what you found most interesting. Please be as specific as possible.” Instructors can use the responses to these quizzes to support a form of just-in-time teaching whose focus is problematic areas reflected in students’ quiz responses.

Though the use of reading quizzes is a common practice in physics PI, CS PI courses sometimes do not use such quizzes (Simon & Cutts, 2012). When CS instructors do administer reading quizzes, research publications tend to jump straight to an analysis of in-class MCQs without reference to the preliminary reading quizzes. For example, Zingaro (2010) note that reading quizzes were used in the style of (Crouch & Mazur, 2001) and offer a sampling of the types of questions included in the quizzes, but say little in regard to correctness, completion, effort, acknowledgment of confusion, and so on. That said, we do know that students find reading quizzes valuable. For example, in work using clickers, occasionally in the PI format, it was found that 76% of Data Structures students agreed that reading quizzes helped them recognize and focus on difficult course concepts (Pargas & Shah, 2006).

One recent paper suggests that the form of reading quizzes in CS should leverage the activity-based nature of our subject (Esper, Simon, & Cutts, 2012). Rather than decoupling reading from experimentation, these authors support students in experiment-based learning where book and programming environment are used concurrently. Interspersed with reading, students are guided
to emulate expert-like behavior: identifying learning goals, exploring small code segments, making
predictions, and so on. Then, at the start of each lecture, students are quizzed on the homework
material before the instructor launches into the first PI cycle. Student usage and valuation of these
exploratory homeworks were promising: 50% of students “almost always” or “always” completed the
homeworks; 58% of students “always” felt that homework helped them understand lecture; and those
who read the text and wrote code before, during or after that reading outperformed the students
who did not write the code. At the same time, only 30% of students completed the homeworks as
intended: flexibly moving between reading, experimenting, and exploring (rather than, for example,
serially reading and then coding).

The work of Esper et al. (2012) is similar to the present work, but with two important differences.
First, those authors used exploratory homeworks rather than standard PI reading quizzes, and it
is the latter that is currently in more common use. In particular, those authors did not seek input
from students on what they find difficult or confusing, which is a key role of PI reading quizzes.
Second, they did not report on actual student performance on the quizzes, instead relying on student
perceptions of the utility of doing the homework. Here, we seek to explore traditional PI reading
quizzes by examining completion, correctness, confusion, and the ways in which these variables relate
to course-based performance.

2.4 Setting

We implemented PI in a small (40 students) remedial CS1 course for Computer Engineers taught
by the primary author at a research-intensive Canadian university. All students in the class were
unsuccessful in completing a traditional (non-interactive) 12-week version of the same course the
previous semester, and require this course to continue in the engineering program. We used C to
teach the major topics of the course: selection, iteration, functions, arrays, recursion, sorting, pointers
and memory allocation (linked lists were taught in the prior offering but not in this remedial offering).
We met for three fifty-minute lectures each week for 10 weeks, and were supported by weekly practical
lab sessions and weekly tutorials.

Our motivation for using PI in this remedial setting was based on a reported interaction between
pedagogy (PI or traditional) and experience (Lasry, Mazur, & Watkins, 2008). That study found that
students with low background knowledge gain as much from PI as students with high background
knowledge gain from traditional instruction. It is likely that students in our remedial CS course lack significant background knowledge; therefore, we believed that PI would be particularly effective for achieving learning gains commensurate with those of the successful students from the semester prior.

Before each class, students were required to read one or two sections of our textbook (Carter, 2008), and complete an online reading quiz, typically with three questions. We used the format suggested in Crouch and Mazur (2001) and described above (i.e. two or three content questions followed by the “confusion question”). 6% of students’ grades were allocated to completion of the quizzes: marks were awarded for effort, not correctness. In general, reading quiz questions were designed to skim the surface of what the students had read, while deeper application of that material was designated for in-class ConcepTests. Figure 2.1 contains one example reading quiz from week 2; the remainder are available on a PI resource page (Zingaro, 2012). An additional 6% of students’ grades were allocated to participation in ConcepTests and, again, marks were awarded for effort, rather than correct answers. Further details of the class offering can be found in (Zingaro, 2010).

1 Is the expression ‘a’ < ‘C’ true (1) or false (0). Please carefully explain your answer.
2 The following program does not run.

```c
#include <stdio.h>

int main (void) {
    int q = 4;
    if (q == 4)
        printf (“The if-statement is running.
”);
    printf (“Another printf.
”);
    else
        printf (“This is the else.
”);
    return 0;
}
```

Try to compile the program. Explain each error that your compiler gives, and suggest a solution.

3 What did you find particularly difficult or confusing about the material you read? If nothing was difficult or confusing, tell us what you found most interesting. Please be as specific as possible: this is meant to help you focus on problematic areas, as well as help me make more meaningful lectures.

Figure 2.1: The fifth reading quiz.
2.5 Data Analysis and Results

2.5.1 Completion and Correctness

In total, there were 27 reading quizzes. On average, students submitted 21.2 reading quizzes, for an average submission rate of 78.5 percent. As will be shown, some of these “submissions” contained empty (or essentially empty) responses.

The reading quizzes contained a total of 52 content-based questions (i.e. not including the “confusion question”). To investigate the quality of the responses submitted, the primary author (also the course instructor) graded each response to a content-based question on a four-point scale:

- G0-Empty: Essentially no response (e.g. submitting “no” or “not sure”, or leaving the response completely blank).
- G1-Incomplete: something submitted, but very incomplete (e.g. answering half of a question or tracing only partway through a program).
- G2-Incorrect: complete response, but incorrect.
- G3-Correct: complete and correct response.

Table 2.1 contains the average per-student percentage of responses that fell in each of these four categories.

<table>
<thead>
<tr>
<th>Category</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>G0-Empty</td>
<td>1.8</td>
</tr>
<tr>
<td>G1-Incomplete</td>
<td>10.5</td>
</tr>
<tr>
<td>G2-Incorrect</td>
<td>37.6</td>
</tr>
<tr>
<td>G3-Correct</td>
<td>50</td>
</tr>
</tbody>
</table>

G2-Incorrect and G3-Correct reflect complete responses to the question. In the case of G2-Incorrect, students worked through a problem with misconceptions that led them toward incorrect conclusions, but the execution was nevertheless complete. For this reason, we argue that 87.6 percent of responses may be educationally useful as they represent students engaging with questions prior to lecture. 12.3 percent of responses, plus the questions not submitted at all (21.78 percent), may suggest inadequate preparation for lecture. Lastly, we see the rate of G3-Correct responses as surprisingly
high: students knew that they were being graded only on submission (not correctness), so there was no immediate mark-based benefit for students to answer questions correctly. In addition, these are students that struggled in the past, and it is encouraging that half of the responses submitted by such “weak” students are correct with no further instructor intervention.

2.5.2 Confusion Question

On all but two of the reading quizzes, we provided the “confusion question” as the final question for students to answer. As indicated in Figure 2.1, we suggested to students that their responses might help the instructor create more accurately-targeted lectures. There is also recent evidence that students who are able to acknowledge confusion or difficulty can be well-positioned to excel in post-secondary courses (Mazur, 2011).

To determine the extent to which students use this question for such purposes, we read through the responses searching for themes. Themes were initially based on the wording of the question itself, and were refined until all data could be captured and no themes failed to represent data. The following themes emerged:

**C0-Interesting:** identifies something interesting.

**C1-Not Interesting:** states that nothing in the reading was interesting.

**C2-Confusing:** identifies something confusing or unclear.

**C3-Not Confusing:** states that nothing in the reading was confusing.

**C4-Difficult:** identifies something that is difficult or “hard”.

**C5-Not Difficult:** states that nothing in the reading was difficult.

**C6-Forward:** identifies something difficult that has not yet been reached in the course. (This is an artifact of the students having taken the course unsuccessfully once before.)

**C7-Problematic:** identifies something that cannot be understood or that something is proving problematic or troublesome. (This likely overlaps with student conceptions of difficulty, though with different word choice. For example, students often used phrases such as “I did not
Figure 2.2: Average per-student response percentages, broken down by codes on the 786 submitted responses to the “confusion question”.

understand” or “I had trouble with this concept”. When they did not explicitly use the word “difficult” or “hard”, we coded the response as problematic rather than difficult.)

C8-Not Problematic: explicitly states that something is understood.

C9-Questioning: poses a question without explicitly indicating confusion or difficulty.

C10-Ambiguous: identifies a topic, but does not indicate whether that topic is interesting, confusing, difficult, etc. For example, a student might indicate simply “recursion”.

C11-Other: comments relating to midterms or exams, or comments too short to otherwise classify (e.g. “so far so good”).

C12-Empty: completely blank submission or a submission containing no information (e.g. “N/A”, “nothing”, “no”).

Figure 2.2 provides the average per-student percentages of these codes in the 25 reading quizzes where the “confusion question” was asked. A total of 786 student responses were coded. The sum of the averages is more than 100% because responses could be coded by multiple codes (e.g. a response that asks a question but also acknowledges confusion).

Striking in this data is the number of questions posed to the instructor by students, an indication that areas of difficulty were not only uncovered but explicitly framed in the form of questions to
which the instructor could respond. As a case in point, the responses to the quiz in Figure 2.1 contained eight instances of students asking questions. Two of these asked about the C `switch` statement that was topical but not tested in this quiz. A further two questions asked about the code that had been supplied, and why the code works at all without a `scanf` statement. The remainder of the questions asked how to chain `else if` branches. In our experience, students simply do not ask about course material to this degree when reading quizzes are not used; in such cases we find the vast majority of questions specifically focused on the students’ current assignment rather than course material per se. We hypothesize that reading quizzes have the desirable effect of encouraging students to ask questions continuously, rather than clustering questions around assignment deadlines.

Unfortunately, the mingling of confusing, difficult, and interesting in the question text also led to 51 ambiguous responses. For example, again in the responses to the quiz in Figure 2.1, one student provided simply “the flowcharts”. This referred to the approach used in our text that depicted control statements in the form of flowcharts, though whether this was problematic or interesting is unclear.

The codes representing students who are not confused or who found nothing difficult mostly reflect a rephrasing of the question text. That is, we received many responses of the form “nothing was confusing or difficult”. Given no further elaboration, it is not clear whether this was an attempt to submit content for the question (rather than leaving it blank), or whether they genuinely believed that nothing was confusing or difficult.

The 122 C11-Other responses were made mostly of high-level comments on the course, students’ favourite C constructs, reflections on the prior course offering, and suggestions for how lectures could be improved (though of course without providing explicit questions).

### 2.5.3 Correlations with Course Performance

Table 2.2 gives the significant Spearman correlations ($p < .05$) between reading quiz codes and final exam grade. (This correlation was used rather than the Pearson correlation because reading quiz codes are ordinal rather than interval.) The final exam was worth 45% of students’ grade. Perhaps unsurprisingly, students submitting many G1-incomplete responses do poorly on the final exam, and those submitting many G3-Correct responses do well. G2-Incorrect was negatively but non-significantly correlated ($r = -.23$) with final exam grade. None of the codes from the “confusion question” were significantly correlated with exam performance, though C7-Problematic approached
Table 2.2: Significant Spearman correlations between reading quiz codes and final exam grade.

<table>
<thead>
<tr>
<th>Code</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content responses submitted</td>
<td>0.32</td>
</tr>
<tr>
<td>G1-Incomplete</td>
<td>−0.37</td>
</tr>
<tr>
<td>G3-Correct</td>
<td>0.51</td>
</tr>
</tbody>
</table>

Table 2.3: Significant Spearman correlations between reading quiz codes and unsupervised grade.

<table>
<thead>
<tr>
<th>Code</th>
<th>Correlation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content responses submitted</td>
<td>0.80</td>
</tr>
<tr>
<td>G0-Empty</td>
<td>−0.35</td>
</tr>
<tr>
<td>G3-Correct</td>
<td>0.72</td>
</tr>
<tr>
<td>C2-Confusing</td>
<td>0.40</td>
</tr>
<tr>
<td>C7-Problematic</td>
<td>0.31</td>
</tr>
<tr>
<td>C9-Questioning</td>
<td>0.57</td>
</tr>
</tbody>
</table>

Students’ unsupervised term work (worth 18% of the final grade) involved the completion of five labs on which students worked alone. Table 2.3 provides the significant Spearman correlations (p < .05) between codes on the reading quizzes and students’ unsupervised term work grade. Number of content responses submitted, a clear effort measure, had the strongest correlation with unsupervised term work. Unlike with final exam score, students willingness to ask questions and acknowledge difficulty and confusion were significantly correlated with unsupervised work. In addition, G2-Incorrect was positively but non-significantly correlated with unsupervised term work (r = 0.29).

2.5.4 Correlations with Clicker Scores

Clicker responses, like reading quizzes, were assessed by contribution rather than correctness. Still, we wondered to what extent scores on reading quizzes correlated with clicker correctness. Our hypothesis was that students who did well on reading quizzes would be well-prepared for lecture and hence correctly answer more clicker questions. We focus here only on solo votes (i.e. those votes occurring prior to group discussion). Our rationale for excluding the group vote is that such scores comingle students’ incoming knowledge and knowledge obtained through group discussions. We found significant positive correlations between number of clicker questions answered correctly and both the number of submitted content responses (r = 0.65) and number of G3-Correct submissions (r = 0.48). There was also a significant correlation between clicker correctness and C9-Questioning (r = 0.31). Interestingly, though nonsignificant, there was a moderate negative correlation (−0.16) between clicker correctness and C3-Not Confusing. There were two other non-significant but positive
correlations with clicker correctness: C7-Problematic \( (r = 0.23) \) and C2-Confusing \( (r = 0.22) \).

2.6 Discussion

The fact that students submitted almost 80 percent of reading quizzes and that 87.6 percent of question submissions were of acceptable quality suggests that students took reading quizzes seriously. There was no obligation for students to answer questions completely; the instructor explicitly told students that any submission counts. It is heartening that students took it upon themselves to put forth effort rather than simply answering for the sake of answering. This corroborates an end-of-term survey finding reported in Zingaro (2010) where 92 percent of these same students agreed with the claim that reading quizzes helped identify difficult concepts in the reading. The current paper strengthens the belief that students find reading quizzes valuable.

The “confusion question” provided useful data in terms of student questions and confusions, though the wording of the question sometimes led to responses that could not be unambiguously categorized. We argue that receiving this quantity of questions and feedback is further evidence that students cared about the quizzes and used the opportunity to communicate with the instructor. The course instructor was careful to address this feedback personally or at the beginning of the next lecture, which we suspect contributed to the quantity of feedback received. Though feasible in small classes, the workload involved in responding personally to quiz submissions in large classes is likely prohibitive. That said, we believe it is important for students to understand that we take quizzes seriously and treat them as a core component of PI practice.

There were no relationships between codes on the “confusion question” and final exam performance. Yet, codes related to confusion and questioning were highly correlated with unsupervised term work, affirming the relationships discussed by Mazur (2011). The reason for this difference is unclear and deserves future consideration. We do note that confusions were largely local (“pointers”) rather than global (“what is the relationship between the heap, pointers, and the memory model”), and so perhaps related more strongly to small-scale assignments rather than the integrative exam. Also, assignments provide students some time in which to acknowledge confusions, obtain answers or clarifications, and then apply that knowledge in the context of their deliverable. Such feedback-driven progress is not possible in an exam setting.

Correlations between reading quiz correctness and clicker questions suggest that the quizzes might
prepare students for and increase learning in the course. Importantly, we cannot rule out alternate hypotheses: it is equally plausible, for example, that stronger students do well on both quizzes and clicker questions, or that those putting forth more effort on the quizzes also put forth more effort toward answering clicker questions correctly. The relationship does give us confidence that students are finding reading quiz questions and clicker questions to be similar in terms of content and/or difficulty.

2.7 Future Work

There are several areas requiring future work. First, we suggest that the “confusion” question be disaggregated to make interpretation of responses more straightforward. For example, the question could be divided into prompts that ask for confusing, difficult, and interesting aspects of the course reading; students could be required to respond to at least one of these prompts. Students could also be provided a space explicitly for asking questions. An analysis of responses to these new questions would be interesting as a comparison to what we have offered here. Second, it will be interesting to compare traditional PI reading quizzes as discussed here and other preparatory assignments such as exploratory homeworks, effectively bridging the span between our work and that presented in Esper et al. (2012). Third, student perceptions of reading quizzes would be useful in the context of our correlation-based causation hypotheses. Do students feel that reading quizzes prepare them for lectures, labs, exams, or the course in general? Do they see relationships between quiz questions and clicker questions? Further student perceptions of reading quizzes are required to determine the accuracy of our correlation-based causation claims.

2.8 Conclusion

Analysis of reading quizzes is underaddressed in the current PI literature in computing. We provide one such analysis here, focusing on completion, correctness, confusion, and relationships to course-based assessments and PI questions. We find that students complete a large portion of reading quiz questions and that their responses suggest a reasonable level of effort and correctness. Students’ responses to the “confusion question” demonstrate similar levels of attention: students ask many questions, acknowledge significant confusion, and note particular difficulties. We demonstrate that
reading quiz performance correlates with performance in other areas of the course, suggesting that reading quizzes target important course material and may confer advantages to students who expend (or are willing to expend) the effort required to answer the reading quizzes correctly. We also find correlations between identifying confusions and questions on the one hand and performance on unsupervised term work and clicker questions on the other. We encourage further investigations into the utility of reading quizzes, their relationship to learning and the rest of the course, and their applicability to other CS courses in which PI is implemented.
Chapter 3

The Value of Instructor Intervention

Note: This paper was accepted in Computers & Education (Zingaro & Porter, 2014b). The second author is Leo Porter. Leo is the colleague mentioned in the paper who helped me validate the isomorphicity of PI questions. The purpose of this paper was to continue answering RQ1, the research question focused on the utility of each PI component. Here, I study the learning gains associated with peer discussion and instructor-led classwide discussion. I use isomorphic questions, placed in one of two configurations, to experimentally measure this learning. At the time this paper was written, isomorphic questions had not been used for this purpose in CS courses.

Research has demonstrated that Peer Instruction (PI) is an attractive pedagogical practice in computer science classes. PI has been shown to improve final exam performance over standard lecture, reduce failure rates, contribute to increased retention, and be widely valued by students. In addition, a recent study using isomorphic (same-concept) questions found that students are learning during peer discussion and not merely copying from neighbors. Though this prior work is useful for evaluating peer discussion, it does not capture learning that takes place after peer discussion when the instructor further expands on the concept through a whole-class discussion. In the present work, isomorphic questions were used to determine the value of a PI question from start to finish: solo vote, group discussion, group vote, and instructor-led classwide discussion. The analysis revealed that the value of the instructor-led classwide discussion was evident in increased student performance over
peer-discussion alone (raw gains of 22% compared to 14%). Moreover, the instructor-led discussion was highly valuable for all groups of students (weak, average, and strong) and was of particular value for weak students. Importantly, the largest gains were associated with more challenging PI questions, further suggesting that instructor expertise was valuable when students struggled.

3.1 Introduction

Peer Instruction (PI) is a pedagogical technique developed in physics that has since been used with considerable success in computing. At the core of this pedagogy is the ConcepTest (Crouch et al., 2007): a multiple-choice question answered by students typically using clickers. Each ConcepTest sets off a well-defined pedagogical protocol: students first answer the question individually (solo vote), then discuss the same question for several minutes with their neighbors, and finally re-vote on the question in light of the group discussion (group vote). Following the group vote, the instructor facilitates a classwide discussion and explanation of the ConcepTest, and can adjust the remainder of the class to target student difficulties.

In physics, it has been repeatedly demonstrated that PI vastly improves student performance on post-course concept inventories (Crouch et al., 2007; Hake, 1998). In CS, there are few concept inventories, and those that exist have not been widely deployed and established (Tew, 2010). Therefore, CS education researchers have used other metrics to measure the effectiveness of PI. PI in computer science has been found to improve final exam performance (Simon, Parris, & Spacco, 2013), reduce failure rates (Porter, Lee, & Simon, 2013), and contribute to improved retention (Porter & Simon, 2013).

In addition to overall student outcomes, the value of PI in the classroom can be measured quantitatively by the shift in student correctness between the solo vote and the group vote (Porter, Garcia, Glick, Matusiewicz, & Taylor, 2013; Simon et al., 2010; Zingaro, 2010). Such numeric gains from peer discussion suggest, but do not imply, conceptual gains. Is peer discussion helping students conceptually, or are students largely copying from neighbors? Recent work by Porter, Bailey-Lee, et al. (2011) used isomorphic questions to verify that students are indeed learning from the peer discussion.

Since PI is squarely a student-focused pedagogy, it is unsurprising that the nascent PI-CS literature has focused on determining the value of the peer discussion portion of PI. However,
measuring gains solely from peer discussion may underestimate the total learning conferred through a PI ConcepTest. Each PI cycle concludes with the instructor providing the correct answer and engaging in a classwide discussion meant to provide students with knowledge uniquely held by a subject matter expert: illuminating each response choice, discussing why the concept is important at large, and aiding students in integrating this concept with other concepts in their construction of increasingly expert-like maps of core disciplinary areas. This “instructor intervention” must be captured if we are to truly evaluate student learning from PI ConcepTests.

In this paper, we offer the first account in CS education of the additional benefits conferred through instructor intervention. We also offer the first account in the sciences of the effect of question difficulty on learning gains conferred through peer discussion or instructor intervention.

We conducted a controlled experiment on a large introductory computer science (CS1) class in order to compare peer discussion alone versus peer discussion combined with instructor intervention. We find statistically significant differences between these two modes, clearly demonstrating the importance of instructor intervention within the peer-based PI framework. In addition, we conduct analyses on student ability groups and find that instructor intervention is particularly useful for low-performing students.

The contributions of this paper include:

- A first CS study that measures both peer learning and instructor-led learning. We compare these results with a similar study in biology (Smith, Wood, Krauter, & Knight, 2011).
- Evidence that instructor-led discussion is valuable for weak, average, and strong students alike.
- Evaluation of question difficulty demonstrating that difficult questions are particularly valuable for student learning.

### 3.2 Background and Related Work

While our focus in this paper is the PI pedagogy, we note briefly that the CS research community is currently investigating many active and collaborative forms of teaching and learning. For example, the flipped classroom, pair programming, and lectures supported by visualizations (Lockwood & Esselstein, 2013; McDowell, Werner, Bullock, & Fernald, 2006; Kaminski, 2008) have all been advanced as alternatives or complements to traditional lecture-based teaching.
3.2.1 Peer Instruction

As described previously, each cycle of the in-class portion of “classic PI” involves students answering a question on their own (solo vote), discussing with their neighbors, and voting again (group vote); see panel A in Figure 3.1. Following the group vote, the instructor leads a classwide discussion related to the core concept and its misconceptions. The instructor may lecture briefly, or ask students to explain why specific distractors were compelling (“Why did you choose A? What misunderstanding might have led you to choose B?”).

Core to the implementation and evaluation of PI is the use of clickers: small devices, similar to television remote controls, that enable students to transmit responses to the instructor’s base receiver (Blasco-Arcas, Buil, Hernández-Ortega, & Sese, 2013). Clickers provide students a low-risk, “fun” technology with which to commit to response choices and engage with the material (Simon et al., 2010; Knight & Wood, 2005). While others have argued that learning is equivalent whether clickers or flashcards are used (Lasry, 2008), clickers afford the immediate generation of accurate response graphs that are of value to both the students and teacher. For example, teachers can use the graphs to facilitate discussion, and students can use the graphs for purposes of formative feedback (Moss & Crowley, 2011). Clickers also generate interaction between students and their peers and instructor, leading to active learning and engagement (Blasco-Arcas et al., 2013). Naturally, clickers as a technology must be paired with an effective pedagogy in order for the positive effects of clickers to be realized. One such pedagogy, and the pedagogy used in the present work, is PI.

Discussion-based pedagogies like PI limit the amount of lecture and increase time spent in discussion and problem-solving. Therefore, to best use available class time, teachers often require students to complete pre-lecture reading (Crouch et al., 2007). In addition, some authors argue that a PI mindset should extend to all aspects of a course, including tutorials and labs (Zingaro, 2010). For this reason, CS researchers have begun a more nuanced inquiry into the effects of PI that go beyond solo-to-group gains. See Zingaro, Petersen, and Craig (2012) and Zingaro, Bailey-Lee, and Porter (2013) for reviews of this work.

Much recent literature suggests that PI is a highly effective pedagogy for teaching CS courses; for example, PI reduces failure rates (Porter, Lee, & Simon, 2013), contributes to increased retention (Porter & Simon, 2013), and yields exam-inferred learning gains compared to traditional course offerings (Simon, Parris, & Spacco, 2013). PI has been used successfully across the CS curriculum,
from introductory courses in C and Matlab (Zingaro, 2010; Lee, 2013) to senior-level courses (Porter, Bailey-Lee, et al., 2011). However, there are no studies that examine the instructor’s impact on learning in a PI course.

Ideally, peer discussion should engage students in deep processing of material, including comparing and contrasting views with peers as they consider each response choice. We believe that such processing sets up a context in which a follow-on lecture would be particularly effective. This belief stems from constructivist learning theory; specifically, studies have shown that students who have actively compared and contrasted material are better primed to learn from instructor explanation (Schwartz & Bransford, 1998). Through peer discussion, students may come to directly understand the relevant concept. However, if they remain confused, then exchanging perspectives and engaging in argumentation with peers remains educationally important. Bjork has used the term “desirable difficulties” (Bjork, 1994) to characterize student struggles that yield payoff at a later time. We suggest that after engaging with difficult questions, students are particularly prepared and motivated to learn from a more coherent discussion led by the instructor.

3.2.2 Isomorphic Questions

To examine the impact of peer discussion alone versus peer discussion and instructor intervention, we use a method similar to that used in a recent biology study (Smith et al., 2011). In computing, our work aligns most closely with that of Porter, Bailey-Lee, et al. (2011). Each of these studies of interest used isomorphic questions to assess the extent to which students’ PI-based learning was generalizable across contexts. Isomorphic questions are designed to test the same concept, but use different “cover stories” or parameter values (Smith et al., 2009).

Porter, Bailey-Lee, et al. (2011) used isomorphic questions in their study of learning from peer discussion. Specifically, a first question was presented, on which students voted individually (Q1), discussed, and voted again (Q1\textsubscript{ad}, for “after discussion”). Then, a second question (isomorphic to the first) was presented, on which students voted individually (Q2). Students did not have the opportunity to discuss the first question between the completion of Q1\textsubscript{ad} and the beginning of Q2, and were not shown the answer to the first question until after Q2.

Note that the terms above will be used throughout the present paper. Q1 refers to the initial vote, Q1\textsubscript{ad} refers to the vote after discussion, and Q2 refers to the individual vote on the second, isomorphic question. The change in correctness from Q1 to Q2 is representative of the amount of
learning that occurred during the PI process.

Porter, Bailey-Lee, et al. (2011) highlighted that one particularly important group of students is those who answer Q1 incorrectly, Q1\textsubscript{ad} correctly, and Q2 correctly. These are students that did not understand the first question, learned from their peers, and were able to apply that understanding to the context of Q2. In the upper-level courses in their study, Computer Architecture and Theory of Computation, respectively, 20% and 13% of students demonstrated these gains.

In addition to these absolute gains, Porter et al. report gains within particular groups of students. They define the Potential Learner Group (PLG) as those students who answer Q1 incorrectly and Q1\textsubscript{ad} correctly. These students appear to learn from the peer discussion, because they correctly answered the same question following peer discussion. That is, they potentially learned from peer discussion, and Q2 can be used to determine the extent to which this learning was generalizable (rather than, say, a result of uncritically copying peers). In Computer Architecture and Theory of Computation, respectively, 76% and 62% of the PLG correctly answered Q2. The clear majority of learning was generalizable to the isomorphic question.

In a recent study in biology (Smith et al., 2011), the authors varied this protocol slightly in order to compare gains from individual parts of the PI process: peer discussion, instructor, and the combination of peer discussion and instructor. That study was conducted in a genetics course for majors and used three modes of isomorphic question administration:

- Peer: students answered Q1 individually, discussed the question, and answered Q1\textsubscript{ad}; the correct answer was shown, and students then answered Q2. (Besides the positioning of the display of Q1’s correct answer, this is otherwise identical to the mode used by Porter, Bailey-Lee, et al. (2011).)

- Instructor: students answered Q1 individually, the instructor led a classwide discussion, and students answered Q2.

- Combination: students answered Q1 individually, discussed the question, and answered Q1\textsubscript{ad}; the instructor then led a classwide discussion, and students answered Q2.

Smith et al. (2011) used Normalized Change (NC) to compare gains made in the three modes. NC measures the amount of learning as a fraction of available learning. (For example, imagine that 60% of students answer correctly in the solo vote and 80% of students answer correctly in the group vote. This is a 20% gain, but NC divides this 20% by the possible amount of learning following the
solo vote. That is, \( \frac{20}{(100 - 60)}\% = 50\% \), so this question has a 50\% NC.) These authors found that the combination mode was more effective in terms of NC than the peer and instructor modes. Furthermore, this finding held across ability groups: the combined mode was best for weak, average, and strong students.

### 3.3 Hypotheses

From the prior work on biology (Smith et al., 2011), we expect:

- CS students will learn more from the combination of peer discussion and instructor intervention than from peer discussion alone.

Moreover, from our experience with PI, we have observed that the instructor performs critical interventions on difficult questions. On such questions, students struggle on the initial vote and often continue to struggle during peer discussion. As argued earlier, we see peer discussion as setting the context within which students can learn a great deal from follow-up lecture. This may be further heightened when students have just finished grappling with difficult questions. Our second hypothesis is therefore:

- Compared to easy questions, the instructor will have a larger contribution to student learning gains on difficult questions.

### 3.4 Method

#### 3.4.1 Study Context

The course for this study was an introductory computer science course (CS1) taught in Fall 2012 at an undergraduate campus of a large Canadian research-intensive university (131 students wrote the final exam). Since 2007, the course has been taught using the Python programming language. Python has recently gained traction as a language for introductory instruction because of its clean syntax and extensive library of functions (e.g. graphics, Internet applications, user interface design, etc.) The course covers traditional CS1 topics in imperative programming, and also spends one week each on sorting, complexity, and object-oriented programming. The course took place over 12 weeks, with 3 50-minute lectures per week. Prior to each lecture, students completed a reading
quiz; the instructor read the responses before class to help shape the lecture. The reading quizzes were marked based on completion (not correctness) and were worth 4% of students’ final grade; in-class clicker participation accounted for a further 5% of students’ grade. The course instructor was a senior education graduate student with significant PI and CS teaching experience, and had taught CS1 using PI several times. The course instructor developed new PI materials for this course offering; we discuss one question later in the paper (Figure 3.5), and all questions are available at peerinstruction4cs.org.

3.4.2 Question Administration

Each lecture contained an average of three PI cycles, with one of the cycles augmented with a follow-up isomorphic question. It was necessary to gain confidence that questions were actually isomorphic and that the difficulty of questions within pairs was comparable (Porter, Bailey-Lee, et al., 2011). To this end, the course instructor sent proposed questions to a colleague who is an experienced CS1 PI instructor. This colleague read each pair of questions; when concerns were raised relating to the questions’ isomorphic nature or relative difficulty, changes were made by the course instructor. Then, to further eliminate within-pair difficulty variance, the course instructor generated a random number prior to each lecture that determined the order in which to present the isomorphic questions.

For each isomorphic pair, three student votes were taken; we use the notation introduced above and refer to these votes as Q1, Q1_{ad}, and Q2, with the first two votes taken on the first question and the third vote taken on the second. Note that the histogram of responses was never shown between Q1_{ad} and Q2.

Our two question modes are similar to the Peer and Combined modes used by Smith et al. (2011); we did not introduce an Instructor mode as our primary interest here is measuring the instructional value of the full Peer Instruction process including both peer discussion and instructor intervention. These two modes were administered as follows (see panel B in Figure 3.1):

- **Peer**: students were shown Q1, individually answered, engaged in peer discussion, and answered Q1_{ad}. Then, students were immediately presented with Q2 (the second question of the isomorphic pair) and voted individually. Note that between Q1_{ad} and Q2, there was no instructor intervention at all, and that the correct answer to the first question was not displayed.
• **Combined**: the treatment of Q1 and Q1\textsubscript{ad} in this mode is the same as in the Peer mode. Specifically, students individually answered Q1, engaged in peer discussion, and answered Q1\textsubscript{ad}. Then, the instructor displayed the histogram for Q1\textsubscript{ad} and proceeded to explain the question, its distractors, and its correct answer. The instructor was careful not to “give away” Q2, even though solid teaching would likely argue for this very comparison (we return to this in the discussion section). Following instructor intervention, Q2 was shown on which students voted individually.

When responding to Q1 and Q2, students worked by themselves, with no help from their peers in the discussion or the chosen response. Therefore, we use Q1 and Q2 as measures of what students individually know, independent of the influence of their group.
At the beginning of the semester, the course instructor generated a stream of random numbers that was to be used each lecture to determine the mode (Peer or Combined) for the isomorphic pair. The instructor checked the random number just prior to class (after slides had been finalized) to minimize mode bias.

### 3.4.3 Data Analysis

As noted by several studies (Simon et al., 2010; Zingaro, 2010; Crouch et al., 2007), PI questions that are too easy have limited learning potential for students. There is no standard cutoff for “too easy”, though Q1 correctness above 70% (Crouch et al., 2007) and above 80% (Smith et al., 2011) have been proposed. For purposes of comparison with the Smith et al. (2011) study, we chose to drop questions where the Q1 correctness was 80% or above. (The rationale here is that we seek to develop challenging questions for our students; when 80% of students answer correctly prior to peer discussion, we have produced a poor question.) Two Peer questions and four Combined questions were dropped according to this criterion, leaving 12 Peer and 12 Combined question pairs as our dataset.

For each remaining question pair, we required that each student vote all three times (Q1, Q1\textsubscript{ad}, and Q2); we removed partial data resulting from students answering only a subset of the votes. In addition, we completely removed a student’s data if they answered two or fewer Peer questions or two or fewer Combined questions. We did this in an effort to obtain more reliable data from students. For example, if a student answered only one Peer question, then both 0% and 100% are unlikely to be reasonable estimates of the students’ knowledge. Our final dataset contains data for 127 students.

Using Q1 percentage correct, we compared the difficulty of Peer and Combined questions. We found no significant difference in the difficulty of the questions used in these modes (paired \(t(126) = 1.7, p = .09\)), suggesting that our Peer-versus-Combined randomization was effective.

In order to assess gains between student ability groups, we require that our questions adequately separate weak from strong students. For each question, we expect strong students to answer correctly most of the time and weak students to answer incorrectly most of the time. We used a measure of question discrimination from Item Response Theory (IRT) that lies between 0 and 1, with higher values indicating greater levels of discrimination. Research argues that questions with discrimination indices above 0.3 adequately separate weak students from strong students (Matlock-Hetzel, 1997). The average discrimination for our Peer questions was 0.34; the average discrimination for our
Combined questions was 0.38. As we are above the cutoff of 0.3 in both cases, we are justified in using Q1 correctness as a measure of student ability.

We divided students into three groups based on Q1 correctness (Smith et al., 2011). Weak students are defined as those students answering up to 33% of Q1 correctly, average students answered between 33% and 66% correctly, and strong students answered more than 66% correctly. This resulted in 13 weak, 81 average, and 33 strong students. These group sizes are significantly unbalanced; we use them only for comparison with Smith et al. (2011) and for data visualization. Rather than use this arbitrary split in a statistical analysis, we instead use the percentage of Q1 answered correctly as each student’s baseline ability. We used the \texttt{nlme} R package (Pinheiro, Bates, DebRoy, kar, & R Core Team, 2013) to test a random-intercept linear mixed-effects model of the effect of Peer and Combined questions on Q2 performance, using Q1 performance as a covariate. A mixed-effects model was used because the data are hierarchical, not independent: each student answers both Peer and Combined questions, so questions are grouped within students. We set $p = .05$ for determining statistical significance.

### 3.5 Results

#### 3.5.1 Potential Learners in CS1

Prior work has examined the use of isomorphic questions in upper-division computing courses (Porter, Bailey-Lee, et al., 2011). As the present paper is the first such study of isomorphic questions in a CS1, we begin by extending the main findings of Porter, Bailey-Lee, et al. (2011) related to Peer questions.

Recall that the Potential Learner Group (PLG) are those students who answer Q1 incorrectly and $Q_1^{ad}$ correctly. We find a raw Q2 correctness of 73% for the potential learners which is comparable to the percentages reported by Porter et al. (76% and 62%). That is, over three quarters of potential learners learned from peer discussion.

Next, consider those students who answer both Q1 and $Q_1^{ad}$ correctly. This is our control group: we would expect these students to answer Q2 correctly as well. However, they may not, and it is therefore advocated by Porter, Bailey-Lee, et al. (2011) that we weight the performance of the PLG based on the performance of this control group. When we do this, we find that the PLG is 92% as likely as the control group to correctly answer Q2. This result is similar to those reported by Porter,
Bailey-Lee, et al. (2011) (85% and 89%). This is strong evidence that the potential learners are learning from the peer discussion; they are almost indistinguishable after the PI process from (i.e. 92% similar to) those students who understood the concept before any discussion.

In this section, we have demonstrated that CS1 students learn from peer discussion. The similarity of our measures to those of Porter, Bailey-Lee, et al. (2011) suggests that the conclusions on two upper-level computer science courses (Computer Architecture and Theory of Computation) may be generalizable to lower-division CS1 courses. We now move to an analysis of the PI process as a whole.

### 3.5.2 Comparison of Peer and Combined Modes

In this section, we investigate the first of our two hypotheses. Do students learn more from the combination of peer discussion and instructor intervention compared to only peer discussion?

Figure 3.2 shows the gain from Q1 to Q2 for each of the three student ability groups and all students. We intentionally provide raw values for these results to allow for more meaningful comparisons. For the error-bars: the Peer and Combined scores are repeated measures on the same students, so traditional between-group error bars are not appropriate. Therefore, we have used within-group error bars (Cousineau, 2005; Morey, 2008) to eliminate between-participant differences. Error bars roughly represent the 68% confidence interval; visually, significant differences between two conditions are represented by error bars that are separated by the average size of the two error bars in question (Cumming & Finch, 2005).

This figure first provides the Q1 data for Peer and Combined modes which demonstrates that Q1 scores were quite similar. This can be seen by comparing “Q1 P” and “Q1 C” in the figure for each pair (Weak, Average, Strong, All).

This figure also provides the improvement between Peer and Combined. The improvement between Q1 and Q2 in each mode represents how much was learned between the start of the initial question and when students are tested on Q2. For example, for Weak students in Peer, 29% (on average) initially respond correctly (Q1). After discussing the question with their peers, 54% respond correctly. This represents a 25% improvement in student correctness. For Weak students in Combined, 41% (64%-23%) of students improve from the combination of peer discussion and instructor intervention.

Evaluating each student ability group and the course overall, the figure shows the differences between gains for Peer and Combined. Weak and average students show the largest gains in Combined compared to Peer; this difference remains substantial with strong students but is less defined than...
Figure 3.2: Student performance on P (Peer) and C (Combined) questions. From left to right, the groups of bars represent weak, average, strong, and all students. Error bars show the within-group standard errors.

Table 3.1: Linear Mixed-Effect model of Q1, mode, and the interaction, on Q2. **p < 0.01

<table>
<thead>
<tr>
<th>Predictor</th>
<th>$\beta$</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.44**</td>
<td>(0.04)</td>
</tr>
<tr>
<td>Q1</td>
<td>0.46**</td>
<td>(0.07)</td>
</tr>
<tr>
<td>Admin_Mode</td>
<td>0.19**</td>
<td>(0.05)</td>
</tr>
<tr>
<td>Q1:Admin_Mode</td>
<td>-0.15</td>
<td>(0.09)</td>
</tr>
<tr>
<td>Log Likelihood</td>
<td>124.97</td>
<td></td>
</tr>
</tbody>
</table>

with the other two subgroups. The rightmost group of bars shows that, across the class, students learn more in Combined compared to Peer, suggesting that instructor intervention confers additional learning beyond peer discussion alone.

To evaluate the statistical relevance of the results from Figure 3.2, we performed multilevel regression analysis on the model we created to determine which inputs (question mode, student ability level, etc.) statistically impact Q2 performance. The results from this analysis appear in Table 3.1. The predictor variables are students’ Q1 performance, the administration mode (Peer or Combined), and the interaction of Q1 performance and mode; the outcome variable is students’ performance on Q2. Each predictor occupies one row in the table, which contains the predictor’s name, beta coefficient, statistical significance, and the standard error of the beta coefficient. Each
beta coefficient provides the increase in Q2 performance for a one-unit increase in the predictor, keeping all other predictors constant. Beginning with the Q1 predictor, we see that an increase of 1% in Q1 score yields a 0.46% increase in Q2 score, and that this relationship is statistically significant. This means that Q2 performance increases as Q1 performance increases. This is not surprising: we expect that students who are more likely to answer Q1 correctly will also be more likely to answer Q2 correctly.

Next, we move to the mode predictor: the table shows that there is a significant relationship between administration mode and Q2 performance; since the coefficient 0.19 is positive, we know that as mode “increases,” so does performance on Q2. In carrying out this analysis, our baseline administration mode was Peer, so we can interpret the mode “increasing” as changing the mode from Peer to Combined. That is, compared to Peer, Combined yields performance increases on Q2. This confirms that adding instructor intervention to peer discussion has a significant effect compared to peer discussion alone.

Finally, we consider the interaction of Q1 and mode. The effect is non-significant ($p = .09$), offering no evidence that the interaction significantly predicts Q2 performance, and that our interpretations of the main effects above are warranted. However, as the interaction does approach significance, it is worth considering what significance of the Q1-mode interaction would have meant. Were it significant, such an interaction might suggest differential effects of the administration mode based on Q1 performance. For example, Combined might help some subgroups of students more than others. Indeed, we see evidence of this tendency in Figure 3.2, where Combined was helpful for all students (main effect) but particularly helpful for weak and average students (evidence of an interactive effect). As such, the near significance suggests that possible interactions are worthy of future evaluation.

In summary, our statistical model confirms what we observe in Figure 3.2. The combination of peer discussion and instructor intervention is superior than peer discussion alone, and this finding holds independent of student Q1 scores. These results support our first hypothesis.

3.5.3 Comparison of Normalized Change between Modes

Normalized change (NC) remains a standard metric in the PI literature for evaluating student learning. As such, in Figure 3.3, we provide the NC from Q1 to Q2 for each of the student groups and the class overall. Note that we have not displayed error bars in this figure, and that we have not statistically analyzed these NC scores. The reason is that NC is a nonlinear computed quantity, not
3.5.4 Question Difficulty

So far, we have compared Peer and Combined for all students and for student ability groups, finding clear learning improvements for Combined. We now move to our second hypothesis to further analyze our findings by question difficulty.

To investigate this hypothesis, we change focus from examining groups of students to examining groups of questions. We define difficult questions as those where Q1 correctness is below 50%, and define easy questions as those where Q1 correctness is at least 50%.\(^1\) This yielded 7 easy and 5

---

\(^1\) Ideally, we would have preferred to use an easy, medium, difficult split where difficult questions are those whose Q1 correctness is below 35% (Crouch et al., 2007). However, we had only one Combined and two Peer questions that would have been classified as difficult according to this split; such small numbers preclude useful analysis.
difficult Peer questions, and 9 easy and 3 difficult Combined questions. There was no statistical difference between the Q1 scores on easy Peer and Combined questions ($t(13.24) = 0.33, p = .75$), or between the Q1 scores on difficult Peer and Combined questions ($t(4.64) = 0.17, p = .87$). That is, as for questions overall, our randomization was effective in balancing both easy and difficult Peer and Combined questions.

For each of the two question difficulties, we examined performance on Q2 for two subgroups of students: those who answered Q1 incorrectly and those who answered Q1 correctly (see Figure 3.4). Most striking in this data is the benefit of Combined when students initially answer Q1 incorrectly or when questions are difficult. On difficult questions, students who answer Q1 incorrectly score 54% on Q2 in Peer compared to 73% in Combined. This is a benefit of almost 20% for Combined. Smaller but substantial gains associated with Combined are found on difficult questions when students answer Q1 correctly; in this case, Combined outperforms Peer on Q2 by 11%. Indeed, the only case where Peer and Combined are comparable on Q2 is when questions are easy and students correctly answer Q1. (In this case, students are over 80% likely to answer Q2 correctly whether or not instructor
Table 3.2: Comparison of Q1, Q1_{ad} and Q2 for Peer and Combined.

<table>
<thead>
<tr>
<th>Mode</th>
<th>Difficulty</th>
<th>Q1 (%)</th>
<th>Q1_{ad} (%)</th>
<th>Q2 (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Difficult</td>
<td>Peer</td>
<td>38</td>
<td>60</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Combined</td>
<td>36</td>
<td>56</td>
</tr>
<tr>
<td>Easy</td>
<td>Peer</td>
<td>68</td>
<td>82</td>
<td>76</td>
</tr>
<tr>
<td></td>
<td>Combined</td>
<td>66</td>
<td>84</td>
<td>84</td>
</tr>
</tbody>
</table>

intervention is provided.) In all other cases, the instructor contributes substantially to learning as measured through Q2 performance.

Surprisingly, peer discussion was also highly useful on difficult questions (see Table 3.2). In Peer, we see a gain from Q1 to Q2 of 25% for difficult questions and 8% for easy questions. As expected from Figure 3.4, these values are substantially larger for Combined; nevertheless, progress is made by peer discussion even when questions are difficult. These results provide support for our second hypothesis: compared to easy questions, the instructor has a larger contribution to student learning gains on difficult questions.

3.6 Discussion

In this study, we have demonstrated the learning gains conferred to students through peer discussion alone (Peer) and through the combination of peer discussion and instructor intervention (Combined). For all students, Combined resulted in significantly larger learning gains than Peer, and a near-significant interaction suggests that Combined may have an even greater effect on weak and average students. In terms of NC, however, we found that strong students proportionally gained the most.

How can these results be remedied? It is our opinion that raw gains are a more useful measure in this context, since NC can be inflated by high Q1 scores. For example, if 10 out of 12 strong students answer Q1 correctly, and one of the two incorrect students then proceeds to answer Q2 correctly, the NC for the strong students will be 50%. This, however, does not communicate the effectiveness of the mode of administration, which in this example helped only one strong student. We see raw gains, at least when group comparisons are concerned, as a more pure measure of the effectiveness of each mode.

Continuing with an examination of question difficulty, we found that both administration modes led to learning gains on both easy and difficult questions. On difficult questions, however, we found
that Combined offered significant benefits over Peer, especially for those students who incorrectly answered Q1. When students struggle with Q1, instructor intervention is argued to be extremely valuable.

These findings, and the design of the study itself, lead to several areas of discussion: the quality of instructor explanation, the usefulness of difficult questions, and the use of Q1 to Q1<sub>ad</sub> as a proxy for “total learning” in a PI cycle.

### 3.6.1 Quality of Instructor Explanation

In Combined, the instructor discussed Q1 following the Q1<sub>ad</sub> vote. Naturally, the instructor was aware that Q2 would directly follow the explanation, and reflection suggests that this constrained his teaching. In standard lectures, we seek to engage students in the types of cognitions that are valued by CS professionals: comparing-and-contrasting, generating multiple solutions, studying multiple examples (Beatty et al., 2006). Frequently, the instructor desired to do this, but often the “what if” or “what would happen when” was exactly represented in the impending Q2. (In fact, several questions were removed from the dataset due to the instructor inadvertently giving away the answer to Q2.)

```python
def mystery(n):
    total = 0
    for i in range(n):
        for j in range(10000):
            for k in range(50):
                total += 1
    return total
```

This algorithm is:

- A. Linear (n)
- B. Quadratic (n<sup>2</sup>)
- C. Cubic (n<sup>3</sup>)
- D. Not one of these three

```python
def mystery(n):
    total = 0
    for i in range(n):
        for j in range(n):
            for k in range(50):
                total += 1
    return total
```

This algorithm is:

- A. Linear (n)
- B. Quadratic (n<sup>2</sup>)
- C. Cubic (n<sup>3</sup>)
- D. Not one of these three

---

Figure 3.5: Two isomorphic questions on complexity. Correct answers in bold.

For example, consider the isomorphic pair in Figure 3.5. These questions address algorithmic complexity in computer science. That is to say, given a problem size of n, what is the upper bound on
the runtime of the algorithm with respect to \( n \). In the question on the left, the nested for loops cause the runtime to be \( n \) multiplied by two constants (10000 and 50). As constants do not grow with \( n \), this runtime is Linear (\( n \)). In the question on the right, we see that the runtime is \( n \) multiplied by \( n \) multiplied by a constant (50). Again, the constant does not grow with \( n \) so the answer is Quadratic (\( n^2 \)).

Following peer discussion of the first question (the question on the left), it would be natural for the instructor to ask, “what would happen if I replaced one of the numeric constants with \( n \)?” However, this is exactly what happens in Q2 (the question on the right). Therefore, rather than explore a concept to its full generality, the instructor often stayed focused on Q1 itself, only expanding the focus following Q2. Once Q2 was complete, the instructor was free to compare Q1 and Q2, add further examples, and otherwise highlight similarities in the isomorphic questions that may not have been apparent. We therefore argue here that the instructor’s explanation following Q1 is not a reflection of the best-quality instruction that could be given, but was constrained by the experimental controls.

One possible change to our protocol would involve a third-party generating the Q2 questions, of which the instructor would remain unaware until after the explanation was complete. However, we argue that many questions would have been “spoiled” by virtue of the instructor essentially answering Q2 before it was asked.

What we are suggesting is that gains due to instructor intervention may in fact be larger than what we have shown here. In essence, we are measuring the effect of an instructor’s intervention when the instructor was constrained in his ability to engage students in discussion of the concept at large.

### 3.6.2 Use of Difficult Questions

The Q1 average over all questions in the study was 57%. The instructor sought to create challenging questions that were within the 35%-70% range advocated in the literature (Crouch et al., 2007). At the same time, the instructor hesitated to intentionally create questions where the Q1 performance would likely be very low, for fear that peer discussion would be fruitless. However, the few extremely difficult Q1 questions in the study do allow us to make tentative suggestions regarding these very questions. Rather than provide little benefit, peer discussion on difficult questions was extremely valuable, leading to an average 25% increase from Q1 to Q2. Adding instructor explanation increased the gains to 42%. For two reasons, therefore, we advocate for using conceptually-difficult PI questions.
First, such questions have the potential to lead to large learning gains, since room for improvement is large. (Easy questions may have large NC, but the number of students benefiting is small.) Second, both peer discussion and instructor explanation combine to produce these large gains, so the use of class time is certainly warranted. Faced with a challenging question, we anecdotally observe that students have longer, more in-depth discussions than those observed on easy questions. After grappling with difficult material, the students may be in a position to learn even more with instructor support. Easy questions, on the other hand, may not engender such discussion and do not maximize the benefits of knowledgeable instructor input.

3.6.3 NC as Proxy for Total Learning

Much current PI literature measures NC between the individual and group vote (what we have been referring to as Q1 and Q1\textsubscript{ad}), and uses this as a measure of student learning. However, those studies that additionally include Q2 sometimes find that Q2 correctness drops below that of Q1\textsubscript{ad} (Porter, Bailey-Lee, et al., 2011). The goal of those studies is to measure student learning from peer discussion, so their Q2 scores represent student learning prior to any instructor effects (much like the Peer mode in the present study). Indeed, we see this drop from Q1\textsubscript{ad} to Q2 in the Peer mode of the present study as well. Here, we find an NC between Q1 and Q1\textsubscript{ad} of 0.38, but a smaller NC of 0.33 between Q1 and Q2.

Does this mean that Q1\textsubscript{ad} is an overestimate of what students learn from PI? Perhaps it is an overestimate of what students learn from their peers (for example, students may choose answers based on peer influence rather than personal understanding). Data from the Combined mode of the present study, however, suggests that Q1\textsubscript{ad} is not an overestimate of total student learning. There, the NC between Q1 and Q1\textsubscript{ad} is 0.41, and the NC between Q1 and Q2 is 0.54. Contrary to what was found in Peer, NC between Q1 and Q2 is in fact larger than NC between Q1 and Q1\textsubscript{ad}.

This result is of important practical value for PI instructors. Low Q1\textsubscript{ad} results can be disheartening for an instructor assuming that Q1\textsubscript{ad} is highly indicative of student understanding at the end of the PI process, especially for conceptually difficult questions. However, our results from Combined show that Q1\textsubscript{ad} is not representative of student understanding after instructor explanation. Therefore, instructors should feel comfortable asking difficult questions despite low Q1\textsubscript{ad} results, knowing that their classwide discussion is likely to raise student understanding above the Q1\textsubscript{ad} threshold. That is, they should view Q1\textsubscript{ad} as an underestimate of student understanding.
3.7 Future Work

In this paper, we have conceptualized instructor intervention as the combination of providing students’ the correct answer along with conducting a classwide discussion of the ConcepTest. It may be worthwhile to disaggregate these effects to determine the relative importance of “providing an answer” (e.g. showing “A” as correct) and “explaining that answer”. It has been shown in biology (Smith et al., 2011) that most of the gain results from instructor explanation (not simply giving the correct answer). In addition, our analysis of difficult questions suggests that it is unlikely that the correct answer would have led to such large learning gains (if students do not understand the question, then they may not be able to abstract from a correct answer to a correct interpretation of why that answer is correct). That said, directly measuring the gains from answers alone would be productive follow-up to the present work.

More broadly, future work should further probe the utility of PI in engendering specific skills among CS students. While prior work has shown PI to be overall “better” than traditional lecture (Simon, Parris, & Spacco, 2013), and the present work shows that the full PI process is advantageous, a remaining goal is a more fine-grained understanding of skill development. Introductory CS students must learn and demonstrate a variety of skills, including code-tracing, code-reading, code-explaining, and code-writing (Venables, Tan, & Lister, 2009). The way in which these specific skills relate to conceptual understanding is an important open question and one that deserves future research attention. As the CS community moves forward in its understanding of PI, it is important to consider the specific ways in which PI helps our students develop conceptual knowledge as well as programming-related skills.

3.8 Conclusion

The current Peer Instruction literature in CS focuses largely on the gains associated with peer discussion. This is to be expected, since peer discussion is a crucial part of PI (Crouch et al., 2007). However, by privileging the discussion portion, other supporting roles in the PI process may not be as well-understood. Most critically, learning gains after peer discussion have not been measured in computing. As such, we have extended previous studies (Porter, Bailey-Lee, et al., 2011; Smith et al., 2011) in order to evaluate instructor intervention in PI. In this study, we demonstrate that instructor intervention is indeed crucial to the success of PI. Students evinced larger gains on isomorphic
questions when discussing with peers and receiving answers and explanations from an instructor (81% correct) than when only discussing with peers (69% correct). Moreover, the benefit of the instructor was heightened further when analyzing gains made on difficult questions. For these questions, initially incorrect students improved to 54% correct with peer discussion alone whereas they improved to 73% correct with both peer discussion and instructor intervention. Our findings also suggest that instructors should view Q1ad correctness as an underestimate of overall student learning, as that measure fails to capture the value of the full PI process. In summary, this work demonstrates that instructor intervention nicely complements peer discussion while acknowledging the crucial role to be played by the domain expert.
Chapter 4

A Link to the Exam

Note: This paper was presented at ITiCSE ’14 (Zingaro & Porter, 2014a). The second author is Leo Porter. Leo is the colleague mentioned in the paper who helped me validate the isomorphism of PI questions. This is the third and final paper that seeks to answer RQ1. Here, I investigate whether gains from the two in-class components of PI — peer discussion and classwide discussion — are visible on the final exam. Taken together, the three papers related to answer RQ1 suggest that reading quizzes are a key component of PI in CS courses, that both peer and classwide discussion combine to yield measurable learning, and that this learning is measurable using final exam scores.

In computer science, the active learning pedagogical practice of Peer Instruction (PI) has been shown to improve final exam performance, reduce student failure rates, and improve student retention. PI consists of two major parts: group discussion and follow-up instructor intervention. We expect that PI performance as a whole will correlate with final exam performance, but it is unclear whether or how each piece of PI is involved in these relationships. In this work, we use isomorphic questions to isolate the effects of peer discussion and instructor intervention, and examine scores on a final exam and its code-writing and code-tracing questions. We find that both pieces of PI correlate with the final exam as a whole, code-tracing question (similar to PI questions), and code-writing question (not similar to PI questions). This is further evidence that both PI components are important to the success of PI.
4.1 Introduction

Concerns that our computer science students are failing at alarming rates (Bennedsen & Caspersen, 2007) and that many of our students are not demonstrating acceptable learning outcomes (Lister, Simon, Thompson, Whalley, & Prasad, 2006) have spurred the CS research community to examine how we might respond as teachers. Prompted by research from other disciplines such as psychology and physics education, CS researchers have begun questioning the lecture as the pedagogical basis of CS courses. This is evident in the recent interest in inverted classrooms (Lockwood & Esselstein, 2013) and in-class collaboration (Kothiyal, Majumdar, Murthy, & Iyer, 2013), each of which decenters the lecture as the core element of teaching. One particularly promising alternative to lecture is Peer Instruction (PI) (Crouch et al., 2007; Simon et al., 2010). In a PI class, students participate in multiple iterations of individually answering a question (using a clicker), discussing that question in a group, answering the question again, and then participating in an instructor-led classwide discussion of the question.

Recent research has shown PI to offer a number of benefits over traditional lecture in CS courses. For example, PI can contribute to substantially lower failure rates (Porter, Lee, & Simon, 2013) and improved retention of majors (Porter & Simon, 2013). In addition, similar to other collaborative learning domains (Kothiyal et al., 2013), students report being particularly engaged in PI classes (Simon, Esper, Porter, & Cutts, 2013).

As the momentum in favour of new pedagogies builds, it is paramount that we measure student learning. A growing body of research therefore seeks to verify that students learn from the PI process. Some of this work uses isomorphic (same-concept) questions to measure performance before and after peer discussion (Porter, Bailey-Lee, et al., 2011). Such questions have been used to demonstrate that students learn from the peer-discussion part of the PI process. Other work shows that students in a PI class outperform students in a matched lecture class (Simon, Parris, & Spacco, 2013; Zingaro, 2014). As yet, no work has investigated links between PI performance and final exam performance. To be sure, we expect such links to exist — we know that PI is correlated with enhanced performance — but it is not clear whether the peer discussion, the instructor intervention, or both, are responsible for these expected links.

We examine student performance during the in-class PI process using isomorphic questions and use regression modeling to determine whether this performance predicts final-exam grades. We
anticipate there to be a great deal of noise between a student demonstrating learning a concept in a class during the early weeks of a term and the final exam months later, due to other in-class content, laboratory assignments, programming assignments, outside-class discussions, studying, programming practice, etc.

Despite this noise, we expect, and find, that students who come to class already prepared to answer questions correctly do better on a final exam consisting of code-tracing and code-writing questions. We also find, controlling for baseline performance, that student learning during the PI process, both from peers and from the instructor, relates to higher exam scores. This is the first work, to our knowledge, that directly connects learning during the in-class PI process with student performance at the end of the term. We discuss the elements of PI that we believe are particularly valuable and address potential threats to validity.

4.2 Background and Literature Review

4.2.1 Active Learning in Computer Science

Computing educators have long used active and collaborative learning techniques outside of lecture, most notably in the use of pair programming (McDowell et al., 2006). Recently, notions of an “inverted classroom” have brought active learning to the forefront of CS lectures as well. An inverted classroom involves asking students to prepare for lecture, perhaps by reading, watching a video, or exploring programming techniques (Lockwood & Esselstein, 2013). The justification is that our contact time with students is minimal, so it should be used in ways that maximize the utility of the instructor. Students know how to read but may not know how to integrate their reading with other disciplinary knowledge, hence the divide between reading before lecture and instructor-led learning during lecture.

Other disciplines, such as physics and psychology, have rich histories of pedagogical interventions that are presently being exercised in computing courses. For example, think-pair-share (TPS) is a popular cooperative learning technique that has students individually ponder, discuss with ad hoc groups, and then share with the entire class. Recent work has found that students are engaged in each of the three phases of the technique (Kothiyal et al., 2013) and that much of this engagement is combined with active learning such as discussing with peers and contributing to discussion. PI has much in common with TPS and other active learning pedagogies through its focus on group
discussion and conceptual understanding.

4.2.2 Peer Instruction

Peer Instruction (PI) focuses students on several conceptual multiple choice questions (“conceptests”) per class meeting. The PI protocol includes the instructor posing a multiple choice question, giving students a minute or two to respond with a clicker, and providing time for students to discuss in groups and submit a second response. Then, the instructor leads a wrap-up discussion of the particular concept, and moves on to a mini-lecture or the next PI conceptest.

Do students learn from PI in CS? It is interesting to trace the progression of research on this question. Initial studies demonstrated that students' more often answered correctly in the second vote (the group vote) compared to the first vote (the individual vote) (Simon et al., 2010; Zingaro, 2010). This might mean that students are learning from group discussion, but it could also mean that students are copying from neighbors. To distinguish these alternatives, other research has used isomorphic questions where students vote a third time on a new question very similar to the first. As students vote individually on this “isomorphic vote”, students are unable to passively copy from neighbors. These isomorphic studies use common terminology that we will also use below, so we introduce these terms:

- Q1: the individual vote on the first question.
- Q1_{ad}: the group vote on the first question. This occurs after students have discussed the question in groups.
- Q2: the individual vote on the second (isomorphic) question.

Porter, Bailey-Lee, et al. (2011) used isomorphic questions to study PI learning in Computer Architecture and Theory of Computation. They particularly focused on those students who incorrectly answered Q1 and correctly answered Q1_{ad}, since these are the students that may have learned (or copied) during the peer discussion. The authors found that 76% and 62% of these students correctly answered Q2, suggesting that real learning was happening during peer discussion.

Of course, the complete PI cycle — including Q1, peer discussion, and Q2 — occurs within a few minutes, and all measurements have been taken during this small time window. That is, learning might be happening, but is the learning enduring? A logical follow-up to the work of Porter,
Bailey-Lee, et al. (2011) is to examine the link between Q2 and the final exam when Q2 represents learning from various parts of the PI process. Students’ correctness on Q2 should correlate with correctness on the final exam to the extent that peer- and instructor-based learning are long-lasting. We follow this line of inquiry in the present paper.

4.3 Method

4.3.1 Study Context

Data for this study comes from a CS1 taught in Python at a large campus of a Canadian research university. The course uses Python 3 and studies traditional CS1 topics in the following order: introduction, functions, booleans, conditionals, while- and for-loops, lists (including nesting and aliasing), dictionaries, file I/O, testing and test coverage, introduction to object-oriented programming, and introduction to sorting and complexity. 131 students wrote the final exam.

The course took place over 12 weeks, with 3 50-minute lectures per week. Prior to each lecture, students were required to read 10-15 pages of the textbook and submit answers to three questions as part of a reading quiz. Reading quizzes are common and recommended for use in PI classes to bootstrap the discussion process in lecture (Crouch et al., 2007; Esper et al., 2012; Zingaro, Bailey-Lee, & Porter, 2013). The reading quizzes were marked based on completion (not correctness) and were worth 4% of students’ final grade; in-class clicker participation accounted for a further 5% of students’ grade. The course instructor was a senior education graduate student with significant PI and CS teaching experience, and had taught CS1 using PI several times. New PI materials were developed for this course offering and are freely available for anyone’s use (Peer Instruction for Computer Science, 2013).

In addition to facilitating PI questions, the instructor live-programmed during lecture to model code-writing. Students also pair-programmed in weekly labs (supported by TAs) and completed two larger programming assignments. This skill-based focus on code-writing was meant to complement the conceptual focus of PI (Zingaro, Petersen, Cherenkova, & Karpova, 2013).
4.3.2 Question Administration

The course instructor developed pairs of isomorphic questions, generally using one pair per lecture. (The other PI questions in each lecture were of the standard PI format, with no isomorphic question.) To verify the isomorphic nature of the questions, the instructor sent the proposed isomorphic questions to a colleague experienced in CS1, PI, and isomorphic question administration. Questions were modified as necessary so that both parties agreed that they were suitably similar and of comparable difficulty. Additional controls (below) were added to help ensure isomorphicity.

For each isomorphic pair, three student votes were taken; we use the notation introduced above and refer to these votes as Q1, Q1\textsubscript{ad}, and Q2, with the first two votes taken on the first question and the third vote taken on the second. Note that the histogram of responses was never shown between Q1 and Q1\textsubscript{ad}.

We are interested in the relationship between Q2 and the final exam score when Q2 represents learning from the full PI cycle and when Q2 represents only the effect of peer discussion. To this end, we ran half of the isomorphic questions as follows:

**Combined:** students individually answered Q1, engaged in peer discussion, and answered Q1\textsubscript{ad}. Then, the instructor displayed the histogram for Q1\textsubscript{ad} and proceeded to explain the question, its distractors, and its correct answer. Following instructor intervention, Q2 was shown on which students voted individually. This mode is identical to the Combined mode of Zingaro and Porter (2014b).

We ran the other half of the isomorphic questions so that Q2 measured only the peer-based learning, not the instructor-imparted learning. That is, these isomorphic questions were run as follows:

**Peer:** students were shown Q1, individually answered, engaged in peer discussion, and answered Q1\textsubscript{ad}. Then, students were immediately presented with Q2 and voted individually. Note that between Q1\textsubscript{ad} and Q2, there was no instructor intervention at all, and that the correct answer to the first question was not displayed. This mode is identical to a mode of Porter, Bailey-Lee, et al. (2011) and Zingaro and Porter (2014b).

See Figure 4.1 for these two administration modes and how they compare to classic PI.

We included the following experimental controls in order to validate isomorphic questions, avoid mode biases, and eliminate poor questions:

- A colleague was engaged in the validation of isomorphic questions (explained earlier).
Figure 4.1: The P (Peer) and C (Combined) administration modes used in this study.

- The presentation order (Q1 and Q2) within each question pair was randomized at the start of class.

- The choice of whether to run an isomorphic pair as Peer or Combined was randomized at the start of class.

- We removed questions that were too easy — operationalizing this as Q1 of 80% or above — as these questions leave little room for learning (Smith et al., 2011).

In the Combined mode, we expect that Q2 will be a close approximation to what students know of the concept being tested, since no further class time was spent explicitly discussing the question’s topic. Most certainly, CS1 topics recur repeatedly throughout the course, so students would have seen the questions from the PI conceptests throughout the semester. But the concept itself was never explicitly tested and discussed again in isolation. In the Peer mode, by contrast, we expect that Q2 performance will less faithfully represent what students ultimately come to know, since Q2 in this mode does not measure learning from instructor follow-up. When correlating PI performance to final exam performance, therefore, we expect that Peer correctness will relate to the final exam score and that Combined will have additional predictive power over and above Peer.

4.3.3 Final Exam Questions

The final exam for the CS1 studied here consisted largely of code-writing questions. Such exams are common in CS1, to the extent that research finds many exams consisting mostly or entirely of
code-writing questions (Petersen, Craig, & Zingaro, 2011). Researchers argue that these exams offer students minimal opportunities to demonstrate elements of knowledge as opposed to coherent and complete knowledge structures. For example, progress on the way to being able to write code might not be evident in students’ responses if we skip the preliminary steps and ask only for written code. To be sure, we are not advocating code-heavy exams: our final exam was produced in such a way for a different project involving between-lecture comparisons on a “traditional” CS1.

That said, the exam does contain some non-code-writing questions, and we used this limited variety to delineate performance on the final exam. We use three final exam measures in our correlations of PI scores and exam scores:

- The final exam score as a whole out of 100. The exam contained ten questions: one tracing, one sorting, one object-oriented programming, one describing code, and six mostly or completely code-writing. The mean was 38.69 with standard deviation 22.63.

- The score on a code-tracing question containing six questions each worth two marks (mean 4.23, standard deviation 2.95). The questions involved short code segments that added up integers in a loop or manipulated a Python list or dictionary; some of the questions involved nesting or aliasing. These questions are in many ways similar to PI questions: they are small, require students to trace/understand existing code (common in PI questions), and focus on single concepts. For example, one of the subparts asked students for the output of this code segment, focusing on aliasing:

```python
composers = ['Kondo', 'Tamura']
composers2 = composers
composers2[0] = 'Ito'
composers2[1] = composers[0]
print(composers)
```
The score on a code-writing question worth 10 marks (mean 3.07, standard deviation 3.03). Students were given a “compressed string” like:

\texttt{abc\#2,3\#5,2}

and had to decompress it to \texttt{abcabcab}. The \texttt{\#x,y} directives mean “go back \(x\) characters and copy \(y\) characters from there”. Note that this question is not similar to PI questions, as it is difficult to ask students to write code in a multiple choice format (Zingaro, Petersen, et al., 2013).

\section*{4.4 Results}

Multiple regressions were used to test relationships between final exam grades and PI performance. We performed three sets of regressions: one for the final exam at large, one for the code-tracing question, and one for the code-writing question. In each case, we included our predictors in three blocks:

**Block 1.** We first added two covariates: the number of Peer questions answered correctly and the number of Combined questions answered correctly. We expect that these two predictors will significantly predict exam score because they represent incoming ability prior to each lecture. The more questions that students answer correctly before any peer discussion or instructor intervention, the better they should do on the final exam.

**Block 2.** In this block, we add the number of Peer Q2 questions that each student answered correctly. If the \(R^2\) change from Block 1 to Block 2 is significant, it suggests that what students learn in the Peer mode (from Q1 to Q2) impacts performance on the final exam. Note that the Block-1 predictors are also included in this model and control for Q1 performance.

**Block 3.** In this block, we add the number of Combined Q2 questions that each student answered correctly. If the \(R^2\) for this model is significantly larger than the \(R^2\) for Block-2, it suggests that Combined questions correlate with exam performance over and above the relationship between Peer and exam performance.

Of primary interest is whether the change in \(R^2\) (i.e. the change in variance explained) is significant when moving from the Block-1 model to the Block-2 model, and from the Block-2 model to the
Table 4.2: Statistical models predicting overall final exam performance

<table>
<thead>
<tr>
<th></th>
<th>Block 1</th>
<th>Block 2</th>
<th>Block 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1 Peer</td>
<td>5.07 (2.01)$^*$</td>
<td>3.87 (1.92)$^*$</td>
<td>2.04 (1.99)</td>
</tr>
<tr>
<td>Q1 Combined</td>
<td>7.24 (2.01)$^*$</td>
<td>4.29 (2.04)$^*$</td>
<td>3.53 (2.01)</td>
</tr>
<tr>
<td>Q2 Peer</td>
<td>7.62 (1.95)$^*$</td>
<td>4.37 (2.23)</td>
<td></td>
</tr>
<tr>
<td>Q2 Combined</td>
<td></td>
<td>6.56 (2.38)$^*$</td>
<td></td>
</tr>
</tbody>
</table>

$^*$ $p < 0.05$

Block-3 model. We find that for each of the three regression analyses (entire final exam, code-tracing, and code-writing), each of the changes in $R^2$ is statistically significant.

These $R^2$ changes are in Table 4.1. The first row of the table gives the $R^2$ values for the final exam as a whole. We see that Q1 explains 21% of the variance, Q2 Peer increases this to 30%, and Q2 Combined increases this further to 34%. In terms of the significance of these $R^2$ changes, we find that the Block-2 model explains significantly more variance than the Block-1 model ($p = .0001$), and the Block-3 model explains significantly more variance than the Block-2 model ($p = .007$). This means that Peer questions correlate with final exam grade (controlling for baseline performance), and Combined questions also correlate with final exam grade (controlling for baseline performance and Peer performance).

The second row of the table gives the $R^2$ values for code-tracing, and they tell a similar story as the row above. The $R^2$ values increase from left to right, showing that adding Q2 Peer and then Q2 Combined both increase the proportion of explained variance. Specifically, the Block-2 model explains significantly more variance than the Block-1 model ($p = .041$), and the Block-3 model explains significantly more variance than the Block-2 model ($p = .023$).

Finally, the third row of the table gives the $R^2$ values for code-writing. Again, the Block-2 model explains significantly more variance than the Block-1 model ($p = .003$), and the Block-3 model explains marginally more variance than the Block-2 model ($p = .055$).

In Table 4.2, Table 4.3, and Table 4.4, we give the coefficients for each block for the overall, code-tracing, and code-writing regressions, respectively.

One might argue that Combined questions explain new variance (over and above Peer questions) simply because adding the Combined questions more accurately models how much students know. That is, perhaps we could have replaced Combined questions with more Peer questions to see the
same effect. While that may be true, we document an interesting finding here to the contrary. If we add the Combined questions to the regressions first, and then add the Peer questions second, the Peer questions do not significantly increase the model $R^2$, though the first is marginal ($p = .053$, $p = .60$, and $p = .113$, respectively). What this means is that, controlling for the Combined questions, the Peer questions do not further improve our final exam predictions. Combined questions seem to subsume the variance explained by the Peer questions, lending support to the hypothesis that measuring the full PI process in the Combined mode affords more predictive accuracy than does the Q2 measure in the Peer mode.

### 4.5 Ancillary Analyses

As Q1$_{ad}$ could represent learning from peer discussion, we decided to investigate Q1$_{ad}$ as a predictor of final exam performance. Q1$_{ad}$ necessarily conflates active learning with passive peer influence, so we suspected that it would have a weak or null relationship with the final exam (over and above the relationship between Q1 and the final exam). Using the Peer and Combined isomorphic questions as above, we performed three regressions: one for the final exam as a whole, one for the code-tracing
question, and one for the code-writing question. In each case, we included Q1 and Q1_{adj} as predictors. Q1, as expected, was a significant predictor in each case, but in no case was Q1_{adj} a significant predictor. That is, Q1_{adj} gives us no predictive power over and above Q1. We then repeated this analysis using the complete dataset of Q1 and Q1_{adj}; i.e. using all PI questions from the semester rather than just the Peer and Combined questions. For the final exam as a whole, code-tracing question, and code-writing question, we similarly see no positive and significant increment in $R^2$ when adding Q1_{adj} to the models.

4.6 Discussion

In this paper, we have shown that the learning conferred through the PI process is evidenced in higher scores on the final exam. Both key portions of what happens during class — peer discussion and instructor intervention — contribute to the relationship between PI correctness and final exam score.

We believe that these findings are both interesting and surprising. PI questions are not like code-writing questions. Indeed, in this PI offering, there were no PI questions that asked students to write code. (The reason is that students used clickers; while such devices allow students to submit numbers or select among multiple choice responses, they do not permit code-entry.) There were questions that asked students to fill-in the missing code and choose the correct code, but this is not the same as writing code from scratch (Denny, Luxton-Reilly, & Simon, 2008). PI questions are much more similar to code-tracing than they are to code-writing. Yet, we find relationships between PI correctness and both code-tracing and code-writing questions on a final exam. The latter relationship is a welcome surprise to us. We expected PI correctness to correlate with similarly small “chunks” on the final exam, but not to correlate much with code-writing questions. It has been argued that code-writing is at the top of a hierarchy of tasks that includes code-reading, code-tracing, and code-explaining (Venables et al., 2009). It is encouraging that PI performance not only correlates with questions similar to PI questions, but also to dissimilar code-writing questions.

One might wonder whether it is simply the number of PI questions being answered, or indeed the number of lectures attended, that is responsible for the observed significant relationships. This is certainly part of the story, because Q1 significantly predicts final exam score. However, recall that we have controlled for Q1 in our regressions, as we investigate the additional predictive power of Q2.
That is, for students who answered the same number of Q1 questions correctly, those who answer more Q2 questions correctly do better on the final exam.

Note that in this paper we are not comparing the effectiveness of PI to some other pedagogical method. In particular, we could imagine using a pre-test, post-test design in a lecture course, possibly finding that post-test scores explain final exam variance over and above pre-test scores. In addition, we cannot preclude the possibility that the students that learn from PI are simply those students that would have learned the material regardless of intervention. That is, the increase from Q1 to Q2 could simply represent the learning that would have occurred by these “potential learners” in some other learning setting (such as self-guided reading or lecture). That said, we believe that PI itself is at least partly responsible for the correlations between learning and exam performance. We found that gains from the Combined mode were related to final exam performance once we had controlled for Peer performance. We deem it unlikely that students in a lecture-only or self-guided configuration would exhibit both the peer-based and instructor-based learning gains that are associated with PI. That is, PI seems to aggregate learning from both peers and instructor, and we suggest that this pairing is powerful for both engendering and holding learning.

4.7 Conclusion

Prior Peer Instruction research has demonstrated that students score higher on the group vote than the individual vote, learn from group discussion as measured by isomorphic questions, and outperform lecture-taught peers. In the present work, we complement these findings with an analysis of the relationship between in-class clicker correctness and scores on the final exam. Using isomorphic questions and two modes of question administration, we find that learning from peers and learning from the instructor are each correlated with final exam scores. Furthermore, instructor-based learning is seen even when controlling for peer learning, suggesting additional benefits over and above the peer portion of PI. That is, while prior work has shown that PI students demonstrate learning when tested immediately after discussion, the present work shows that relationships between PI learning and performance are long-lasting. PI learning correlates with both code-tracing and code-writing on exams, even though the latter are far-removed from the types of questions we ask using PI. Future research should continue with a more fine-grained analysis of final exam performance, using exams that tap the wide array of question types that CS1 students should be able to answer. In addition,
we urge the community to investigate the mechanisms through which PI learning correlates with final exam performance in order to deepen our understanding of potential causal processes.
Chapter 5

Peer Instruction Contributes to Self-Efficacy

Note: This paper was presented at SIGCSE ’14 (Zingaro, 2014). The self-efficacy scale used in this study was not included in the paper for space reasons; it has been reproduced in the thesis appendix. This paper begins the study of RQ2, which is focused on whether PI leads to better learning outcomes than traditional courses. This paper is the core response to RQ2: it measures whether students’ exam grades and/or self-efficacy differ having taken a PI or lecture-based offering of CS1.

Recent work in computing suggests that Peer Instruction (PI) is a valuable interactive learning pedagogy: it lowers fail rates, increases retention, and is enjoyed by students and instructors alike. While these findings are promising, they are somewhat incidental if our goal is to understand whether PI is “better” than lecture in terms of student outcomes. Only one recent study in computing has made such a comparison, finding that PI students outperform traditionally-taught students on a CS0 final exam. That work was conducted in a CS0, where the same instructor taught both courses, and where the only outcome measure was final exam grade. Here, I offer a study that complements their work in two ways. First, I argue for and measure self-efficacy as a valued outcome, in addition to that of final exam grade. Second, I offer an inter-instructor CS1 study, whose biases differ from those of intra-instructor studies. I find evidence that PI significantly increases self-efficacy and suggestively increases exam scores compared to a traditional lecture-based CS1 class. I note validity concerns of such an in-situ study and offer a synthesis of this work with the extant PI literature.
5.1 Introduction

CS1 (Computer Science 1) is the first course taken by CS majors at the post-secondary degree level. As this course serves as entry into the CS curriculum, it has received significant attention by the CS education community. A recurring concern in the contemporary CS1 research is that students are not learning what we expect and that disproportionate numbers of students worldwide are failing CS1 (Bennedsen & Caspersen, 2007). Bimodal grade distributions (Robins, 2010) and assertions that programming skill cannot be taught (Dehnadi, Bornat, & Adams, 2009) have quickened the pace with which student’s grade-based outcomes have been measured and predicted. In general, this program of research seeks to predict students’ CS1 grade or final exam grade through characteristics such as prior programming experience (Hagan & Markham, 2000), math ability (Byrne & Lyons, 2001) and learning style (Hudak & Anderson, 1990), to name a few. The breadth of student predictors is both broad and creative, and serves as a testament to the effort put forth by the research community to thoroughly explore the hypothesized link between “what students have” and whether they succeed.

However, while the sheer number of studied predictors is comprehensive, two other areas of the research have gone largely unquestioned. First, the research narrowly defines success in CS1 as the student’s score on an exam or the course as a whole. It rarely acknowledges the importance of self-efficacy as a valued outcome in and of itself. Second, it rarely acknowledges the potential importance of pedagogy in the link between student-based predictors and outcomes. This is a particularly important gap given the recent interest in Peer Instruction (PI) and other active learning pedagogies for teaching CS courses.

This paper contributes in two ways to the gaps mentioned here. I offer a comparison of two sections of a CS1, one taught using PI and the other taught traditionally. In this way, I can compare outcomes from courses that are otherwise held constant (subject to validity concerns as described later). Second, I take both final exam grade and post-course self-efficacy as independent, valued outcomes of CS1. This work complements a similar study of a CS0 that compared PI to a traditional section (Simon, Parris, & Spacco, 2013). In that work, the PI section scored significantly higher on the final exam than the traditional section. In the present work, I find suggestive (but not statistically significant) evidence of the same finding. However, as I also measure self-efficacy, I document that students in the PI section show significantly higher self-efficacy than the control section. The benefit of a broad conception of “success” is evident in that the self-efficacy increases would have been
missed through a sole examination of final exam grade.

5.2 Background and Literature Review

5.2.1 Peer Instruction

Peer Instruction (PI) is a pedagogical technique developed in physics that has since been used with considerable success in CS. At the core of this pedagogy is the ConcepTest (Crouch et al., 2007): a multiple-choice question answered by students typically using clickers. Each ConcepTest sets off a well-defined pedagogical protocol: students first answer the question individually (solo vote), then discuss the same question for several minutes with their neighbors, and finally re-vote on the question in light of the group discussion (group vote). Following the group vote, the instructor facilitates a classwide discussion and explanation of the ConcepTest, before dynamically adjusting the class based on student performance. The instructor may lecture briefly before or after a ConcepTest, but the vast majority of lecture time engages students in discussions with peers. The role of the instructor is to help foster substantive discussions of core course concepts, and to focus students on understanding each distractor in addition to answering the question correctly (Simon et al., 2010; Zingaro, Bailey-Lee, & Porter, 2013). This downplaying of lecture means that instructors cannot “cover” as much material, so students are typically required to read one or two textbook sections before class and complete a small preparatory reading quiz (Crouch et al., 2007; Zingaro, Bailey-Lee, & Porter, 2013).

In a taxonomy of observable learning activities, Chi (2009) argues that “interactive” learning activities may be most powerful for engendering learning. Such activities are characterized by learners dialoguing and discussing concepts, where each peer makes substantive contributions to the discourse. Simply grouping students for PI does not guarantee that their discussions will be interactive as described here, but recent work suggests that students see PI as more interactive than traditional lecture (Simon, Esper, et al., 2013). We therefore expect PI students to exhibit strong gains, and for the most part this is borne out by the literature. For example, PI reduces failure rates (Porter, Lee, & Simon, 2013), contributes to increased retention (Porter & Simon, 2013), and yields exam-inferred learning gains in CS0 courses (Simon, Parris, & Spacco, 2013). Students learn from their discussions with peers (Porter, Bailey-Lee, et al., 2011) and students and teachers alike value the pedagogy for its effects on learning and argumentation (Porter, Bailey Lee, Simon, Cutts, & Zingaro, 2011). See
recent reviews (Zingaro et al., 2012; Zingaro, Bailey-Lee, & Porter, 2013) for further commentary on these and other benefits.

### 5.2.2 Self-Efficacy as Valued Outcome

The most observable measure of course success, and certainly one of the most important, is student grade. The vast majority of CS studies concerned with measuring outcomes have operationalized success as course grade or final exam grade. Further, these studies tend to study lecture-based offerings of CS1. The typical finding is that students with prior programming experience earn higher grades compared to inexperienced students. This finding is robust in the sense that even a small amount of prior programming experience is helpful (Hagan & Markham, 2000). Having taken a CS course (Evans & Simkin, 1989), studied a programming language (Hagan & Markham, 2000), or dabbled in programming as a hobby (Holden & Weeden, 2003) all positively predict performance in a CS1. To be clear, there are isolated studies where this relationship is deemed not to hold, but the relevant features of those examples have resisted capture or replication. For example, in an objects-first CS1, no prior experience gap was evident (Ventura, 2003). Yet, in other “objects-first” courses, the gap is back (Bennedsen & Caspersen, 2005). I argue that a well-defined pedagogical shift — from traditional to PI — may yield more reliable results than particulars of the content being taught.

Most relevant to this discussion, Simon, Parris, and Spacco (2013) have recently reported a comparison of a PI section of a course to a traditional section. To determine whether PI students demonstrated enhanced learning, the sections’ final exam grades were compared. They found that PI students scored almost 6% higher on the common final exam than traditional students.

While grade is an important outcome, it is not the only important outcome. Elsewhere, I have argued for the importance of measuring CS students’ interest and enjoyment of CS to complement grade-based outcomes (Zingaro, in submission). Students take CS courses for a variety of reasons including job prospects, challenging themselves with difficult material, and understanding societal trends and changes (Jenkins, 2001; Fisher, Margolis, & Miller, 1997). These are only peripherally-related to the grades students earn. There are other outcomes that may suggest that students are getting what they want from our courses, and we must measure these outcomes in order to broaden our understanding of student success.

One such outcome measure that may only partially overlap grades is self-efficacy. Self-efficacy is
the most-studied sociocognitive attribute of CS1 students, but it is typically studied as a predictor of (grade-based) success, not as a legitimate outcome in itself. It is defined as the conviction that one can successfully orchestrate behavior to accomplish something (Bandura, 1977). Self-efficacy expectations are differentiated from outcome expectations, the latter of which refer to the instrumental relationship between a behavior and an outcome. Bandura (1977) explains that efficacy judgments help determine whether coping behavior will be initiated, the amount of effort expended, and level of perseverance in the face of obstacles. It is for this reason that I take self-efficacy as an important measure of what students “get” from CS1. If students' self-efficacy is strengthened, then this is valuable whether or not it manifests immediately in heightened grades. For example, students with high self-efficacy following CS1 may be more likely to continue to a second CS course, where perhaps grade will catch-up to their belief in what they can do. Of course, self-efficacy expectations in and of themselves are not sufficient to produce successful behavior; but in the presence of sufficient skill and motivation, efficacy expectations are crucial in shaping consequent action. The most powerful mediator of self-efficacy is performance accomplishments, though vicarious experience, verbal persuasion, and associated physiological states also play a role.

The only widely-used self-efficacy scale for computer programming is that developed by Ramalingam and Wiedenbeck (1998). This scale contains 32 questions primed to object-oriented programming in C++. Students rate on a seven-point scale their confidence that they could carry out specific and relevant programming tasks. These authors administered the scale once prior to any instruction, and again after the final lecture. Factor analysis on the pre-self-efficacy administration suggests that this scale is composed of four factors: independence and persistence, complex programming tasks, self-regulation, and simple programming tasks. In the traditionally-taught CS1 courses, self-efficacy increases from pre-course to post-course (Ramalingam & Wiedenbeck, 1998; Ramalingam, LaBelle, & Wiedenbeck, 2004).

5.2.3 Hypotheses

PI courses give students considerable opportunities to experience small successes. Each class has students discuss and answer three or more important but small questions. If self-efficacy is tied largely to performance accomplishment, then it should be that PI increases self-efficacy beyond that of a traditional course offering. In addition, as PI has shown to improve grades over a traditional offering, I hypothesize that PI students' final exam scores will be higher than those of traditionally-taught
students. My first two hypotheses are therefore:

- PI students leave CS1 with higher self-efficacy than traditionally-taught students.
- PI students will earn higher grades on a final exam than their traditionally-taught counterparts.

In addition, I suggest that links between previous programming experience and gender on the one hand and outcomes on the other might be moderated by pedagogy.

As noted earlier, the literature shows strong links between prior programming experience and grades, but that work was conducted solely in lecture sections. To the extent that PI gives students of lesser experience opportunities to learn from experienced peers, I hypothesize that the gap generated by prior experience might be smaller in the PI section.

In an ethnographic study of gendered experiences in CS, Margolis and Fisher (2002) argue that contextualized learning is important for women’s success. They describe the computing-related narratives of many male college students as approximating an in-born, magnetic attraction to computers. These males are attracted to the physical computer for its own sake, in contrast to females who often understand computers as serving larger societal goals. Women’s stories run counter to the idea that CS majors “hack for hacking’s sake”. They are motivated by a desire to help others, an appreciation of the versatility of CS, and encouragement from family and friends. Large, impersonal and competitive lectures disadvantage women who are bolstered by social and academic support. Margolis and Fisher (2002) give three suggestions for creating what they call a course in “contextualized computer science”: situating technology in realistic settings, making connections to other disciplines, and using diverse problems and teaching methods. I contend that PI addresses, at least partially, the first and third of these guidelines, and therefore that the gender gap (if one exists in my setting) will be reduced in the PI section. My second set of hypotheses is therefore:

- PI will reduce the prior experience gap.
- PI will reduce the gender gap.

5.3 Method

I report on a CS1 taught in Fall 2012 at an undergraduate campus of a large Canadian research-intensive university. Two sections of the course, taught by different instructors, were offered: a PI
offering and a traditional offering. The course covers traditional CS1 topics in imperative programming, and also spends one week each on sorting, complexity, and object-oriented programming. The course took place over 12 weeks, with three 50-minute lectures and one lab session per week. The course has been taught in Python 2 since 2008 and the two Fall 2012 instructors worked together to revamp the course for Python 3.

In the PI section, the instructor began the course by introducing the rationale for using PI, covering some of the research findings and goals for the peer discussions. Prior to each lecture, students completed a reading quiz; the instructor read the responses to help shape the following lecture. The reading quizzes were marked based on completion (not correctness) and were worth 4% of students’ final grade; in-class clicker participation accounted for a further 5% of students’ grade. Each lecture was focused on three to four ConcepTests, with mini-lectures interspersed when planned by the instructor or required based on question performance. The course instructor was a senior education graduate student with significant PI and CS teaching experience, and had taught CS1 using PI several times.

The traditional section was taught by an instructor with significant CS and Python teaching experience. This section used the same labs, assignments, midterm, and final exam as the PI section. The two sections of the course were synchronized both in the topics to be covered each class and the examples used to teach those topics. The PI instructor used multiple choice questions and the PI process, while the traditional instructor introduced and worked examples through a lecture format. Importantly, both instructors were teaching in their preferred modes (PI or traditional) and developed and followed agreed-upon plans for each lecture.

As the traditional offering did not have reading quizzes or class participation marks, those students were required to submit three small exercises throughout the semester that were each worth 3%. The PI instructor used some of the very same exercise questions in the students’ reading quizzes so as to attempt to equalize exposure to these small programming questions.

Consenting students were asked to respond to two surveys: one at the start of term and one in the last two weeks of classes. On the first questionnaire, an initial measure of self-efficacy was collected, along with the number of prior CS courses taken. Prior programming experience was collected because of its reliable role in predicting student grade (e.g. (Hagan & Markham, 2000; Morrison & Newman, 2001)). On the end-of-term questionnaire, a post-self-efficacy measure was obtained. The self-efficacy measure was adapted from Ramalingam and Wiedenbeck (1998) by replacing “Java”
with “Python”. In this way, I hoped to measure the domain-specific self-efficacy of students’ Python programming ability, not a global personality trait (Ramalingam & Wiedenbeck, 1998). Student final exam grades were obtained as the second outcome measure.

5.4 Threats to Validity

As a quasi-experimental study, there are threats to the validity of the results here. I survey these threats, and in some cases describe how they were partially controlled:

**Different Instructors.** This is an inter-instructor study, where one instructor taught the PI section and the other taught the traditional section. This is to be contrasted with an intra-instructor study where the same instructor teaches both sections (Simon, Parris, & Spacco, 2013). These designs each have limitations that are only partially-overlapping. For example, an instructor may prefer one mode over the other, so intra-instructor studies force the instructor to teach in the mode they disprefer. In both designs, the quality of instruction in one section may be better than that in the other section. Inter-instructor studies, as the present one, successfully engage instructors in teaching in their preferred modes, but do so through a loss of control of general instructor effectiveness. For example, perhaps the PI instructor is globally “better” or overall works harder than the traditional instructor, or vice versa. In the present study, both instructors were teaching the Python 3 CS1 for the first time and developed their own lectures following the same teaching plan. Post-course student comments were positive in both cases, but of course such evaluations cannot disentangle instructor effects from pedagogy effects.

**Different Students.** The other obvious confound is that qualitatively different students enrolled in the two sections. However, the two sections were offered back-to-back, the two instructors were unknown to the students, and the students were unaware that the sections would use different pedagogies. Students could have switched sections early in the term, but anecdotally this was observed very infrequently. As an additional check, I found that pre-self-efficacy and prior experience measures did not statistically differ per section ($p > .05$ in both cases), giving credence to the suggestion that students were similar.
5.5 Results

Across both sections, 221 students wrote the final exam. Data for the present study includes 109 (49%) of these students: these are the students that responded to both questionnaires and took at least 40 seconds to respond to the post-self-efficacy questionnaire. (The self-efficacy scale contains 32 questions; 40 seconds was used to remove submissions where students were unlikely to have read many of the questions at all.) The Cronbach’s reliability of the self-efficacy questionnaire was 0.98.

Multiple regressions were used to test the predictors of post-self-efficacy and exam grade. Regressors were entered in two blocks. In the first block, main effects of gender, section, and prior experience were added. In the second block, interactions between section and gender and between section and prior experience were added. These models allow for the exploration of the four hypotheses of this study.

**Post-Self-Efficacy** The initial models failed to reject a test of non-constant variance; a square root transform on post-self-efficacy corrected the problem. Using a nested chi-square test, I compared the predictive power of the block-1 model (containing only the main effects) with the block-2 model (additionally containing the interactions). A non-significant result \( p = .65 \) shows that the block-2 model is no better than the block-1 model. That is, there are no significant interactions between pedagogy and gender, or between pedagogy and prior experience.

The block-1 (main effects) model of self-efficacy is given in the left-hand column of Table 5.1. I find a significant effect of section \( (p = .015) \), such that PI students’ self-efficacy \( (5.13) \) was higher than the self-efficacy of traditionally-taught students \( (4.65) \). A significant effect of gender \( (p = 0) \) shows that females have lower post-self-efficacy \( (4.12) \) than males \( (5.32) \). Finally, prior course-based programming experience was associated with higher post-self-efficacy \( (p = .003) \).

**Exam Grade** Using a nested chi-square test, I compared the predictive power of the block-1 model (containing only the main effects) with the block-2 model (additionally containing the interactions). A non-significant result \( p = .74 \) shows that the block-2 model is no better than the block-1 model. That is, there are no significant interactions between pedagogy and gender, or between pedagogy and prior experience. (This mirrors the finding for post-self-efficacy.)

The block-1 (main effects) model of final exam grade is given in the right-hand column of Table 5.1. Section is insignificant \( (p = .16) \), suggesting that PI and traditionally-taught students did not differ
on final exam grade. PI students were predicted to score 2.4% higher on the exam, but this difference was not statistically significant (c.f. Simon, Parris, and Spacco (2013)). Using the full student roster (not just the subset of students that responded to the questionnaires), PI students scored 4.4% higher, but a t-test remained non-significant ($p = .10$). Moving on, I find a significant effect of prior experience ($p = .0198$), such that students who took prior programming courses performed better on the final exam. A near-significant effect of gender ($p = .06$) suggests that females performed more poorly than males when controlling for prior experience. On the final exam, males scored an average of 61% and females scored on average 52%.

### 5.6 Discussion

I now return to the four hypotheses of the study.

As predicted, PI students’ post-self-efficacy was higher than that of students in the traditional section. This is perhaps due to the numerous opportunities for quick, accurate feedback in the PI class, where students could experience small successes before tackling larger labs and assignments (Ramalingam & Wiedenbeck, 1998).

However, the second hypothesis — that PI students would also score higher on the final exam — was not supported. Considered together, this suggests an apparent contradiction: if PI students believe they can organize activity toward success, why were they no more successful on the final assessment? I suspect that this may have to do with the type of final exam that is typically administered in CS1s. Prior research shows that CS1 exams typically involve several large, integrative code-writing questions (Petersen et al., 2011). Students are required to synthesize several concepts in order to arrive at a coherent program to solve a problem, and this tends to limit opportunities for
students to demonstrate what they know. (That is, there is a concern that these integrative questions measure all-or-nothing.) The instructors of the present CS1 intentionally designed such an exam to correspond to this CS1 practice. That is, the exam was mostly composed of integrative code-writing questions. This is to be contrasted with the earlier study by Simon, Parris, and Spacco (2013) where a PI section of a CS0 outperformed a traditional section. In that context, the final exam contained many multiple choice questions in addition to some short answer questions. What I am arguing is that the elevated self-efficacy of PI students did not present in terms of higher grades because the students may not yet have the required practice and experience to successfully orchestrate the mechanics required for integrative code-writing questions. Such self-efficacy, however, may prove useful for these students going forward.

The third and fourth hypotheses were not supported. I found overall evidence that males performed better than females, and that those with prior CS course experience performed better than those who hadn’t taken a CS course. However, the use of PI did not moderate these relationships. That is, gender- and experienced-based gaps existed in this CS1, and they were not lessened by a change in pedagogy.

Overall, PI suggestively (but non-significantly) led to increased final exam scores, and significantly contributed to post-self-efficacy. While these results are more equivocal than those previously reported for a CS0 (Simon, Parris, & Spacco, 2013), they do suggest the importance of multiple outcome measures to provide more nuanced understandings of pedagogical interventions. In future work, I seek to study a more balanced CS1 exam to determine (a) whether PI students outperform traditionally-taught students on some tasks and, if so, (b) precisely where the differences lie. That is, I do not argue here that PI did not help students conceptually, only that students may not have had the opportunity to demonstrate this understanding on the type of CS1 exam in common use. Of course, it may be that PI is more effective in a CS0 compared to a CS1, particularly when many questions on the final exam are multiple choice as in the Simon, Parris, and Spacco (2013) study. Questions of when and in what ways PI helps our students across different courses are exciting possibilities for future research.
5.7 Conclusion

When outcomes include more than final exam grade, our understanding of course success correspondingly broadens. In this CS1 study, I examine outcomes of final exam grade and self-efficacy for students taught using Peer Instruction (PI) and those taught in traditional lecture style. I find significant gains in self-efficacy for the PI students, but no significant evidence that final exam scores differed (though PI students did score 4.4% higher). I offer that the increased self-efficacy is a win in itself, and urge the community toward a multi-faceted conception of success in CS1. Students earn more than grades in our courses: they possibly become interested in CS, enjoy the lectures, gain self-efficacy, and so on. To truly understand the effect of a pedagogical shift requires an understanding of more than grade-based outcomes.
Chapter 6

A Study of Pedagogy and Achievement Goals

Note: This paper is single-authored and is currently in-submission to a peer-reviewed educational journal. It continues the response to RQ2. Rather than focus on exam grades for all students, it examines links between student goal strivings and exam grades, and how this link may be impacted by pedagogical approach. It also studies interest and enjoyment as outcome measures in addition to grades. The underlying theory for this paper is achievement goal theory as explicated in the educational psychological research literature. The core thesis is that student goal strivings are known to predict important outcome measures in psychology, so such goals may be powerful in CS as well. In addition, the ways that goal strivings relate to outcome measures may change depending on pedagogical approach. Of specific interest to the PI vs. lecture argument, PI may shift the goal context such that goal-outcome relationships are altered. For example, while students of particular goal strivings may be successful in lecture courses, they may not be the same students that are successful in PI courses. The inconclusive results from Section 5 further motivate the present study, as it is important to understand whether stronger results can be obtained by disaggregating effects across particular types of students.

Computer Science 1 (CS1), the first course taken by CS majors, has traditionally suffered from high failure rates. Efforts to understand this phenomenon have considered a wide range of predictors of CS success, such as prior programming experience, math ability, learning style, and gender, with
findings that are suggestive but still inconclusive. The current quasi-experimental study extends this research by exploring how the pedagogical style of the course (traditional or Peer Instruction), in combination with student achievement goals (mastery goals vs. performance goals) relates to i) exam grades, ii) interest in the subject matter, and iii) course enjoyment. The research revealed that students with performance goals scored significantly lower on final exams in both the lecture and Peer Instruction conditions. However, students with performance goals reported higher levels of subject matter interest when taught through Peer Instruction. Students with mastery goals, in both conditions, scored significantly higher on the final exam, had higher levels of interest, and reported higher levels of course enjoyment than their performance-oriented counterparts. The results suggest that Peer Instruction may improve the level of subject-matter interest for some students, but that the choice of pedagogy had no discernible affect on final exam grades.

6.1 Introduction

CS1 (Computer Science 1) is the first course taken by CS majors at the post-secondary degree level. As this course serves as entry into the CS curriculum, it has received significant attention by the CS education community. A recurring concern in contemporary CS1 research is that students are not learning what we expect and that disproportionate numbers of students worldwide are failing CS1 (Bennedsen & Caspersen, 2007). The discovery of bimodal grade distributions (Robins, 2010) and assertions that programming skill cannot be taught (Dehnadi et al., 2009) have quickened the pace with which students’ grade-based outcomes have been measured and predicted. In general, this program of research has sought to predict students’ CS1 grade or final exam grade based on an examination of student characteristics such as prior programming experience, math ability, learning style, and gender. While the breadth of studied predictors has been extensive, one area of research is notably lacking: the importance of pedagogical style in the link between student-based predictors and outcomes. This paper begins an investigation into this gap using achievement goals as a guiding framework. Achievement goals reflect student desires in attaining or demonstrating competence, and have been shown to predict grades, interest, and enjoyment in subject matter. Invoking a “matching hypothesis” (Harackiewicz & Elliot, 1993), I explore how the pedagogy used to teach CS1 may interact with student achievement goals to predict interest in subject matter and exam grades.

To study the impact of pedagogy and achievement goals on outcomes, I report on a comparison of
two sections of a CS1 course taught by two different instructors. One section used Peer Instruction (PI): an interactive pedagogy quickly gaining traction among CS educators. The other section was taught using a traditional lecture style. In all other respects, efforts were made to lockstep the delivery, timing, and assessment of both sections. Questionnaires were used to obtain achievement goal measures at the start of the semester and measures of interest and enjoyment at the end of the semester.

6.2 Background and Literature Review

In the following subsections, I briefly review the existing CS education literature linking student characteristics to grade-based outcomes, describe the PI pedagogy, and introduce relevant pieces of achievement goal research.

6.2.1 Predicting Student Performance in CS1

Why do some students succeed in CS1 and others fail? Some have argued that the tightly-interconnected nature of CS concepts sets up a context within which students are driven to extremely low or extremely high grades, with little variance between these extremes (Robins, 2010). Others argue that our assessments do not accurately capture gradations in student ability, because they repeatedly test the same skills (Scott, 2003; Petersen et al., 2011). Though such disciplinary concerns are undoubtedly important, I focus here on that large segment of the literature concerning relationships between student characteristics and later success.

Overall Ability

There is strong evidence that students of high overall ability perform well in CS1. For example, most SAT-like tests contain a mathematical-ability component, and it is commonly assumed that mathematical ability is a precursor to success in a CS program. Indeed, two studies (Byrne & Lyons, 2001; Bergin & Reilly, 2005b) find correlations between science/math components of the Irish leaving certificate (an entry college entrance exam) and performance in a CS1. Bennedsen and Caspersen (2005) found that math grades from high school were a significant indicator of success in an objects-first CS1, explaining approximately 15% of the variance. However, it seems we have no evidence that mathematical ability has predictive power over and beyond that of general intelligence.
Prior Programming Experience

Generally, students with prior programming experience outperform those without such experience. Prior course-related study (Wilson & Shrock, 2001; Morrison & Newman, 2001) or experience with a programming language (Evans & Simkin, 1989; Hagan & Markham, 2000) are typical metrics used to differentiate those with experience from those without. I note, however, that prior experience is sometimes found not to predict final marks. For example, in a study of 15 factors in an Irish object-oriented (OO) CS1, previous programming experience and previous non-programming computer experience were not significantly correlated with course performance (Bergin & Reilly, 2005b). The authors suggest that this might be country-specific, since students cannot study exam-level programming in Irish high schools. Ventura and Ramamurthy (2004) argue that prior programming experience is not required in their graphical, design-based, objects-first CS1. They asked students to specify their number of years of experience with various object-oriented, scripting, and imperative languages. When students were split on having no experience or more than no experience, no difference was found in performance on lab, exam, or overall course grade. This remains the case when students are split based on whether they know C++ and whether they know Java.

Self-Efficacy

Self-efficacy is the most-studied sociocognitive attribute of CS1 students. It is defined as the conviction that one can successfully orchestrate behavior so as to produce a desirable outcome (Bandura, 1977). Students’ self-efficacy increases from pre- to post-CS1 (Ramalingam & Wiedenbeck, 1998) and such increases are generally associated with performance gains. Research shows that self-efficacy at the end of a CS1 is positively correlated with course grade (Ramalingam et al., 2004) and that early measures of self-efficacy predict final exam performance (Bergin & Reilly, 2005a). Others have found self-efficacy to be a null predictor (Wilson & Shrock, 2001) or (in the case of non-majors) even a negative predictor (Wiedenbeck, 2005) of course grade.
Other Factors

Research finds a variety of other student characteristics to be relevant to CS1 success. Most such findings, similar to those reported above, are equivocal, yield contradictory research evidence, and seem to depend on the particulars of individual studies. For example, learning style based on Kolb’s learning style inventory was found to be a weak predictor in one study (Byrne & Lyons, 2001) but a robust predictor in another (Hudak & Anderson, 1990). Similarly, the importance of affective factors is unclear and depends on the particulars of how the factors are operationalized (Wilson & Shrock, 2001; Ventura, 2003; Rountree, Rountree, & Robins, 2002; Bergin & Reilly, 2005a; Bennedsen & Caspersen, 2008). Finally, the relationship between gender and course performance remains unclear: gender is often not a significant predictor (Byrne & Lyons, 2001; Wilson & Shrock, 2001; Bennedsen & Caspersen, 2005; Ventura, 2003; Quade, 2003; Rountree et al., 2002), though sometimes females perform more poorly than males (Bergin & Reilly, 2005b; Goold & Rimmer, 2000).

6.2.2 Achievement Goals and Student Interest

Research suggests that students enter CS for varying reasons, including interest in and enjoyment working with computers and a desire to obtain a well-paying computer-related job (Fisher et al., 1997). Such variety is also evident in the ways that students describe their motivation for taking a programming course. In one study (Jenkins, 2001), 20% of students were motivated by the opportunity to learn something new and 15% were motivated by specific course content. Examining students’ motivation for the degree program at large, only 3.5% of respondents cited achievement-based goals, far below the 40% of students motivated by career prospects and 36% of students motivated by learning itself. There is also evidence from psychology suggesting that course interest is a significant predictor of number of future courses taken in the discipline, but that there is no such link between grades and future courses (Harackiewicz, Barron, Tauer, & Elliot, 2002). Finally, recent evidence in CS suggests that using pedagogical features that motivate students increases retention (Porter & Simon, 2013).

The goals adopted by students can have dramatic effects on both course performance and intrinsic interest (Harackiewicz et al., 1998). Achievement goal theory studies those achievement-related goals that reflect the desire to develop, attain, or demonstrate competence (Harackiewicz et al., 1998). Research suggests two particularly important types of achievement goals: mastery goals and
Mastery goals are self-referential, focusing on the task at hand and one’s past performance. Students with such goals try to learn as much as possible, and are thought to seek challenge and persist through difficulty. There is much to suggest the adaptive nature of mastery goals. For example, students that adopt mastery goals are thought to involve themselves deeply in learning processes, use effective study strategies, and believe that ability increases through effort. Indeed, much early literature argued that mastery goals should be the sole type of goal adopted by students (Senko, Hulleman, & Harackiewicz, 2011).

The effects of performance goals, however, have been much contested. Students with performance goals are concerned with normative comparisons of their ability to that of others. This focus on “winning” suggests several negative effects, such as sacrificing deep learning and focusing only on what will be tested. However, recent evidence suggests that some performance goals can be adaptive when disaggregated from the larger performance-goal construct. Specifically, when students focus on outperforming others rather than avoiding doing worse than others, and on normative comparisons rather than demonstrating ability, performance goals can have positive effects (Senko et al., 2011).

A multiple goals perspective acknowledges the possibility that mastery and performance goals can each contribute to desirable interest and performance outcomes. For example, Harackiewicz, Barron, Tauer, Carter, and Elliot (2000) used a longitudinal design in order to investigate the relationship between achievement goals and interest and performance outcomes. Using questionnaires, they obtained students’ achievement goals (mastery and performance) two weeks into an introductory psychology course and final grade and interest at the end of that course. Students who adopted mastery goals were more interested in psychology and enjoyed lectures more than students who did not adopt mastery goals. On the other hand, students who adopted performance goals earned higher grades in the course and higher semester GPA than students who did not adopt performance goals. Each type of goal was therefore indicative of a desirable outcome. As Harackiewicz et al. (1998) explain, outperforming others is not inconsistent with attaining task mastery, and this is borne out by data showing that students can simultaneously adopt both goal strivings.

In addition to the independent effects of mastery and performance goals, there is also evidence that such goals may interact with task context to influence interest or performance (Harackiewicz & Elliot, 1993). The current focus of PI as a viable CS1 pedagogy suggests the importance of studying

---

1 The achievement goal literature uses the word adaptive to indicate positive or beneficial effects of goals.
goal effects in both traditional and PI contexts. After introducing PI, I return to the achievement goal literature to hypothesize on possible links between pedagogical style and course outcomes.

### 6.2.3 Peer Instruction

Peer Instruction (PI) is a pedagogical technique developed in physics that has since been used with considerable success in CS. At the core of this pedagogy is the ConcepTest (Crouch et al., 2007): a multiple-choice question answered by students typically using clickers. Each ConcepTest sets off a well-defined pedagogical protocol: students first answer the question individually (solo vote), then discuss the same question for several minutes with their neighbors, and finally re-vote on the question in light of the group discussion (group vote). Following the group vote, the instructor facilitates a classwide discussion and explanation of the ConcepTest. The instructor may lecture briefly before or after a ConcepTest, but the vast majority of class time engages students in discussions with peers. The role of the instructor is to help foster substantive discussions of core course concepts, and to focus students on understanding each distractor rather than simply answering the question correctly (Simon et al., 2010; Zingaro, Bailey-Lee, & Porter, 2013). This downplaying of lecture means that instructors cannot “cover” as much material, so students are typically required to read one or two textbook sections before class and complete a small preparatory quiz (Crouch et al., 2007). Evidence suggests that students take these quizzes seriously and provide quality responses (Zingaro, Bailey-Lee, & Porter, 2013).

Much recent literature suggests that PI is a highly effective pedagogy for teaching CS courses. For example, PI reduces failure rates (Porter, Lee, & Simon, 2013), contributes to increased retention (Porter & Simon, 2013), and leads to increased grades compared to traditional course offerings (Simon, Parris, & Spacco, 2013). In terms of the in-class components of PI, group discussion and instructor follow-up are each important for maximizing student learning (Zingaro & Porter, 2014b).

### 6.2.4 Context as Achievement Goal Moderator

Past studies of predictors of success in CS1 have largely focused on lecture-based courses. What we know about these predictors may change as the CS education community continues to invest in pedagogies such as PI.
Achievement goal theorists argue that multiple levels of goals can be simultaneously operative in a given situation, and that higher-level goals can be moderated by lower-level goals. Two levels are of particular importance: the purpose goal and the target goal. Purpose goals represent the overall reason for task engagement (Harackiewicz & Elliot, 1998); for example, students may adopt mastery goals or performance goals (or both) as explained earlier. Target goals, on the other hand, are more concrete: they are task-specific and outline the behaviours necessary to attain such goals. The matching effect of Harackiewicz and Elliot (1993) argues that outcomes will be optimized when purpose and target goals “match” in the sense that they are oriented to the same end state (Harackiewicz et al., 1998). Evidence for the matching effect comes from several studies (Sansone, Sachau, & Weir, 1989; Harackiewicz & Elliot, 1993, 1998). For example, Sansone et al. (1989) recruited participants to play a fantasy computer game under one of two purpose goals. Some participants were instructed to become engaged in and enjoy the game (a neutral purpose goal), while others were instructed to score as many points as possible (a performance purpose goal). Half of the participants in each purpose goal were provided scoring tips to help them quickly accumulate points, while the other half of participants were not provided with these tips. The tips are congruent with the performance purpose goal (scoring as many points as possible) but not with the neutral purpose goal (because the tips are irrelevant when scoring is not a concern). As predicted by the matching hypothesis, intrinsic motivation was optimized under the performance purpose goal when tips were provided, and optimized in the neutral purpose goal when tips were not provided.

In the present study, I operationalize the purpose goal as students’ goal strivings related to performance or mastery. Further, the pedagogy used to teach CS1 can instigate target goals, orienting students to those behaviours that are expected by their teachers. In a traditional, lecture-based CS1, students are implicitly asked to attend lectures, take notes on lectures, and remember and learn from what the instructor said. In PI classes, students are much more oriented toward peer discussion, negotiation of ideas, and shared responsibility for learning. These differing contexts highlight different targets for behaviour, and may therefore interact with students’ higher-level purpose goals to predict grades or interest.

6.2.5 Research Questions

- RQ1: What is the relationship between performance and mastery goal orientations and CS1 exam grade? Does this relationship change depending on pedagogical style? As described
above, some work in introductory psychology has shown that performance goals predict grades but mastery goals do not (Harackiewicz et al., 2000). However, in some courses requiring deep understanding of course material, it is mastery goals, not performance goals, that are related to achievement (Senko et al., 2011). CS1 exams require the simultaneous use of many concepts in order to answer integrative code-writing questions (Petersen et al., 2011). I therefore hypothesize that mastery goals, with the consequent focus on deep study strategies, will be related to final exam grade. Performance goals are hypothesized to have null or negative effects on grade, and this is likely to be independent of pedagogical style.

• **RQ2:** What is the relationship between performance and mastery goal orientations and interest/enjoyment in CS? Does this relationship change depending on pedagogical style?

Following from prior literature, I hypothesize that mastery goals will be related to interest and enjoyment in CS (Harackiewicz et al., 2000). I also hypothesize an interaction between pedagogy and performance goals on interest/enjoyment. Specifically, for those students motivated by normative comparisons to others, PI may provide an environment where they can assess the extent to which they are “winning” against their peers. They may ascertain, through discussions, that they know more than their peers or that they are well-positioned to outperform peers on course assessments. Of course, the intent is for PI discussions to be collaborative and mutually rewarding, so any benefit to performance-oriented students must be understood in relation to potential social consequences; I return to this in the discussion.

### 6.3 Method

#### 6.3.1 Study Context

I report on CS1 taught in Fall 2012 at a large Canadian research-intensive university. Two sections of the course, taught by different instructors, were offered: a PI section and a traditional lecture section. The course covers CS1 topics in imperative programming using Python, and also spends one week each on sorting, complexity, and object-oriented programming. The course took place over 12 weeks, with three 50-minute lectures and one lab session per week.

In the PI section, the instructor began the course by introducing the rationale for using PI, covering some of the research findings and goals for the peer discussions. Prior to each lecture,
students completed a reading quiz; the instructor read the responses to help shape the following
lecture. The reading quizzes were marked based on completion (not correctness) and were worth 4% of students’ final grade; in-class clicker participation accounted for a further 5% of students’ grade. Each lecture was focused on three to four ConcepTests, with mini-lectures interspersed when planned by the instructor or when student performance on a question was poor. The course instructor had significant experience teaching CS courses, and had taught CS1 using PI several times.

The traditional section used the same labs, assignments, midterm, and final exam as the PI section. The two sections of the course were synchronized both in the topic to be covered in each lecture and the examples used to teach that topic. The PI instructor used multiple choice questions and the PI process, while the traditional instructor introduced and solved examples through a lecture format. The traditional instructor, like the PI instructor, had been teaching CS courses for several years. Importantly, both instructors were teaching in their preferred modes (PI or traditional) and developed and followed agreed-upon plans for each lecture.

As the traditional section did not have reading quizzes or class participation marks, those students were required to submit three small exercises during the semester that were each worth 3%. The PI instructor used some of the very same exercise questions in the students’ reading quizzes so as to equalize exposure to these small programming questions.

6.3.2 Questionnaire Administration

At the start of the semester, students in both sections were invited to participate in the research study. Students were introduced briefly by both instructors to the field of Computer Science Education, and were told that the purpose of the study was to understand whether and why particular types of students succeed in CS courses. Students were also sent an email with the same information (including the link to the first survey), and the information was included in the lecture slides posted for the first lecture. A small course credit was given to students who completed both waves of the study, including those students who did not consent to the use of their responses in data analysis.

Goals Wave The first questionnaire was made available at the start of the semester and remained available for two weeks. The items (all seven-point, from “not at all” to “very”) were intended to measure students’ adoption of mastery and performance goals, and are based on questionnaire items from a prior study of college student goals (Harackiewicz et al., 2000). Both the mastery (α = .83)
and performance ($\alpha = .86$) scales proved quite reliable. This questionnaire also asked students for demographic information and to indicate their performance expectations for the course. The relevant questionnaire items appear in the appendix.

Note that the achievement goal literature distinguishes between two goal valences, approach and avoidance, so that in fact there are four combinations of goals: performance-approach, performance-avoidance, mastery-approach, and mastery-avoidance. I have focused on performance-approach (striving to outperform others) and mastery-approach (striving to develop intrapersonal confidence) and have not collected performance-avoidance (striving to avoid appearing incompetent) or mastery-avoidance (striving to avoid the loss of skill) measures. The avoidance strivings are uniformly associated with negative results for psychology students, including low interest and performance (Senko et al., 2011); I had no reason to believe that such goals would serve students well in CS.

**Interest Wave** In week 10, students were reminded in class that the study contained two questionnaires and that the second questionnaire would now be made available. Again, students were emailed a link to the questionnaire. This questionnaire contained 7-point items assessing students’ interest in CS and enjoyment of the specific course and are based on items used by Harackiewicz et al. (2000). The questionnaire was accessible after students had received most of their marked term work but before the study break leading to the final exams. The interest scale proved highly reliable ($\alpha = .94$), but the reliability of the enjoyment scale is low ($\alpha = .65$). In retrospect, the poor reliability of the enjoyment scale is perhaps unsurprising, because two of the statements invoke an affective dimension (“I like”), and one of those statements mixes enjoyment of the course with positive affect toward the professor. In addition, statement 4 on the interest scale (“I am enjoying this computer science class very much”) seems to fit more cleanly on the enjoyment scale. Removing that question from the interest scale has no appreciable effect on the findings. Moving that question from the interest scale to the enjoyment scale does increase the reliability of the enjoyment scale ($\alpha = .68$), but the interpretation of results is unchanged. I therefore proceed using the scales as originally developed by Harackiewicz et al. (2000).

**Grades** As a measure of course performance, I use students’ final exam grade. This is preferred to course grade as a whole because students worked with partners on labs and assignments, so those measures necessarily conflate ability with the ability of their partner and effort in general. Final
exams are far from perfect measures of ability (Petersen et al., 2011; Senko et al., 2011), but do match the usage in the vast majority of those CS1 studies concerned with student performance.

6.3.3 Data Analysis

Table 6.1 contains the means and standard deviations for the continuous variables; Table 6.2 contains the zero-order correlations below the diagonal and associated p-values above the diagonal. Across both sections, 221 students wrote the final exam, and 129 (58%) provided all of the required data for the study. There was no significant difference between the two CS1 sections in performance goals, mastery goals, or performance expectations obtained using the first questionnaire (all $p > .05$).

Multiple regressions were used to test the effects of goals on final exam grade, interest in CS, and enjoyment in CS1. Dichotomous variables (gender and section) were added to models using $(-1, 1)$ contrasts. Initial models were constructed to include section (PI or traditional), gender (male or female), performance goals, mastery goals, and all two- and three-way interactions between these variables. Continuous variables were centered but not standardized. Backward stepwise elimination using the AIC criterion was then used to remove non-significant predictors. One observation from each section was deleted because it was a strong outlier with large Cook’s distance, and substantially altered the interpretation of the model coefficients. In addition, the data from ten students who spent less than 30 seconds on one or both questionnaires were removed as it is unlikely students could read and respond accurately in that time. (Re-running the models from this section with the dubious data included does not significantly change any of the reported findings.) None of the three models violated assumptions of constant variance or autocorrelation (Fox & Weisberg, 2011). The residuals of the enjoyment model were non-normal according to a Shapiro-Wilk normality test. While the non-normality was correctable using a power transformation and did slightly improve the model fit, it did not change the significant coefficients. Therefore, to facilitate comparisons with other models,

<table>
<thead>
<tr>
<th>variable</th>
<th>range</th>
<th>mean</th>
<th>sd</th>
</tr>
</thead>
<tbody>
<tr>
<td>mastery</td>
<td>3.2-7</td>
<td>5.9</td>
<td>0.9</td>
</tr>
<tr>
<td>performance</td>
<td>1-7</td>
<td>4.8</td>
<td>1.3</td>
</tr>
<tr>
<td>interest</td>
<td>1.2-7</td>
<td>5.2</td>
<td>1.4</td>
</tr>
<tr>
<td>enjoyment</td>
<td>2.3-7</td>
<td>5.7</td>
<td>1.1</td>
</tr>
<tr>
<td>exam</td>
<td>6.5-85.5</td>
<td>44.2</td>
<td>19.37</td>
</tr>
</tbody>
</table>

Table 6.1: Range, mean, and standard deviation for continuous variables.
Table 6.2: Correlations (below diagonal) and p-values (above diagonal) for continuous variables.

<table>
<thead>
<tr>
<th></th>
<th>mastery</th>
<th>performance</th>
<th>interest</th>
<th>enjoyment</th>
<th>exam</th>
</tr>
</thead>
<tbody>
<tr>
<td>mastery</td>
<td>-</td>
<td>0.14</td>
<td>0.00</td>
<td>0.02</td>
<td>0.04</td>
</tr>
<tr>
<td>performance</td>
<td>0.13</td>
<td>-</td>
<td>0.70</td>
<td>0.18</td>
<td>0.00</td>
</tr>
<tr>
<td>interest</td>
<td>0.53</td>
<td>-0.04</td>
<td>-</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>enjoyment</td>
<td>0.21</td>
<td>-0.12</td>
<td>0.42</td>
<td>-</td>
<td>0.17</td>
</tr>
<tr>
<td>exam</td>
<td>0.19</td>
<td>-0.30</td>
<td>0.34</td>
<td>0.13</td>
<td>-</td>
</tr>
</tbody>
</table>

I use the untransformed enjoyment scores. The final models contained data for 63 students in the PI section and 56 students in the traditional section.

6.4 Results

Table 6.3 contains the coefficients, standard errors, and significance symbols for each of the three regression models.

6.4.1 Final Exam Grade

The overall model was significant \( f(7, 111) = 2.97, p = .007 \). Mastery goals were significantly related to final exam score \( t = 2.20, p = .03 \), such that students who adopted mastery goals scored higher on the final exam than those students who did not adopt mastery goals. On the other hand, performance goals \( t = -3.73, p = .0003 \) were associated with poorer performance on the final exam. No other main effects and no interactions with pedagogical style were significant. Across pedagogy and gender, mastery goals were adaptive and performance goals were maladaptive in terms of final exam grade.

6.4.2 Interest

The overall model was significant \( f(10, 108) = 8.33, p = 0 \). Mastery goals were positively and significantly related to interest in CS \( t = 6.24, p = 0 \). Performance goals were negatively related to interest, though unlike for final exam grade, the relationship with interest is non-significant \( p = .12 \). There was a main effect of gender \( t = -3.51, p = .0006 \), such that females (4.48) were less interested in CS after the course than males (5.61).

However, these main effects must be understood with respect to the significant interactions. First, there was a significant interaction between section and performance goals \( t = 2.27, p = .025 \). To
break down this interaction, interest was predicted by section for performance goals at the mean, one standard deviation below the mean, and one standard deviation above the mean; other variables were fixed at their means or baseline values. Table 6.4 displays the pattern of means. For the PI section, interest increases as performance goals increase; for the traditional section, however, interest decreases as performance goals increase. That is, performance goals are positively related to interest in the PI section but negatively related to interest in the traditional section. To determine whether these simple slopes of performance on interest were significant by section, I carried out separate regressions for each section (Aiken & West, 1991). In both cases, the overall models were significant (PI: \( f(5, 57) = 9.43, p = 0 \); traditional: \( f(5, 50) = 6.39, p = .0001 \)). As expected from Table 6.4, there was a significant negative effect in the traditional section of performance goals on interest \( (t = 2.47, p = .017) \); however, the association between performance goals and interest in the PI section was not significant \( (p = .63) \).

<table>
<thead>
<tr>
<th></th>
<th>Exam Grade</th>
<th>Interest</th>
<th>Enjoyment</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>58.97***</td>
<td>5.10***</td>
<td>5.78***</td>
</tr>
<tr>
<td>section</td>
<td>0.13</td>
<td>0.05</td>
<td>0.28**</td>
</tr>
<tr>
<td>performance</td>
<td>-4.86***</td>
<td>-0.12</td>
<td>-0.14</td>
</tr>
<tr>
<td>mastery</td>
<td>4.34*</td>
<td>0.75***</td>
<td>0.25*</td>
</tr>
<tr>
<td>gender</td>
<td>-1.51</td>
<td>-0.38***</td>
<td>0.10</td>
</tr>
<tr>
<td>section:performance</td>
<td>1.14</td>
<td>0.18*</td>
<td>0.13</td>
</tr>
<tr>
<td>section:gender</td>
<td>-0.53</td>
<td>-0.09</td>
<td>-0.15</td>
</tr>
<tr>
<td>mastery:gender</td>
<td>0.07</td>
<td>0.07</td>
<td>0.27*</td>
</tr>
<tr>
<td>section:mastery</td>
<td>0.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>mastery:performance</td>
<td>0.13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>section:mastery:gender</td>
<td>-0.23*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

R²                     | 0.16       | 0.44     | 0.23      |

*** \( p < 0.001 \), ** \( p < 0.01 \), * \( p < 0.05 \), . \( p < 0.1 \)

Table 6.3: Multiple regressions for exam grade, interest, and enjoyment. \( a : b \) is used to denote the interaction between \( a \) and \( b \).
Chapter 6. A Study of Pedagogy and Achievement Goals

88

Table 6.4: Effect on interest for low, medium, and high levels of performance goals.

<table>
<thead>
<tr>
<th></th>
<th>performance goals</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Low</td>
</tr>
<tr>
<td>PI</td>
<td>5.55</td>
</tr>
<tr>
<td>Traditional</td>
<td>5.73</td>
</tr>
</tbody>
</table>

Returning to the main interest model, there was also a significant three-way interaction between section, mastery goals, and gender ($t = -2.00, p = .048$). This suggests that the relationship between mastery goals and gender in the PI section differs from that in the traditional section. However, predicting interest at the mean of mastery goals and one standard deviation above and below the mean, the overall relationship between mastery goals and interest holds across genders and section. For the PI section, one standard deviation increase in mastery goals led to 0.89 increase in interest for males and 0.59 increase in interest for females. In the traditional section, one standard deviation increase in mastery goals led to 0.34 increase in interest for males and 0.88 increase for females. That is, mastery goals in the PI section are more potent for males, and mastery goals in the traditional section are more potent for females, but the overall positive relationship between mastery goals and interest is not changed by this analysis.

6.4.3 Enjoyment

Considering the low reliability of the enjoyment scale reported earlier, analysis in this section should be assumed less robust than those from the previous sections. The overall model was significant ($f(7, 111) = 4.84, p = 0$). For the first time, there is a main effect of section ($t = 2.90, p = .005$). Those in the PI section (5.41) enjoyed the CS1 more than those in the traditional section (5.00). Two other main effects were significant: performance goals were negatively related to enjoyment ($t = -1.89, p = .06 \text{ NS}$) and mastery goals were positively related to enjoyment ($t = 2.28, p = .02$). The interaction of section and performance goals approached significance ($p = .07$); if significant, its interpretation would mirror that given for the interest regression above (i.e. performance goals are unrelated to enjoyment in the PI section, and negatively related to enjoyment in the traditional section).

The only outright significant interaction is between gender and mastery goals ($t = 2.54, p = .01$), and the interpretation of this interaction qualifies the main effect of mastery goals. Predictions using mastery goals at the mean and one standard deviation below and above the mean show that the
relationship between mastery goals and enjoyment is weak for males but substantial for females. For males, a one standard deviation increase in mastery goals led to a 0.01 increase in enjoyment, whereas for females such a change in mastery goals led to a 0.45 increase in enjoyment. That is, mastery goals are related to enjoyment for females but not males.

6.4.4 Perceived Competence

So far, and contrary to a multiple goals perspective, performance goals seem rather maladaptive in the CS context. In the traditional section, performance goals are associated with poor final exam grade and lower levels of interest and enjoyment. In the PI section, performance goals are associated with poor final exam grade and are not correlated with interest and enjoyment. In this sense, it seems that PI may protect against some negative influences of performance goals, but that performance goals are undesirable overall.

However, achievement goal theorists (Harackiewicz et al., 2000; Senko et al., 2011) caution that effects of performance goals should be investigated in the presence of perceived competence. Specifically, performance goals may be neutral or positive for students with high perceived competence but maladaptive for students with low perceived competence.

To determine whether the non-negative effects of performance goals in the PI section are retained when considering competence, I added a measure of perceived competence to an interest model for the PI section. Specifically, I regressed interest on mastery goals, performance goals, expected grade, and the interactions between expected grade and mastery/performance goals. On the first questionnaire, students were asked to state the grade that they expect to receive (A, B, C, D, or unsure). Due to the low numbers of responses for C (4), D (2), and unsure (7), I collapsed expected grade to “A” and “not A”. As those of highest competence may be immune to negative effects of performance goals (Harackiewicz et al., 2000), collapsing across levels, though artificial, does capture the two levels (“highest” and “not highest”) of importance when examining competence moderators of performance goals.

The overall model was significant ($f(5, 57) = 6.91, p = 0$), but the interaction between performance goals and expected grade was not significant ($p = .84$). Performance goals remained non-significant ($p = .74$) and mastery goals remained significant ($t = 5.43, p = 0$). Therefore, the neutral effect of performance goals in the PI section does not reflect a spurious link between perceived competence and interest (Senko et al., 2011).
6.5 Discussion

I now return to the research questions guiding this study. What types of goals are optimal in CS contexts? Do goal relationships depend on pedagogical style?

Goals and Grades (RQ1) The model for final exam grade is the most parsimonious of the models in this study. There were no gender or section effects. Students endorsing mastery goals scored significantly higher on the exam; students endorsing performance goals scored significantly lower. This result is discordant with work in psychology (Harackiewicz et al., 2000), where it is frequently found that performance goals are a powerful positive predictor of final exam grade.

On the one hand, the negative effects of performance goals are surprising. Some have argued that CS courses are rather defensive, rife with competitiveness, “show-offs”, and demonstrations of superiority (Garvin-Doxas & Barker, 2004; Margolis & Fisher, 2002). Such descriptions seem to accord more with a performance goal than a mastery goal, where students are largely concerned with outperforming peers. One might expect that students who adopt performance goals would be well-positioned to succeed in a discipline that is allegedly so competitive, but the findings suggest otherwise.

On the other hand, a close study of typical CS1 exams clearly shows the unique ways with which we assess students. CS1 exams, including the one used in this study, are largely focused on code-writing (Petersen et al., 2011). The majority of the questions are large and integrative, requiring students to synthesize many concepts to correctly write a solution to a programming problem. For example, to ask students a question requiring a loop also necessarily involves asking about variables, expressions and assignment, in addition to more complex structures such as selection. As argued in the literature (Robins, 2010; Petersen et al., 2011), one mis-step along the way can prove fatal to a student’s ability to write correct programs. Ignoring the obvious disciplinary threats to the diagnostic, formative, and summative utility of such final exams, why might such assessments disadvantage performance-goal individuals? One possibility is that thoughts of outperforming peers focus some students on local concerns (“I want to do better than 80% of students on the next test”) rather than global concerns (“I want to understand how the loop stuff from last week relates to the conditional stuff from last month”). Such locality, of course, is not conducive to performance on final exams whose questions tend toward comprehensiveness.

In fact, this line of thinking may also highlight why PI did not improve matters (recall that
performance goals were equally maladaptive for grades in the PI section). The PI questions used were focused on single concepts — indeed, the focus on core concepts is critical to PI's effectiveness (Crouch et al., 2007). But if performance-oriented students do not naturally coalesce ideas over the long-term, the use of PI may not help them do so.

Goals and Interest/Enjoyment (RQ2) As for final exam grade, mastery goals significantly and positively predicted interest in CS. However, the effects of performance goals differed by pedagogical style. Performance goals were negatively related to interest in the traditional section but had no relationship to interest in the PI section.

Why should PI attenuate the performance goal-interest relationship in this way? If the matching effect is operative in this context (Harackiewicz & Elliot, 1993), it would suggest that the goals afforded by PI may be more congruent with performance goals than are those goals associated with traditional lecture. Indeed, PI gives students many opportunities to discuss with peers and assess their relative performance. For example, if a performance-oriented student answers a question correctly and many of their peers do not, they may understand this as a success and become increasingly interested in the course. The double use of discussions as both learning and normative contexts is not inherently undesirable, particularly if normative comparisons spur students to become more interested in a subject or work harder to earn better grades. However, there is evidence that performance goals may undermine collaborative learning (Senko et al., 2011). Focused on correct answers and good ideas, performance-oriented students may be less tolerant to “bad ideas”, may choose their favourite partners, or offer guarded opinions (Senko et al., 2011). So, while performance-driven students may become interested in CS when taught using PI (or at least do not become uninterested!), the important open question is: what is the effect on their PI partners? Further research is required for understanding the possible risks of performance goals on broader PI culture.

The findings from the enjoyment model are similar to those just described for the interest model. As stated earlier, the low reliability of the enjoyment scale limits confidence in those results. However, of particular interest is the main effect of section on enjoyment, showing that PI students enjoyed CS1 more than their traditionally-taught peers. From the interest model, however, we see that PI students did not develop stronger general interest in CS than their peers. Though disciplinary interest, not “catch” interest, is typically implicated in students’ academic decisions (Harackiewicz et al., 2000), further work is necessary to understand the effects of increased enjoyment on students’
propensity to continue studying CS.

### 6.6 Limitations of the Research

Throughout, I have claimed comparisons between a PI section and a traditional section of CS1. While the sections were taught by different instructors, I have reason to believe that the pedagogy, rather than any instructor effect, accounts for this study’s findings. The instructor’s followed the same plan for each lecture, using the same live-programming approach, examples, and topic sequence. The only difference in each class was that the students in the PI section engaged in collaborative discussion using clickers, whereas the instructor of the traditional section worked similar examples for the students. In addition, student course evaluations for both instructors were equally positive, and these instructors were teaching in their preferred pedagogies.

I also believe that the students were similar across the two sections. Both sections were taught back-to-back (minimizing a time-selection bias); the students were unfamiliar with the instructors and their teaching methods (reducing instructor-selection bias); and there was no difference on mean performance goals, mastery goals, or performance expectations at the outset of the course.

I welcome replications of this study that use intra- and inter-instructor designs. Intra-instructor studies have their own confounds (what if the instructor prefers PI but is forced to teach traditionally?) and so can provide non-overlapping confirmation or refutation of the findings presented here.

Finally, the present study is correlational; it is not an experiment that allows claims of causation. It could be, for example, that some unmeasured variable both produces mastery goals in students and high grades in those same students. While this is possible, the study is based on achievement goal research, offering a theoretical basis for the findings presented here.

### 6.7 Implications for Research and Practice

Across the sciences, we are moving from lecture-based courses to courses that actively involve students in their learning (Knight & Wood, 2005; Kay & LeSage, 2009; Crouch et al., 2007). This practical pedagogical shift should be met by a corresponding shift in research that re-examines what we have established in lecture-based offerings. I contend that pedagogy may interact with student-based factors to influence attitudes, grades, learning, interest, self-efficacy, and other constructs known to
be important for students’ academic success. Indeed, in the present study I find that pedagogical style moderates the relationship between student goal strivings and interest in a first CS course. It is necessary to re-examine what we know about goal strivings by including pedagogy in our models.

While it is unlikely that teachers can guide students toward more adaptive achievement goals in general, it seems plausible that we can impact specific student behaviours that might emanate from those goals. For example, I hypothesized above that performance goals might be maladaptive in CS because they focus students on local, rather than integrative, conceptions of the subject matter. We might therefore encourage students to examine weekly readings in light of what they learned earlier, perhaps through explicit scaffolding of this skill using reflection and comparison exercises. In addition, we might better communicate to students that CS concepts are highly interrelated, and that these concepts cannot truly be understood in isolation. For example, students may study chunks of material, such as variables, booleans, conditionals, and loops, without making the deep connections between these topics necessary for writing working programs. It is worthwhile to demonstrate to students that all of these concepts must be understood in relation to one another; learning each topic on its own is necessary but not sufficient for becoming an effective programmer. In addition, given the negative pattern of results associated with performance goals, we might communicate to students that a competitive focus on grades is not necessary for success in CS courses. Prior research suggests that students are aware of the defensive climate sometimes associated with CS courses (Garvin-Doxas & Barker, 2004), where students show-off what they know to make impressions on peers or the teacher. This, in turn, may make others feel that CS is not for them. Focusing students on mastery goals, rather than performance goals, may help show students that CS courses can and should be cites of collaborative learning, not competition. Further research is necessary to explicitly disentangle the relationships between goals, competitiveness, and interest and grades in CS courses.

Finally, as students may exhibit broad conceptions of success and failure that go beyond course grade, we must in turn study conceptions of success that similarly go beyond grades. Grades are undoubtedly the most important and visible marker of success, but understanding grades in relation to other outcomes can lead us to a more nuanced understanding of outcomes and how those outcomes relate to students’ incoming attitudes and goals.
6.8 Conclusion

Much CS literature investigates the question: who will succeed in CS1? The present study examined how two different pedagogies, moderated by student achievement goals, related to multiple outcome measures (grade, interest, and enjoyment). The results revealed that students with performance goals (i.e., those who strive to outperform their peers) performed poorly on the final exam relative to students with mastery goals. This was true in both the traditional lecture and Peer Instruction conditions. Performance-focused students were also less interested in CS than their peers in the traditional lecture section, but not in the Peer Instruction section. Students with mastery goals scored higher on the final exam in both sections and exhibited higher levels of interest and enjoyment. The pedagogical style of the course had no measurable impact on the outcomes of students with mastery goals. The results suggest that i) there may be some advantages to the use of Peer Instruction as a pedagogical tool, particularly for individuals with performance goals; and ii) students who enter CS1 with mastery goals appear to be more successful than individuals who are more concerned with competition or acquiring grades.
Questionnaire Items

6.8.1 Mastery and Performance

The following questions make up the mastery and performance scales, as asked of students on the start-of-course questionnaire. mastery/performance labels were not included. These questions are slightly modified (i.e. changing the word “psychology” to “computer science”) from Harackiewicz et al. (2000).

Indicate the extent to which each statement is true of you from 1 (not at all true) to 7 (very true):

Mastery:

• I want to learn as much as possible in this class.

• In a class like this, I prefer course material that really challenges me so I can learn new things.

• The most important thing for me in this course is trying to understand the content as thoroughly as possible.

• Understanding computer science is important to me.

• I like it best when something I learn makes me want to find out more.

• In a class like this, I prefer course material that arouses my curiosity, even if it is difficult to learn.

Performance:

• It is important for me to do better than other students.

• My goal in this class is to get a better grade than most of the other students.

• It is important for me to do well compared to others in this class.

• I want to do well in this class to show my ability to my family, friends, advisors, or others.

• Getting a good grade in this class is the most important thing for me right now.

• It is important for me to establish a good overall grade-point average, so my main concern in this class is getting a good grade.
6.8.2 Perceived Competence

On the start-of-course questionnaire, students were asked (from Rountree et al. (2002), where the question was the best predictor of course performance):

What grade do you expect to get in [CS1 course code]?

- A
- B
- C
- D
- Unsure

6.8.3 Interest and Enjoyment

The following questions make up the interest and enjoyment scales, as asked of students on the end-of-course questionnaire. Interest/enjoyment labels were not included. These questions are slightly modified (i.e. changing the word “psychology” to “computer science”) from Harackiewicz et al. (2000).

Indicate the extent to which each statement is true of you from 1 (not at all true) to 7 (very true):

Interest:

- I think what we are learning in this class is interesting.
- I think I will be able to use what I learn in this course in other courses.
- I would recommend this class to others.
- I am enjoying this computer science class very much.
- I think the field of computer science is very interesting.
- This class has been a waste of my time.
- I'm glad I took this class.
- I think the course material in this class is useful for me to learn.
- I would like to take more computer science classes after this one.
• I am more likely to register for another computer science class because of my experience in this course.

Enjoyment (first two questions reversed):

• The lectures in this class really seem to drag on forever.

• I don’t like the lectures very much.

• I like my professor.
Chapter 7

General Discussion and Conclusion

What is the value of PI as a pedagogical approach for teaching CS1? I investigated this general question by separately focusing on two subquestions. First, is there evidence that the individual components of PI are valuable? Second, is there evidence that PI as a whole contributes to learning? In this section, I synthesize the presented papers on each of these research questions.

7.1 RQ1: Value of PI Components

I began with a focus on individual PI components to collect evidence on the value of each piece. I wondered whether particular components of PI, transferred and adapted from physics education, were equally valuable in the CS context. This inquiry may suggest ways in which PI may differ (or should differ) when used in CS, and may offer hypotheses for understanding any overall effects of PI.

7.1.1 Reading Quizzes

PI researchers will agree that it is not possible to cover as much material in class meetings due to the time allocated to individual reflection and group discussion (Crouch et al., 2007). One particular challenge for CS educators is that “covering” content in class is related to both conceptual knowledge and practical skill. For example, in a week on loops, it is important for CS educators to cover definitions, syntax, and semantics of loops, but also to model and demonstrate effective use of loops in the students’ repertoire of programming skills. Time taken from these activities to do PI is perhaps one reason for the slow adoption of PI over the past few years.
In physics, this time-allocation dilemma is addressed through reading quizzes. Students read conceptual material before class, then attend class to deepen their conceptual knowledge and apply that knowledge to solve practical problems. When implementing PI in CS, it was necessary to off-load some of the coverage in lecture to make time for student discussion; however, reading quizzes had not yet been adapted from physics. I therefore developed a set of reading quizzes, in addition to a complete PI-based offering of the course, and studied the ways in which reading quizzes were being used by students and their linkages to other areas of the course. As hinted above, my reading quizzes were both skill- and conceptually-based. Students read sections of the textbook to gain conceptual and definitional knowledge, then answered a mix of conceptual and implementation-level questions before class.

Chapter 2 contains the results of my study of reading quizzes. Overall, it seems that reading quizzes can be transferred from physics to CS, given the adaptations discussed above (i.e. including both conceptual and skill components). Almost 88% of responses were “reasonable”, in the sense that students answered correctly or gave complete but incorrect responses. (Recall that students were awarded marks for any solution, and that I was careful to mention this in-class. I told students that I was concerned with effort, not submitting correct responses.)

Perhaps more interesting than the overall quality of responses is the qualitative coding of the “confusion question” and the relationship of these codes to other course assessments. For example, students’ recognizing (and communicating) their confusion was associated with increased performance on course labs. As this study is not an experiment, it is possible that acknowledging confusion and doing well on labs are both produced by some other quality, such as good study habits. However, such relationships between confusion and later success do echo findings by Bjork (1994), who note that the conditions for successful training performance are not necessarily the conditions that yield post-training success. In particular, trainees should be exposed to difficulties in their training; not everything should run smoothly while learning. In one of Bjork’s (1994) examples, subjects were asked to learn the content of a technical article, and were provided an outline that did or did not follow the organization of the article. Compared to those given the consistent outline, subjects given the inconsistent outline performed more poorly on verbatim recall of the article but performed better when asked problems involving inference or understanding. Perhaps students confused by reading quiz material will similarly be well-positioned to remedy that confusion in time to perform well on course assignments. Furthermore, students were not provided the correct answers to the
reading quizzes. The pedagogical value of this choice can be debated, though it may be that reducing feedback during initial learning can in fact be desirable in the longterm (Bjork, 1994).

I conclude that reading quizzes served their purpose of preparing students for lecture, identifying student difficulties and confusion, and engaging students in critically questioning the course material that they were learning. Given suitable attention to conceptual and implementation concerns, reading quizzes appear to be useful in CS PI contexts much as they are useful when learning physics.

7.1.2 Peer Discussion and Instructor Follow-up

Past investigations of PI in CS courses have focused on the brief time between students’ first and second responses to a conceptual question (one response before group discussion and one response after group discussion) (Simon et al., 2010; Zingaro, 2010). Large gains were taken as measures of student learning. However, this metric does not tell us what happens after class is over, or how long students will retain those gains. In fact, the metric fails to distinguish learning from simple copying. One previous study (Porter, Bailey-Lee, et al., 2011) used isomorphic questions to demonstrate that learning had both taken place and persisted. However, that work did not include CS1, and did not consider the effects of the instructor on learning.

In Chapter 3, I similarly used isomorphic questions to investigate students’ ability to transfer learning to isomorphic situations. The placement of the isomorphic question allows for measuring the peer discussion part of PI, or both the peer discussion and instructor follow-up portions. I find evidence that both of these components of PI contribute to learning, insofar as that learning manifests in the ability to answer an isomorphic question. In addition, gains are strongest for difficult questions, and this is true for both peer discussion and instructor follow-up.

In practice, this means that learning gains, as typically measured, are likely underestimates of what students will ultimately learn through the PI process. Students learn from and can demonstrate understanding of the targeted instructor feedback provided after each PI cycle.

In addition, the use of isomorphic questions themselves has implications for how CS instructors understand conceptual change in their students. Two broad theories of conceptual change are knowledge-as-theory and knowledge-as-elements (Ozdemir & Clark, 2007). The first of these argues that knowledge exists as highly-organized schemata that can be used to consistently interpret situations, and that, at any moment, knowledge exists as a coherent whole. The second views knowledge existing as somewhat-independent elements that exhibit high contextual-sensitivity.
Knowledge-as-theory researchers, echoing ideas from Piaget, seek revolutionary changes in knowledge structures, whereas knowledge-as-elements researchers seek evidence for gradual evolutionary changes in knowledge. The latter perspective suggests that what we learn through interaction with the world is not organized and structured into a form reminiscent of a theory, but is loosely connected and activated depending on context. Multiple conflicting ideas can coexist, so that the student may arrive at correct reasoning in one situation but incorrect reasoning in other conceptually-related situations (diSessa, 1993).

There is evidence for the knowledge-as-elements perspective in the CS threshold concepts literature. A threshold concept such as program dynamics encompasses many other concepts such as function calls, recursion, loops, and so on. Lacking a complete understanding of program dynamics, a student may be able to answer a loops question, but not be able to answer a question with a function call inside of a loop. That is, knowledge structures can be context-dependent as students grapple with the full generality of important concepts (Sorva, 2010). By posing isomorphic questions to students, important learning opportunities may present themselves when a student answers the first of a pair correctly but the second incorrectly. Assuming that such questions are conceptually isomorphic, answering one incorrectly is a signal that the concept has not been grasped in full. Therefore, the use of isomorphic questions is advantageous outside of its utility as a research instrument. Teachers can use isomorphic questions to detect fragile understanding in students, particularly those students who often answer one question correctly and the other incorrectly. As students struggle to form a coherent understanding of programming, we should expect half-formed, half-correct, and half-understood concepts to sometimes lead to correct reasoning and other times to incorrect reasoning. Isomorphic questions increase the number of test points with which we can assess the extent to which our students are working with consistent mental models of programming.

In Chapter 4, I demonstrate that performance on isomorphic questions is also associated with scores on various types of final exam questions. Students who come to class prepared (and can correctly answer PI questions on their own) naturally perform well on many types of exam questions. Further, those students who demonstrate learning from peers or from the instructor also exhibit gains on the final exam. Note that these gains were found on code-tracing questions (which are similar to PI questions), and also on code-writing questions (which are very different from PI questions). This can be interpreted in a variety of ways. Perhaps it is that those students who learn from PI are those students who can learn from any reasonable way of teaching; that is, student ability to learn underlies
both PI and exam performance. Or, it could be that, as indicated in recent research (Venables et al., 2009), all programming-related skills exist in a hierarchy where the mastery of lower-level skills is required for higher-level skills. For example, doing well on PI questions means that students can effectively trace, describe, discuss, translate, and reason with code. These skills are prerequisite to being able to write code, so it follows that effective PI performers would also be effective exam performers.

Taken together, the findings from the studies in this section bode well for the effectiveness of PI in CS1. The majority of students complete their pre-class reading quizzes and provide thoughtful and complete responses to the questions. Reading quiz responses correlate with other course assessments, help identify student difficulties, and engage students in reflective questioning. Isomorphic questions demonstrate that students are learning from peers and the instructor. Linkages between PI and the exam take the findings from the isomorphic questions and extend them across the semester.

### 7.2 RQ2: Comparison of PI and Lecture

In Chapter 5, I described the “learning competition” between lecture and PI offerings of a CS1 course. The results are suggestive but not clear-cut. PI students earned higher marks on a common final exam, outperforming the lecture students by 4.4%. This was suggestive but not statistically significant ($p = .10$). This grade difference is almost half a letter grade, so it is meaningful in terms of the grades that students earn. In addition, PI led to significant increases in self-efficacy, which as I have argued is valuable in itself. Therefore, PI leads to statistically significant gains in self-efficacy, but the findings with respect to final exam scores are inconclusive. These positive findings are broadly consistent with the large base of PI research in physics (Crouch & Mazur, 2001) and the small but growing body of research in CS education (Simon, Parris, & Spacco, 2013).

It is interesting to speculate on what it means for PI to increase final exam grades by 4.4%. The score obtained on a final exam depends on many factors. Pedagogical approach is one of these factors, along with time spent studying on one’s own, attending labs, working on assignments, talking to the instructor, studying with peers, solving problems from previous exams, reading the textbook, and so on. I had no control over whether students in the PI section communicated and studied with students in the lecture section. Indeed, considering all of these factors, PI or the matched lectures reflect only a small portion of the time commitment made by these students, and there is precedent
to believe that this dilutes the actual effect of pedagogical intervention. For example, one recent study finds enormous effect sizes when comparing interactive teaching approaches to traditional lecture approaches in physics (Deslauriers, Schelew, & Wieman, 2011), though this study took place over a single week. Such a brief timespan limits the effect of external influences on student learning, bringing the effects of pedagogical change into sharper focus.

In Chapter 6, I investigated how achievement goals interact with pedagogical approach to predict student outcomes. I again found evidence (statistically significant, this time) that PI is superior to lecture. In the lecture section, those students striving for interpersonal performance attainment became less interested in CS than their mastery-focused counterparts. In the PI section, however, both performance- and mastery-focused students became interested in CS. Therefore, with no detriment to the mastery-focused students, PI increased interest for the performance-oriented students.

Unfortunately, PI had no discernible effect in some other areas of investigation important to the current body of CS education research. There was no evidence that PI narrowed the gap between experienced and inexperienced students, or between males and females, or between exam scores of performance vs. mastery students. My investigations suggest that PI leads to moderate learning across the board and increased interest and enjoyment for particular students. This is sufficient to recommend the continued use and further research of PI as a viable CS1 pedagogy.

7.3 Limitations of the Research

7.3.1 Core Concepts, and how to Test Them

There is an assumption in the PI literature that each PI Conceptest targets one core or important disciplinary topic. In physics, the way to do this seems straightforward: look at one of the standardized concept inventories, themselves often multiple choice, and create questions that target the same concepts as those standardized tests. While this may evoke ideas of “teaching to the test” or “objectives-based learning”, the fact is that physicists have codified the important concepts and there is some disciplinary agreement on the goals of a “physics 1” course. By contrast, there is considerable variety in the concepts deemed to be under the purview of CS1. Substantiating this claim requires little more than enumerating some of the many studies that have tried to determine the most important or difficult concepts according to instructors and students (Schulte & Bennedsen, 2006; Robins, Haden, & Garner, 2006; Tew & Guzdial, 2010; Goldman et al., 2010). Part of the
disagreement appears to be related to whether teachers believe that CS1 should be taught using an object-oriented language or an imperative language. For example, imperative teachers rate selection and repetition as highly relevant, whereas object-oriented teachers rate these concepts much lower than inheritance and polymorphism (Schulte & Bennedsen, 2006). This said, there is agreement that some topics deserve to be in every offering of CS1; for example, selection and iteration, simple data structures, and parameters (Schulte & Bennedsen, 2006).

This last point is strongly made by the work of Tew (2010), who developed a language-independent assessment of fundamental CS1 knowledge. The first step in developing such an assessment was to determine whether there really were common CS1 topics covered regardless of language choice or teaching approach. Through an analysis of textbook and curricular guidelines, the author found a small core of ten topics around which seemingly disparate courses conceptually hang. However, as mentioned in the preceding paragraph, outside of these core topics, there is little agreement as to the contents of a CS1.

This lack of focus is coupled with a lack of standardized, available tests of student conceptual understanding. The test of conceptual understanding developed by Tew (2010) is complete and validated but is not generally available to the research community. Unfortunately, this leaves CS education researchers with few evaluative options. I have chosen to use my final exam as a proxy for student learning in CS1. Based on my understanding of the literature, and my experience teaching CS1 many times, I developed (with my co-instructor) an exam that was typical of CS1 exams. The exam was in no way validated: it was used as a standard part of the course evaluation. The exam was also based on our own understanding of core CS1 topics, again of which there is no widespread agreement. Previous work in PI, and my conceptual framework, suggests that the results I found here would not depend on the choice of final exam. However, replication is necessary, particularly when standardized assessment mechanisms become available.

### 7.3.2 Intra- or Inter-Instructor

In the quasi-experiment portions of my studies (Chapters 5 and 6), I compared a PI offering to a lecture offering of a CS1. The two sections were taught by different instructors, and I suggested in the papers that intra-instructor studies might be run to strengthen or oppose my findings.

There is precedent to believe that the instructional strategy has a greater impact than the instructor. In fact, this is the source of an interesting controversy caused by a 2011 article in
the journal Science (Deslauriers et al., 2011). The original article compared outcomes from two pedagogical approaches in physics: lecture and “deliberate practice”, the latter of which engages students in activities that exercise expert-like reasoning and problem-solving. There were two course sections that shared all assessments, but for one week late in the term they used different pedagogical styles and were taught by different instructors. There was no significant difference in initial scores on an electricity and magnetism concept test, scores on two midterms, rates of attendance, level of engagement, or attitudes toward physics. The instructors developed a multiple choice test that they used to measure student learning. (This is similar to the setup of the quasi-experiments reported in the present thesis.) The core finding was that students scored an average of 41% in the control section and 74% in the deliberate practice section. The effect size was 2.5 standard deviations, which is an enormous effect size for interventions of this type. The article produced considerable debate, including one response by Torgerson (2011), who claimed (among other methodological problems) that the teacher effect was not controlled. Perhaps it was the teacher, not the pedagogical differences, that led to some or all of the 2.5 standard deviation increase in learning? In response, the original authors (Deslauriers & Wieman, 2011) marshal evidence from prior work showing that the teacher rarely has an effect independent of the pedagogical approach being used. In addition, the authors argue that it is unnecessary to use validated assessment instruments because they do not typically exist for very many topics, and because they are not the tests typically used by teachers in real classroom settings. Finally, and in contrast to much “hard science” work, there are ethical concerns associated with replication, especially for those pedagogical approaches backed by considerable theoretical and empirical evidence.

Given such ethical concerns and the positive effects of PI described in this work, it is unclear whether the study should be replicated prior to the availability of standardized and widely-used assessment instruments. If the work is replicated using another course final exam, then it makes sense to try an intra-instructor study where the same instructor teaches both sections of the course. Unfortunately, this brings its own set of biases, including an instructor who is likely biased toward one pedagogical style or the other.

Though there is precedent in the literature to pit one pedagogical approach against another to determine which one “wins” (Deslauriers et al., 2011; Simon, Parris, & Spacco, 2013; Hake, 1998), the merits of running such competitions have been challenged. Of particular relevance is a piece by Wise and O’Neill (2009) arguing that “horse races” between constructivist and instructionist
courses are unproductive. The argument to which this work applies is that between those who advocate inquiry-based classrooms and those who advocate considerable guidance through up-front instruction. Comparing test grades achieved through differing pedagogical approaches may tell us which group of students performed better on that test, but says little of the causal processes important for interpretation of results. As Wise and O’Neill (2009) explain, the pedagogical foundations of teaching techniques go beyond questions of quantity of guidance and scaffolding and include goals for learning transfer and the ways that problems are structured. Such epistemological beliefs are bound together into a pedagogical approach which, if compared wholesale to another pedagogical approach, cannot be unbound when seeking causal explanations. Therefore, while I do offer such a comparison of PI and traditional lecture in this thesis, I also present evidence suggesting the utility of individual PI components. Such focused studies of component pieces of pedagogical approaches may be useful for identifying the particular features of pedagogical approaches that contribute to learning outcomes. Unfortunately, such componentizing was studied only in the PI section, not the lecture section. The reason is one of data-generation: PI naturally produces data (reading quizzes, clicker responses) whereas lectures do not. Understanding whether and where PI students make their gains, compared to where lecture students make their gains, has therefore not been achieved. Such fine-grained comparisons would be useful for disaggregating overall effects of pedagogical approaches and working inductively to understand how pieces fit (or do not fit) together to support desired learning.

7.4 Future Research

7.4.1 PI as Data-Gathering

This thesis has taken PI, the pedagogical approach, as the object of inquiry. As mentioned, future work may be warranted to replicate my findings, especially as validated assessment tools for CS learning become available. Beyond that, there is considerable opportunity to learn from PI as a data-gathering technique. By virtue of its reliance on clickers, PI endogenously generates naturalistic classroom data that can be used for fine-grained tracking of students as they progress through the term. Indeed, such analyses are largely independent of comparisons between PI and lecture. Nonetheless, if future work corroborates the findings here and PI becomes increasingly used, it will be a happy coincidence that a pedagogy shown to be valuable for student learning also leaves data
traces that are valuable as objects of inquiry.

In recent work (Porter, Zingaro, & Lister, 2014), we have begun answering the question: what can we learn from PI data itself? Specifically, that work uses PI data as fine-grained approximations of student understanding, and tracks this understanding throughout the semester. Elevated CS1 failure rates and suggestions that students who fall behind stay behind led us to use PI data to examine the extent to which early performance is correlated with exam scores. We found that student clicker scores in the first three weeks correlated highly with scores on final exam questions. These week 1-3 scores were stronger predictors than any other later tri-weekly period, even for exam questions based on material taught late in the term. Those in the bottom quartile on weeks 1-3 clicker performance were very unlikely to score above the median on key final exam questions.

The utility of these findings bodes well for the use of PI data to respond to long-standing CS education questions. For example, are there important gender differences in the ways that students learn or fall behind in CS courses? Do students with no prior experience catch-up to those with prior experience as the course progresses? For perhaps the first time, we have hundreds of datapoints, not a handful of datapoints, that can be applied in relevant data analyses.

A particularly intriguing possibility is the use of PI data to inform targeted interventions to help students overcome early difficulties. Notions that students spiral rapidly to the top or bottom of a CS1 grades distribution (Robins, 2010) suggest that we must act quickly in order to intervene before it is too late. Traditional CS1 offerings, with one or two midterms, may not provide useful data sufficiently early to make interventions meaningful. With PI, clicker data is available immediately. We have already observed that particular questions asked early in the semester can predict scores on exam questions (Porter et al., 2014). In particular, there are “easy” questions that, if answered incorrectly, predict low performance on core exam questions: correctly answering these clicker questions is necessary but not sufficient for passing the exam. Students who fail to correctly answer these “easy” questions are at risk of failing the course; now, with PI, these students can be identified immediately.

There are several lines of research necessary for producing interventions that work. First, we must further research these “necessary but not sufficient” questions, understand why they are so important, and be able to generate such questions for more CS1 topics. Second, we must decide what to do once we determine the students who are at risk. Offering extra help (review sessions, practice problems, office hours) does not tend to work (Deslauriers, Harris, Lane, & Wieman, 2012), so we cannot simply “add more” and expect weak students to avail themselves of the additional
resources. One encouraging study suggests that interventions focused on specific study strategies can be effective, particularly when exams are aligned with learning goals (Deslauriers et al., 2012). In that study, students who performed poorly on the first midterm were asked to meet the instructor or were sent an email containing studying advice. Results are not conclusive, but it appears that meeting face-to-face with the instructor was superior to both failing the midterm alone and failing the midterm in combination with receiving the study-advice email. Armed with early and frequent PI data, how might we further help weak students? Might we be able to intervene on a content level by virtue of our ability to notice content deficiencies and misconceptions very early? Are those students we identify through PI data those students with poor study strategies? These are all open questions. I am excited by the possibilities open to computing educators who embrace PI data (or classroom data more generally) as a powerful formative feedback mechanism.

### 7.4.2 PI as Scripted Inquiry

While recent studies suggest that collaborative pedagogies are useful for teaching and learning science, there is evidence that students may not automatically engage in productive collaboration (Kollar, Fischer, & Slotta, 2007). For example, students may pose arguments that they and their partners do not resolve, rely on extraneous cues rather than discuss core content, or passively agree with others (James & Willoughby, 2011). Such behaviours may limit what students learn about scientific argumentation and thereby limit their acquisition of scientific knowledge itself (Kollar et al., 2007).

Collaboration scripts have been offered as one means to address these challenges. These scripts scaffold the discursive processes in which students should engage as they work together and hold considerable promise particularly in technology-rich learning environments (Slotta, 2013). The focus is on the collaboration process itself, not explicitly on conceptual knowledge. Scripts structure students’ cognition in ways that facilitate valid scientific reasoning; conceptual understanding of a particular scientific domain can then be acquired with the help of such scientific processes.

Scripts can be coarsely categorized by their level of structuredness: some scripts impose few constraints whereas others carefully orchestrate the particular activities, the times at which these activities should occur, and the particular students that should act at each step of the activity. There is some evidence that collaboration processes are particularly improved by highly-structured scripts, though this is to be balanced with the dangers of “over-scripting” (Dillenbourg, 2002) and artificializing the collaborative inquiry.
PI shares some of the above features of collaborative scripts. It dictates when students are to work alone and when students are to collaboratively work toward a solution. It orchestrates exactly how much time students are to spend on each activity. Finally, it suggests the importance of argumentation, discussion, and building consensus before committing to a response choice. As far as scripts go, however, this is considerably unstructured. Students are not given much advice on how to argue convincingly, and the several minutes of discussion for each question are unstructured in the sense that students can use the time as they wish.

Viewing PI as a script is useful in that it makes available a host of hypotheses and related findings from the collaborative scripting literature. For example:

- What are the ways in which PI can be made more highly-structured, and is such structure associated with increased conceptual- or argumentation-related gains? One approach may be to script the discussion process itself. Student discussion time could be split into three one-minute chunks corresponding to argument, counterargument, and reply synthesis (Kollar et al., 2007). That is, student A would be given the first minute to offer their perspective; student B would then have one minute to do likewise; then both students would work toward consensus in the final minute. The script would change accordingly for groups of three students.

- How do students' internal scripts affect what is learned through PI? An internal script consists of relevant procedural knowledge that students possess prior to invocation of any collaborative script. For example, students will hold conceptions of appropriate and inappropriate forms of argumentation and persuasion that can be generally applied to a collaborative setting (Kollar et al., 2007). Such internal scripts likely impact what is discussed, how turn-taking is negotiated, and affective dimensions of relationship-building and conflict resolution. Little work has examined questions of effective PI groups (Alvarado, Lee, & Gillespie, 2014) or profitable characteristics of students' internal scripts.

- How should student knowledge contributions be aggregated? Using a clicker system, student responses are automatically aggregated based on response choice and displayed as a histogram. In the pedagogical scripting literature, aggregate representations are chosen based on the likelihood that they can help the teacher and students explore important patterns that may lead to further insight (Slotta, Tissenbaum, & Lui, 2013). It is important here to distinguish PI the pedagogical method from clickers the technology: we can have the former without the
latter, and so it makes sense to investigate new ways of aggregating student responses that may not involve clickers. This is particularly important in CS, where we have found clickers to be limiting input mechanisms, especially where code-entry is concerned (Zingaro, Petersen, et al., 2013). An important outcome of CS1 is that students be able to write code. Yet, clickers do not support code-entry, and so our questions necessarily cannot ask for students to write code from scratch. We can get quite close to the same skills by asking students to read and reason with existing code, or reorder code so it works correctly, or find bugs in existing code, etc. But what we cannot effectively do with clickers is acquire student code snippets and aggregate them into a form suitable for instructor response and furthering the dialogue. One way to aggregate student code is to use a web interface rather than clickers, where students submit their code using their own connected devices (Zingaro, Petersen, et al., 2013). Student code can be run through a test suite, and submissions that pass the same set of test cases can be deemed equivalent as far as aggregation is concerned. There is much more work to do in this area, including a determination of whether such test-case-based aggregation is useful and meaningful in the first place.

- How might the PI script change depending on the structuredness of the domain being learned? Many CS courses, such as software design and data structures, are well-positioned to ask students questions about ill-defined problems. For example, “which design is best”, or “discuss possible data structures for solving this problem” may not have a right answer per se, but may instead require social negotiation of strengths and tradeoffs of various approaches. Collaborative learning researchers note that solving such ill-structured problems is often facilitated by providing such examples in the learning process itself (Kapur, 2008). Compared to well-structured learning exercises, ill-structured ones will likely lead to poorer short-term performance. However, such “productive failure” is often seen when students later tackle ill-structured problems on their own. A particular question for PI implementation is how to engage students in discussions of ill-structured problems. Certainly the notion of one correct answer will have to be abandoned. But so might the ways in which responses are aggregated and the form of instructor follow-up.
7.5 A Personal Reflection

Reflecting on the teaching process itself, independent of the research, brings forth memories of the extensive time and energy required to design and offer the PI section of CS1. In many ways, I have come to understand that lecturing is far easier than PI. PI required me to author 35 reading quizzes, write hundreds of clicker MCQ questions, solve numerous technical clicker problems, read reading quiz responses before class to make just-in-time changes, think quickly on my feet in response to unanticipated directions of inquiry, and sometimes generate new questions on the spot. In traditional lecture settings, instructors can carefully curate what happens, what the students see, and how much time is spent on each topic. Barring unexpected technical trouble, the teacher can make relevant pacing and coverage decisions to make things run quite cleanly. PI is less linear, less structured, less familiar, and hence more contingent and unknown. I continue to find it difficult to feel properly prepared for a PI class, perhaps because one cannot actually anticipate what will happen and how to best respond. Not all relevant pedagogical decisions can be bound prior to class; some of them must be decided in situ as dictated by student learning.

One recurring fear of mine relates to students performing dismally on an important PI question. This in itself is OK: good PI teaching would then suggest developing an isomorphic question (or having one prepared in advance). But what then if students do not improve on the isomorphic question, even with the support of peers and the instructor? Another isomorphic question, perhaps, and then another one? But what to do when this throws off timing for future lectures and other course decisions? “Giving up” and moving on feels disingenuous, particularly because one benefit of clickers (for the instructor and thereby for the students) is that they tell the instructor when to slow down and set a new pace. Pace-setting and contingent instruction are core benefits of clickers according to students (Han, 2014), and these are “sells” that I use for student buy-in. Realities of post-secondary education, of course, mean that this pace-setting can go only so far. I can remember a particularly difficult question on ragged two-dimensional lists that I thought was core to student understanding of lists, nesting, and order of subscripts. I reasoned that if I could help students toward understanding this question then it would go a long way toward clearing up errors such as off-by-one indexing, using column-order indexing instead of row-order indexing, and so on. I had also prepared an isomorphic question for the topic, as I had anticipated trouble. And I was right: on the initial question, students scored 40% on the individual vote and 50% on the group vote.
Things didn’t improve much on the isomorphic vote. Now what? What I did was to tell students that I had to move on, but that they should revisit the questions again on their own if they didn’t understand them. And then, at the time, I felt as though I had given up on teaching this particular topic properly, for the sake of moving on with the curriculum and keeping to a reasonable pace. However, a more balanced view is in order. First, perhaps the question was simply not effective: too difficult, not appropriately timed, not motivated properly. Second, and in spite of any concerns about the question, it may have in fact succeeded. It gave students evidence, in the form of wrong answers, uniform-distribution response graphs, and probably some evident instructor exasperation, that they did not understand something that I obviously thought was important. This in itself can be motivating. Unfortunately, I have no happy conclusion to this story: no clear end-of-semester data point with which to remeasure this learning. But I do have a realization that not all conceptual learning must conclude before the associated class concludes. It may be sufficient to show students that they lack important understanding, and leave them to resolve the conflicts. This is certainly preferable to students unaware of their conceptual difficulties in the first place.

This thesis offers evidence that PI is valuable for student learning. Being an early PI adopter, it is perhaps expected that I would have by now developed PI offerings for all of the courses that I teach. However, I have not. (I’m working on it.) Several of my courses are still lecture-based, owing entirely to the investment required to create a PI course from scratch.

Fortunately, this situation is changing. I and my colleagues have contributed all of our PI resources to a central website. Anyone can visit peerinstruction4cs.org and gain access to a variety of PI offerings, including CS1, CS2, Operating Systems, Architecture, and more. It is our hope that the effort required to generate a single PI offering can be leveraged to benefit the community at large. In fact, the PI course that has served as the subject of this thesis is available in full from that website.

7.6 Conclusion

The CS1 educational research literature is replete with examples of students performing poorly on tests, demonstrating fragile conceptual knowledge, and failing CS1 courses at alarming rates. Some authors (Dehndadi et al., 2009) have gone so far as to suggest that there are two types of students — those that can program and those that cannot — and that these students can be identified and binned prior to any instruction in CS1. These authors have since relaxed this assertion, but the
general sentiment from this type of work is that students will succeed or fail based on criteria that are beyond our purview.

An alternate perspective is more empowering. Laurillard (2002) argues that “university teachers must take the main responsibility for what and how their students learn” (p. 22), and that “teachers create the choices open to [their students]” (p. 22). One way in which teachers assume responsibility for learning is through their choice of pedagogical approach.

In this thesis, I investigated one such pedagogical approach in one computer science course: Peer Instruction (PI) in CS1. I studied the adaptation of PI from physics to CS, including the use of reading quizzes, the role of peers, the value added by the instructor, and the relationship between PI questions and final exam scores. I then studied the effect of PI on student outcomes, including grades, self-efficacy, enjoyment, and interest, as well as possible differential effects on students of different genders or prior experience levels.

Reading quizzes, a PI best-practice from physics, were shown to transfer well to the CS context when adapted to use code-reading and code-writing exercises. Placing isomorphic questions at various points in the PI process demonstrated that peer discussion alone, instructor follow-up alone, and the combination of the two was effective for engendering learning. The gains on these isomorphic questions were later positively associated with exam scores.

In terms of the effectiveness of PI, I showed through a quasi-experiment of two CS1 sections that PI is positively related to some outcomes and unrelated with other outcomes. PI increases self-efficacy and enjoyment over standard lecture, and these two outcomes are known to predict persistence and later achievement. In addition, PI increases interest for performance-oriented students, though there is no such positive effect on mastery-oriented students. Finally, PI students scored almost 5% higher than the control section on a CS1 final exam that was written to be similar to typical CS1 exams, though this difference was not statistically significant. By most accounts, therefore, PI is shown to be superior to lecture and certainly never inferior to lecture. The data collected in this thesis is further evidence of the usefulness of PI in CS1.

There is little work that examines the use of PI in other CS courses. Is PI equally effective in theory courses, in mathematical courses, or in design courses? How are questions to be developed in each of these cases? Do reading quizzes remain useful tools for preparation? There is much work to do. The core contribution of my thesis is that this work is worth doing. I suspect that our present use of PI is not wholly optimized for the CS context. We have taken a pedagogical approach from
physics and, in general, have used the approach as-is. This thesis discusses the ways that reading quizzes, peer discussion, instructor explanation, and isomorphic questions are useful to students’ learning, but the pedagogical approach remains a close descendant of physics-based PI. In many ways, this is good news. Using PI in a recognizable form means we can study physics education literature to get a sense of the effectiveness of PI. We can replicate studies from physics as a check on our PI progress. In other ways, however, we should expect our incarnation of PI to differ based on differing teaching contexts. Code-writing is an important example of a CS skill that we have not explicitly assimilated into the PI protocol. Given the encouraging findings of this thesis, it is time to explore the ways that we can take ownership of PI for our own academic discipline and shape the pedagogy in new ways.
References


References

Cambridge: MIT.


problems. In *Proceedings of the fourth international workshop on computing education research* (pp. 113–124). ACM.


References

123


References


Appendix A

Research Ethics Board Materials

My research was approved by the Research Ethics Board of University of Toronto, protocol reference number 27982.

A.1 Approval Letter

Following is the text reproduced from the protocol approval letter.

PROTOCOL REFERENCE # 27982

August 2, 2012

Dr. James Hewitt

DEPT OF CURRICULUM, TEACHING & LEARNING
OISE/UT

Mr. Daniel Zingaro

DEPT OF CURRICULUM, TEACHING & LEARNING
OISE/UT

Dear Dr. Hewitt and Mr. Daniel Zingaro,

Re: Your research protocol entitled, "Predictors of success across pedagogy in Computer Science 1"

ETHICS APPROVAL

Original Approval Date: August 2, 2012

Expiry Date: August 1, 2013
Continuing Review Level: 1

We are writing to advise you that the Social Sciences and Humanities Research Ethics Board (REB) has granted approval to the above-named research protocol under the REB’s delegated review process. Your protocol has been approved for a period of one year and ongoing research under this protocol must be renewed prior to the expiry date. Any changes to the approved protocol or consent materials must be reviewed and approved through the amendment process prior to its implementation. Any adverse or unanticipated events in the research should be reported to the Office of Research Ethics as soon as possible. Please ensure that you submit an Annual Renewal Form or a Study Completion Report 15 to 30 days prior to the expiry date of your current ethics approval. Note that annual renewals for studies cannot be accepted more than 30 days prior to the date of expiry. If your research is funded by a third party, please contact the assigned Research Funding Officer in Research Services to ensure that your funds are released.

Best wishes for the successful completion of your research.

Yours sincerely,

Margaret Schneider, Ph.D.,
C.Psych
REB Co-Chair

Sarah Wakefield, Ph.D.
REB Co-Chair

Dean Sharpe, Ph.D.
REB Manager

OFFICE OF RESEARCH ETHICS
McMurrich Building, 12 Queen’s Park Crescent West, 2nd Floor, Toronto, ON M5S 1S8 Canada
Tel: +1 416 946-3273
Fax: +1 416 946-5763
ethics.review@utoronto.ca

http://www.research.utoronto.ca/for-researchers-administrators/ethics/

A.2 Informed Consent

Following is the text reproduced from my letter of informed consent.
Dear Student:

As was mentioned in your CSC108 lecture, I am conducting a study of characteristics of students taking CSC108, and measures of interest, enjoyment, and performance in the course. My long-term goal is to improve the course by identifying what can be done to increase enjoyment and performance for all students.

Whether you participate in this study or not, you will be asked to fill out two surveys: one now, and one close to the end of the semester. The total time required will be less than an hour.

You will receive a 1% bonus on your overall course grade for filling out these surveys, as I think it is useful for you to be introduced to the type of research happening in computer science education. The first survey must be filled by Sept. 16, and the link is at the bottom of this email.

If you wish to participate, I will include your numeric survey responses in my study data. For purposes of data analysis, I will link your survey responses (and PI responses, if you are in my section) to final grade. Following course completion, all data will be made anonymous, so that I have no link between you and the survey and performance data from the course. Only aggregate statistics will be published with no identifying information. And, as you would expect, none of your data will be made available to anyone. It will be held strictly confidential. Further, the anonymous data will itself be destroyed within five years.

You have the right not to participate in this research. If you do not wish to participate, you will still receive the 1% bonus as long as you fill out the surveys. However, in this case, your data will be immediately discarded and none of your data will be used in my study. Think of the surveys as a walk-through of the study, with no data saved. Additionally, you may contact Gerhard (the other CSC108 instructor) to withdraw your decision to participate at any time; if you withdraw your participation, your data will in no way be used in the study. Whether you participate or not, rest assured that your decision in no way affects your performance in the course. During the course, I will not have access to your survey responses or your participation/non-participation decision.

As a participant in the study, you have the right to receive a copy of the research findings. Once the research is complete, I will provide to you a small pamphlet by email outlining the study’s findings. In addition, you will be able to download the full report for further reading.

Should you have any further questions concerning your rights as a research participant, you may contact the Office of Research Ethics at ethics.review@utoronto.ca or 416-946-3273.

Daniel Zingaro, Instructor Email: daniel.zingaro@utoronto.ca You may use this contact infor-
mation to request a copy of the research findings after the research is complete (fall 2014). To withdraw from the research study at any time, please email Gerhard (gerhard.trippen@utoronto.ca).

Please click the following link to fill out the first survey. You will be presented a consent form, with two options corresponding to participating and not participating in the study.

[link deleted; survey no longer active]
Appendix B

Self-Efficacy Instrument

In Chapter 5, a self-efficacy scale was revised and used in data collection. The scale does not appear in the paper, but is reproduced below. It is based on the scale of Ramalingam and Wiedenbeck (1998).

Rate your confidence in doing the following Python programming related tasks by using a scale of 1 (not at all confident) to 7 (absolutely confident). If a specific term or task is totally unfamiliar to you, please select 1.

- Not at all confident
- Mostly not confident
- Slightly confident
- 50/50
- Fairly confident
- Mostly confident
- Absolutely confident
- I can write syntactically correct Python statements.
- I understand the language structure of Python and the usage of the reserved words.
- I can write logically correct blocks of code using Python.
• I can write a Python program that displays a greetings message.

• I can write a Python program that computes the average of three values.

• I can write a Python program that computes the average of any given number of values.

• I can use built-in functions that are available in the various Python modules.

• I can build my own Python modules.

• I can write a small Python program given a small problem that is familiar to me.

• I can write a reasonably sized Python program that can solve a problem that is only vaguely familiar to me.

• I can write a long and complex Python program to solve any given problem as long as the specifications are clearly defined.

• I can organize and design my program in a modular manner.

• I understand the object-oriented paradigm.

• I can identify the objects in the problem domain and declare, define, and use them.

• I can make use of a pre-written function, given a clearly labeled declaration of the function.

• I can make use of a class that is already defined, given a clearly labeled declaration of the class.

• I can debug (correct all the errors in) a long and complex program that I had written and make it work.

• I can comprehend a long, complex multi-file program.

• I could complete a programming project if someone showed me how to solve the problem first.

• I could complete a programming project if I had only the language reference manual for help.

• I could complete a programming project if I could call someone for help if I got stuck.

• I could complete a programming project once someone else helped me get started.

• I could complete a programming project if I had a lot of time to complete the program.
• I could complete a programming project if I had just the built-in help facility for assistance.

• I could find ways of overcoming the problem if I got stuck at a point while working on a programming project.

• I could come up with a suitable strategy for a given programming project in a short time.

• I could manage my time efficiently if I had a pressing deadline on a programming project.

• I could mentally trace through the execution of a long, complex, multi-file program given to me.

• I could rewrite lengthy confusing portions of code to be more readable and clear.

• I can find a way to concentrate on my program, even when there were many distractions around me.

• I can find ways of motivating myself to program, even if the problem area was of no interest to me.

• I could write a program that someone else could comprehend and add features to at a later date.