The University of Texas at Austin, Office of Graduate Studies

Science Education

Perceptions of Teaching and Learning Automata Theory in a College–Level Computer Science Course

a thesis

by

Phoebe Kay Weidmann

submitted in partial fulfillment of the requirements
the degree of

Ph.D

Perceptions of Teaching and Learning Automata Theory in a College–Level Computer Science Course

Abstract

This dissertation identifies and describes student and instructor perceptions that contribute to effective teaching and learning of Automata Theory in a competitive college–level Computer Science program. Effective teaching is the ability to create an appropriate learning environment in order to provide effective learning. We define effective learning as the ability of a student to meet instructor set learning objectives, demonstrating this by passing the course, while reporting a good learning experience.

We conducted our investigation through a detailed qualitative case study of two sections (118 students) of Automata Theory (CS 341) at The University of Texas at Austin taught by Dr. Lily Quilt. Because Automata Theory has a fixed curriculum in the sense that many curricula and textbooks agree on what Automata Theory contains, differences being depth and amount of material to cover in a single course, a case study would allow for generalizable findings. Automata Theory is especially problematic in a Computer Science curriculum since students are not experienced in abstract thinking before taking this course, fail to understand the relevance of the theory, and prefer classes with more concrete activi–ties such as programming. This creates a special challenge for any instructor of Automata Theory as motivation becomes critical for student learning.

Through the use of student surveys, instructor interviews, classroom observation, material and course grade analysis we sought to understand what students perceived, what instructors expected of students, and how those perceptions played out in the classroom in terms of structure and instruction. Our goal was to create suggestions that would lead to a better designed course and thus a higher student success rate in Automata Theory.

We created a unique theoretical basis, pedagogical positivism, on which to study college–level courses. Pedagogical positivism states that through examining instructor and student perceptions of teaching and learning, improvements to a course are possible. These improvements can eventually develop a "best practice" instructional environment. This view is not possible under a strictly constructivist learning theory as there is no way to teach a group of individuals in a "best" way. Using this theoretical basis, we examined the gathered data from CS 341.

Our classroom observations revealed several useful instructional techniques. First, an overview lecture should be given so that students have a schema by which to pigeonhole concepts during the semester. Second, using a course webpage to post solutions to homework helps reduce the time between which students complete an assignment and when they receive feedback.

The interview data suggested that Dr. Quilt's instructional strategies, thus perhaps other Automata Theory instructors' strategies, were strongly influenced by how she learned the material. She covered roughly the same material in CS 341 that she was taught. Dr. Quilt also strongly believes that learning abstract material comes through practice, which is how she mastered the concepts.

Student survey responses indicated a love/hate relationship with the topics in CS 341. Students disliked the abstractness of the material, but liked the challenge of solving problems. The responses also suggested that the material at the end of the course should be given more lecture time by covering finite state machines faster. Students expressed that finite state machines were easier to learn.

Using end−of−course grades, we analyzed student performance by sub−dividing students into four categories: ultra, high, average, and low performers. Using these categories, we took a holistic view of the survey data to find specific characteristics that could predict performance. Attendance was important for success. We also found that past performance in prerequisite theory courses was a statistically significant indicator for success in CS 341. This work concludes with a summary of suggestions for Automata Theory instructors aimed at improving student learning experiences and success.

The Dissertation Committee for Phoebe Kay Weidmann
certifies that this is the approved version of the following dissertation:

Perceptions of Teaching and Learning Automata Theory in a College-Level Computer Science Course

Committee:

Perceptions of Teaching and Learning Automata Theory in a College-Level Computer Science Course

by

Phoebe Kay Weidmann, M.S., B.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 2003

For those who believed I could do it.

Acknowledgments

There are several people who I have had the privilege of knowing and who have greatly impacted my work here at UT. I could not have done this process alone. To those of you that are not specifically listed here, know that your contributions have not been forgotten, but that I wanted to finish writing this dissertation quickly. There is just no way I could possibly do justice and thank every person who helped me along the way although, in my heart, you are all very special.

With the brevity of this section in mind, I'd first like to thank three specific professors for cultivating my interest in Computer Science: Mohamad Gouda, Allen Emerson, and Jeff Brumfield. Sometimes you are lucky enough to have good teachers that make material interesting, and sometimes you are lucky enough to have that material brought to life. Each of these professors made theoretical Computer Science topics come to life for me. Later I would work briefly with Mohamad Gouda and get my feet wet in the PhD process. I would have never even entered the PhD program except that Allen Emerson personally made sure that I was accepted into the program, even with my not so stellar GRE scores. And I could not have finished this degree without Jeff Brumfield taking pity on me and hiring me as an assistant when I was no longer able to work as a TA or GRA (yep...been here that long). He also patiently waded through every Education term paper, every cover letter, and every draft of my dissertation just to keep me sane. Jeff also had a very odd sense of motivation. At one point we were cleaning rooms in RLM because we were upset at the janitorial staff and we ended up scraping gum off of desks. Whenever I would look as if I

was slacking on my thesis writing, Jeff would bring in a scraper and wave it around and say, “write or scrape gum!” Truly effective technique. He was also responsible for introducing me to Sharan Happel, an assistant to another Dean in Natural Sciences (there are many). She has been a true cheerleader and believed I could do this even when I was in doubt. In fact, she started calling me Dr. Weidmann about a year before I finished and grin so wide while saying it that I would just have to laugh every time.

As all students have experienced, especially those in graduate school, there were several times when I thought I just couldn’t make it through the program. I would vacillate between thinking that I didn’t have what it took, wasn’t smart enough, or just didn’t fit in with what was expected in order to succeed. There were two people who were pivotal in getting me through these periods of self-doubt: Samuel Guyer and Nina Amla. Sam, my old office mate, was always ready with a pithy comment to make me feel better about my lack of self efficacy, or was available for listening with a sympathetic look to my long rants about specific woes while going through graduate school. Nina took a bit more direct approach and told me about how the world worked in graduate school and that I wasn’t allowed to give up. They are both invaluable to me and I owe them more than I’ll ever admit to their faces.

In fact both Sam and Nina were beyond supportive of my switch from Computer Science to Computer Science Education. If I remember correctly, Sam said that it was where I belonged all along when he saw the huge transformation of lifted spirits and enthusiasm after I began my new direction. Nina, in her more direct fashion, told me that the people in the Science Education department were just too nice for me not to enter and pursue a degree there. These two were not the only people who supported my decision to switch. Vicki Almstrum, one of the members of my committee, actually did a little hop and said that she had been waiting for me to “see the light.” I had not experienced that sort of enthusiastic support from a faculty member in my entire research career up to that point.

The switch led to a new faction of friends over in “my other world.” I was frustrated the first semester when I was taking the necessary education courses. The language was different, everyone sat in circles, and lectures were more group discussion than imparting of information. I was lost until I met Magnia George who actually came from the hard sciences as well and spoke in a way that I understood. She convinced me that I could actually make it in this new strange world and that I did indeed have a place in Education. I feel like I have known her forever even though I will have only known her for a couple of years when this dissertation is accepted. I also gained other good friends in La Vergne Lestermeringolo and Bhaskar Upadhyay who taught me that group projects could actually work and be fun. Both patiently pulled me out of the "do everything yourself" mode that I had acquired over in the Computer Sciences Department and helped me re-learn how to function academically on collaborative tasks. Due to these friendships, I joined a graduate student group called "Dissertations Anonymous." The members of the group were: Magnia George, Barbara Austin, Janice Meyer, Bhaskar Upadhyay, Yu-Mei Lee, Eunmi Lee, Luis Tinoca, and Brian Fortney. We met every Wednesday the year that I was collecting data and writing this dissertation to eat junk food and chat about current results from research in the group. They were a lot of help in nailing down important issues and wording significant results.

These little stories do not even begin to scratch the surface of the people I would like to thank. In particular, there is the lunch bunch which consisted of (over the years): Jeff Thomas, Yannis Smaragdakis, Daniel Jiménez, Ramgopal Mettu, Emery Berger, Sam Guyer, Brendon Cahoon, and Xianglong Huang. They kept me laughing on a daily basis and shared with me their trials and tribulations so I didn't feel so alone. Daniel in particular is the best proof reader I've ever had. He has this uncanny ability to read and critique anything I do and still make me feel like I did a great job, even when it sucks. In fact, I guess I should mention that Daniel, Sam, and I had a breakfast/coffee group going for about a year as well which was a wonderful way to start the day.

Then there is the Scuba gang. I have taught Scuba at UT for the past 7 years, and all of the instructors have gotten a sense for when things are going well and they could pal around with me and when things are not going well and they should just let me get in the pool and chill out. But they too were always making me laugh and forget about any hardships. In particular Jay Reichman, Walter Hokanson, Laura Albright, and Cuatro Kowalski were willing to arrange for a show of excitement or feign outrage depending on what the situation called for.

There were also a number of friends that even though they were out of town were only a phone call, letter, or email away for some friendly chat and encouragement. Some I have known longer than I have been at UT and their continued patience with me being a "gradual" student is greatly appreciated.

Speaking of being a "gradual" student, I have been fortunate to find committee members that would let me set fire to the normal pace in which to finish a PhD in the Science Education department. Without their advice and help I would not have been able to finish this work in only two years. I've had a lot of fun challenging old ideas and learning new perspectives and ways of thinking/writing from each of them. Dr. Bethel was always giving me pep talks and friendly advice that would generally end with both of us smiling. When I would pass Dr. Barufaldi in the hall, he would sarcastically say something to the effect of he had never seen such bad work. And then, as if anyone could misinterpret his obvious joke, he would school his face into a very serious fatherly look and say, "no, really, it's just fabulous." Dr. Drake, although half way across the country, could always give me warm fuzzies with her superb advice and guidance over email. She has the same ability as Daniel to make me feel good about my work even when it may not be warranted. Dr. Rich is probably the only other person in the world that talks as fast as I do and we have had awesome conversations about almost everything. Without her specific knowledge of Microsoft Access I would probably still be analyzing survey data. Dr. Almstrum, who was my first champion in this dissertation process, has continued to be nothing but supportive and sweet, even when I would dump proofing and other requests in her lap with only a 36

hour notice. I swear I'll work on my procrastination tendencies!

There is also the "dissertation anonymous" group who read through my survey responses to help me assign meaningful codes to the free responses, and the CS Education group who listened to progress reports and helped validate themes found in my data.

And last but definitely not least there is my family to thank. I am fortunate enough to have several relatives in town and when we get together for dinner all troubles seem to melt away. Amy, my sister, has several times come to the rescue and helped me around the house when I've gotten overwhelmed with school, or sometimes she would just come over and rub my feet...now that's dedication! Garth, my brother, would constantly offer to beat up any person giving me trouble. Kim, my aunt would always have uplifting advice for getting through the process as she herself has two Masters degrees. My uncles Eric and Duke kept me laughing and on my toes with our constant competition of wits. My aunt Monica would always tell any stranger who would listen how smart I was. And my two little cousins Mara and Leah would remind me how fun it was to be young at heart.

Then there were the family members that were out of town but in no way less supportive. Constant emails from my mother, Cheryl, would make a pleasant break in the work day; and my dad would send funny jokes (he doesn't type very fast so actual correspondence was right on out of the picture...we call him "twinkle finger" due to his one finger hunt and peck methodology when typing) and would call and inquire, "How's school PK?" In fact I think he also offered to beat up people causing me trouble...hmmm. My aunt Evelyn would remind me that getting a PhD (she herself has one in music) took forever and that I was doing fine. All my other aunts, uncles, and cousins would send letters and make phone calls to make sure I was still alive and feeling good about what I was doing. And at the end of the day I could always enjoy coming home to a purring cat, Kaelin.

And last but definitely not least, I need to thank my husband, Henry Pierluissi. He put up with all my bad moods, listened to all the downs and encouraged all the ups. And although he probably thought that I would never really finish, he continued to support me.

I did it, baby!

PHOEBE KAY WEIDMANN

The University of Texas at Austin

May 2003

Perceptions of Teaching and Learning Automata Theory in a College-Level Computer Science Course

Publication No. ______________________

Phoebe Kay Weidmann, Ph.D.
The University of Texas at Austin, 2003

Supervisor: Lowell J. Bethel

This dissertation identifies and describes student and instructor perceptions that contribute to effective teaching and learning of Automata Theory in a competitive college-level Computer Science program. Effective teaching is the ability to create an appropriate learning environment in order to provide effective learning. We define effective learning as the ability of a student to meet instructor set learning objectives, demonstrating this by passing the course, while reporting a good learning experience.

We conducted our investigation through a detailed qualitative case study of two sections (118 students) of Automata Theory (CS 341) at The University of Texas at Austin taught by Dr. Lily Quilt. Because Automata Theory has a fixed curriculum in the sense that many curricula and textbooks agree on what Automata Theory contains, differences being depth and amount of material to cover in a single course, a case study would allow for generalizable findings. Automata Theory is especially problematic in a Computer Science curriculum since students are not experienced in abstract thinking before taking this course, fail to understand the relevance of the theory, and prefer classes with more concrete activities such as programming. This creates a special challenge for any instructor of Automata

Theory as motivation becomes critical for student learning.

Through the use of student surveys, instructor interviews, classroom observation, material and course grade analysis we sought to understand what students perceived, what instructors expected of students, and how those perceptions played out in the classroom in terms of structure and instruction. Our goal was to create suggestions that would lead to a better designed course and thus a higher student success rate in Automata Theory.

We created a unique theoretical basis, pedagogical positivism, on which to study college-level courses. Pedagogical positivism states that through examining instructor and student perceptions of teaching and learning, improvements to a course are possible. These improvements can eventually develop a “best practice” instructional environment. This view is not possible under a strictly constructivist learning theory as there is no way to teach a group of individuals in a “best” way. Using this theoretical basis, we examined the gathered data from CS 341.

Our classroom observations revealed several useful instructional techniques. First, an overview lecture should be given so that students have a schema by which to pigeonhole concepts during the semester. Second, using a course webpage to post solutions to homework helps reduce the time between which students complete an assignment and when they receive feedback.

The interview data suggested that Dr. Quilt’s instructional strategies, thus perhaps other Automata Theory instructors’ strategies, were strongly influenced by how she learned the material. She covered roughly the same material in CS 341 that she was taught. Dr. Quilt also strongly believes that learning abstract material comes through practice, which is how she mastered the concepts.

Student survey responses indicated a love/hate relationship with the topics in CS 341. Students disliked the abstractness of the material, but liked the challenge of solving problems. The responses also suggested that the material at the end of the course should be given more lecture time by covering finite state machines faster. Students expressed that finite state machines were easier to learn.

Using end-of-course grades, we analyzed student performance by sub-dividing students into four categories: ultra, high, average, and low performers. Using these categories, we took a holistic view of the survey data to find specific characteristics that could predict performance. Attendance was important for success. We also found that past performance in prerequisite theory courses was a statistically significant indicator for success in CS 341.

This work concludes with a summary of suggestions for Automata Theory instructors aimed at improving student learning experiences and success.

Table of Contents

Acknowledgments **v**

Abstract **xi**

List of Tables **xix**

List of Figures **xxi**

Chapter 1 Introduction **1**

1.1 Automata Theory in Computer Science curricula 3

1.2 History of Automata Theory at The University of Texas at Austin 4

1.3 Problems with Automata Theory 5

1.4 Purpose of detailed case study 7

1.5 Brief introduction to three formalisms in Automata Theory 9

1.6 Outline of dissertation 13

Chapter 2 Literature Review and Related Work **15**

2.1 Student perceptions on learning 15

2.2 Instructor perceptions on teaching 16

2.3 Motivation of students 20

2.4 Intellectual maturity 22

2.5 Learning styles 26

2.6 Computer Science curriculum and structure 30
2.7 Teaching Automata Theory 32
2.8 Lecture-style teaching 34
2.8.1 Effective material use 39
2.8.2 Defining excellence in instruction 43
2.9 Theoretical models found in Science and Education 46

Chapter 3 Methodology **47**
3.1 Pedagogical Positivism 47
3.2 Research questions 51
3.3 Data-gathering techniques 55
3.4 Specific data gathered 58
3.5 Data analysis and validation techniques 60
3.6 Limitations of the study 63

Chapter 4 Findings **65**
4.1 General overview of CS 341 66
4.1.1 Classroom descriptions 66
4.1.2 Lecture content 67
4.2 Interviews 70
4.2.1 First interview details 71
4.2.2 Summary: First Interview 86
4.2.3 Second interview details 87
4.2.4 Summary: Second interview 95
4.2.5 Third interview details 96
4.2.6 Summary: Third interview 102
4.3 Surveys 103
4.3.1 First survey details 104

4.3.2 Summary: First survey 125
4.3.3 Second survey details 125
4.3.4 Summary: Second survey 140
4.3.5 Third survey details 142
4.3.6 Summary: Third survey 158
4.4 Classroom observations 159
4.4.1 Attendance during lecture 160
4.4.2 Student-instructor interactions 161
4.4.3 Pace and coverage of lectures 163
4.4.4 Voluntary workload 166
4.4.5 Required workload 167
4.4.6 Examinations 169
4.4.7 Summary: Classroom observations 170
4.5 Material analysis 171
4.5.1 CS 341 course packet 171
4.5.2 Summary: Material analysis 173
4.6 Student performance typing 174
4.6.1 Population characteristics and academic background 175
4.6.2 Performance on and reactions to the examinations 176
4.6.3 Student learning acquisition and class pace perceptions 178
4.6.4 Out-of-class learning 180
4.6.5 Student involvement in lecture 184
4.6.6 Summary: Student performance typing 186
4.7 Answers to Research Questions 187

Chapter 5 Discussion, Conclusions, and Implications 191

5.1 Effective practices 192
5.1.1 Use of overview lectures 192

5.1.2 Use of example problems in lecture 193
5.1.3 Use of solution explanations in lecture 193
5.1.4 Use of questions in lecture 194
5.1.5 Use of reasonable student learning objectives 195
5.1.6 Use of material for assumed background knowledge 195
5.1.7 Inclusion of programming projects 196
5.1.8 Use of course webpage 197
5.1.9 Making grading algorithms known to students 197
5.1.10 Use of student lecture notes 197
5.2 Creating a more effective learning environment 198
5.2.1 Designing materials 198
5.2.2 Motivating good study habits 206
5.2.3 Creating a better lecture environment 209
5.2.4 Soliciting student feedback 213
5.3 Recommendations for Future Research 214
5.3.1 Contribution of pedagogical positivism 214
5.3.2 Discussion of research methodology 214
5.3.3 Future work 219
5.4 Conclusions 223

Appendix A Glossary of Terms **226**
A.1 Computer Science terms 226
A.2 Education terms 233

Appendix B Course Description **236**
B.1 Course Description 236
B.2 Class information 237

Appendix C Course topics outline **240**

Appendix D Sample Test and Homework Questions **243**

D.1 Sample test questions 243

D.2 Sample homework questions 245

Appendix E Programming project descriptions **248**

E.1 First programming project: A finite state machine simulator 248

E.2 Second programming assignment: Using lex and yacc 251

Appendix F Instructor Interview Questions **261**

F.1 First interview questions 261

F.2 Second interview questions 263

F.3 Third interview questions 264

Appendix G Student survey content **266**

G.1 First survey 267

G.1.1 Samples of online formatting for first survey 272

G.2 Second survey 273

G.2.1 Samples of online formatting for second survey 279

G.3 Third survey 280

G.3.1 Samples of online formatting for third survey 286

Appendix H Sample data from classroom observations **288**

H.1 Sample field notes 288

H.2 Sample of compiled spreadsheet information 291

Appendix I Post-hoc SIGCSE member survey **293**

Bibliography **297**

Vita **311**

List of Tables

1.1 Comparison of passing rates between LIN 340 and CS 341. 6

4.1 Timeline showing introduction of topics in each lecture. 68
4.2 Summary of survey distribution. 104
4.3 Summary of survey response rates. 104
4.4 Students' thoughts on how studying with other people could benefit learning in CS 341 (based on 65 responses). 110
4.5 Students' thoughts on how studying with other people could help them cope with the material in CS 341 (based on 67 responses). 111
4.6 Students' grading preferences (based on 104 responses). 113
4.7 Students' comments about the usefulness of the practice material provided in the course packet (based on 102 responses). 119
4.8 Summary of performance categories used for student typing. 175
4.9 Demographics and academic background of students by performance category. 176
4.10 Survey-reported perceptions on examinations by performance category. . . 177
4.11 Perceptions on course grade by performance category. 178
4.12 Performance on survey items related to course material by performance category. Table reports the percentage of students answering each item correctly.179

4.13 Students' views of pacing and course difficulty over the semester by performance category. 180
4.14 Completion of voluntary reading assignments by performance category . . 181
4.15 Completion of voluntary homework sets by performance category. 182
4.16 Students' views of course-specific learning structures by performance category. 183
4.17 Students' learning preferences by performance category. 184
4.18 Participation patterns by performance category. 185
4.19 Students' reported interest in subject matter by performance category. . . . 186

List of Figures

1.1 Simplified coke machine function described using a finite state machine. . . 10

1.2 A push-down automaton that recognizes non-empty strings in which the number of *a*'s match the number of *b*'s and *a*'s precede *b*'s.The notational standard "x/y/z" means that the current input symbol is "x", a "y" should be popped from the stack, and a "z" should be pushed onto the stack. 11

1.3 A Turing machine that takes as input an arbitrary equation $A + B = C$, and then checks for accuracy. In the figure, Calculate B and Calculate C have the same expansion as Calculate A except for the number under consideration. 12

3.1 A model of the three components underlying student and instructor perceptions in CS 341. 50

4.1 Sample of histogram shown to students to allow for comparison of outcomes on examinations. 85

4.2 The non-deterministic finite state machine presented on the second survey to test recognition of non-deterministic functions. 127

4.3 Percentage of student-reported completion on voluntary homeworks. 128

4.4 Section 1 attendance by week. (Missing data for week 4.) 161

4.5 Section 2 attendance by week. (Missing data for weeks 6 and 14.) 162

4.6 Number of student-instructor interactions in Section 1 by week. 163

4.7 Number of student-instructor interactions in Section 2 by week. 164
4.8 Lecture slide count by week. 165
4.9 Lecture example count by week. 166
4.10 Number of voluntary problems assigned each week. 167
4.11 Number of pages in textbook assigned each week. 168
4.12 Number of pages in supplementary reading section of the course packet assigned each week. 169

5.1 Scan of a page from the lecture slide portion of the course packet after modification. 200
5.2 Scan of a page from the lecture slide portion of the course packet before modification. 201
5.3 An example of a scaffolded solution. 205
5.4 An example of a hint solution. 205
5.5 A modified version of the model first presented in Chapter 3 after the conclusion of the CS 341 case study. 218

G.1 Formatting of online survey introduction. 267
G.2 Sample one of online formatting for first survey. 272
G.3 Sample two of online formatting for first survey. 273
G.4 Sample one of online formatting for second survey. 279
G.5 Sample two of online formatting for second survey. 280
G.6 Sample one of online formatting for third survey. 286
G.7 Sample two of online formatting for third survey. 287

H.1 Spreadsheet tracking slides covered in each lecture of CS 341. 291
H.2 Spreadsheet tracking student participation in first section of CS 341. 292

Chapter 1

Introduction

This dissertation identifies and describes student and instructor perceptions that contribute to effective teaching and learning of Automata Theory in a competitive college-level Computer Science program. Effective teaching is the ability to create an appropriate learning environment in order to provide effective learning. We define effective learning as the ability of a student to meet instructor-set learning objectives, demonstrating this by passing the course, while reporting a good learning experience.

We conducted our investigation through a detailed case study of CS 341 - Automata Theory at The University of Texas at Austin. Because Automata Theory has a fixed curriculum in the sense that many curricula and textbooks agree on what Automata Theory contains [24], differences being depth and amount of material to cover in a single course, a case study would allow for generalizable findings. An understanding of what students perceive, what instructors expect of students, and how those perceptions play out in the classroom in terms of structure and instruction may lead to a better designed course and thus a higher student success rate in Automata Theory.

Automata Theory may be especially problematic in a Computer Science curriculum since students are not experienced in abstract thinking before taking this course, fail to understand the relevance of the theory, and prefer classes with more concrete activities such as programming. This creates a special challenge for any instructor of Automata Theory as motivation becomes critical for student learning. Another challenge is convincing students who view Computer Science as coding to do paper and pencil homework. Practice is necessary for mastery of the material taught in Automata Theory. Without sufficient incentive to do practice problems outside of class, students will tend to devote their time to other concerns.

This work examines these and other issues and offers general solutions based on findings from the case study of CS 341. To develop a context for this research, we will define Automata Theory, describe its role in Computer Science, give a brief history of Automata Theory at The University of Texas at Austin, and present some basic formalisms that are introduced in Automata Theory.

Automata theory is the study of abstract computing machines "which follow a preset sequence of instructions automatically" [33]. In layman's terms, this means that Automata Theory studies computability: what is or is not computable and the theoretical complexity of problems that are computable. Students of Automata Theory learn how to classify computational problems using formal notation, understand the hierarchy of solvability of computational problems, and generate hypotheses about computations formally. These hypotheses require logic and formalisms introduced during the course. *Formalisms* are theoretical constructs that are used to show different classifications of computation. The three main formalisms are *finite state machines*, *push-down automata*, and *Turing machines*, all of which are described briefly later in this chapter. In addition to these formalisms, students are introduced to the idea of reduction proofs and complexity classes. These two concepts are useful in general programming and advanced theoretical courses in Computer Science. Another goal of the course is to have students become comfortable with formal notation since it plays a large role throughout Computer Science. Formal notations are used as a language in which program specifications and requirements can be stated clearly and unambiguously.

1.1 Automata Theory in Computer Science curricula

Automata Theory is the study of computation through abstractions. As such Automata Theory is the basis on which Computer Science is built. It is therefore important that any Computer Science curriculum include at least some of the topics from Automata Theory. This theoretical basis is the foundation that makes the study of computation a science, differentiating it from engineering or programming as an area of study. The authoritative source on computing curricula is the Computing Curriculum published in 2001 (CC 2001) [1] by the joint committee of the Institute of Electrical and Electronics Engineers–Computer Society (IEEE-CS) and Association for Computing Machinery (ACM). IEEE-CS and ACM are the leading organizations in Computer Science. Many top conferences are under their umbrella. CC 2001 states that Computer Science graduates should have the following ability:

> Appreciation of the interplay between theory and practice. A fundamental aspect of computer science is the balance between theory and practice and the essential link between them. Graduates of a computer science program must understand not only the theoretical underpinnings of the discipline but also how that theory influences practice. [1]

Automata Theory is one course that could be used to fulfill this requirement and thus a good candidate for undergraduate degrees in Computer Science (CS).

Automata Theory generally consists of a single upper-division course in the undergraduate CS curriculum (SIGCSE member survey) although many of the topics could easily be incorporated into lower-division courses or courses earlier in the degree program [58, 60, 101]. The historical placement of Automata Theory as a upper-division course is influenced by the desire for students to have a strong mathematical background, some experience with basic logic and proof techniques, and an introduction to programming [29]. The course itself serves as a prerequisite for advanced theory courses in computing and formal methods. It is also a valuable prerequisite to compiler design.

Many of the concepts taught in Automata Theory are necessary for graduate courses in computing. Automata theory topics are also present on the Computer Science Graduate Record Examination (GRE), a topic-specific examination required for admittance in many Computer Science graduate degree programs. A graduate student with no exposure to Automata Theory would most certainly have a difficult time completing graduate-level theory courses.

1.2 History of Automata Theory at The University of Texas at Austin

When Computer Science became a degree program at The University of Texas at Austin (UT) it was only offered with a Bachelor of Arts option. Automata Theory was not originally included among the subjects taught for the degree. Circa 1982 the Computer Sciences Department at UT decided to include more theory in the degree requirements. Automata Theory was chosen as one of the theory classes to be integrated into the degree program. However, a Bachelor of Arts degree at UT required only 120 hours, not all of which could be dedicated to the major area of study. For any Bachelor of Arts option, UT set a limit on the number of courses in any one department that could be counted toward the degree. With the limited number of CS hours in a Bachelor of Arts degree and the knowledge that Automata Theory should be a required area of study, the CS Department decided that Automata Theory should be required of their students, but offered through the Linguistics Department, as LIN 340. This made sense for two reasons: (1) Historically, linguists developed many of the formalisms taught in Automata Theory, and (2) the Linguistics Department included several experts in formal language theory who wanted to teach the course.

A drawback to out-sourcing the teaching of Automata Theory to the Linguistics Department was that the CS Department had limited influence over the content that was taught and no influence over the passing requirements for the course. Although the course was taught by both Linguistics and CS faculty, the passing rate was dictated by the Linguistics Department's standards. These standards tended to pass students with a weak understanding of the material. Another hurdle that arose over time was the movement of instructors. The year after the LIN 340 requirement was put into place, two of the three faculty members in formal language theory left the University. The one remaining instructor taught the course consistently almost every semester until his retirement. When the Linguistics Department announced the retirement of this last LIN 340 instructor, the Computer Sciences Department decided to re-adopt the course. The re-adoption process took over a year to implement. The limitation of hours that could be applied to CS courses under a Bachelor of Arts degree was no longer necessary since the CS Department had created a Bachelor of Science option. By re-adopting the course, the CS Department regained control over passing standards as well as curriculum coverage. The passing standards in the CS Department did not allow students with a weak understanding of the material to pass the course. The course, CS 341 - Automata Theory, was first taught in Fall 2000, replacing LIN 340.

The primary instructor of this "new" course was Dr. Lily Quilt, who had taught LIN 340 for several years. Her continued teaching was one reason behind the smooth transition during the re-adoption. In fact, most of the materials Dr. Quilt used in her CS 341 course were unchanged from when she taught LIN 340. Even with well developed materials and experience in teaching Automata Theory, Dr. Quilt faced challenges in teaching CS 341. Students were not well prepared by the prerequisite theory courses for studying the material in Automata Theory. Additionally, end-of-course surveys indicated that students did not like the course.

1.3 Problems with Automata Theory

During the last two years LIN 340 existed 864 students completed the course. Because the class was restricted upon registration and required of all CS majors, the majority of the students taking this course were CS majors. Of those students taking the course, only

Course	Time	# students	“C” or better	repeating
LIN 340	Fall 1998 – Spring 2000	864	75%	25%
CS 341 (all sections)	Fall 2000 – Spring 2002	718	68%	32%
Dr. Quilt’s CS 341	Fall 2000 – Spring 2002	502	66%	34%

Table 1.1: Comparison of passing rates between LIN 340 and CS 341.

25% did not receive a grade of “C” or better. A grade of a “C” or better was required to have the course count toward degree requirements [36]. In the last two years during which Automata Theory was taught as CS 341, the repeat rate (those students not receiving a “C” or better) rose from 25% to 32% (see Table 1.1). This rise may have been due to departmental passing standards, Computer Science centric teaching staff, or the large gain in popularity of pursuing a Computer Science degree which brought with it many unprepared students.

Due to the high repeat rate, CS 341 gained a reputation among students and advisors for being the last “weed-out course”. This weed-out reputation was due to several factors, one being that CS Department policy states that courses can only be taken twice, and if not passed on the second attempt, the student is not allowed to continue in the CS degree plan [36]. As LIN 340, the course was not subject to this “two strike rule” and students could repeat the course as many times as necessary, which affected their GPA but not their eventual completion of a CS degree.

Another factor that leads many students to dislike this course is that it is abstract. Based on student survey data gathered during this work, CS students may be attracted to the field because they want to program, design systems such as video games, or think that having a degree in CS will make them stronger contenders in the job market. Students with such a focus may find the topics in Automata Theory irrelevant for Computer Science. Findings in Computer Science Education also suggest that undergraduate students are not well equipped to think abstractly [4, 29]. If given the choice, these students would probably not elect to take CS 341. The fact that Automata Theory is required for the degree can be a great frustration to such students.

What these students do not understand is that Automata Theory contains a core set of ideas that will persist in Computer Science. Programming languages will change, technology will evolve, but the underlying theory will never age [101]. It is one of the few stable subjects in Computer Science. Another advantage to taking Automata Theory is that the material gives students practice in how to break down complex problems into simple components as well as using logic to communicate. This type of problem solving is useful when developing large programs, whether the task be coding or design. Students are often unaware of this advantage since the problems they encounter in class are generally either superficially generated or already broken into pieces. During a single semester students may not have time to use the concepts in other courses or time to reflect about the usefulness of the theory in the Computer Science field.

This student perception of Automata Theory as useless for Computer Science pursuits is not unique to UT. In a poll of Special Interest Group in Computer Science Education (SIGCSE) members, 38% of the respondents stated that the students attending their institutions question the utility of the course in the study of Computer Science. (Appendix I details the results of this informal survey.) In essence instructors of Automata Theory are teaching unmotivated learners. When students view a course as nothing more than a hurdle to overcome, it creates a setting conducive to non-successful student performance and learning. These observations motivated this investigation into the workings of CS 341.

1.4 Purpose of detailed case study

The apparent trouble of teaching and learning CS 341 - Automata Theory at UT inspired the detailed case study of this course. We were interested in understanding student perceptions of learning this material, as well as the instructors' perceptions of their role in teaching the course and how both these perceptions play out in the classroom.

Two aspects of the Automata Theory course contribute to making it a worthy target for this research.

(1) The topics covered in Automata Theory have been agreed upon in the CS community for about 30 years (first interview with Dr. Quilt). Many textbooks on the subject have a similar outline of topics, varying only in depth of coverage and how much material after Turing machines are included [63, 67, 70, 74, 79]. This "fixed" curriculum implies that observed difficulties in teaching and learning Automata Theory at UT are probably not unique, and that any improvements found for CS 341 might apply to other college-level courses focusing on the same topics.

(2) Automata Theory is very different from other Computer Science courses. Student activities for the course do not necessarily include programming; instead paper and pencil homework is often required. Students in Computer Science may not be accustomed to this type of work. In addition, paper and pencil homework often has slow turn-around due to the time needed for grading, so feedback is not immediate. Programming courses offer immediate student feedback via computer use [97]. When students take CS 341 during the same semester they take other programming-intensive classes, students may focus their energies on programming projects as they are more familiar with this type of work and feedback and progress is immediately observable.

An understanding of how students perceive their learning of topics in this course, combined with an understanding of the instructor's goals, may lead to strategies that would help inform curriculum designers and instructors of Automata Theory. In order to recognize good practices and suggest improvements based on our case study of CS 341 we need to answer the following questions:

1. How do instructor perceptions shape teaching and learning in CS 341?

2. How do student perceptions shape teaching and learning in CS 341?

3. How do the materials used in CS 341 help or hinder teaching and learning?

In summary, this work focuses on understanding students' perceptions on learning the topics presented in CS 341, instructors' perceptions of and goals for teaching Automata Theory, and how those perceptions play out in the classroom.

1.5 Brief introduction to three formalisms in Automata Theory

The three basic formalisms introduced in CS 341 are: *finite state machines*, *push-down automata*, and *Turing machines* ([67] and classroom observation notes). We describe these formalisms briefly since an understanding of these concepts is necessary in order to interpret the findings of this work. The reader does not need to be familiar with computational theory to understand this presentation as the concepts are described at a high level.

The first and most basic formalism is the *finite state machine*. A finite state machine can be seen as a finite directed graph. Each node in the graph represents some internal state of a system. Each arc of the graph represents a computational step. The "start state", or initial state of the system is denoted by two lines that come together in an upside-down "v" shape. Final states, or states that signify successful completion of the computation are denoted by two concentric circles. Many interesting problems require only this basic formalism to describe their function. One area of computation that is nicely described by finite state machines is specific pattern recognition. For instance, a coke machine must recognize when it has received enough money to dispense a soda. Figure 1.1 shows how a simplified coke machine can be described using a finite state machine.

Finite state machines, although useful, are too simple to describe arbitrary counting. For instance, suppose a given string *aaabbb* must be checked to make sure that the number of *a*'s and *b*'s match. If only one such string is given, then it would be simple to construct a finite state machine similar to the coke machine example. If the goal was expanded to check whether arbitrary strings have a matching number of *a*'s and *b*'s, with the constraint that all *a*'s must precede *b*'s, a finite state machine could not be constructed for this task because this would require an infinite graph to map all the possibilities. Instead this type

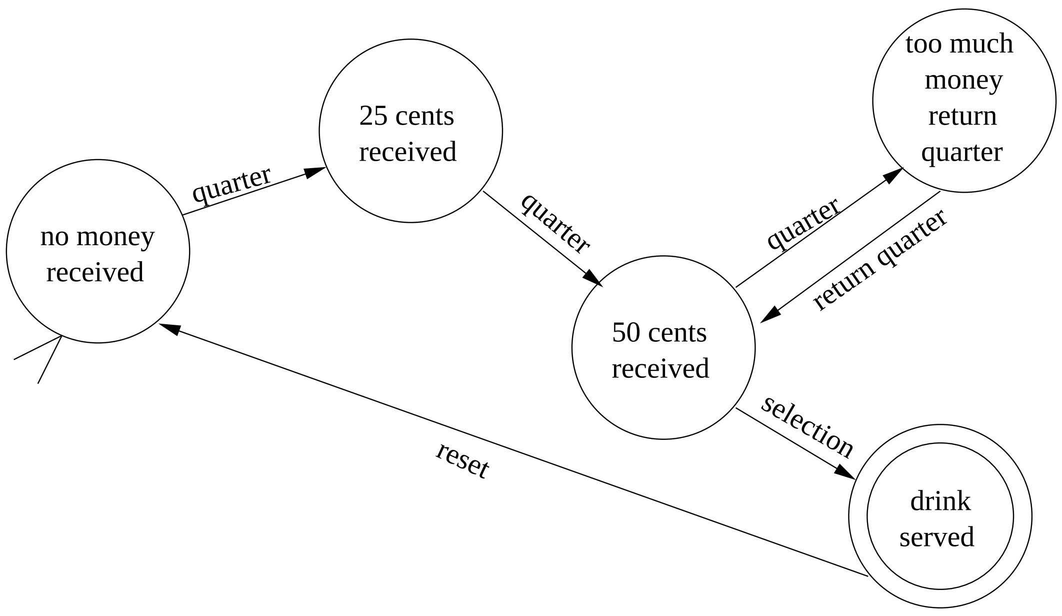


Figure 1.1: Simplified coke machine function described using a finite state machine.

of problem requires a more powerful formalism called a *push-down automaton*. A push-down automaton has a finite directed graph as before, with the addition of a stack to help its computation. The stack is used to "count" necessary elements, such as the number of *a*'s encountered, by pushing each *a* onto the stack. When a *b* is encountered, an *a* that was previously pushed can be popped from the stack. In this way, at the end of the string, if the stack is empty and in processing the string no ordering violations occurred, then the string has met all necessary requirements. Figure 1.2 is a pictorial representation of this push-down automaton.

Although the matching *a*'s and *b*'s string example is simple, the formalism is quite powerful. Push-down automata are necessary for compiler construction and complex pattern recognition. However, push-down automata are not powerful enough to construct all functions, e.g. a push-down automata cannot check that the answer "*C*" of $A + B = C$ for an arbitrary *A* and *B* is correct. For this type of problem we need the third formalism

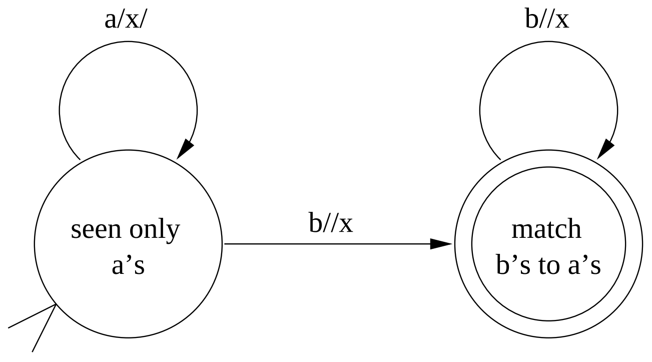


Figure 1.2: A push-down automaton that recognizes non-empty strings in which the number of *a*'s match the number of *b*'s and *a*'s precede *b*'s.The notational standard "x/y/z" means that the current input symbol is "x", a "y" should be popped from the stack, and a "z" should be pushed onto the stack.

introduced in most Automata Theory courses, the *Turing machine*. A Turing machine can be thought of as a push-down automaton with two stacks. The two stacks are used to juggle information for an ongoing computation. There is an equivalent definition in which the two stacks are replaced by two infinite tapes with a single read head on each tape. Thus, a Turing machine has a finite directed graph that keeps track of internal state and two infinite tapes on which intermediate computations can be written. With a Turing machine, it is possible to check $A + B = C$. For our discussion, we consider *A*, *B*, and *C* to be natural, base 10 numbers.

The Turing machine is initialized with the string $A + B = C$ on the first tape. The first tape is scanned from left to right. When the scanning process encounters the first number *A*, it transposes *A* into a unary representation on the second tape. When the $+$ is encountered, the machine temporarily ignores the character. The second number *B* is transposed into unary representation on the second tape. The $=$ character is copied onto the second tape without any modification. When the last number *C* is encountered, it is transposed into unary on the second tape directly after the $=$ character. The second tape now contains a unary number followed by $=$ followed by a unary number. It is now a simple task to make sure that the number of unary digits on either side of the $=$ character match. If they do, then the equation is correct. If the unary strings do not match, the equation was incorrect. Figure 1.3 shows a Turing machine that checks correctly formatted strings of the form $A + B = C$.

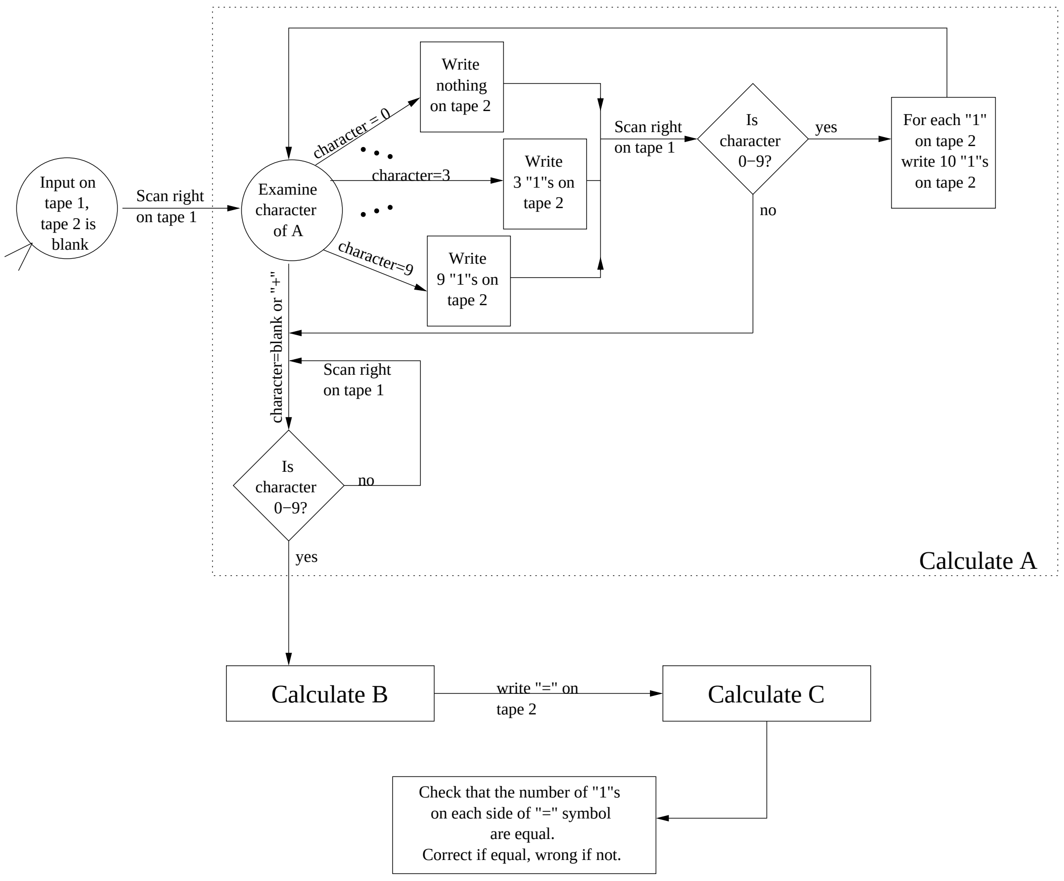


Figure 1.3: A Turing machine that takes as input an arbitrary equation $A+B = C$, and then checks for accuracy. In the figure, Calculate B and Calculate C have the same expansion as Calculate A except for the number under consideration.

It is counter-intuitive that these three simple formalisms, finite state machines, push-down automata, and Turing machines, can describe all possible computations and yet this is true. In fact, Alan Turing created the Turing machine formalism in 1936, preceding the existence of modern computing machines [61]. The power of these formalisms is the key insight that all students of Automata Theory must come to understand. Once they have gained this insight, students may better appreciate the role of Automata Theory in Computer Science.

1.6 Outline of dissertation

The remainder of this document is structured as follows: In Chapter 2 the literature that influenced our study of Automata Theory is summarized. The primary focus in on past ideas in teaching Automata Theory as well as general teaching techniques for lecture style classes and college-level computer science/engineering courses. We also present previous work that was based on instructor and student perceptions.

Chapter 3 explains the methodology used for this research. Our theoretical perspective is based upon pedagogical positivism, a term we created to combine constructivism and positivism in a teaching environment. This model provided the foundation for our research questions and the basis for testing these questions through qualitative techniques. The data collection techniques used in this research were surveys, interviews, and classroom observations. The theoretical perspective, rationale for techniques used, and explanation of the specific data we gathered are described in detail.

In Chapter 4 we describe the findings that emerged from analyzing the data. Each interview is individually presented in a narrative style. The student surveys are presented next, one at a time, with reference to specific interview commentary as appropriate. We intertwine the survey and interview findings with an analysis of classroom observation data. Lastly we present our analysis of the course material and present an ethnographic-styled student performance typing: a reporting style that looks at a specific population over time.

Chapter 5 is dedicated to creating sense out of the facts and high level analysis presented in Chapter 4. In this last chapter we discuss the aspects of our case study that will create a better and more productive learning environment for students. We conclude by suggesting future research and summarizing the implications of our work.

Several appendices are included to help readers understand this document, our methodology, and our findings. Appendix A provides a glossary of common CS and Education jargon used in this dissertation. Appendices B and C contains the course description and timeline of discussion topics during CS 341 provided to students via the course webpage the semester we studied Automata Theory. Appendix D includes sample test and homework question given to students, and Appendix E details the two programming projects that were assigned during the study of CS 341. Appendices F, G, and H contain the interview questions, survey items, and samples of observation data used for our case study. Appendix I describes the post-hoc SIGCSE survey items and results.

Chapter 2

Literature Review and Related Work

With most research, an original study or idea is influenced by what has come before. Many fields of educational research and current practices were examined for this work. Each section of this chapter addresses a specific conceptual unit. The conceptual units are (in order): student perceptions on learning, instructor perceptions on teaching, motivation of students, intellectual maturity, learning styles, Computer Science curriculum and structure, teaching Automata Theory, and lecture-style teaching. We conclude this chapter by presenting theoretical models found in Science and Education to lay the groundwork for introducing the theoretical basis of our research, pedagogical positivism, which is presented in Chapter 3.

2.1 Student perceptions on learning

In the book *Talk About Leaving: Why Undergraduates Leave the Sciences* [100], students in Engineering, Science, and Mathematics were asked to describe the factors that contributed to their staying in or leaving these majors. In this four-year, multi-institutional study, 794 undergraduates were asked to describe their learning experiences and how those experiences influenced whether they persisted in or left Engineering, Science, and Mathematics. A striking aspect of the book was the in-depth insights that were possible due to the inclusion of the students' accounts of their learning experiences. Although most of the commentary presented in the work was drawn from transcripts of focus group discussions, the power of the students' own comments contributed to a compelling final tale. To capture this same richness of data, the student surveys designed for our study included several open-response items.

A second book, *Unlocking the Clubhouse* [77], focused on women's experiences in Computer Sciences. Qualitative data gathered from over 230 interviews of men and women students were used to investigate factors that retained or repelled women in Computer Science. This four-year study tracked changes in perceptions over time through repetitive interviews of undergraduates, informal interviews with graduate students and faculty, classroom observations, surveys of the entire incoming classes from 1995–1997, online bulletin board discussions, student journals, and a focus group. The multiple sources of data contributed to the exceptionally strong conclusions of this work. The implications from the study were used to modify the learning environment in the Computer Science Department at Carnegie Mellon University. The changes led to better success in attracting and retaining women in the department and the authors suggested that other universities could also benefit from implementing similar changes in their departments. These works influenced the inclusion of multiple sources of data for triangulation as well as the importance of student responses in creating suggestions for better learning environments.

2.2 Instructor perceptions on teaching

In many ways, instructors are responsible for creating the environment and rules by which students learn. Polman [86], studied a high school science class by investigating an instructor's view of teaching and how those perceptions played out in classroom activities. The majority of Polman's work took place over a 6 month period in which he observed classroom activities on a daily basis. Polman's understanding of the instructor of the High School science class had come through prior interviews and working as an aide in the same class prior to the observation period. Polman's primary focus was to understand how the teacher's work in the classroom helped or hindered high school students in successfully completing an inquiry-based project. Though this focus, Polman was able to elucidate instructor activities that would help other educators successfully include this type of activity in their classes. Polman's work inspired the instructor focus of this study as well as influenced the importance of instructor activities in suggesting improvements for learning in Automata Theory.

Studies have shown that professed instructional goals are not always carried out in the classroom [73, 102]. Lederman studied five high school biology teachers "to investigate the relationship of teachers' understanding of the nature of science and classroom practice and to delineate factors that facilitate or impede a relationship" [73]. In using classroom observations, open-ended questionnaires, interviews, and instructional artifacts he came to the conclusion that "teachers' conception of science do not necessarily influence classroom practice" [73]. What was critical in influencing classroom activities was "teachers' level of experience, intentions, and perceptions of students" [73]. A study by Simmons [102] produced similar results. In her 3-year exploratory study of 69 pre-service secondary science teachers, interviews and video portfolios were used to contrast classroom practice with reported teaching beliefs. Her study found that "observations of...teaching practices contrasted starkly with teacher beliefs" [102]. These works influenced the inclusion of both instructor interview and classroom observations in our study.

Bryan [25] supported the need for multiple sources of data in order to understand teacher beliefs. Bryan's study was a year-long case study of a senior elementary education major. During her study, Bryan collected data from non-participant observations, copies of work turned in during a method-course her participant was taking, and 6 semi-structured interviews. A key finding from her case study was the role that previous learning experiences played in forming beliefs about teaching.

In a literature review, Cochran [32] explored the nature of pedagogical content knowledge with learning environments. In her review, Cochran suggested that a teacher's pedagogical content knowledge, what is known about teaching applied to specific content, is important. She expressed that pedagogical content knowledge is what creates science teachers and not just scientists. She pointed out that "An experienced science teacher's knowledge of science is organized from a teaching perspective and is used as a basis for helping students to understand specific concepts" [32]. It is this knowledge that influences the effectiveness of teaching.

Instructor expectations of students also influence learning environments. In a often cited study from 1981, Anyon [6] investigated how high school teachers could influence classroom dynamics and student perceptions simply through what they felt students were capable of learning. In her work, Anyon examined teacher and institutional expectations of students at four elementary schools. Data was gathered from classroom observations; interviews of students, teachers, principals, and administrative staff; and analysis of classroom materials. Her conclusion was that attitudes of the teachers and institutions affected student self-efficacy and performance.

The work done by Sadker and Sadker [98] supported Anyon's findings. Sadker and Sadker showed how teachers could shut down productive learning tendencies in women by subtle instructional cues used in daily interactions with female students. In their multi-year, multi-institution study they gathered data from teacher and student interviews, as well as classroom observation. They found that in many instances teachers were unaware of how their teaching mannerisms influenced student behavior.

In addition to classroom mannerisms, epistemological beliefs about teaching have been shown to affect teaching activities in the classroom [56]. In a study of 35 science teachers Hashweh found that teachers holding constructivist beliefs were more likely to:

(a) more likely to detect student alternative conceptions;

(b) have a richer repertoire of teaching strategies;

(c) use potentially more effective teaching strategies for inducing student conceptual change;

(d) report more frequent use of effective teaching strategies; and

(e) highly valuate these teaching strategies compared with teachers holding empiricist beliefs [56].

These conclusions were based on the analysis of a three-part questionnaire addressing teacher epistemological beliefs.

Ben-Ari [14] theorized that a constructivist view supports learning in Computer Science when explicitly taught models precede abstractions. In his discussion on the role of constructivism in the Computer Science Educational realm, Ben-Ari built a case for the importance of a constructivist learning model in teaching Computer Science. He warned that under this view "models must be explicitly taught, models must be taught before abstractions, and the seductive reality of the computer must not be allowed to supplant construction of models" [14]. This work suggests the importance of inter-twining theory with concrete exercises.

Vermunt [111] combined student and instructor data to form a coherent picture of the differences between teaching and learning paradigms. In her work Vermunt noted that "teaching does not automatically lead to learning" [111]. Vermunt constructed a basis by which to compare learning and teaching activities in order to explore the congruence and friction between student-regulation and teacher-regulation of learning. The outcome was an analysis of teaching activities and the learning functions they achieve. The analysis defined three levels of instruction: strong teacher-regulation, shared-regulation, and loose teacher-regulation. Vermunt did not state which instructional level was appropriate. Instead the work was designed to be informative. Vermunt acknowledged that "there is always a task division between teachers and students in fulfilling learning functions" [111]. Vermunt's work was used to analyze Dr. Quilt's teaching and learning beliefs and activities.

We drew on several concepts from the works in this section for the focus and analysis of our data: instructors are key in developing learning environments, several sources of data should be used in uncovering instructional beliefs, and student expectations are an important influence on teaching activities. Instructors are largely responsible for creating student learning environments. Suggestions for improvements in learning may need to focus on instructor beliefs and activities. Stated beliefs and observed actions may not always align. To uncover information about beliefs and how those instructional beliefs are applied in the classroom, both interviews and classroom observations are necessary. An important aspect of instructor beliefs is their expectations of students. Instructors exhibiting constructivist beliefs are more likely to produce flexible learning environments that accommodate a larger number of students. Constructivist beliefs have been argued to be an effective epistemological basis for Computer Science instruction [14].

2.3 Motivation of students

One way in which teachers can influence students to take part in "fulfilling [their own] learning functions" [111] is through motivation. Boekaerts [21] summarized the latest research on the effect of interest in classroom learning environments. Boekaerts presented two main points in her paper: (1) the more interested students are in a subject, the more they are motivated and willing to work in order to learn, and (2) interest and intrinsic motivation are not always one and the same. This suggests that students with no intrinsic motivation to study Automata Theory could still become interested in the subject.

Another implication of Boekaerts work is that extrinsic motivators can be used to increase student interest. One such motivator that plays a central role in college courses is grades. Lin's [75] work examined five college-level courses to determine whether intrinsic or extrinsic motivation factors influenced positive student performance. Students in the five courses were surveyed using a pre-designed questionnaire. The outcome of the study was that students with medium levels of extrinsic motivation were more likely to out-perform other students with respect to end-of-course grades, suggesting that grades can play a moderate role in enhancing student performance. Lin suggests that "teachers need not eliminate all motivation for good grades in order to achieve both cognitive and lifelong learning goals" [75].

The importance of grades in student and instructor views of learning is supported by Pollio [85]. In his study, Pollio had 396 undergraduate psychology students and 157 university instructors take surveys designed to investigate the relationships between grade-oriented and learning-oriented views of education. The conclusion was that both students and instructors preferred to view themselves as learning-oriented, when in fact most were still grade-oriented. This would support the use of grades as a motivator in learning Automata Theory.

In addition to grades, well deserved verbal praise during lecture can inspire students to do more homework [54]. Hancock conducted a study of 61 West Point students to investigate the effect of well-deserved verbal praise on amount of out-of-class work. On pre-defined days chosen at random before the study began, students started class by filling out a time card indicating how much time they had spent outside of class on that course. There were two treatments in the study. In the first treatment, the instructor simply collected the time cards. In the second treatment students who indicated they had spent more than an hour studying outside of class were given verbal praise. Hancock found that students in the verbal-praise treatment scored better on the course final examination. This study suggests the importance of instructor interaction with students during class.

Another factor that enhances student motivation and performance is self-efficacy in computer usage [27, 87]. Chalupa [27] found that for 74 students in a computer applications course, there was a significant correlation between students' beliefs of self-efficacy in computer usage and their final course grade. Potosky [87] ran a similar study with 56 newly hired computer programmers to investigate the effects of self-efficacy ratings on programming capabilities.

Motivation can be improved by allowing students to choose what they study. Polman's [86] work explored the advantages of allowing students to define their own projects within teacher-set guidelines. Although it would be hard to apply this source of motivation to a lecture-based class, the idea of allowing students to choose their own programming projects had possible applications for our study.

The aspect of these works that were most influential in our research was that motivation was a natural way to get students to do out-of-class work. The motivation did not need to be intrinsic, rather extrinsic motivators such as grades and verbal praise have been shown to positively motivate students. These ideas, combined with our data, helped to form suggestions for better student motivation in Automata Theory.

2.4 Intellectual maturity

In order for a student to learn more material, he or she must first have the ability to learn at all. This requirement goes beyond motivation to an intellectual capability to process the new material. In a literature analysis, Tall [105] examined several educational theories addressing intellectual ability with regard to mathematical thinking. In his article, Tall wrote about the relevance of Piaget's theory of development [65] in understanding how students learn to think about mathematics. Piaget surmised that children underwent four stages of cognitive ability:

> The first is the sensori-motor stage prior to the development of meaningful speech, followed by a pre-operational stage when the young child realizes the permanence of objects, which continue to exist even if they are temporarily out of sight. The child then goes through a transition into the period of concrete operations where he or she can stably consider concepts which are linked to physical objects, thence passing into a period of formal operations in the early teens when the kind of hypothetical "if-then" becomes possible [105].

Piaget's theory states that in order to understand a new concept the learner must transition from an old knowledge base to a knowledge base that includes the new information.

> During such a transition, unstable behavior is possible, with the experience of previous ideas conflicting with new elements. Piaget uses the term *assimilation* to describe the process by which the individual takes in new data and *accommodation* the process by which the individual's cognitive structure must be modified. He sees assimilation and accommodation as complementary. During a transition much accommodation is required [105].

Tall used this idea to describe the process by which students are "told" new information. Instead of allowing for erratic thoughts, mathematical notions are generally fed to the student in succinct steps; steps that have been clarified and smoothed-out by an instructor. This leads to a disconnect between "intuition" and "rigor" and generally in teaching mathematics "rigor" is focused on in the classroom.

The idea of scaffolding can be effective in helping a student assimilate new ideas. "Scaffolding is an instructional technique whereby the teacher models the desired learning strategy or task, then gradually shifts responsibility to the students" [99]. The origin of scaffolding is commonly attributed to Vygotsky. In a book published in 1962, Vygotsky introduced the term "Zone of Proximal Development" (ZPD) [68]. Scaffolding is used to allow learners to reach learning goals that are within their ZPD.

> Vygotsky defined the Zone of Proximal Development as "the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under the guidance or in collaboration with more capable peers." This [sic], the ZPD is the difference between what children or students can accomplish independently and what they can achieve in conjunction or in collaboration with another, more competent person. The Zone is created in the course of social interaction [68].

The main contribution of the ZPD is that learning can be furthered by instruction.

Gagné, an instructional psychologist, examined the learning process and defined nine events that must take place in order for instruction to be effective. The nine instructional events are [80]:

1. Gaining attention. To ensure reception of coming instruction we give the learner a stimulus.

2. Tell the learners the learning objective. Tell the learner what they will be able to do because of the instruction.

3. Stimulating recall of prior learning. Ask for recall of existing relevant knowledge.

4. Presenting the stimulus. Display the content.

5. Providing learning guidance.

6. Eliciting performance. Ask the learner to respond, demonstrating learning.

7. Providing feedback. Give informative feedback on the learner's performance.

8. Assessing performance. Require more learner performance, and give feedback, to reinforce learning.

9. Enhancing retention and transfer to other contexts. Provide varied practice to generalize the capability.

According to Gagné's theory, these instructional events must be followed in a strict order. The formulation of these nine events is dependent on what type of learning is to transpire. According to Gagné, there are eight categories of learning [80]:

1. Signal Learning. The individual learns to make a general, diffuse response to a signal. Such was the classical conditioned response of Pavlov.

2. Stimulus-Response Learning. The learner acquires a precise response to a discriminated stimulus.

3. Chaining. A chain of two or more stimulus-response connections is acquired.

4. Verbal Association. The learning of chains that are verbal.

5. Discrimination Learning. The individual learns to make different identifying responses to many different stimuli that may resemble each other in physical appearance.

6. Concept Learning. The learner acquires a capability of making a common response to a class of stimuli.

7. Rule Learning. A rule is a chain of two or more concepts.

8. Problem Solving. A kind of learning that requires the internal events usually called thinking

An instructor should first identify what learning task they are attempting to teach. In order for students to learn the task, all previous levels of understanding must be established before instruction starts. In order to assess previous knowledge, both formative and summative evaluation should be used [80].

More recently, Lawson, Alkhoury, Benford, Clark, and Falconer [72] explored the intellectual development of college students. In testing 663 non-science majors enrolled in a biology course, it was found that students did indeed display different intellectual levels, suggesting that "intellectual development continues beyond the 'formal' stage [Piaget] during the college years, at least for some students" [72]. Lawson, et. al., found that three types of scientific concepts existed: descriptive, hypothetical, and theoretical. The three levels are of increasing difficulty for students at all developmental levels. These findings imply that "college science instructors should not only concern themselves with introducing new terms/concepts... they should also concern themselves with student thinking abilities, that is, with students' continued development" [72]. Additionally, courses should "start with descriptive concepts and progress to hypothetical concepts, and then to theoretical concepts...Such changes may not only help students better understand concepts and promote their intellectual development, they might also help solve the widespread program of college student attrition from the sciences" [72]. The advice about sequencing instructional activities to better facilitate student learning was used in this study to situate Dr. Quilt's observed teaching activities.

2.5 Learning styles

In addition to mental maturity, "students have different learning styles—characteristic strengths and preferences in the ways they take in and process information" [46]. There are many learning style questionnaires and tests used to help students understand what styles suit them and how those preferences can be used to build better personal learning strategies. In a survey of learning styles, Felder [46] describes four of these models and how they relate to Engineering education. The four models presented are: Myers-Briggs Type Indicator (MBTI), Kolb's Learning Style Model (Kolb), Herrmann Brain Dominance Instrument (HBDI), and the Felder-Silverman Learning Style Model (F-S).

The MBTI is based on Carl Jung's theory of psychological types. To complete the MBTI, a student answers a series of questions that classify them into one of 16 possible categories [81]. Using this model, students may be [46]:

- extroverts (try things out, focus on the outer world of people) or introverts (think things through, focus on the inner world of ideas)
- sensors (practical, detail-oriented, focus on facts and procedures) or intuitors (imaginative, concept-oriented, focus on meanings and possibilities)
- thinkers (skeptical, tend to make decisions based on logic and rules) or feelers (appreciative, tend to make decisions based on personal and humanistic considerations)

- judgers (set and follow agendas, seek closure even with incomplete data) or perceivers (adapt to changing circumstances, resist closure to obtain more data)

Felder notes that:

> Engineering professors usually orient their courses toward introverts (by presenting lectures and requiring individual assignments rather than emphasizing active class involvement and cooperative learning), intuitors (by focusing on engineering science rather than design and operations), thinkers (by stressing abstract analysis and neglecting interpersonal considerations), and judgers (by concentrating on following the syllabus and meeting assignment deadlines rather than on exploring ideas and solving problems creatively) [46].

The Kolb learning style is similar in structure to the MBTI. Again students answer a series of multiple-choice questions to be grouped into a specific category. However, where MBTI had 16 categories into which a student might be placed, the Kolb has only four categories. The category definitions for the Kolb are [46]:

- Type 1 (concrete, reflective). A characteristic question of this learning type is "Why?" Type 1 learners respond well to explanations of how course material relates to their experience, their interests, and their future careers. To be effective with Type 1 students, the instructor should function as a motivator.

- Type 2 (abstract, reflective). A characteristic question of this learning type is "What?" Type 2 learners respond to information presented in an organized, logical fashion and benefit if they have time for reflection. To be effective, the instructor should function as an expert.

- Type 3 (abstract, active). A characteristic question of this learning type is "How?" Type 3 learners respond to having opportunities to work actively on well-defined tasks and to learn by trial-and-error in an environment that allows them to fail safely.

To be effective, the instructor should function as a coach, providing guided practice and feedback.

- Type 4 (concrete, active). A characteristic question of this learning type is "What if?" Type 4 learners like applying course material in new situations to solve real problems. To be effective, the instructor should stay out of the way, maximizing opportunities for the students to discover things for themselves.

With regard to the Kolb and instruction, Felder notes:

> Traditional engineering instruction focuses almost exclusively on formal presentation of material (lecturing), a style comfortable for only Type 2 learners. To reach all types of learners, a professor should explain the relevance of each new topic (Type 1), present the basic information and methods associated with the topic (Type 2), provide opportunities for practice in the methods (Type 3), and encourage exploration of applications (Type 4). The term "teaching around the cycle" was originally coined to describe this instructional approach [46]

The next learning style model discussed by Felder is the HBDI. In this model, students are put into one of four categories dependent on "their relative preferences for thinking in four different modes" [46]. The categories have names "based on the task-specialized functioning of the physical brain" and are [46]:

- Quadrant A (left brain, cerebral): Logical, analytical, quantitative, factual, critical;
- Quadrant B (left brain, limbic): Sequential, organized, planned, detailed, structured;
- Quadrant C (right brain, limbic): Emotional, interpersonal, sensory, kinesthetic, symbolic;
- Quadrant D (right brain, cerebral): Visual, holistic, innovative.

Felder states that Engineering professors, on average, are more aligned with Quadrant A [46].

The last model presented in Felder's survey of learning style indicators is the Felder-Silverman. The model has five categories [46]:

- sensing learners (concrete, practical, oriented toward facts and procedures) or intuitive learners (conceptual, innovative, oriented toward theories and meanings);
- visual learners (prefer visual representations of presented material–pictures, diagrams, flow charts) or verbal learners (prefer written and spoken explanations);
- inductive learners (prefer presentations that proceed from the specific to the general) or deductive learners (prefer presentations that go from the general to the specific);
- active learners (learn by trying things out, working with others) or reflective learners (learn by thinking things through, working alone);
- sequential learners (linear, orderly, learn in small incremental steps) or global learners (holistic, systems thinkers, learn in large leaps).

In order to address all types of learners in the Felder-Silverman model, Felder suggests Engineering professors follow nine instructional strategies [46]:

1. Teach theoretical material by first presenting phenomena and problems that relate to the theory (sensing, inductive, global).
2. Balance conceptual information (intuitive) with concrete information (sensing).
3. Make extensive use of sketches, plots, schematics, vector diagrams, computer graphics, and physical demonstrations (visual) in addition to oral and written explanations and derivations (verbal) in lectures and readings.
4. To illustrate an abstract concept or problem-solving algorithm, use at least one numerical example (sensing) to supplement the usual algebraic example (intuitive).

5. Use physical analogies and demonstrations to illustrate the magnitudes of calculated quantities (sensing, global).

6. Occasionally give some experimental observations before presenting the general principle, and have the students (preferably working in groups) see how far they can get toward inferring the latter (inductive).

7. Provide class time for students to think about the material being presented (reflective) and for active student participation (active).

8. Encourage or mandate cooperation on homework (every style category).

9. Demonstrate the logical flow of individual course topics (sequential), but also point out connections between the current material and other relevant material in the same course, in other courses in the same discipline, in other disciplines, and in everyday experience (global).

If an instructor could know the learning styles of their students, he or she would be in a position to develop a learning environment better suited to those students [28, 48]. If student learning styles are not known, instructors can follow the nine instructional strategies derived from the Felder-Silverman model to facilitate most students' learning. These works influenced our inclusion of learning preferences in student surveys as well as follow-up questions as to how these preferences affected other learning activities.

2.6 Computer Science curriculum and structure

Each learning institution is responsible for their own curriculum standards and structure. However, in an attempt to create a cohesive framework for learning objectives in undergraduate Computer Science degrees, a joint committee of the Institute of Electrical and Electronics Engineers-Computer Society (IEEE-CS) and Association for Computing Machinery

(ACM) developed a computing curriculum standard [1]. IEEE-CS and ACM are the leading professional organizations in Computer Science. Many top conferences in Computer Science and Computer Science Education are sponsored by these two organizations. The latest version of the Computing Curricula (CC 2001) was created by professionals in Computer Science and Computer Science Education reviewing past standards and how those past standards needed to change in order to make current recommendations for content in a Computer Science degree more appropriate to the latest developments and industry needs. The standard is based on findings in the literature, and web-based surveys of computing educators and professionals [1].

Computing Curricula 2001 (CC 2001) is a resource for institutions providing baccalaureate degrees in Computer Science. CC 2001 suggests specific "knowledge units" that should be contained in every CS undergraduate program. In addition, the document provides suggestions on how these specific "knowledge units" [1] might be approached by an instructor. Knowledge units are labeled as either necessary or optional to allow institutions to create unique programs for their specific student populations, while at the same time providing some sense of cohesion between degree programs.

CC 2001 labels some topics in Automata Theory as required and others as optional [1]. Required concepts that are part of Automata Theory include (with their respective knowledge units in parenthesis): basic logic and proof techniques (discrete structures); fundamental computing algorithms and basic computability (algorithms and complexity); and abstraction mechanisms (programming languages). The heading "Automata Theory" is suggested as an optional topic under the algorithms and complexity knowledge unit.

With Automata Theory topics spread across different knowledge units, the suggestion that earlier topics in the course—such as regular expressions, normal forms, and finite state machines—could be moved to more introductory courses in the Computer Science degree program has been made [8, 9, 58, 60]. In fact, Shackelford [101] suggests that the CS undergraduate curriculum differs from other sciences which tend to follow a breadth-followed-by-depth approach. Shackelford suggests that CS should integrate theory and practice early in the degree program instead of offering theory in disconnected courses taken largely at the end of the degree.

Currently Automata Theory is generally taught as a cohesive unit whose topics have been agreed upon for over 30 years [24]. Many textbooks [63, 67, 70, 74, 79] and syllabi [7, 41, 66, 94] have similar layout and curricula. In the current consensus, Automata Theory introduces three formalisms: finite state machines, push-down automata, and Turing machines. Each of these formalisms have associated material such as production grammars, decision procedures, equivalent language definitions, and proof techniques. This amount of material makes coverage of Automata Theory in a single semester an aggressive endeavor. If some concepts could be taught in introductory computing courses [101], it would allow more time for presenting the more difficult topics in Automata Theory.

2.7 Teaching Automata Theory

The literature includes little material on teaching or learning Automata Theory (or Formal Methods, or Computability Theory, other common names for the course). This lack of published work on this specific course may be due to several factors: it appears late in the curriculum ([36], SIGCSE member survey), it is more aligned with mathematics than with traditional Computer Science courses [53], CC 2001 lists the course as optional [1], and the majority of industry jobs do not focus on formal methods per se, although formal methods are used in many applications [31, 34, 108].

While there are no studies about the course itself, some researchers have investigated how to teach specific topics in Automata Theory. Chau and Winton [29] describe the usefulness of tools created in the APL programming language for lecture demonstration of automata. Similarly, the concept of demonstrating automata through the use of coding in Scheme, a Lisp-like programming language, is described by Wagenknecht and Friedman [112]. Researchers at Duke University created a Java-based tool for allowing pictorial representation of the three formalisms in Automata Theory [19, 53, 64].

The Java-based tool developed at Duke University, dubbed JFLAP [96], was created to enhance learning in Automata Theory. Due to the bulk of pencil-and-paper assignments, "this course can be more difficult for some students than most of the other computer science courses which have hands-on interaction in the form of programming" [53]. The JFLAP user interface uses graph structures to allow users to enter specifications of the formalism under study. This feature may make JFLAP especially useful to students who are visual learners. The step-through capability allows the tool to be used in lecture for demonstrating the step-by-step sequence that the formalisms follow when computing a particular example. Rodger [97] reported that she used JFLAP to exemplify Automata Theory formalisms and create a more interactive lecture with students.

DEBBIE (DePauw Electronic Black Board of Interactive Education), a more comprehensive instructional tool, was developed and used in teaching Automata Theory at DePauw University [16]. This system uses an electronic whiteboard to capture and distribute instructor notes to laptop computers in an electronic classroom. Each laptop allows students to add to instructor-supplied images (note-taking) as well as produce and submit material back to the instructor (presenting work). This type of electronic dialogue during lecture creates a very dynamic learning environment, but requires a large hardware and software overhead.

Automata theory strives to teach students to understand and create abstractions and formal proofs (first interview with Dr. Lily Quilt). As early as 1976 Berztiss [17] warned Computer Science Educators of the importance of including abstraction and proof construction as part of a Computer Science degree. He stated that the study of algorithms had become "heavily dependent on proofs" and that students should be taught "the value of abstraction in the clarification of problems so that poor formulations gradually disappear" [17]. Requirement of an Automata Theory course would provide a setting that could accomplish both goals.

Harrow [55] suggests specific types of proofs to which students should be exposed as part of Automata Theory, including: proofs using the Myhill-Nerode theorem, proofs using the Pumping Lemma, and diagonalization proofs. In addition to exposure to proofs, Harrow states that it is the goal of Automata Theory to give students "an appreciation of...things that can and cannot be done [computed/proven], and hopefully some intuition as to how" [55].

There are several types of proofs in Automata Theory: direct proof (e.g., classifying a language definition), proof by contradiction (e.g., proving that a language definition is not part of a specific classification), and reduction proof (e.g., a particular language definition describes an uncomputable function). The last proof technique, reduction, relies on a high level of abstraction in order to solve. In her paper, Hazzan [57] discussed how students go about understanding abstractions and how those efforts affect learning outcomes. Hazzan's suggested that examples should use familiar concepts before introducing more abstract concepts to facilitate student understanding.

2.8 Lecture-style teaching

Although many colleges and universities rely on lecture, the intent and acceptance of lecture is regarded differently by discipline [84]. In his literature review of the latest findings on perceptions of lecture-style teaching, Pollio combined and presented information about the differences in lecture-style teaching in Natural Sciences (which includes Computer Science) and the Humanities. Pollio examines earlier work from Biglan [18], Erdle and Murray [43], Feldman [50], Kolb [69], and Pohlmann [82]. The synthesis of these works stated that students appreciated the ability of teachers to "explain things clearly" in Natural Sciences [84]. This appreciation is not just applicable to lecture content, but is also relevant to expectations set for the course [84]. When teaching objectives are clear, students have an easier time constructing efficient learning schemas [47].

Lecture is conducive to creating a transmission-based learning environment. In a transmission-based learning environment, students are passively told information. In this type of learning environment, participants can easily lose interest. One way of combating this “minds-off” side effect of lecture is for the instructor to question students at strategic points during the lecture [83]. In observing college professors and their use of questions during lecture, Pollio found that science instructors were less likely to question students than professors in Law [83]. Science instructors tended to use questions as a way to elicit past knowledge. With this type of questioning, the science instructor already knows the answer, and there is an “air of power” associated with the process [83]. Law professors, in contrast, displayed a use of questions intended for seeking “answers for advancement of knowledge” [83]. Felder expressed the opinion that questions facilitate learning in a lecture setting [45]. He stated that “a good question raised during class...can provoke curiosity, stimulate thought, illustrate the true meaning of lecture material, and trigger a discussion or some other form of student activity that leads to new or deeper understanding” [45].

Supporting the use of good questions, survey results from 465 academic members of the American Economic Association (AEA) by Benzing and Christ found that economics and finance instructors thought learning occurred when students were “actively engaged in or out of class” [15]. This engagement could be in the form of class discussions or “asking questions in an open lecture environment” [15]. The majority of the AEA respondents reported teaching in an exclusive lecture environment although many reported that they had begun to change their methods to allow for “more class discussion, use of questions, and group activities” [15].

Another method for involving students during lecture is the use of examples to teach concepts. Renkl [91] found that the use of examples is an expedient learning vehicle for novices in well-structured fields of study such as Computer Science. Renkl stated that the motivation behind the use of examples in lecture is to have the students simultaneously work through the steps of a problem with the instructor. However, “the extent to which learners profit from the study of examples depends heavily on how well they explain the solutions of the examples to themselves" [91]. In order to facilitate this self-explanation, examples in class should be followed up by similar homework problems where the student is required to use their knowledge from the solution in class to construct their own answers. This facilitates a deeper understanding of the material presented in lecture and creates an environment in which knowing is a "process rather than a product" [26].

An advantage to using a lecture approach is the ability of a single professor to address a large number of students simultaneously. In part of a study designed to investigate the delivery of course materials, student support, and student assessment in a large class, Hicks [59] found that in a large room, the use of machine-produced slides is a better delivery mechanism for information than hand-written notes. The course used for Hick's study consisted of 310 undergraduate students in an Accounting and Finance course at the University of Newcastle upon Tyne. The results showed that the use of machine-produced slides, such as those created by PowerPoint, and microphone usage were critical if all students were to hear the instructor and see the lecture materials. The course was set up so that all class materials were posted online for ease of delivery and all assessments were online multiple-choice examinations. In a post-course survey, students gave feedback on teaching effectiveness. The large class survey results were compared to previous survey results of the same instructor when fewer students were in the course. Hicks concluded that responses on both surveys were "similar," suggesting that the size of the class did not seem to bother students when slides and microphones were used during lecture and class materials were posted online.

A drawback to large lecture classes is the lack of personal contact time between instructor and student [59]. One method for combating the lack of individual contact in large classes is to have supplementary discussion sections. Ramakrishna [89] found that discussion sections that focused on homework completion seemed to engage students more than traditional discussion sections that focused on transmission of additional lecture material. In an attempt to improve performance in an Advanced Database course, Ramakrishna restructured his course to include a discussion session. The discussion session replaced individual tutorial sessions that had been offered in previous instances of the course. The session was designed to allow students to receive extra practice in solving problems for required homework in the course. In a post-course survey, more than 75% of the students who took the survey (28) agreed strongly with the statements that the extra work helped them understand the subject matter better and that they liked the extra tutorials. Ramakrishna also stated that discussion sessions designed to facilitate homework completion motivated students to attend the tutorials [89]. This last statement implies the importance in motivating attendance for learning.

A number of researchers have found a correlation between lecture attendance and higher grades. In investigating how attendance patterns affected course grades, Van Blerkom [110] surveyed 140 undergraduates on academic perseverance and self-efficacy. The questionnaire answers were correlated with both attendance and performance in classes. One result was a low correlation between attendance and better grades. Raphelson [90] examined the relationship between lecture slide recognition and course grades for 77 undergraduate psychology students. He found a significant correlation between the scores on three course examinations and the students' ability to recognize lecture slides on an unannounced recognition examination. In his conclusion he stated:

> "the obtained correlations between the examination scores and Slide Recognition scores were mediated by other variables (e.g., class attendance)...Students could not recognize a slide if they were not present when it was shown...It is also true that class attendance plays some role in exam performance per se, especially in a lecture class" [90]

In an attempt to produce criteria by which instructional activities could be evaluated, Felder, Rugarcia, and Stice [49] enumerated eleven criteria for effective course instruction and eight methods by which the criteria could be evaluated. Their eleven criteria for effective instruction are [49]:

1. The course contributes toward published program goals.
2. The course has clearly stated measurable learning objectives.
3. The assignments and tests are tied to the learning objectives and are fair, valid, and reliable.
4. Appropriate methods have been devised to monitor the effectiveness of the instruction.
5. The learning environment is appropriate.
6. The instructor has appropriate expertise in the course subject.
7. The instructor communicates high expectations of students and a belief that they can meet those expectations, interacts extensively with them inside and outside class, conveys a strong desire for them to learn and motivates them to do so.
8. The instructor seeks to provide an education in the broadest sense of the work, not just knowledge of technical content.
9. The instructor integrates teaching with research.
10. The instructor continually attempts to improve the course by updating the content and/or making use of new instructional materials and methods (including applications of instructional technology).
11. The students achieve the learning objectives.

Felder, et. al., stated that evaluation of the above goals could be accomplished by using the following methods [49]:

- Student performance in the course.
- Student end-of-course ratings.

- Student surveys, or interviews directed at specified criteria.
- Retrospective student ratings of the course.
- Alumni ratings of the course and instructors.
- Peer ratings of classroom instruction, learning objectives, assignments, and tests.
- Evaluations submitted by external referees.
- Self-evaluation by instructors.

This work was instrumental to our evaluation of CS 341.

The work in this section influenced the make-up of instructor interview questions, student survey questions addressing lecture, and information being tracked during classroom observations, as well as the analysis of our gathered data. In particular, the importance of the following three ideas were influential: clear student expectations, use of questions and examples during lecture, and student attendance.

2.8.1 Effective material use

In a class with a large number of students (greater than 150), the management and creation of materials becomes important because it is unlikely that all students can interact with the instructor during lecture. The use of lecture slides has been shown to make lecture material visible to a large room [59]. Ralphelson found that the use of slides correlates with higher examination grades and that the use of slides "does not interfere with class learning" [90].

Another technique shown to positively affect learning is note-taking. Beecher [13] stated that an instructor should produce materials and organize lectures to facilitate good note-taking techniques. Two good techniques for facilitating effective note-taking are to provide students with "handouts that give students room to add notes" and "announcing explicitly the precise role that lectures play in the course" [13].

Materials can also include assessment and motivational tools in addition to lecture slides and class notes. Quizzes have been shown to raise student incentive to study [109]. In investigating the relative effectiveness of using weekly quizzes on students' motivation for studying, Tuckman [109] found that when students were motivated through quizzes, their examination scores were substantially better. To investigate this, he conducted two studies on students in an educational psychology course. In the first study, 109 junior and senior college students were split into three groups. In group one, students followed a prescribed study technique for reading class material and completing homework. Students in group two were not told about any study technique; instead they completed a quiz at the beginning of each class. Group three was the control: students did not receive any advice on how to study, nor were they given any quizzes. After five weeks students in all three groups took the same cumulative examination. The analysis of examination scores showed that students in the second group (i.e., with quizzes) scored significantly better than students in the other two treatment groups. In a follow-up experiment designed to replicate the previous investigation, Tuckman introduced a moderator variable of grade point average within each of the three treatments. The findings from the second study showed a significant increase examination performance in students who were motivated via quizzes and had low incoming grade point averages (GPAs). Both of these studies suggest that incentive to study is more important than prescribing study methods, and that motivation for studying is most important to lower performers.

With the ease of access to the web, class web pages have become useful instructional tools. In 1999, Brawner, Felder, Allen, and Brent [22] conducted a survey of 511 Engineering instructors to investigate the current teaching practices of college Engineering faculty. They found that 96% of the respondents used email to communicate with students, 75% sent information to the whole class using electronic means, more than 60% reported posting syllabi and assignments, and more than 40% posted lecture notes and problem solutions [22]. This type of web use requires a fully networked computing environment.

Students and instructors need access to the networked materials both on and off campus in order for this type of instructional methodology to be productive.

Ritter and Lemke [95] found that students expressed appreciation for course materials posted on a course website. The findings were based on survey data from 256 students enrolled in an Introductory Physical Geography course. The course consisted of a combination of lecture and laboratory activities. The course website included "lecture outlines, diagrams, photos, exam reviews, practice tests, lab answer keys, online readings, and links to relevant material published on the Internet by others" [95]. Conclusions of this study revealed that students expressed that the "use of email encouraged student-faculty contact" as well as prompt feedback [95]. Students also reported that the materials posted on the Internet enhanced their learning and allowed "more efficient use of time in and out of the classroom" [95].

The use of individualized assignments may enhance student learning [62, 107]. Toothman investigated whether homework questions specifically designed to target "each students' areas of weakest performance would help the student perform better in the future" [107]. Four hundred students enrolled in an Introduction to Computing course were divided into three treatment groups. The treatment groups differed in what type of questions were assigned on the last two homework problems during the course. Group one received a set of two additional questions appended to each of the homework assignments. The additional questions were based on "analysis of the students' previous quiz performances on a range of topical areas" [107]. Group two received an additional two questions on the homework assignments; however, the questions were not targeted for individual student needs. Instead the additional questions were targeted toward overall class weaknesses. Group three was the control. Students in group three did not receive any additional homework questions. The use of questions designed to help individual student weaknesses (group one) showed significant improvement on similar questions on the final examination. However, these findings were only significant for students who were "neither particularly strong nor particularly weak" on the topic [107]. For students who demonstrated significant fundamental weaknesses, Toothman hypothesized that "stronger, more intensive intervention" was necessary [107].

The University of Texas at Austin (UT) provides a homework service that completely individualizes homework problems for each student [62]. The individualization is not influenced by individual weakness, but rather to provide each student a unique set of problems to work on:

> "The Homework Service provides inventoried 'problem banks' with more than 22,000 different questions for high schools and colleges in the subjects of algebra, pre-calculus, finite math, calculus, physics (in English and Spanish), physical science, and chemistry. The unique, algorithm-based problems ensure that questions are different, and that every student receives a different version... Students download the assigned problems, solve them, then enter the answers into the program and receive immediate right/wrong feedback. Explanations are provided to students after the due date." [62].

The homework service can also be used by teachers to provide practice problems by allowing explanations to be provided immediately upon completion of the answer (informal discussion with Dean of Information Technology). This service is not only available to college-level courses. Many high school classes take advantage of the service: "More than 13,000 students at 250 different schools nationwide took advantage of the program during the fall 2000 semester" [52]. If Automata Theory problems could be incorporated into the UT homework service problem bank, students taking Automata Theory could use the system to gain examination practice problems and immediate feedback on homeworks.

The key ideas that influenced our work were the need for machine-produced materials for large classes, that materials should provide a structure to help students take in-class notes, and the increased flexibility in student use that web-based materials could produce. By far the most influential notion from this section is the use of quizzes for formative assessment and motivation for student attendance. The suggestions for motivating better attendance and more interaction during lecture in CS 341 made strong use of the notion of quizzes.

2.8.2 Defining excellence in instruction

Obtaining excellence in instruction should be the goal of every educator. A preponderance of literature exists on suggestions for instructional strategies and changes aimed at improving student learning. Sadker and Sadker [98] suggest that teachers attempt to engage an equal number of girls and boys in question answering during lessons to improve overall class involvement. Margolis and Fisher [77] suggest that teaching assistants for college-level Computer Science be trained in pedagogy and diversity to improve overall effectiveness of instruction. Excellence in instruction for Sadker and Sadker, as well as Margolis and Fisher, is defined by active involvement and inclusion of both men and women during the instructional process.

In addition to large studies, the literature contains many single-focused self-reports on positive changes in instructors' own pedagogy. Aguirre [3] teaches college-level Sociology to "mass classes," a term he uses to indicate lecture-styled classes that consist of 100 or more students. Based on student exit surveys, Aguirre claims that a mass class is effective when the instructor creates an environment that is "reflective." The goal of a reflective mass class is to "engage students in a discussion of how they interpret the world around them, [and] of how they make sense [of the lecture] by thinking out loud" [3]. One way in which to get the students to reflect is to ask questions or to use examples that students can comment on. "The use of current examples...allows [the instructor] to ask questions of students that are focused on critical thinking rather than factual knowledge...One result is that students are more willing to involve themselves, via their responses, to questions in the instructional process" [3]. Aguirre obtained positive student feedback on class exit surveys when he taught mass classes reflectively.

In addition to equity and student feedback, much work has been done to define specific objectives by which to gauge instructional excellence. Felder [46] created nine instructional strategies that would produce better learning environments for Engineering classes. These categories are discussed in Section 2.5 of this chapter. Weiss and Weiss suggest the following four criteria for creating "accomplished teaching" regardless of the specific subject to be taught [114]:

1. Teachers are committed to students and their learning.
2. Teachers know the subjects they teach and how to teach those subjects to students.
3. Teachers are responsible for managing and mentoring student learning.
4. Teachers are members of learning communities.

Chickering and Gamson state seven principles that define "good practice in undergraduate education" [30]:

1. Encourage contact between students and faculty.
2. Develop reciprocity and cooperation among students.
3. Use active learning techniques.
4. Give prompt feedback.
5. Emphasize time on task.
6. Communicate high standards.
7. Respect diverse talents and ways of learning.

Garcia-Barbosa and Mascanzie cite Svinicki's [104] "six central ideas of cognitive learning theories and their implications for instructors" [51] in discussing good practices for college-science teaching assistants. The six central ideas are:

1. Instructors should emphasize and be clear about the information to be learned and why it is important.
2. Learners and instructors should strive to act on new information to make such information more understandable. This includes the use of numerous examples, illustrations, extended assignments, and references to previous content and experiences.
3. Learners commit information to long-term memory according to their current understanding of major concepts. The instructor should facilitate ways of organizing knowledge by demonstrating ways they have organized content for greater understanding. Instructors can also encourage learners to create their own ways of organizing new information.
4. Learners need to revisit and re-evaluate new knowledge with an emphasis on refining and revising concepts that should be retained. Instructors can offer opportunities for learners to self-check and evaluate their understanding of new concepts and ideas.
5. Transfer of learned material is not automatic but can be facilitated by continued application of new knowledge and by applying it to new situations and contexts.
6. Learning is greatly enhanced when people are aware of specific strategies that work for them. Learners also benefit by monitoring their use of such strategies. Instructors should emphasize strategies that help translate new information into memory.

Garcia-Barbosa and Mascanzie suggest that in addition to the listed six central ideas, "student beliefs and perceptions do influence their ability to learn in college situations" [51]. The influence of students' beliefs on learning supports the importance of student feedback for formative and summative assessment of teaching effectiveness.

The works presented in this section give examples of how the literature addresses instructional excellence: either a specific goal is reached (e.g., gender equity), student feedback is positive after a pedagogical change is instantiated, or specified instructional objectives are met. The criteria posed in these works were influential in creating the focus of our surveys, defining what we were looking for during classroom observations, and influenced how we defined "best practice" under pedagogical positivism.

2.9 Theoretical models found in Science and Education

A prevalent theory in science is logical positivism [115]. Logical positivism states that "science should be based on concrete sensations rather than upon abstract words. This [means] that only the past [is] a certainty" [76]. Logical positivism does, at its basis, allow for preponderance of ephemeral evidence (such as individual perceptions) as verification [37] but most positivists would not agree with this elastic definition. Logical positivism is commonly viewed in a very critical light [35] as being too focused on mathematical-like facts. The elegant aspect of logical positivism is that the data is verifiable: since there is a single truth, once a truth is known, others can verify that the fact or finding is true.

At the other end of the spectrum, a great deal of research in Education is based on constructivism. A constructivist epistemology (the basis of constructivism) states that each individual creates their own truth [12] based on personal prior experiences. In a constructivist epistemological setting "verification" of a fact can never take place. If meaning is made by individuals, and individuals are unique [35], then how can one person verify another's data?

A recent version of constructivism is social constructivism. Social constructivism posits that the "analysis of 'knowledge' or 'reality' or both as contingent upon social relations, and is made out of continuing human practices, by processes such as reification, sedimentation, [and] habitualization [115]." The more modern definition of social constructivism states that "knowledge is the result of social interaction and language usage, and thus is a shared, rather than an individual experience [40]."

In defining the theoretical basis of this research, pedagogical positivism, we rely on both positivism and constructivism. Pedagogical positivism and the research model that emerged from this theory are presented in the next chapter.

Chapter 3

Methodology

This research investigated student perceptions of learning Automata Theory, instructor perceptions about teaching Automata Theory, and how these perceptions affected interactions in the classroom through a detailed case study.

The foundational task in any case study is to understand the current environment. Through examination the researcher can build a picture of what currently works and what needs modification or attention [49]. Without understanding the current environment thoroughly, suggestions used to improve learning is a form of action research [2] which was not the goal of this study.

In this chapter, the first section describes the theoretical basis of our research, pedagogical positivism. The second section describes the research questions that were generated from our theoretical model. The next three sections describe the data gathering techniques used, the specific data collected, and the analysis of the collected data.

3.1 Pedagogical Positivism

This dissertation research is based on a newly invented theoretical model we call *pedagogical positivism*. The term pedagogical positivism was created to describe a theoretical model that borrows ideas from both logical positivism and constructivism. Pedagogical positivism is a philosophical stance that believes for a specific population of students, there is a definable instructional method that is better than others. Pedagogical positivism allows "best practice" to be defined through research of a teaching environment.

The theory supports individually constructed facts that are verifiable through socially accepted means such as repetition of observations, repeatable experiments, or preponderance of evidence. If 51 out of 100 people stated "there is a God" the result under pedagogical positivism would be the statement "there is widespread belief in God." This theoretical model supports interviewing, observation, and survey techniques, as well as positivist-style data collection activities such as grade analysis and printed material evaluations.

A reasonable objection to pedagogical positivism is that it blurs together two well understood and time-honored theoretical paradigms: positivism and constructivism. The motivation for defining a new theoretical basis was to support our study of a Computer Science course through an educational lens. Computer Science as a field of study is generally aligned with positivists' beliefs. In Computer Science there is a single right answer to many activities: a program has a correct output; when comparing two algorithms, one clearly has better performance. Education, on the other hand, is currently aligned with a constructivist belief: meaning is created by individuals. The combination of positivism and constructivism allowed our study to express notions of "best practice" and to comment on which style of presenting material is "better" than another. In a constructivist-only environment, any such claims would be invalid. Because our goal is the improvement of teaching and learning topics in Automata Theory, we needed a basis on which to claim certain techniques needed improvement or currently worked with the given student population. Pedagogical positivism allows for these type of findings. Findings of this type, however, must be tempered by existing instructional goals. It may be the case that what students like and what is good for them are opposing views. For instance, if most students in Automata

Theory do not see the need for the course, it may not be prudent to remove the course from the degree program.

But what is a preponderance of evidence? In creating pedagogical positivism we have chosen to allow researchers to define their own level of acceptable evidence. The level of acceptance, similar to an α-level in ANOVA statistics, is individual to the research being conducted. Our goal is to be able to define "better practice" for a population of students. Thus we have chosen findings to be valid when majority agreement on a certain statement is observed. Majority agreement was chosen because it is similar to how collaborative actions are taken. If a majority consensus is achieved, the group generally follows the suggestions of the consensus. In the same way, if our data shows a majority agreement on any one statement, we have a consensus in understanding the views of the individuals commenting on the statement, and thus would be valid in acting as if the statement was representative of the group.

Although the definition is new, the idea behind pedagogical positivism is not. Bayesian logic was created with the idea that decisions could be based on the probable outcome of an event, or the use of prior events to predict future events [11]. Using this type of logic is similar to using pedagogical positivism. A model based on Bayes' Theorem was not sufficient because the observable facts in traditional Bayesian logic have known a fixed probability before computing the probable outcome. Pedagogical positivism has the elasticity to encompass previously unknown perceptions.

A case study based on pedagogical positivism will need to examine both static and dynamic variables, as the static variables may influence the perception of any dynamic variables. For a college course, static variables include location in which the material is taught, time of day when teaching is conducted, duration of classes, and material used and covered during lecture. The label "static" denotes that they are stable over a single semester. Dynamic variables include lecture content, instructor interactions with students, and student attitudes and beliefs about their learning progress in the course. Dynamic variables have the ability to change at any point during a single course offering. Both static and dynamic factors must be examined in turn.

For our case study, the static and dynamic variables are summed up in the pictorial model presented in Figure 3.1. The three spheres represent the three areas that contribute variables in CS 341, whether they be static or dynamic. The content in each sphere represents the information we gathered from each area in order to illuminate the perceptions influencing learning in Automata Theory.

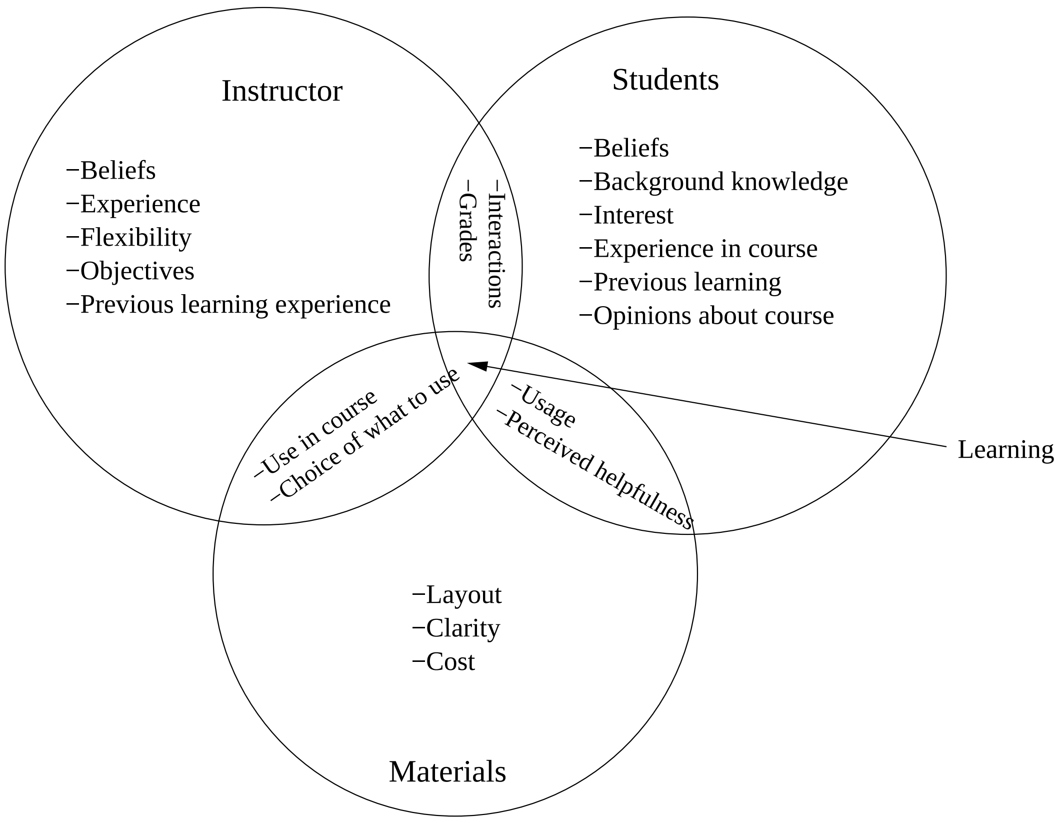


Figure 3.1: A model of the three components underlying student and instructor perceptions in CS 341.

3.2 Research questions

Based on the theoretical model of pedagogical positivism, the research questions framing this research were:

- How do instructor perceptions shape teaching and learning in Automata Theory?
- How do student perceptions shape teaching and learning in Automata Theory?
- How do the materials used in Automata Theory help or hinder teaching and learning?

These research questions were then operationalized into sub-questions around which data were collected. We first present the questions that were created for the case study of CS 341 taught by Dr. Lily Quilt followed by a discussion of the purpose behind each question.

1. How was the course created? Were any standard curricular goals used?
2. What does the instructor see as the goal(s) of this course?
3. What types of questions does the instructor ask the class during lecture?
4. How do the students view learning the material?
5. What do the students think is missing from the course?
6. What do the students think is good about the course?
7. What motivates students to do homework?
8. Are students attending and benefiting from discussion sessions?
9. Are students coming to class?
10. What is the pace of class?
11. What drives student involvement in lecture?

12. What examples are used in class?

How was the course created? Were any standard curricular goals used? Based on conversations with several professors in Computer Sciences, it became clear that the creation of classes is generally somewhat ad-hoc. Course creation tended to fall into one of three categories: (1) the course was created by committee and then each instructor customized it to cover material they thought was important to the specific topic; (2) the course was created to mimic the instructor's learning of the same material [42]; or (3) the course was created due to the individual instructor's interests.

Some courses are not hurt by this ad-hoc style of curricular design. In fact some courses have a continually changing focus due to the advancement of research and technology. But the content of some courses is more static, meaning that the knowledge for that field has been relatively unchanged for some time. Some logic and theoretical courses fall into this category; CS 341 is one such example. Textbooks on Automata Theory tend to follow the same basic outline and focus, although the depth of coverage and level of rigor varies [67, 70, 74, 79]. Part of considering this research question was to investigate the last time these "agreed upon" topics [24] and ordering were examined for effectiveness in student understanding in CS 341.

Some prerequisite courses for CS 341 are not of a static nature. To build a full picture, the study must also investigate what the instructor of CS 341 assumes about the students' prerequisite knowledge and how that prerequisite knowledge is reviewed (if at all).

What does the instructor see as the goal(s) of this course? This research question focuses on what the instructor sees as her job in the classroom [42, 56]. Evidence related to this research question will uncover some of the instructor's beliefs about the material, her role in the student learning process, and what she thinks is important for students to know when they complete the course [102]. This information will allow us to see whether the instructor's goal(s) for the course are aligned with students' views.

What types of questions does the instructor ask the class during lecture? Instructor cues can encourage or discourage students in a lecture-style class to participate during the lecture [15, 83]. If the instructor does not encourage questions then students may not participate because they feel it is inappropriate or not required. On the other hand, if the instructor encourages student responses by asking questions, then students may believe it is acceptable to spontaneously comment or ask questions during lecture [84]. A related issue is whether the instructor encourages student interaction through the use of rhetorical or non-rhetorical questions during her lectures. Investigating this may reveal whether student participation is driven by instructor cues. By examining typical instructor questions with respect to Bloom's taxonomy [20], the type of information the instructor is seeking will be clearer. The types of questions an instructor asks certainly influence the number of students who participate in class and whether their participation is active mental engagement or passive answering of rote questions.

How do the students view learning the material? Students' perceptions about learning in CS 341 can be compared with instructor perceptions on student performance to see whether the instructor's expectations and the current student experiences are in alignment [85]. Data gathered to answer this research question will also be useful in gaining a better understanding of the reasons for interactions: whether interactions arise because students are confused or interested in the subject. The answer to this research question provides insights into student participation in the classroom.

What do the students think is missing from the course? Sometimes the best insight into how to fix an existing course comes from the criticism of those involved in it. The students themselves can think about what the course lacks and suggest specific changes for improving the course.

What do the students think is good about the course? A researcher can come into an existent program, study it thoroughly, yet overlook activities that are effective in helping the current student population to learn. To avoid this situation students should have an opportunity to discuss what they enjoy about the course in its current form.

What motivates students to do homework? CS 341 covers theoretical topics, and thus there is very little programming. In addition, graded homeworks are not weighted very heavily. However, the students are given a large set of voluntary homework and reading for the course. The instructor believes that good performance in this course hinges upon students doing the voluntary homework, while students' motivation to do voluntary work may differ. In order to see how students' homework efforts align with instructor expectations for this course it is necessary to understand what motivates students to work on the assignments. [54, 107, 109].

Are students attending and benefiting from discussion sessions? The discussion sessions for CS 341 are not mandatory, but were created to benefit and bolster student learning [89]. In order to ascertain student benefit from this type of activity, the first question to ask is, "Are they are attending?" The next part of this research question is, "Are the attending students benefiting from the extra time and input from the discussion leader?"

Are students coming to class? Attendance has been shown to positively correlate with good grades in college-level classes [110]. However, some students may feel that since the lecture slide material is part of their course packet, they do not need to attend lecture. Understanding why students do or do not attend lecture regularly may help shape suggestions on what materials should be handed out and how lectures should be structured.

What is the pace of class? The rate at which information is transmitted to students during lecture may affect understanding. Understanding in CS 341 is assessed through grades that students receive on homeworks and examinations. To understand whether the speed of lecture affects performance, both lecture pace and grade information should be examined. Lecture pace also provides a basis for comparing the amount of information covered in a single semester of CS 341 with the amount of information covered in other CS courses.

What drives student involvement in lecture? Reasons students may become more involved in lectures could be because they enjoy specific topics, interact more when the material is concrete [1, 88], have an interactive personality type, are worried about performance on an examination, or are concerned about course grades. Answering this research question may reveal which factors, such as instructor questions, concrete examples, or student personalities, encourage active participation in class.

What examples are used in class? Not all students learn in the same manner [46, 48]. Some students may be verbal learners, while others may be more visual. In order to relate lecture material to students' self-perceived learning styles, it is necessary to observe how many pictorial and verbal examples are used in each lecture. Investigating the types of examples used in lecture will indicate whether multiple learning styles are supported in the current course [44, 46].

3.3 Data-gathering techniques

The evidence needed to answer the research questions in the previous section requires information that is found either in the material used in the course, with the instructor, with the students themselves, or in the interactions of the instructor with the students. Each of these four areas requires different techniques to extract information. These techniques are listed and their use motivated in this section.

The primary materials used in CS 341 included a textbook [67], a course packet, and a course website. We obtained a copy of the textbook and the required course packet. As supplemental handouts, including homework and programming assignments, were distributed in class we retained a copy. The website from the Fall 2002 semester was available for general access during the course. In having a copy of the textbook, course packet, class handouts, and access to the course website, we had full access to the same materials to which each student had access. As the semester progressed, we kept a record of the number of pages relevant to each lecture, the material covered in each class, and how much homework, both voluntary and required, was assigned each week. When observing student interactions in lecture, we recorded the specific material students used (if any) for the interaction.

Materials, being static, were easy to obtain and understand. It was more difficult to understand the choice of what materials to use, how they were used and interpreted by students, and how they were used and revised by the instructor. To understand the rationale behind the choice of materials used in CS 341, we interviewed the instructor of the course. In the Fall 2002 semester, Dr. Lily Quilt taught two sections of CS 341. In the first interview with Dr. Quilt we investigated her rationale for choosing and creating the specific material used in CS 341 this semester and how she arrived at those choices. In addition to asking questions about materials in the class, we used interviews to investigate the instructor's point of view on lecture content, delivery, and interactions with students. The interviews were a good method for uncovering Dr. Quilt's beliefs about how the material could best be learned, student prerequisites, and student learning in CS 341 [42]. Beliefs held by an instructor can influence the way lecture is structured and students' attitudes toward learning expectations in the course [6]. Through discussion with the instructor, we could also uncover the rationale behind grading strategies, homework content, and examination content. Understanding the grading system for this course was important since grades provide students with tangible evidence of whether or not their learning of the material was acceptable. The psychological impact of doing well with respect to grades also influences students' study behavior, as well as their motivation and attitudes [77, 85].

In order to obtain a broad view into student behavior, motivation, and attitudes, we collected data from the students themselves using on-line surveys. Surveys allowed students to express their opinions about their progress, what they thought of the course, and what they thought of the material. The alternative approach of face-to-face interviews was eliminated due to concerns that students might feel embarrassed about speaking aloud to a person they did not know well. Another advantage of surveys was students could take more time in answering the questions. They had a chance to reflect without the added pressure of the interviewer looking at them waiting for a response. The current CS undergraduate population includes many individuals for whom English is not their first language. Surveys gave ESL students as much time as they wanted to understand each question. Additionally, transcripts of recorded conversations with individuals who have heavy accents can suffer from inaccurate transcriptions due to misinterpretation of badly-phrased or hard-to-understand spoken English. Surveys avoid at least some of this type of ambiguity. A side benefit of studying junior or senior college-level computer science students was that all students had already acquired extensive experience with using the computer and communicating via email, another form of asynchronous discourse. Time to reflect and comfort with asynchronous communication enhanced the richness of the data collected through surveys conducted on-line. The student surveys were designed to gather data on each student's understanding of their progress in the course, their view of how effectively they learned each topic in CS 341, and their motivation to do homework, study for examinations, and attend lectures.

To interpret the data gathered via student surveys, it was necessary to understand the learning environment in which the students were situated. The primary learning environment that was common to all of the students was the lecture. In order to understand the lecture setting, we attended half of the lectures for each section of CS 341. Observation notes captured student behavior during the class (i.e., who asked questions and what topics elicited discussion) as well as the speed, pace, and delivery of the material in lecture. The researcher conducting the classroom observations had previously taken Automata Theory at UT. This helped with the depth and accuracy of field notes generated from the observations. Having the researcher observe the students' learning environment gave us a better structure with which to interpret student survey data and instructor interview commentary.

3.4 Specific data gathered

To capture the data discussed in the previous section, we read the material required for CS 341, conducted instructor interviews, conducted student surveys, and attended lecture to capture classroom interactions. We obtained and reviewed the material required for CS 341 before classes started so that we were up to speed at the beginning of the semester and could then observe and understand classroom interactions from the first class day. This preparation also simplified the task of recording specific references to material during the initial instructor interview.

The first interview with Dr. Quilt was conducted before the semester began. Dr. Quilt was asked to sign a consent form allowing the interview transcripts to be used in this research, which she did prior to our first interview. The timing of the initial interview was based on Dr. Quilt's schedule in order to ensure that she had at least one hour to sit and talk. By scheduling the interview before the first day of classes, we gained better insights into the choices she had made for the upcoming semester without the technicalities of teaching the course influencing Dr. Quilt's thinking. The interview was transcribed and given to Dr. Quilt so she could confirm the accuracy of the transcription. In order to understand and track her decisions about issues such as the pace of the class and changes in the course syllabus, we conducted two additional interviews with Dr. Quilt, the second approximately mid-semester and the third at the end of the semester. These follow-up interviews were valuable for the purpose of generating data that revealed insights into the instructor's patterns of beliefs about the material and students, as well as her reactions to how the semester was progressing.

Each interview used a pre-established list of questions that were generated a week before the meeting was to take place. The follow-up interview questions were influenced by prior interview transcripts as well as classroom observation data.

In addition to the formal interviews, Dr. Quilt and the researcher frequently chatted informally about the day-to-day progress of CS 341 and student reception of lecture topics.

This information was not transcribed or used in a direct way in this research. However, the informal conversations did influence the questions asked during the second and third interviews, as well as the structure and nature of the student survey questions.

It is important to note that Dr. Quilt played a vital role in this research. Not only was she one of the variables under study, she was also a member of the dissertation committee. Her role as a committee member was an important mechanism for ensuring the credibility and accuracy of the research. In order to avoid having knowledge gained from her role as a committee member unduly influence the data collection, Dr. Quilt was not informed of any themes that emerged from the data until after the end-of-semester grades had been submitted and the data collection phase was complete. In discussing the situation, Dr. Quilt felt that her unfamiliarity with qualitative research methods would help prevent unintentional bias in her interview answers because she was unaware of any research goals aside from understanding how CS 341 currently operated and how it might be improved. Dr. Quilt also agreed that with the exception of the interview transcripts, she would not validate any raw data until the data collection phase was completed. With these safeguards in place, the data was more authentic and representative of a generic Automata Theory instructor.

In order to gather student-centric data, we conducted on-line student surveys. The purpose behind the student surveys was announced to each section of Dr. Quilt's CS 341. Students were told that the surveys were being used to gather student opinions on the activities and presentation of the course in order to create improvements for future offerings. Each survey posting was announced to the students via an email message sent to the class list in addition to being posted on the "announcements" section of the class webpage. The first survey was given on the twelfth class day, a date close to the start of the semester but after the University deadline for easy adding and dropping of classes. To track changing attitudes and beliefs of the students, we conducted a total of three surveys: One at the beginning of the semester (twelfth class day), one mid-semester, and one just prior to the end of the course. Each survey was developed using the SurveySuite^{TM} [103] tool and then uploaded to a local CS website. All surveys began with a cover letter describing to students that participation in the study was voluntary and that once the survey responses were submitted, they were implying consent to have their anonymous responses included in this work.

Similar to instructor interview questions, student survey questions were developed prior to each survey. Survey questions were influenced by past student responses, classroom observations, and instructor interview commentary.

As foreshadowed, another type of data analyzed for this research came from classroom observations. At the start of the semester, Dr. Quilt introduced the observing researcher to both sections of CS 341. Dr. Quilt explained the role of the researcher's presence in lecture as recording events in order to understand what transpired during lecture. The field notes created during each lecture were used to track specific aspects of classroom behavior, such as student interactions in each section, examples used in lecture, number of slides and topics covered in each lecture, attendance, and amount of voluntary and assigned homework given each week.

The current assessment of student learning in CS 341 was in the form of grades. We collected scores on homeworks, projects, and examinations after the final examination was graded. Students were asked to sign a consent form to have their grades in CS 341 used as part of this study. Grade information was collected only from students who had voluntarily signed the consent form. Because letter grades are broad categories of student performance, we also collected raw scores as well as the rubrics used to develop the scores. During analysis, raw scores were used in conjunction with survey data to understand how students' perceptions of learning aligned with their ability to demonstrate that learning.

3.5 Data analysis and validation techniques

The type of data gathered for this study included textbook and course packet critiques, three instructor interviews, three student surveys, classroom observations, and student grades.

Qualitative research methods admit the possibility of researcher bias, that is, the individual interpretation of data such as student responses or interview commentary. To minimize this bias, qualitative research uses a methodological approach to validation that is commonly referred to as triangulation [38]. Triangulation occurs when several sources arrive at or imply a single conclusion. One method for triangulation is to collect multiple types of data that look at various aspects of the same phenomenon. We have employed this technique by gathering data on materials, the instructor, and the students.

Another important aspect in minimizing the effect of researcher bias is to establish validity through agreement of conclusions by multiple people [38]. Results are validated when other researchers inspect the same raw qualitative data and arrive at similar conclusions. Pedagogical positivism validates data when a majority of the researchers reach the same conclusion.

For the data gathered in this study, two main analysis techniques were used. For open-response data, such as interview transcripts or essay items on the surveys, we used grounded theory's open coding technique to categorize the data [38]. In some cases, the categories were defined in advance and responses were pigeon-holed into these categories. For non-directed interview questions and open-response survey items, the categories evolved through a successive process of grouping similar responses into reasonable categories. The categories were reviewed for soundness by other researchers. Once the categories had been derived, descriptive statistics were sometimes used to show patterns and themes in the data. For multiple-choice responses on student surveys, descriptive statistics were used without coding. The remainder of this section describes specific techniques used for each type of data collected.

In analyzing the textbook and course packet, qualitative observations were compiled into a Microsoft Excel^{TM} spreadsheet (e.g., number of examples contained on a page of course notes). Other types of observations about these materials, such as number of student questions asked during class about a specific page of course notes, were recorded in the classroom observation field notes. Quantitative data created to describe the course materials were given members of the Science Education Student Dissertation Group (of which the researcher is a member) for validation.

As each interview was completed, it was transcribed and given to Dr. Quilt. She reviewed the transcript in order to check for accurate transcription. The final transcript of each interview was imported into HyperResearch^{TM} [93], which was used as a support tool for encoding the interview responses. Codes were produced by using key words that appeared in the transcript text. Refinement codes were produced on a second pass by grouping all the key word codes into related categories. The intensive scrutiny that HyperResearch imposed on the coding process was beneficial in understanding the overall themes in the data. The themes were written in prose with supporting interview quotes and given to Dr. Quilt as well as to a second researcher familiar with qualitative data methodology. Only themes validated by both Dr. Quilt and the second researcher were published.

The online survey data was downloaded into an Excel^{TM} spreadsheet. The multiple-choice items were numerically coded in order to run descriptive statistics from within Excel. The open-response items were exported into Microsoft Access^{TM}, a database program that is part of the Microsoft office software suite. In Access, columns were added to the database for coding the existent data. These new columns held the codes assigned to the open-response items. The codes were created by first giving the raw survey data (with student names omitted) to other graduate students in Science Education, who each created their own set of codes. All coding schemes were compared to what the researcher had created and combined into a final scheme. Once the codes were established, the data was grouped into categories by entering a code into the coding column. Access queries were used to discover themes found in the data. Themes were written up using student quotes and given to two other researchers. Only validated themes were published.

The field notes created during each lecture were compiled into several Excel spreadsheets in which specific aspects of classroom behavior were analyzed separately. Spreadsheets were created to record student interactions in each section, examples used in lecture, number of slides and topics covered in each lecture, attendance, and amount of voluntary and assigned homework. After the semester ended, these data were examined for patterns. The patterns were given to Dr. Quilt for her feedback. The feedback was incorporated and the modified results were published.

Raw scores from students who consented to having their grades in the course used as part of this research were collected and stored in an Excel spreadsheet. The grades were used in conjunction with survey data to individually examine how student perceptions changed with regard to performance. The raw course scores were used to subdivide the students into four performance categories. These categories were used to re-examine specific behaviors and perceptions that students in different performance categories had in CS 341. All patterns were discussed with Dr. Quilt and adjusted based on her feedback.

When a combination of data was appropriate for understanding certain themes, we reported related themes from different data sources. For example, when examining student perceptions (based on survey data) it was sometimes appropriate to state related interview perceptions (based on interview data). This technique allowed a better integration of findings from the separate data. The findings that resulted from the data collection and analysis described in this chapter are presented in Chapter 4.

3.6 Limitations of the study

Our study of CS 341 gathered data during the Fall 2002 semester. The semester comprised of 15 weeks of classes followed by a final examination. Although our focused research questions had been prepared before the semester began, specific survey and interview questions were finalized within a few days before deployment. Later surveys and interviews were customized to extract and expand on themes found in earlier data. One drawback to creating the surveys in this manner was the lack of continuity in tracking student perceptions over the entire semester. For instance, the first survey did not ask students to report whether they attended lecture, which prevented us from comparing student responses over the entire semester with classroom observations. This oversight could have been avoided had all three surveys, at least in part, been developed together. Another drawback was the way we scheduled classroom observations. We observed Section 1 on every Tuesday and Section 2 on every Thursday. With this observation schedule it might be the case that some of the interaction patters we observed were due to the differing days of the week and not differing section personalities. We do not think this was influential due to informal conversations with Dr. Quilt, but cannot rule out the possibility.

Due to the newly created admissions filter, we may have studied two different student populations. The Computer Sciences Department put in place an admissions filter in the Fall 2000 semester. That meant that in the Fall of 2002, juniors had been admitted into the degree program after the filter had been activated but seniors had not. This possible dual population may have biased our data in that senior students may not have had sufficient skills to enter the program had they applied for the degree when the admissions filter was in place. Some universities may more highly filter their students, and in these situations our findings may not apply as strongly. Automata Theory students who have entered Computer Science through a filter may be better equipped to study and reason about abstract material. However, we feel that even if the our findings are not as appropriate to the new population of students, the conclusions may still be helpful in creating a better environment, motivating students to do work and attend lecture, and improving the materials used in the course.

Chapter 4

Findings

This section focuses on the results that emerged from examining the data gathered for this study. We first present findings from the instructor interviews, followed by findings from the student surveys, classroom observations, materials critique, and student performance typing. For the casual reader, this chapter is rather weighty. If the goal is to gain a high level of understanding of the data gathered, we suggest reading only the summaries at the end of each section. The Summary Sections describe the most significant findings from the data. If a particular fact is interesting to the reader, he or she is advised to look at the section immediately preceding the summary for the details supporting the finding.

Our research resulted in a rich set of data. We wanted to use as much of the data as possible in our goal to obtain a broad understanding of the perspectives on both teaching and learning in Automata Theory. In support of this decision, most of the gathered data has been presented in this chapter. We first present an overview of the course under study by describing the physical settings and general content for the two sections of CS 341. Second, the instructor interviews are presented individually. When interview data is related to earlier statements, we point them out. Next we report on the three student surveys. When relevant we show the overlap between student and instructor perspectives. After the student surveys, we fold the classroom observation section into the building story. When the observations support or disagree with instructor or student perspectives, we explain the relationship.

The last two sections, material analysis and student performance typing, are somewhat outside of the accumulating story as they are findings that student and instructor perceptions were not necessarily based upon. Instead their use influences instructor and student perspectives on other aspects of the course. Due to this observation, we treated these sections somewhat differently than the previous data. The materials analysis section uses student commentary and our own expertise in order to critique and suggest changes that would improve the material used in CS 341. The student performance typing takes raw numeric scores (no semester grades are used) to look at trends among students that are ultra, high, average, and low performers in the course. This analysis was used in combination with instructor expectations to look at alignment between student perceptions and instructor views on learning.

4.1 General overview of CS 341

In order to situate the findings in this chapter, we first present a general overview of the physical location of each section of CS 341 and the lecture topics covered in the 15 week semester, which began on August 28, 2002 (Wednesday) and ended on December 6, 2002 (Friday). The course syllabus can be found in Appendix D.1.

4.1.1 Classroom descriptions

In the Fall of 2002 Dr. Quilt taught two sections of CS 341. Section 1 was taught in a large room that contained over 150 seats in a pie-shaped stadium-seating arrangement. The individual chairs had pull-out writing surfaces that are slightly larger than a regular textbook. The instructor was situated on a stage area that was located at the inner tip of the pie-wedge. A large screen could be pulled down at the back of the stage area, over the blackboard, as surface on which to project slides from an overhead projector. There was a rolling chalk board and a permanent white board located to the side of the pull-down screen to facilitate hand-written instructor notes. Dr. Quilt used the overhead projector exclusively in this section, using blank slides to create instructor comments. Section 1 met on Tuesdays and Thursdays from 9:30am until 11:00am. We observed Section 1 on every Tuesday.

Section 2 was taught in a medium sized room that contained approximately 50 seats. Each seat had a fixed writing surface that was kidney shaped and slightly larger than a piece of paper. The room was laid out in a rectangular shaped stadium-seating configuration. The instructor was located at the ground level in the front of the chairs. There was more than 5 feet separating a small table on which the instructor kept lecture notes from the first row of student seats. A pull-down screen was located approximately 5 feet beyond the table on which overhead slides could be projected. A permanent blackboard was located to the left of the pull-down screen. Two additional rolling chalk boards complemented the instructor area. Dr. Quilt used the overhead projector for lecture slides, and commonly used the permanent blackboard for instructor notes during lecture. Section 2 met on Tuesdays and Thursdays from 11:00am until 12:30pm. We observed Section 2 on every Thursday.

4.1.2 Lecture content

The topics of each lecture throughout the semester appear in Table 4.1. The labels in Table 4.1 are used in the remainder of this section. The published outline is included in Appendix C.

label	**Tuesday lecture topics**	**Thursday lecture topics**	label
(1.1)	*semester had not begun*	Big picture part 1	1.2
2.1	Big picture part 2	Strings and Languages	2.2
3.1	Regular languages (RL) Regular expressions FSM	Finite state machines (FSM) Non-deterministic FSM	3.2
4.1	Equivalence of RL and FSM	Closure of FSM on set operations	4.2
5.1	Regular grammars Pumping lemma for RL Non-regular languages	State minimization of FSM Pumping lemma for RL	5.2
6.1	Myhill-Nerode Theorem Equivalence sets of strings	Context-free (CF) grammars Parsing	6.2
7.1	Ambiguous grammars	Push-down automata (PDA)	7.2
8.1	Relation of PDA to CF grammars Chomsky normal form Greiback normal form	Top down parsing Look-ahead parsers Chomsky normal form Deterministic PDA	8.2
9.1	Bottom-up parsing LR parsers	Lex and Yacc introduction Pumping lemma for CF languages	9.2
10.1	CF closure theorems Pumping lemma for CF languages	XML programming language Decision procedures	10.2
11.1	Examination review	Turing machine (TM) notation Yields function	11.2
12.1	TM short-hand notation Recursive (R) languages Recursively-enumerable (RE)	Turing enumerable Partially recursive (PR) functions Equivalence of TM definitions	12.2
13.1	Non-deterministic TM Unrestricted grammars	Computing with grammars R and PR functions Context-sensitive languages Church's thesis Universal Turing machine	13.2
14.1	TM as transformers Halting problem Post-correspondence problem Non-RE languages Reduction proofs	*Thanksgiving holiday* *(no classes held)*	(14.2)
15.1	Reduction proofs Rice's theorem	Undecidable problems Review	15.2

Table 4.1: Timeline showing introduction of topics in each lecture.

In Table 4.1, all the lectures labeled with ".1" were observed in Section 1, ".2" lectures were observed in Section 2. The semester started with Lecture 1.2 since the Fall 2002 semester stared on a Wednesday and thus Thursday was the first day CS 341 met.

The first two lectures (1.2, 2.1) were dedicated to giving the students an overview of the theory they would learn over the semester. The rationale for the "big picture" introduction is discussed in Section 4.2.1, which reports the results from the first interview with Dr. Quilt. The two "big picture" lectures give students a brief high-level view of the material Dr. Quilt would cover over the semester. After the big picture lectures, Lecture 2.2 was dedicated to defining strings and languages, the building blocks used by the formalisms in the course.

The next four lectures (3.1, 3.2, 4.1, 4.2) introduced regular languages and finite state machines. Lectures 5.1-6.2 introduced non-deterministic finite state machines, non-regular languages, and the pumping lemma, as well as the basics of context-free languages. The first midterm was held Tuesday night after Lecture 6.1.

After the first midterm, eight lectures (7.1-10.2) focused on context-free grammars, normal forms for grammars, and parsing. One lecture (11.1) was completely dedicated to the second examination review; no lecture slides were used. Instead the lecture was used to solve example problems posed by the students or created by the instructor. The second midterm was held on the Tuesday night after Lecture 11.1.

The last eight lectures (11.2-14.1, 15.1, 15.2) focused on Turing machines, recursive and recursively enumerable languages, and undecidable problems in computing. The Thanksgiving holiday was observed on Thursday of the 14th week. The final examination was held after the 15th week at one of two times: December 11, 2002 from 2:00-5:00pm or December 14, 2002 from 9:00am-12:00pm. Although the University schedules separate final exam times for each section of CS 341, Dr. Quilt allowed students to attend either examination period. The majority of the students (90%) chose the second time offering (from observations).

In addition to lectures, a weekly discussion session was offered every Wednesday evening for two hours. This session was conducted by a teaching assistant from Dr. Quilt's CS 341. The same teaching assistant taught all discussion sessions. The sessions were designed to allow students to dynamically work through example problems or to discuss topics they had not understood from lecture.

Before each of the two midterms and the final examination, there was an extended problem session open to all students taking CS 341, regardless of instructor. The review sessions, each of which generally lasted four hours, were held the Sunday evening prior to the examination.

Student grades in CS 341 were based primarily upon the students' performance on the two midterms (a total of 40%) and the final examination (30%). Projects and required homework made up the remaining 30% of the grade. The published grading policy was part of the course syllabus and can be found in Appendix B.1.

The information in this section assists the reader in creating a groundwork on which to understand the remainder of this chapter. We present the data from the three instructor interviews in the next section.

4.2 Interviews

During the semester, all interviews with Dr. Quilt were held in her office. Generally she sat in a chair facing us, so that we sat face-to-face. Often she would either prop her feet up on the small table next to us or sit with one of her legs tucked under her on the chair. It was a very comfortable atmosphere. Before and after every interview we took a few moments to chat about life in general.

Dr. Lily Quilt is a petite woman who loves to wear brightly colored, loose-fitting clothes. When speaking to Dr. Quilt, one is struck by her high energy level. She talks quickly, often chuckling to herself when talking, and readily smiles when listening to others.

4.2.1 First interview details

The first interview was held on August 21, 2002, before the Fall 2002 semester began. The interview was held in Dr. Quilt's office in the early afternoon and lasted for a little over an hour. Questions used for this interview are located in Appendix F.1. This interview lasted longer than the later two because we were seeking information on course set-up issues by which to understand classroom details. All quotes in this section are transcriptions of portions of the interview; all transcriptions were reviewed and approved by Dr. Quilt.

Belief about Automata Theory curriculum

The interview started with Dr. Quilt describing her perception of Automata Theory and how she developed her curriculum. Her view was that CS 341 is unique in the sense that the curriculum choices are set in stone.

> "This course differs from almost all CS courses in the fact that the notion of what is an Automata Theory course has been pretty much set in concrete since the late 60's when Hopcroft and Ullman wrote their first book. You decide how deeply to go into material, you decide how many proofs to require of the students based on what they know and what you think they can do, but the material itself is almost the only stable class in computer science."

Previous learning experience influencing teaching

Because of this unique characteristic of the course, she has based her teaching of Automata Theory on how she was taught the course in college.

> "I teach roughly the same material, although at a different level of depth, that I learned when I took this class in 1972."

Dr. Quilt took Automata Theory as a graduate student, thus "roughly" translates into less depth and more accessible materials in CS 341 so that the course is better suited to undergraduates.

> “I would use the same book except that, well it’s out of print, and it’s not got much prose. It doesn’t explain much. Its just a theorem/proof theorem/proof type of book. So I use a slightly easier-for-the-students-to-get-to sort of book.”

Dr. Quilt’s inclusion of the two overview lectures stemmed from her desire to engage students who might not simply accept topics as they are introduced in class.

> “[Students] don’t have a sense of the theory we are going to build. So what I do is spend the first week essentially telling them everything I’m going to be telling them all semester really fast.”

Creation and use of materials in Automata Theory

At the time of this interview, Dr. Quilt had taught Automata Theory for over five years, and over that time had created a large collection of teaching and student materials. When asked how these materials were originally created, she said that she started from materials developed by other instructors. This helped her prepare for her first semester of teaching the course. In light of this, she has willingly shared all her course materials with other instructors of CS 341. When asked if she was unique in this, she said no. This sharing of materials helped keep new instructors in sync with other sections of CS 341. Dr. Quilt felt that this tradition of sharing also helped reduce the sense new faculty members might have of being overwhelmed with material preparation for their first semester of teaching.

Inclusion of programming projects

Even with the use of shared materials, the four sections of CS 341 differed slightly. Dr. Quilt required two programming projects in her course, something that the other instructors of Automata Theory did not consistently include. Dr. Quilt’s rationale for including these assignments was to tie programming, something she felt the students were comfortable doing, with the theory presented in Automata Theory. The first project focused on non-determinism. Non-determinism is a term used to indicate that programs have a choice to make with no criteria on which to base the decision (i.e., the decision is made arbitrarily). For instance, when a person approaches an intersection he or she could choose non-deterministically to go left or right if no other information was given about the consequences of either decision. This choice would of course impact future decisions, but at the exact moment of turning, the choice to go either left or right has equal probability of happening. In the first project, students were to simulate a non-deterministic finite state machine in order to explain how a computer could display this "choice" behavior.

> "[The first] project is to actually code a simulator for a non-deterministic finite state machine. A lot of the time people think non-determinism is magic. And I want to dis-abuse them of that idea."

The second project focused on parsing a body of text using standard Unix tools.

> "When you get to context-free languages there is the issue of how far do you go into parsing. And I [include this project] because although we have a compilers course in the department, it's not required and not that many people take it. So I thought every CS person is going to have to build front ends to things, like boolean query front ends to whatever, and they need to know that there are tools so they don't have to start from scratch to deal with parenthesized boolean queries. So I put a `lex` and `yacc` project into the class just because I thought this is something you shouldn't graduate with a CS degree and not know of the existence of so you can find it when you need it."

For those readers not familiar with Unix, `lex` and `yacc` are tools used to parse input. `Lex` is a lexical analyzer: a program that scans input and groups specific patterns into boxes or tokens. For instance, `lex` may scan the text from a document and group letters into words and white space into separators. `Yacc` takes the boxes or tokens that `lex` produces (words and white space) and creates user-defined structures from them. For instance, `yacc` might be responsible for creating sentences that contain specific syntactic elements such as a noun followed by a verb.

Perceptions on students' motivation

The biggest challenge Dr. Quilt faced was how to motivate her students to practice problem-solving outside of class. She felt that it was only through practice that students would learn the material in Automata Theory.

> "I was not a mathematician so doing the proofs was not natural to me and I did definitely learn through practice. I got better when I practiced. So I do feel when I stand up and say to the students that there is no magic here, you can't learn the violin without practice, you can't do this without practice no matter how talented you are, that I'm speaking from experience."

Dr. Quilt was self-motivated to practice, partly because of personality and partly because study habits in a graduate student are more mature than a normal undergraduate. She assumed one of her duties as an instructor was to provide the motivation for students to practice homework problems.

> "The biggest thing I've been playing around with, and I'm going to try a different approach this semester...how to get the kids to do the work! Now, CS students are not used to taking a theoretical class that has homework where they sit around and stare at walls and solve problems. They are used to a lot of coding, they know how to do that. But this material is hard and you have to practice. I know no way to learn how to solve these problems short of practice. You just have to do it for hours a week, and I have to figure out how to get them to do it."

Because this practice was so critical to learning in her course, she had tried various methods for motivating students to do the voluntary, or non-graded, homework.

> "That's the biggest thing I've played around with, is how to get them to understand from the beginning that if they don't do the problems on a regular basis they are not going to learn this stuff. There's no magic."

Dr. Quilt admitted that she had to temper ideas of student motivation with the reality of cheating. Playing with motivation to encourage students to do homework had to be balanced with the acknowledgment that anything done outside the classroom might give rise to cheating.

> "I told you that first I tried quizzes...they were taking up too much class time. So then I tried homeworks...and there are homeworks with answers in the packet that they have. I give them lots of problems to practice on but I didn't collect them. So after I dropped the quizzes I did nothing. That was the worst possible time. Nobody did anything. The last year or so, I've done homeworks that I collect and grade, that don't count for a whole lot, because I can't be sure that people did the work on their own. This time what I'm going to try, since they don't count for that much anyway, and since people don't always do the work on their own, I'm going to try telling them that they can work with other people."

The group work idea was new to the Fall 2002 semester that was about to begin. In the past, Dr. Quilt had focused on solitary learning. After talking to other professors, Dr. Quilt realized that she was hurting students who learned better in a group setting. Since she was skeptical about the increased chances of cheating under this new homework policy, the homework counted as only 10% of the total semester grade.

Management of prerequisite knowledge

Ensuring that students would do their homework was only one problem that Dr. Quilt faced. There was also the issue of students coming into her course with weak background knowledge. CS 341 requires as prerequisites two earlier theory courses, two semesters of calculus (which is part of the basic mathematics requirement for CS), and three semesters of beginning Computer Science courses that focus on basic data structures and programming. However, because courses differ from semester to semester due to different instructors, students entering CS 341 could have widely differing backgrounds. When asked if this would be a problem she said yes.

> "It's a big problem. And it's a problem because the courses are not very standard. I want students to know about proofs and logic. I want them to know that if I say $p \longrightarrow q$[1] that does not mean that $\neg p \longrightarrow \neg q$[2] but it does mean that $\neg q \longrightarrow \neg p$! Some of them don't. I want them to know about sets and relationships among sets. I want them to know that languages are sets of strings, because that's our fundamental object. And I want them to know about functions and relations and they know relatively little."

Although she acknowledged that many students might have missing prerequisite knowledge, Dr. Quilt stated that it was mostly up to the students to get themselves up to speed in order to be prepared for the material in CS 341.

> "I can't go back all the way and do it...In the supplementary material I've written about a twenty-five page 'this is everything about discrete math from a discrete math class that you really need to know for this class' and the first homework set is problems about that. And then I do a pre-test, I think it's about the second week. The problems cover that material. The test is self-graded and I tell the students, 'If you can't do these problems, then you don't have the background for this class.' I don't spend a lot of class time on it, I do give them material by which they can learn it and then a way to test whether they have."

[1]This notation describes a conditional logic statement. It reads, "If p then q."
[2]The symbol $\neg$ means negation, thus $\neg p$ is read "not p."

For the students who felt they lacked the knowledge to continue in the course Dr. Quilt was willing to let them drop. In fact, she structured the course so that the first examination results were available before the "Q" drop date (the last day a student can drop a course without affecting their grade point average). This policy allowed students who were poorly prepared to leave the course without failing.

> "I always make sure that the first midterm is before the Q-drop date. Now what they could also do is struggle to catch up, but, I do get people dropping at that point."
> *Interviewer*: And do you give them Q-drops then?
> "Yeah. I don't make anybody stay who doesn't want to... Now, computer science has a two time repeat rule. So at that point if they do drop that's one of their two shots. So they've used up one of their shots but they haven't messed up their grade point average."

Prior experience with students' learning

Other than insufficient background knowledge, we asked Dr. Quilt if she anticipated any problems that students would have with the material in CS 341.

> "Far and away the conceptually hardest bits are the ones at the end. The part about unsolvability... The hardest concept people have is doing reduction proofs in the end."

When asked how she had structured the course to better address this known area of student learning difficulty, she stated that she had tried to compress the earlier material in order to give more time to the difficult topics at the end.

> "I squeezed out minimization of finite state machines, because they do, well, a different algorithm, but an algorithm for minimizing finite state machines in the EE class, in order to get more time at the end. What I wish I could do is

> take some of this hard stuff and not have it come right at the end, but I don't know how to do that. It builds, like any of these theories, you sort of build up to it and the worst part comes at the end. I have squeezed some of the earlier easier material in order to make more time at the end."

Discussing how to spend more time at the end of the course when the topics became harder led us to discuss timing of topics during the semester. Dr. Quilt admitted that when she teaches more than one section in a single semester it is imperative for her to ensure that the schedule remains very close to the timeline published in the syllabus at the beginning of the semester.

> "Last semester is the only time I've taught only one section of the class, and I was much more flexible. So if on a particular day people were asking questions about something I'd kinda let it go and then hurry to catch up. Typically, every other time I've taught two sections, and I have to keep them in sync. Otherwise I'll forget where I am, I tell the students they can go to either time, ah, I'll make myself crazy. I have to basically, within a couple of slides, end up at the same place in both classes...that means I can't be flexible."

This notion that teaching multiple sections of a class creates less flexibility is an interesting observation, which we explore further in Chapter 5. In the Fall 2002 semester, Dr. Quilt was teaching two sections of CS 341. When asked for her initial plan for the semester, she responded that it was based on her experience from the previous times she had taught the course.

> "I publish the syllabus that's up there [on the class webpage] now, and I plan, that's my guide so that I know I won't run out of time. So I won't let myself get more than like half a lecture off from that. Because if I do, the problem is that the way I'll always get off is to be behind and then I won't have enough [time] for the hard stuff at the end."

Structure of lecture

With the published syllabus as the guideline, Dr. Quilt prepared overhead slides (transparencies) for each lecture.

> "They aren't PowerPoint, I use real slides. But yes, I do use real slides. The students have copies of them to take notes onto."

Dr. Quilt does not use PowerPoint or other technologies because the rooms in which she has taught do not offer consistent technology support beyond an overhead projector.

> "I've never gotten tied to what type of classroom I have to be in. And in fact often I have two classes that may not be in the same classroom. So, and not enough of our classrooms have visual capabilities for projecting or getting to the web and I have just chosen not to be one of the people who is limited by that. If I could be sure that I would always get a room that enabled me to do that, that would be cool. And that's why I don't have PowerPoint. I mean, I am low tech. It's on slides and paper and I can go and you can assign me to any room and I'll use it."

The use of slides, regardless of how they are created, has an unfortunate consequence; it leads to lectures that are primarily transmission-based. In a transmission-based learning environment the learner passively receives information instead of actively engaging in the learning process.

> "I would like to do way less lecturing and have discussion. But the only way I could do that is to assume they read the book first. We have to cover the material, so either I have to stand up and give every definition, and every piece of important information, and the first three simple examples, or they have to read that in the book, in which case we can come in and discuss the more interesting bits. My stance is that they won't read it in the book. So I have to pretty much in the class time that I've got, cover all the material."

She was open to constructing a more interactive lecture and welcomed the idea.

> "If I were to ask you what was the thing I wish I had a better answer to it would be this, this thing is too much of a lecture class, but given that I have to get through the amount of material I have, you don't cover as much in discussion mode. So could I really induce people to read the basic stuff and then let's just come in and do interesting stuff in class, as opposed to me having to spoon feed the basic stuff in class. I have not figured out how to do that."

Dr. Quilt admitted that if students were sufficiently motivated and read the materials outside of class time, prior to coming to lecture, she would change the course to be more interactive.

Changing the way CS 341 is taught, for example to be more interactive, may require changes to the materials used in the course. Materials prepared using electronic media are easier to update than hand-written materials. Dr. Quilt produced her slides from Microsoft Word documents, so in a sense her transparencies were just as easy to maintain as PowerPoint slides. Additionally, having the slides in an electronic format allowed her to print the slides she used and hand them to students so that they could take notes on the slides themselves. She hoped this helped students listen more in lecture, although she realized that many students assume since they have the lecture material they do not need to come to class.

> "No. Students don't come. So that's why I have very much worried about this."

She tried to stress the importance of attendance in the first weeks of class, explaining that she discusses material not on the slides and uses examples not contained in the notes, but attendance still suffered. After the second week, attendance dropped 20% in Section 1 and never recovered. Section 2 had better attendance until the ninth week of classes when it then dropped 14%.

View on changing materials for CS 341

When we asked about the flexibility of changing textbooks for the course, Dr. Quilt admitted that this was a large undertaking. Not only would she have to change the slides, but more importantly she would need to revise the large number of practice problems.

> "Nearly all books use the same conventions and definitions for finite state machines. But for both push-down automata and Turing machines there is not a single standard definition. All are provably equivalent but they are different. And so yes, if I were to switch to using this other book I would have to go though all of my...not just my slides, all of the worked out homework problems and do them over again. If there were a different definition for PDAs for example. And I admit that that is a daunting task."

Much of the work in revising the practice problems came from changing the notation used in the provided answers. When we suggested that perhaps the practice problems did not need completed answers, Dr. Quilt was quick to defend her decision to include the answers as part of the practice problem sets.

> "When I was a student, taking math classes, I was sitting in my room at night on my bed doing my problems and it made me absolutely crazy that I could not check my answers. Not that I needed to find out the answers, although that's sometimes hard. So I've done a problem. I don't want to go do another one and another one and another one not even knowing if I'm on the right track. It made me crazy. And I vowed that if I ever had anything to do with teaching that I was going to give people the ability to check their answers before they did five problems in a row wrong, or totally bang their head."

Again, we see the influence of Dr. Quilt's own learning experiences on her pedagogy.

Creation of out-of-class learning opportunities

In addition to practice problem sets and lectures, Dr. Quilt set up her course to offer other opportunities for learning the material outside of class time, including office hours, email, weekly discussion sessions, and examination reviews.

> "I have my office hours and I encourage them to email me in the evenings when they are working on their homework."

She admitted that she has generally received more email than office visits, but thought this was natural since this arrangement allowed students to ask questions as they worked on the material as opposed to a fixed time during the day. Dr. Quilt also has had one of her teaching assistants oversee a weekly problem-solving session. The session is scheduled for two hours every week, dependent on the teaching assistant's personal schedule and classroom availability. The time for the discussion session was set during the first week of classes. The sessions are not designed to substitute for lecture. In addition, before every midterm, Dr. Quilt had set up a longer, more intensive problem-solving session focused on working problems similar to what was to appear on the examination. These sessions were held on Sunday evenings so as not to conflict with other course offerings. Each review session generally lasted more than four hours.

> "I have a problem session where one of the TAs goes for like 2 or 3 hours, has some prepared material, but mostly the students are supposed to come in and say OK I don't know how to do problem so and so, I don't understand how this happened on the slide, and someone who's different from me, whose approach to solving the problem is not necessarily mine may give a different treatment and say OK let's try to solve this."

The problem sessions were not part of the published course schedule. These four mechanisms (office hours, email, weekly discussion sections, and examination reviews) provided students with several opportunities to seek help in understanding the material outside of lecture.

Learning objectives for students

When asked for the learning goals of her course, Dr. Quilt rattled off a list of topics with which she wanted students to be familiar:

> "There is a set of things I want them to learn. So for example I want them to learn how to write regular expressions. I want them to learn to write finite state machines, context-free grammars, PDAs. I do try to get them to write some unrestricted grammars. I sketch out how some Turing machines would work, I don't make them write them out in excruciating detail. And I want them to be able to prove things about whether a language is or is not in one of these classes."

But overriding these course-specific topics, she wanted students to be familiar with formal mathematical notation.

> "What I really do, throughout the whole thing, is to try to get them just to feel comfortable with formal notation. Which many of them are not."

When asked to clarify if the homeworks were intended to increase topic-specific knowledge, or whether they were aimed at increasing familiarity with formal notations, Dr. Quilt agreed that the homeworks might accomplish both of these goals. However, she described the main focus of the homeworks as a means for immediate assessment of student learning.

> "The point of the homework to turn in is more for me to tell whether they have learned anything, than to be the actual learning vehicle. I see the homeworks with the answers as, in some ways, the primary learning vehicle. So this is also more of the evaluation vehicle although they do learn things by it."

The use of homework assignments as assessment tools made it imperative that the homework problems were planned well in advance. When asked if homework problems had ever been re-used, Dr. Quilt answered affirmatively.

> "I reuse the homeworks totally. They don't count for that much and coming up with good problems is too hard. Simple, elegant, good problems...there are not that many. If I couldn't reuse them, I'd be done for. Couldn't do it six years in a row."

Since the examinations were also used as assessment tools, we discussed the related issue of whether examination problems were re-used.

> "Here's the deal. I do reuse problems. If they have looked at every exam I've ever done, and memorized the solutions to every problem, then they will probably be able to get through. I try to make new things, but they may be able to get through a big chunk of it. On the other hand, unless they have a photographic memory, the only way to recall old solutions is to have learned something. So now my compromise, because I still don't want them to feel like, 'I'm going to get in there and I'm going to panic if I forget the definition of the pumping theorem,' is that they can come with one piece of paper with anything written on it that they want. So this allows them to bring definitions, and that kind of thing, but not the answers to every previous problem."

Although an interesting student artifact, we did not examine students' cheat sheets in this research. In an attempt to appease student anxiety in recent semesters, Dr. Quilt designed the examination problems to be easier than the homework. In earlier semesters, when the opposite relationship held, students were not happy. With the shift to make the homework problems harder than the examinations, Dr. Quilt hoped that students would feel motivated to work the voluntary homework problems. Additionally she announced in class several times that if the students worked all the voluntary homework problems, they should have no trouble with the examinations.

Description of grading algorithm

Dr. Quilt reported that she has used assessments in CS 341 primarily for grade definition, rather than for teaching feedback. We asked Dr. Quilt what algorithm she used for grade determination. She said that her general practice had been to delay the detailed criteria of final course evaluations ("A" versus "C") until the very end of the course. Instead, during the semester she produced histograms of student scores to allow students to compare their progress during the semester to the overall performance in the course. A sample histogram is shown in Figure 4.1. When final grades were to be determined, student performance on

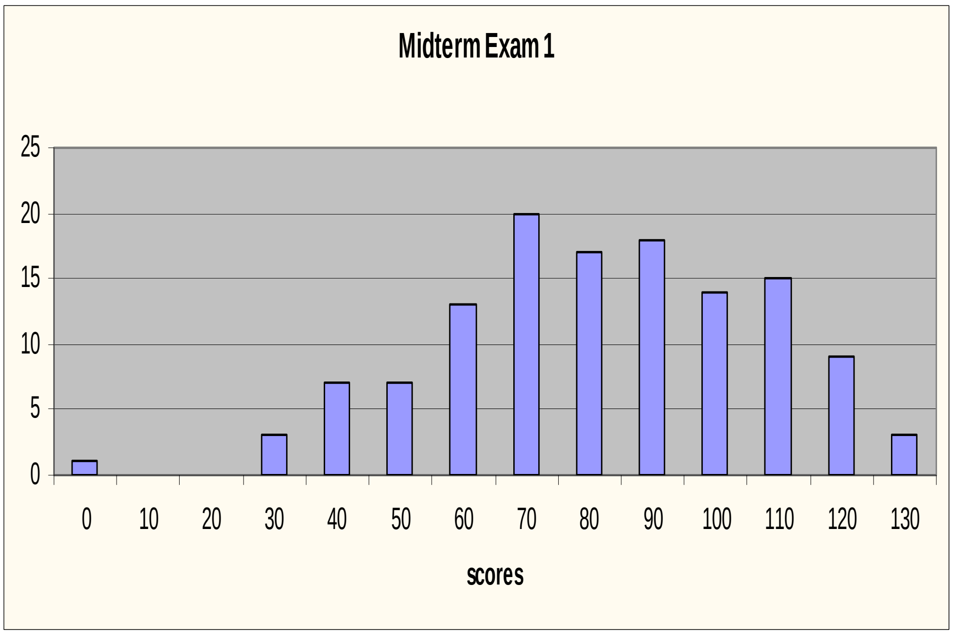


Figure 4.1: Sample of histogram shown to students to allow for comparison of outcomes on examinations.

the final examination would be critical.

"I don't assign A's B's C's to a particular midterm or whatever. What I do do is put up the histogram. So people can tell that this exam was just way too long and nobody finished. Then everyone can tell that the highest grade was a 70 and I got a 69 so hey! I'm doing great! Or, I'm in trouble. So that's the feedback that they get. At the end, I apply my weighted formula and I get the sum for everybody and I work by that. And I have a rough idea of around 15% A's and maybe 20% B's. But then I look for breaks and then what I do is I look at around the people...I draw a tentative line. And then I say, OK, who are the people who are near that line. At that point I have their final exams, and I actually go back and re-look at what they wrote. In other words, you can imagine two people who got the same score, one of them made a stupid mistake on a problem, misread the problem or something and got no points, but did everything else great. Somebody else didn't do anything else right, they just hacked away at it and got partial credit here and partial credit here and partial credit here. I'd rather give the better grade to the first person, who appears, the things they did they knew they had it right, but just like misread a question versus these other people. So then I'll adjust the line, particularly the C-D line. I just read the whole final exam for the gestalt, and then say, am I willing to swear that you know this stuff and don't need to take this class again."

4.2.2 Summary: First Interview

From the first interview, we learned that Dr. Quilt has several teaching beliefs that are grounded in her own experiences in learning Automata Theory. First, when teaching the course she starts with several overview lectures in order to clarify the theory that the semester will build. Second, she relies heavily on having students practice to learn the material in this course. We also saw that Dr. Quilt is very concerned with motivating students to practice. Not only has she made a large body of practice problems available to students;

she has introduced discussion sessions and examination reviews to facilitate practice and to answer student questions.

The first interview revealed that Dr. Quilt has several unanswered questions about her teaching role in CS 341. How can she better motivate students to do homework? How can she make lectures more interactive and still cover the necessary material? Can she find a way to introduce the harder concepts, which are normally taught at the end of the semester, during the earlier parts of the semester? These questions and the specifics of the course were the key points we took from this first interview.

4.2.3 Second interview details

The second interview with Dr. Quilt was conducted in her office on November 4, 2002, the day before the second examination. Questions used for this interview are listed in Appendix F.2. This interview lasted only about 15 minutes for two reasons: her schedule at that time in the semester was particularly busy and the fact that we talked casually after class quite often. The purpose of this interview was to see how the course differed from Dr. Quilt's initial expectations and to investigate her mid-semester views on student learning.

Description of how course had changed from original schedule

When asked about which aspects of the course differed from her original planning, Dr. Quilt admitted that the pace was faster than planned in the initial schedule of topics.

> "I'm a little bit ahead and it may be because I don't get as much interaction in class so I cover more material in lecture, and I'm also...I intentionally got rid of some stuff, for example the details of what the yields relation is, a couple of the very detailed things I'm cutting out each time I teach the class in an attempt to leave more time for the more important stuff. So I am doing a few things at a higher level without going through every formal detail."

View on students' understanding formal details of material

When asked if students were expected to study and understand the formal detail on their own, she said no.

> "I want to make it available to the students who are theoretically inclined and might actually have some intellectual curiosity and notice, but the average students, if I get this much most of the time, then that's not important for them to get."

Specifically when she herself learned Automata Theory, she had wanted the details so that she could read more about topics that interested her. This desire influenced her inclusion of formal details in student materials.

> "I'm imagining that if I were a student in the class I'd want to know where to get or find these things out."

Even when she did not cover the formal details in class, the course notes and practice problems provided additional information about the formal material not discussed at length during lecture.

Perception of student learning

In describing student learning, Dr. Quilt seemed to base her opinions solely on examination performance.

> "Well, of course we have an exam tomorrow and we'll have a better answer. I will tell you that on the basis of the first midterm there were a few people of course who did excellently...the average was lower than I would have hoped. I think we have a lot of students who still really haven't got it. And the sad thing is that they don't know they haven't got it. It would be one thing if they kinda realized they didn't get it but they are so sloppy as thinkers, so they don't know they've messed up."

When asked if the poor performance on the first examination was troubling given that the second examination would be harder, she said that she was indeed worried. She stated that the students were well versed in factual knowledge, but lacked the ability to make subtle distinctions that would lead to proofs that are absolutely correct instead of partially correct.

> "When they thought they had solved a problem (on the first examination), they did it in a muddled fashion. For example, they didn't understand the difference between a string and a language. You know, people would write things like, 'this string is regular' [instead of this language is regular], or they thought that regular languages were finite [and could not be infinite]. It's a very basic muddly thing."

When asked to be specific about what the students were and were not learning well, Dr. Quilt again based her perceptions on the first examination performance. She felt that students were understanding mechanical procedures but not understanding the types of problems that required abstract thinking.

> "On the basis of the first exam, they are better at mechanical procedures. Like for example, I'm betting that, there is a question on the exam tomorrow that says to convert to Chomsky-Normal form, they'll get that. They know how to memorize algorithms. So the same like last time, convert this non-deterministic machine to a deterministic machine. The things that will be hardest will be abstract things, for example there's a question on the exam that says are the CF [context-free] languages closed under this new operation that I define for you. So they are going to have to think it through. I have not given them a procedure for solving this problem. I've given them some similar kinds of problems, they know what closure means, they know that to prove it they have to show some sort of constructive algorithm for generating, and to disprove it they have to come up with a counter example. So I've given them that. But that's a very

> broad outline of how to solve the problem. Can they do that? Only the best students."

The idea of whether abstract thinking can be taught more algorithmically is discussed in Chapter 5. At the time of the second interview, Dr. Quilt had not thought of any algorithmic approach aside from giving students examples and similar practice problems.

> "I don't know how to teach abstract problem solving except you have to do a few and then ask people to practice. And it's very much like teaching art. You do a few and you ask people to practice and some people have talent and end up producing wonderful paintings and some people don't."

The result was that the exams included mostly procedural problems; problems that test specific knowledge taught in the course, and only a few abstract thought problems. She admitted that the criteria for passing the course would allow successful completion even by students who did not know how to solve abstract problems. Such students would not receive high marks, but could still pass without the ability to think abstractly.

> "I'm admitting that we are going to give C's in this class to students who don't have the ability to do abstract problem solving that I would like."

During the first interview, Dr. Quilt mentioned that many students come to office hours before examinations because students generally wait to review for examinations until the last minute. An apparent goal of these last-minute reviewers is to understand what they did wrong on past examinations so as not to repeat errors. This second interview was the day before the second examination, so we asked if Dr. Quilt had seen more students in her office hours. She said no. In fact she reported a lower rate of student interaction through email than she had experienced during previous semesters. A few students had been coming regularly to her office hours to ask how to solve voluntary and required homework problems, but aside from these few individuals, she had very few office hour visits from students.

Use of examination grades for student advising

If a student who had performed poorly on the first examination came to her office seeking advice, Dr. Quilt began the dialogue by looking at where he or she had lost points on their first examination.

> "What I've basically told people is let's look at why you did badly on the exam. Did you do badly because you made a bunch of silly mistakes and you were just careless and you didn't bother to check, or did you do badly because you had a fight with your roommate and didn't sleep the night before? In either of those cases...call that case one. Case two, you did badly because you really are confused. You can say, 'Yeah, I guess I really don't understand.' In case two, you are probably in trouble and you should drop because the next hunk of material is going to build on this. It will be similar but harder. So the context-free pumping theorem has two regions to pump instead of one, for example. If you didn't get the regular one, you're not going to be able to get this one. So you are probably behind and you should probably drop. In case one, the numbers are such that that exam is only 20% of the grade. You could get a zero and still get an 80% in the class. You are not hosed by the numbers. You are only hosed if you don't get it. So you've got to ask yourself whether you think you've got it. You could make it up in the numbers. And so I would try to talk to them a bit about that."

This excerpt shows that Dr. Quilt considered what each student brought into the course with them from their personal lives. It also demonstrates that she had the desire to be honest with students, yet encourage the ones who had a realistic chance to pass the course.

Perceptions on the two sections of CS 341

At this point in the second interview we had completed the set of questions addressing examinations. The next part of the interview focused on differences between the two sections of CS 341 and how those differences affected Dr. Quilt's teaching. In the first interview, Dr. Quilt indicated that section alignment was critical; we asked how closely aligned the two sections were at this point in the semester.

> "I think they are pretty much identical. We know that the two classes have very different personalities. And so in terms of how much excitement they feel about the subject, it may be different, but in terms of the material being covered, it's the same."

The "personality" of the two sections differed in Dr. Quilt's perception. Section 1 was much more interactive. Section 2 was quieter and there were generally fewer student questions. At one point early in the semester, Dr. Quilt actually announced to Section 2, "It is OK to talk in this class!" hoping to stir up some feedback (the comment did not even get many laughs). When asked if her perception of the better level of interaction in Section 1 was due to just a few people or many, she replied that only a few people participated.

> "In the morning section...if you get four or five people in the room who are really talking, even out of 80 that's pretty good. The others you can see are alive and paying attention. So there's that. In the second section, there are days nobody does [talk]."

Dr. Quilt realized that she was more enthusiastic about teaching Section 1 due to the handful of people who were talkative.

> "One of the things that I sense that I do, I've become aware of it fairly recently and I'm trying to decide whether I should stop...I know who's paying attention and the people I feel I'm having a personal dialogue with and I tend to speak to them! And look at them! And I don't even look at everybody else. I've started trying to consciously stop that, but it's attractive to talk to the people who are going to talk back! But I'm trying to stop that, and maybe get other parts, regions of the room involved."

Use of questions during lecture

One of the techniques Dr. Quilt used to get the students involved during lecture was to ask questions. However, many times her questions were simply rhetorical. When asked why she did not wait for students to answer, she said that time was an issue and her intention in posing the question was to evoke student mental engagement.

> "I'm not necessarily waiting for them to answer because I know that it would take too long. If I really waited for them to answer on every one. Sometimes I do. But it's my sense that even a rhetorical question makes people feel they are expected to be thinking, if not talking. I don't care if they answer out loud. I care if they answer for themselves. I care if they stop being in passive mode and try to think it through. If they are shy and they don't say it, although it's nicer to have some interaction, the real thing is get out of passive mode. I just hit you with a question. Most of us speak the language in such a way that when someone asks a question you sort of go, 'Ummm...I guess I'm expected to think about this.' So it's mostly–what I'm trying to do is get a sort of compromise between spending the amount of time it would take to really interact, which I can only afford if I believed they would read outside of class time, because I wouldn't be able to cover every piece of material in class."

Again Dr. Quilt's expressed belief that students will not work on material outside of class enters the picture.

> "I've taken the view that says if I need you to learn it I will cover it in class. I still need you to practice it, but there is no fact that I need you to know that I have not covered in class. I could go the other way and say, 'I'm expecting you to get the facts out of the reading and then we're just going to do examples in class.' But I think that that would probably be a disaster, because I don't think they would read! So the point of the questions is to sort of say, 'By the

> way, you should, if you've been following you should know the answer to this.' Most of the questions that I ask, as you know, are simple. I don't ask complex ones. I ask ones that I think if you've been following you should get them. So its sort of, 'Have you been following? You should get this."'

Thus Dr. Quilt's intention was that the questions should serve as a verbal thinking cue for the students, and also serve as a mental test for students to gauge whether their studying had been effective.

Beliefs about CS 341 reputation

The last topic covered in the second interview was the reputation of the course. From the student surveys, it was becoming apparent that many students considered CS 341 as the last "weed-out course" in the Computer Science degree. When Dr. Quilt was asked to comment on the reputation of CS 341 as a weed-out course, she acknowledged that it might deserve the reputation.

> "It is. It shouldn't be. In the sense that I think philosophically weed-out courses have to happen early in the program. I don't think you have a right to lead people down a path, to invest four years of their life and then suddenly find out that they can't do it."

Dr. Quilt admitted that although many educational systems have a required capstone programming course at the end of a Computer Science degree, she did not find that to be fair. However, she did admit that Automata Theory was similar to a capstone course in several ways:

> "This is in a way a capstone theory class. If you really can't get it, this is when we are going to find out about it. But my own view is that we should make the pass standards on the earlier classes like PHL 313K and 336 higher. So that if somebody is really not able to do the kind of abstract thinking that we think

> you need to do for this CS program, not all CS programs, but this one, that we tell you by the end of your sophomore year."

Belief that admissions filter would help admit able students

PHL 313K is a basic logic course and CS 336 is analysis of programs, also a theory course. Both of these courses are prerequisites for CS 341. Dr. Quilt hoped that the admissions requirement the Computer Sciences department put in place starting in Fall 2001 would begin to have the effect of identifying misplaced students earlier in the process. Under the new requirement, students wanting to major in CS must pass basic programming, basic logic, and some of the primary mathematics requirements with a sufficiently high grade point average in order to pursue a Computer Science degree. Dr. Quilt stated that perhaps this newly imposed requirement might identify students who were unable to master the material in CS 341 before they attempted to take it.

> "I think we are going to see something interesting in a year or so once the true effects of this admissions filter begin to trickle in because we may find that we have weeded and that it really is true that kids who get this [far] can do this. The reason it's too early to tell is that although kids who are now juniors have had to get through that process, a bunch of people put this off [until they are seniors]."

4.2.4 Summary: Second interview

One salient point from the second interview was that Dr. Quilt relies heavily on examinations to judge student learning. Performance on examinations determines the largest part of the final grades in this course and distinguishes between students who can and cannot think abstractly. She admitted that she does not have an effective way of teaching abstract thinking aside from giving the students examples and having them practice solving those problems that require abstract thinking. She also expressed the belief that the ability to think abstractly is what defines the best students in the course.

At this point in the semester (week 11 of 15) Dr. Quilt had covered the material faster than she had originally anticipated and was pleased since this would allow more time to be spent on the harder topics near the end of the course. She also stated that she was interacting less with students during office hours and in email than she had in previous semesters. Two key outcomes of this interview was Dr. Quilt's definition of an interactive class and her use of rhetorical questions. She labeled Section 1 as "interactive" due to four or five individuals asking questions and speaking up in class. We also learned that her use of rhetorical questions is meant to stimulate student thinking so that students would not passively absorb the lecture material. These were the key points we took from the second interview.

4.2.5 Third interview details

The third interview with Dr. Quilt was conducted in her office on December 19, 2002. At this time the semester had ended, all examinations had been graded, and course grades had been turned in. Questions used for this interview are listed in Appendix F.3. The interview lasted about 15 minutes. The purpose of this interview was to investigate how Dr. Quilt viewed the semester as a whole, with specific attention to how the classes had progressed after the second examination.

Perceptions on student learning

In previous interviews, Dr. Quilt had made clear the importance she placed on how students performed on examinations. This observation led us to ask her to describe her reaction to the second and final examinations. She responded that student performance on the second examination was similar to what she had observed in past semesters. However, on the final examination she noticed an improvement in student ability.

"I guess it was about typical. I will say, I didn't notice this so much on the

> second exam, but on the final I do think there were fewer people than usual who were hopeless. I'm guessing that this is because we are beginning to see the effect of the department admission filter."

Dr. Quilt does not attribute this improved student performance to her own teaching, but rather believes that the departmental admissions filter is was having an effect. We suggested that perhaps the perceived improvement in performance was due to the fact that she had adjusted the schedule at the end of the semester to allow more time to cover the harder material, which was focused on in the final examination. She did not agree with this theory and again attributed the higher level of success to the admissions filter.

> "I got about the same amount of time at the end as last time. It's more than I had a couple of years ago. I've been slowly shrinking both the finite state stuff and the parsing stuff to try to make more time at the end. So, I think it's just that some of the weak students who in the past were able to get to where they would take this class and struggle mightily with it, have been told after 315, 'You're not a CS major."'

CS 315 is the second programming course in the bachelor's degree requirement, generally taken in the second semester of a student's freshman year. Under the new rules, a student would apply for admission into a CS degree program at about the time they completed CS 315.

Belief of needing more time for harder topics

The next part of the interview focused on the course. We asked if the extra time that had been created in the course schedule around the time of the second interview had helped as students tried to understand the concepts, and whether Dr. Quilt was happy with the new timing of topics. She expressed dissatisfaction, saying that she would like even more time at the end.

> “No [the students are not understanding] and there are two issues there. One is that it just comes at the end when everybody wants the course over, but I can’t think of any real way to change that without making that whole logical structure of the progression of the concepts so weird that it’s worth it. <*sigh*> And this stuff is the most abstract, and so, I don’t know, I’m thinking that I should try to get at least one more lecture, if I could push it one more lecture back, then maybe that would help because I could spend more time in class just doing more problems.”

The stressed importance for student practice for mastery is again apparent from this excerpt. During the second interview, Dr. Quilt admitted that she did not have a better way to teach abstract thinking other than repeated exposure. The above statement from the third interview indicated her desire to add yet another lecture to help expose students to more examples in order to facilitate learning. When asked how another lecture would be possible given the amount of material that needed to be covered, Dr. Quilt was unsure.

> “I think I’d try to go through finite state machines faster.”

Finite state machines are presented during the first third of the course.

Student learning objectives

Because the semester had ended, we asked Dr. Quilt what she hoped the students had taken from this course. She responded quickly on not only topic familiarity but on skills that she hoped students now possessed as a result of taking the course.

> “OK, so at the highest level, there are two things. Number one the content of this material and I’ll go into that a bit more in detail here in a second, and two is just familiarity with the notation and thinking abstractly. You know, the notion of functions on languages. Do they really need functions on languages? No, but do they need functions on complex things, complex sets, sets of strings

> that are themselves concatenations of basic data types, you know, thinking in those terms of hierarchical levels of building abstract things on top of other abstract things. Do I hope they got 15 weeks more sophisticated at doing that, and reading the mathematical notation, you know, 'The set of all x's such that there exists a y such that xy is...,' whatever, those are the kind of ways specs are going to be written all their life. So there's that general sophistication that's number one. And it is my bet that nearly everybody got some better at it. I don't know how you could not get some better at it. Some of them are not very good at it, but I would be amazed if they weren't better at it than they were 15 weeks ago. So there's that. Then there's specific content. Do I want them to know about finite state machines and regular expressions and how to parse context-free languages and that it doesn't take much to have a completely general-purpose computing engine and there are things that you can't compute, and that problem reduction is a way to say things about various kinds of problems where you know something about some other problem. Most of them got most of those things at some level."

It is interesting to note that these expectations are reasonable. Students should be 15 weeks more experienced and be familiar with topics. She did not mention mastery of material or complete understanding.

Grading policy

The semester grades had already been computed at the time of this interview. We asked Dr. Quilt to describe her end-of-course grading strategy. She responded that this semester she gave "great grades." A higher percentage of students passed the course than in earlier semesters.

> "You know it's always arbitrary. What percent do you give A's to. This time that there was a 20 point spread between the top A and the lowest A. That seems

> pretty big. But part of it was who the people who thereby scraped by with A's are, and there were a couple of cases who I knew knew the stuff, they've been coming to see me a lot, and I just kinda wanted to reward what they did. And pretty much the same thing happened at the bottom of the C's. It was almost driven by, I've gotta make these few people pass, and so I drew the line so that would happen."

We see that familiarity with students affected Dr. Quilt's decision in drawing grade cut-offs. When asked why she felt comfortable breaking away from her former grading patterns, when fewer people passed, she again cited the departmental admissions filter.

> "But there was also this sense that this big tail of the distribution is shrinking. So if the CS admission process does its job, then everybody ought to be capable of passing. Now that's not to say that out of 120 people in a given semester you are not going to get 10 who just decide to blow it off. Because their girlfriend dumped them or you know, whatever. They're just not going to pass this class or any other. There's that. But there should no longer be this set of people who just flat out can't learn this stuff. So I have this sense that I should be able to pass more people."

Taking this admission process to the extreme would mean that everyone taking CS 341 had the ability to pass. Dr. Quilt expressed that she would be comfortable passing all students in the course if they earned it.

Reflection on changes for future offerings of CS 341

When asked to reflect about future offerings of the course and to describe possible changes, no matter how unrealistic they may be, Dr. Quilt expressed a desire to have the weekly discussion section as part of the required course commitment.

> "You know what I would do if they'd let us, is I'd have as an official part of the class a problem session for an hour every week scheduled into the thing so

> no one could say, Oh I couldn't come. Where we use lecture to present new material and we use the problem session, maybe we even require attendance or something, to just work problems. Because, the people seem to get a lot out of that. But the way it is now, it has to be scheduled on a sort of ad hoc basis and then people say they can't come."

Unfortunately, University regulations on course numbering and scheduling make this change almost impossible. Aside from the old mantra of "practice is how students learn the material in this course," Dr. Quilt expressed that the unlikely solution of a weekly mandatory discussion session would solve her dilemma of wanting to both lecture and work with students more interactively.

> "I feel very torn on whether the class time should be spent as lectures or the class time should be spent interactively doing problems and there's not enough time for both and still to cover this much material. And I think we need to cover this much material. This is a class where there is sort of an agreement out there about how much can and should be covered in a semester, I can't dump any of the topics. I wish I could count on them to just read the stuff and then we would come in and interact in class, but I think that would probably be a horrid failure."

From this statement we can see that Dr. Quilt remains convinced that she cannot provide enough motivation to compel students to do homework that would allow more flexibility in lecture.

Belief about changing materials used in CS 341

One historical aspect of the course about which students complained was the textbook. They stated that it was not informative or worth the time to read. When we asked Dr. Quilt about the textbook she readily admitted that she knew the students hated the book. She had chosen the current textbook because she felt it was “somewhat readable” compared to others. When asked to hypothesize about the type of textbook to which she might change, Dr. Quilt said that the textbook should be more advanced than the one in current use.

> “I think if we are going to go any direction we should go not toward novice centric. We should be moving more toward intense. Again, remember that the weak third of the students are going away. So if anything we should be...there’s a real sense that the book we are using now is too novice...that in fact what we are doing is that we are dumbing it down and that what you want to do pedagogically is teach this much and hope they learn this much [indicates lesser amount]. If you only teach this much [the lower hand] then they’ll only learn this much [gestures even lower]. And that we should move the other direction.”

Dr. Quilt’s apparent belief that the new admissions filter will be an effective weed-out for this course was the driving force behind this new book discussion. If the weaker students were no longer enrolled in CS 341, it makes sense to make the material more challenging. The presence of weaker students in her course was one of the reasons she chose to keep the current textbook, which she believed was more novice-centric.

4.2.6 Summary: Third interview

This interview revealed a continued frustration with how to teach abstract material and how to find more time for topics that occur at the end. Dr. Quilt’s devotion to the use of practice problems remained, and she expressed a desire to work through more examples in lecture. We also observed a shift in her expectations of student ability in her course due to the effect of the admissions filter in admitting stronger students, thus preventing the weaker students from ever enrolling in CS 341. To stimulate stronger students, she suggested that making the course more challenging may be the best pedagogical adjustment to the course. These were the key points from the third interview. In Chapter 5, we discuss the overall themes from the interviews and how the data from these interviews align with the other data.

4.3 Surveys

In this section we present the data from the student surveys. The surveys consisted of yes/no, multiple-choice, and open-response items. Appendix G lists the items used on the three surveys. We present descriptive statistics to summarize the multiple-choice items and quotes to illustrate themes based on open-response items. Student quotes have been modified for correct spelling (to account for typos) and the instructor name has been changed to Dr. Lily Quilt to maintain anonymity. Clarifications, when needed, appear in square brackets, [...], to distinguish our comments from student responses. Student names are kept anonymous. For each survey, student identifiers were assigned in a manner that avoided possible identification of the student. These identifiers were not based on alphabetic sorting of the class roster, course ranking, or section assignment, and numbers were re-distributed for every survey; thus it is unlikely that student #5 on the first survey is also student #5 on the second survey. When we provide direct quotes, we include student identifiers. The identifiers were to illustrate that a wide number of student responses were used as the basis of results in this section. When a specific student number is used twice in the same survey, the quotes can be attributed to the same student.

We conjecture that the richness of the responses to the open-response items in the student surveys is due to the student population we were studying. Computer Science juniors and seniors are articulate and comfortable using online communication. This fact alleviated the need for follow-up student interviews.

The description of each survey begins with the dates they were posted online, the period during which students could complete the survey, and the number of responses we obtained. All three surveys appear in Appendix G. Table 4.2 states the specific location within Appendix G for each survey and summarizes the dates for each survey. Table 4.3 summarizes the number of responses received on each survey.

Following administrative details, each survey's results are organized into cohesive areas. Each area is presented in a separate sub-section which simplifies reference to partic-

	Appendix	Date posted	Announced deadline	Actual deadline	Timing in course
survey 1	G.1	Fri, Sept 13	Sat, Sept 21	Mon, Sept 23	12th class day
survey 2	G.2	Fri, Oct 18	Sun, Oct 27	Mon, Oct 28	week 7 of 15
survey 3	G.3	Mon, Dec 2	Mon, Dec 9	Tue, Dec 10	week 14 of 15

Table 4.2: Summary of survey distribution.

	# responses	# continuous
survey 1	103 of 118	—
survey 2	101 of 118	95 of 118
survey 3	95 of 118	83 of 118

Table 4.3: Summary of survey response rates.

ular areas of reader interest.

4.3.1 First survey details

The first survey was posted online Friday, September 13, 2002. The students were told they had until Saturday, September 21st to complete it. The survey was actually taken offline on Monday, September 23, 2002. This timeframe was chosen so the first survey was available directly after the 12th class day so as to avoid adds and drops changing the student population in the course. Out of the possible 118 students, 103 completed this survey, yielding a response rate of 87%.

The rest of this section is broken into the following categories: population characteristics, academic background, learning preferences, and perceptions of CS 341. The first survey is summarized following these findings.

Population characteristics

One goal of the first survey was to profile students participating in our study of CS 341. Students were asked for their age, ethnicity, gender, and maturity in the CS program. We also asked what students thought they would do after graduation and how CS 341 might help them achieve this goal.

Of the 103 students who participated in the first survey, 48 classified themselves as white or Caucasian; 38 classified themselves as Asian, which included Indian (south Asian), Chinese, Vietnamese, and Korean; 2 classified themselves as African-American or Black; 6 as Hispanic; 7 as a mix of ethnicities; and 2 did not enter their information. Females accounted for only 18% of the respondents. With 72% of the participants being either white or Asian, this population is similar to other CS courses at UT [113]. The average age of the respondents was 21.8 years and the majority expected to graduate in the 2002-2003 academic scholastic year, implying that most of the students in CS 341 were seniors. Almost all of the students reported full course loads (average number of hours taken was 12.8 college credits, with an average of 8 credit hours in CS). About half (52%) of the students reported working at a job an average of 17.3 to 22.9 hours a week. These students stated that their future goals was to get a job (81%), mainly in a Computer Science field or with a technical company.

> "Get a job writing code for a small- to medium-sized company. I don't really want to work for a monster company." (survey 1: student 76)

> "I want to get a job but I am not sure exactly in what capacity that would be or what company I would work for." (survey 1: student 59)

About a third of the students (30.4%) expressed an interest in going to graduate school either right away or after working for a while.

> "Get a job for a while and then go to graduate school." (survey 1: student 71)

> "Go to graduate school and get a job in Microsoft! ;)" (survey 1: student 53)

When asked to hypothesize how Automata Theory would be helpful to them, students reported several advantages, thus the percentages for advantages are not mutually exclusive. Many students (36.8%) stated that they thought CS 341 would help them improve logical thinking.

> “It will increase my problem solving skills and help me understand logic-related material that I may encounter.” (survey 1: student 7)

> “A lot of it. Seriously. What I don’t use explicitly will still benefit the way I think and approach problems. That’s what I think after our brief overview of the topics to be covered in the class. We’ll see later what I think depending on how well the prof presents the material and upon closer examination how useful the material is.” (survey 1: student 30)

> “logics is a very important part of computer science since just about everything in CS is logical.” (survey 1: student 10)

Some students (23.3%) thought the course might improve their programming skills.

> “I believe that this material should help me have a better understanding of concepts which will aid me throughout my programming career.”
> (survey 1: student 14)

> “Helps thinking/problem-solving process, will help with my programming ability and knowledge of computational processes.” (survey 1: student 44)

> “Definitely, despite my dislike of logic courses, I know having a deeper understanding of the designs of some of these systems will definitely help, whether it be text processing, compilers, etc.” (survey 1: student 101)

Other students (17.4%) thought that Automata Theory would improve their theory knowledge.

> “Firm foundation in theory. More confidence with dealing with the abstract ideas of cs.” (survey 1: student 18)

> “Because the theory is more universal than just learning a language or how to write a specific type of program, and therefore it will be the overall concepts that will probably help most” (survey 1: student 20)

Fewer than half (41.7%) either did not know how the material in CS 341 would be helpful to them or said that the course would not be helpful in any way.

> “I don’t think it much helpful after graduation unless I go to graduate school.” (survey 1: student 103)

> “Won’t help at all. I believe that many classes I’m required to take are only to provide job security for some teachers. Teachers Unions do have a lot of influence. Why do we college students have to take English, after all we’ve taken it for 12 years in grade school.” (survey 1: student 86)

> “I don’t. It’s a waste of time. The cs department spends too much time with theory and not enough time with application. I’ve never needed the material taught in this class before and I’ve been in the real world. It’s mostly for academics.” (survey 1: student 15)

Academic background

The previous section established the general background of students participating in the first survey. This section focuses on their academic background. In this part of the survey we included items to investigate how many students were repeating the course, why students had taken Dr. Quilt’s section of CS 341, and how well they did in their prerequisite theory courses.

Only 15% of the students reported that they were repeating the course. Most (regardless of repeating) had chosen to take Dr. Quilt’s section of CS 341 due to ease of scheduling.

> "Didn't really care as long as it fit into my schedule and didn't conflict with my other CS classes(only 1 time slot for those other classes)"
> (survey 1: student 10)

> "It was the one that fit best into my schedule - keeping classes blocked together around lunch." (survey 1: student 13)

> "The section was open and didn't conflict with other classes."
> (survey 1: student 15)

Some students had heard that Dr. Quilt was a good instructor and therefore specifically took her section.

> "CS 341 is probably one of the most important CS classes and I knew Dr. Quilt would do a good job of teaching it." (survey 1: student 11)

> "Having already taken CS341 with Dr. Quilt. I felt that it would be better to take it with her again. Also, her teaching style is very effective."
> (survey 1: student 3)

> "A friend who took the class last semester (Spring 2002) said there is more programming in Quilt's section. He cited since I am an above average programmer I would benefit from this. I choose the 9:30 section b/c the later section conflicts with another class, M 340L." (survey 1: student 30)

Most of the students in Dr. Quilt's course reported having taken both of the prerequisite theory courses during the previous two years. The average reported grade in the basic logic course, PHL 313K, was 2.88, or a high C average. The average reported grade in the analysis of programs course, CS 336, was 3.08, or a low B average. This would suggest that these students had a reasonable background in theoretical knowledge, although they were a bit weak on logic.

Dr. Quilt had hypothesized during the first interview that the students would have a weak logic background. To test this hypothesis, the first survey contained several logic questions for the students to answer. One open-response item asked students to describe an equivalence relationship. We "graded" each response on an A,B,C scale according to how accurately the description captured the key points from a discrete mathematics textbook [5] definition. The average was 2.26, or mid B level of response. The next open-response item asked specifically for the three properties that an equivalence relation must have; 93% of the students answered correctly. Most students were also able to answer correctly a multiple-choice item on identifying the converse of a conditional operation (84% students answered correctly) and an open-response item about cardinality (93% answered correctly). The last multiple-choice logic item, although it appeared to be a question about CS 341 content because it used phrases from the "big picture" lectures, simply tested logical manipulation of subsets. On last logic item, 82% of the students answered correctly. The high success rate on these survey items would suggest that the students in Dr. Quilt's courses had sufficient background knowledge. However, excluding the definition of an equivalence relation, only 61% of the participants answered the other four background survey items correctly. This suggests that Dr. Quilt's hypothesis that the students come into CS 341 with weak logic backgrounds may be correct.

To help students with weak backgrounds, Dr. Quilt has designed the first homework and readings in the student packet to facilitate review of requisite materials. To investigate whether this had been effective, we asked the students if they had been doing the homework. Fewer than half (43%) of the students reported having done any homework by the time of the first survey. The students who reported doing homework estimated spending approximately thirty to ninety minutes per week reviewing the material or working problems in the course packet. In the background section, 40 students reported that they had received a "C" average in one or both prerequisite theory courses. However, only 18 of these 40 students (45%) reported doing the review homework. The review would be most helpful to these 40 students. This means that 55% of the students possibly needing review were not proceeding in a satisfactory way for reaching the necessary background understanding. This could be due to the fact that the survey was very early in the semester, and perhaps students needing review were planning to do the review homework at a later date. We cannot rule this out. But it is apparent that the background review material was not being utilized the way that Dr. Quilt intended.

Learning preferences

Because homework was such a critical learning vehicle for CS 341, we included a section on the first survey to investigate students' preferred study habits and grading preferences. The first survey item asked whether the students preferred to study alone or with other people. Almost three fourths of the students (72.8%) reported that they study alone. A follow-up question asked students to hypothesize about the benefits of studying with other people even if they chose to study alone. The responses fell into two broad categories: How studying with others would benefit learning the material and how it would help with the class in general. Most students reported benefits that were appropriate for both these categories. Tables 4.4 and 4.5 summarize these findings. It is interesting that even while the majority of students admitted a preference for studying alone, they recognized multiple advantages in studying with other people.

Category	# responses
Learn through explaining material	20
Different way of thinking	16
Another point of view	12
More information	8
Deeper understanding of material	5
Motivation to study on own	4

Table 4.4: Students' thoughts on how studying with other people could benefit learning in CS 341 (based on 65 responses).

Category	# responses
Help understanding	30
More confidence	12
Help solving problems	11
Correction of mistakes	5
Get answers	4
Reminder of forgotten facts	2
Get advice	2
Material made more concrete	1

Table 4.5: Students' thoughts on how studying with other people could help them cope with the material in CS 341 (based on 67 responses).

The next set of survey items investigated what motivated students to do homework, and when they generally scheduled and completed the work. We found that 51% of the students reported at least one positive motivator for doing homework.

> "Desire of knowledge for knowledge's sake, enjoy learning, good grade, appreciate a challenge." (survey 1: student 104)

> "Intellectual curiosity, subject interest, something to do, I've been in school (and paid/ am still paying for) almost 10 years and it's about time I got off my butt and finished" (survey 1: student 76)

However, the vast majority (83%) reported a strong negative motivational factor for doing homework.

> "Feeling like I have to study. If not, I will fail in this class."
> (survey 1: student 103)

> "I have to study; as Dr. Quilt said on the first day, this isn't easy stuff, it's something you have to keep practicing at to get more efficient"
> (survey 1: student 101)

> "I want to pass this subject so that I can graduate." (survey 1: student 88)

> “in order to make a good grade, I must study. I need good grades to get into grad school/get a good job. This does not apply to all of my classes”
> (survey 1: student 79)

The fear of failing or doing poorly in the class was a prevalent theme in student responses. This would suggest that if the homework contributed more to the overall course grade, greater student motivation for doing the homework could be a natural outcome. When the students were asked about how they scheduled time for their homework, 67% admitted to procrastinating until deadlines at least some of the time.

> “I always procrastinate and do it before it is due. For some reason I am incapable of starting any earlier. I imagine this has something to do with overestimating my abilities and underestimating the problem at hand.”
> (survey 1: student 21)

> “I try and keep up with it on a regular basis, usually trying to do it at work and late at night. But more than half of it is done the night before.”
> (survey 1: student 33)

Others admitted that they try to work on homeworks in a steady progression in order to keep abreast of the material.

> “I start it early, work on it little by little, and get it done by due date.”
> (survey 1: student 4)

> “try to do it early or at least some of it, so i can get some sleep the night before its due.” (survey 1: student 65)

And 3% of the students admit that at times they do not do homework.

> “usually I tried to finish my hw couple days before it’s due. But if I have too many hw that due on the same day, I might not be able to finish them all” (survey 1: student 49)

"also depend on the subject. some i finish the day it is assigned some i don't do at all." (survey 1: student 10)

The last general aspect we asked students to consider was how they preferred their grades to be determined. Because assessment in most college-level courses is tied to grades, the responses to this open-ended survey item would indicate whether students agree with the particular grading policy in CS 341. Table 4.6 summarizes student comments about grading preferences. The preference for giving more weight to homework differs from Dr. Quilt's

More weight given to the homework	40
Mostly by examinations	33
Other	(29)
Mix between examinations and homework	14
Projects listed explicitly	13
No homework as part of grade	1
Attendance	1

Table 4.6: Students' grading preferences (based on 104 responses).

current policy in CS 341, where homework accounts for only 10% of the grade. It is also interesting that these students seem to draw a distinction between coding (projects) and homework.

To understand the students' views on learning activities in general, students were asked to choose activities that help them learn. The four choices listed on the survey were: reading, listening, homework, coding. An open-response area allowed students who had additional ideas to elaborate on learning activities that they found helpful. Listening was the most popular learning mechanism (84%) followed by homework (68%). This would suggest that to these students lecture is critical. It also suggests that for these students, problem-solving through homework may not be as effective as discussion sessions. The ranking of the last two categories was close, with coding (59%) slightly more popular than reading (56%). Other forms of learning listed in the open-ended part of the survey item were: group work, discussion with peers, looking at examples, and seeing the problems in pictures.

Given that the majority of the course grade in CS 341 was determined by examinations, we asked the students to hypothesize what grade they would receive in the course. Most of the students (54%) predicted that they would earn a B. A third of the students (33%) hypothesized they would receive an A. Out of the remaining students, 11% thought they would earn a C, and 2% did not answer. In the Student Performance Typing Section of this chapter, this best guess at performance is compared to the final assigned scores to see whether students had an accurate view of their theoretical abilities.

Perceptions of CS 341

The next section of the first survey investigated what the students thought about participating in class, what type of class they preferred, what they thought of the materials provided in CS 341, and how they utilized other learning opportunities that were set up for the course. Throughout this section, the textual multiple-choice options are converted to a numeric Liker-style scale. This translation eased the analysis and interpretation of student responses by providing a basis for calculating mean values.

The first survey item addressing specific information about CS 341 asked whether students would like to participate during lecture. After conversion, the 3-point Likert scale equated 2 with “yes, on a regular basis” and zero with “no, I prefer to listen and observe.” The average value of 0.617 can be interpreted as meaning that for the most part, students wanted to participate only on occasion, or that they would rather listen and observe in lecture. The next survey item asked whether the students had participated in class up to the time of this survey. Only 36% of the students perceived themselves as having participated in lecture. Because we did not ask them to define participation, it is not clear whether "participation" was seen as attendance, or as speaking out in class.

Preferences regarding participation are one aspect in understanding students' ideal class environment. We also asked for the type of class they preferred, the converted 3-point Likert scale equated 2 with being "straight lecture" and zero with "small group interactions." The mean of 1.17 suggests that on average students preferred lecture with some whole-class discussions. This finding, on first consideration, seems to contradict students' reported reluctance in participation. Perhaps the respondents prefer to listen when other students discuss the material in lecture. A follow-up open-response item allowed students to fill in their own class style preferences. The open-response answers included: lecture with very little discussion, straight lecture with small group projects, lecture without overhead slides, smaller classes for more teacher interaction, and Moore method [92] courses.

A feature of Dr. Quilt's sections of CS 341 is that the students purchase the lecture slides as part of a required packet of materials for the course. Dr. Quilt created this packet with the hope that students would take better notes and listen more during lecture instead of trying to copy everything from the slides. An open-response survey item investigated whether the students appreciated this material or thought it was unnecessary. The students overwhelmingly (93%) liked having the slide materials available.

> "I'd have to concentrate wholly on note-taking and the notes I take do not actually go into my head at the time if I didn't have the slides."
> (survey 1: student 4)

> "I like being able to listen and not have to worry about copying all the slides. It is nice to be able to write in my own notes on top of the slides."
> (survey 1: student 9)

> "slides are good since they are mostly printed. thus sparing the students having to decipher a messy prof's handwriting (of course this doesn't apply to all profs) some have great handwriting =)" (survey 1: student 10)

This would suggest that Dr. Quilt's rationale behind providing the notes aligns with what the students appreciate about the course packet. However, even though students appreciated having the slides, they still acknowledged that having the notes may sometimes interfere with learning the material.

> "i like them because i don't like taking notes. i dislike them because i know taking notes is one way of learning the material." (survey 1: student 11)
>
> "Sometimes, you pay less attention" (survey 1: student 3)
>
> "It saves writing, and reduces tedium. On the minus side it's harder to be involved in the class, but I think the pluses outweighs the negatives."
> (survey 1: student 25)

The 7% of the respondents who stated that they did not appreciate the slides being printed out for them stated that it hindered their learning, or that they did not use the notes to help them learn.

> "I like to write things out - it helps me remember better."
> (survey 1: student 26)
>
> "I find that having the slides causes me to not pay as much attention"
> (survey 1: student 37)
>
> "It feels like we're reading the text in class. Plus I find that less sticks with me after a lecture, and I'm more confused with the material than I think I would be if it was more 'traditional'." (survey 1: student 51)
>
> "I don't take notes. Taking them usually makes me have to pay attention. Also it makes me learn what I'm writing down." (survey 1: student 64)

Another theme of having the notes emerged; the students equated having the notes as having the lecture and were therefore less worried about strict attendance.

> “i can still take notes on the examples and other things, but I don’t have to write down everything, and I don’t have to worry about accidentally missing a few details, as well as the fact that its easier to make up for missing a day of class”
> (survey 1: student 20)

> “They are incomplete. If you miss a lecture you may think that you have the material anyways, but it isn’t true, the slides can be missing vital parts.”
> (survey 1: student 37)

Because students have the notes, they may be lured into thinking they have complete lectures. This may be one contributor to the observed poor attendance in CS 341 lectures.

The next goal in the first survey was to assess how the course was going. Recall that students took this survey after only five of twenty-eight lectures had been held. Dr. Quilt had given the two “big picture” lectures, providing a complete overview of the course, and had begun defining the underlying concepts for understanding finite state machines (for an introductory overview of finite state machines, see Chapter 1). For rating the usefulness of the big picture lectures, the converted 4-point Likert scale equated 3 with “very helpful” and zero with being no help. The average response was 1.98. This suggests that the big picture lectures were helpful to most students.

Dr. Quilt has said that two of the reasons that students should come to lecture is to get help in completing examples that are partially worked out in the slides and to see additional examples not contained in the notes. We asked students if they found the examples worked in lecture to be helpful to their understanding of the material. The converted 4-point Likert scale equated 3 with very helpful and zero with no help. The average student response was 2.57. This suggests that students perceive the examples as integral to learning the material. This outcome dovetails with the preference of listening for learning reported earlier. This average rating also suggests that Dr. Quilt is correct in assuming that students value working the problems during lecture. Recall that during the third interview Dr. Quilt stated she would prefer to dedicate more lecture time to solving problems with the students.

The use of prepared packets with lecture slides by instructors and the resulting assumption that students take fewer notes can sometimes lead to a very fast paced lecture.

> "If they are going to use slides, I like to have them. I prefer for the professor to not use slides because I feel it forces them to go more slowly (and hence, more thoroughly) through the material." (survey 1: student 83)

For the multiple-choice item used to investigate the students' perception of the current pace of lecture, the converted 5-point Likert scale equated 4 with "too fast," and zero with "too slow." The average rating of 2.40 suggests that most students were happy with the lecture pace set in the beginning of the course. One factor that may contribute to students' perceptions regarding pace is the difficulty of the material: Easier material can be covered more quickly. To understand whether this acceptance of the current pace of class was because students found the material at this point was easy, students filled out a a multiple-choice item that rated the complexity of the current topics. The converted 4-point Likert scale equated 3 with "extremely frustrating," and zero with "very easy." The average student rating of 1.53 suggests that students found the material to be challenging but not overwhelming. The combined factors of students' support of including lecture slides in the course packet and their acceptance of the current pace of coverage and complexity of material makes including slides in the packets seem worthwhile.

As part of the course packet, Dr. Quilt included a large problem bank complete with answers and supplementary reading. Her intention was that students should work on or read the extra materials voluntarily in order to improve their understanding of topics taught in this course. The interviews revealed that inclusion of this extra material in the course packet stemmed from her own learning experience. To investigate whether students appreciated having these materials and or were using them, an open-response item asked how the extra materials helped students learn. Table 4.7 summarizes students' comments about the extra materials in the course packet.

In addition to the course packet, students were expected to acquire a required text-

Very useful - use frequently	43
Somewhat useful - use occasionally	39
Neutral - OK to have them, but do not necessarily use them	10
Not useful at all	0
Other	(10)
Useful but not using yet	7
Use only when doing required homework	2
Not sure of usefulness	1

Table 4.7: Students' comments about the usefulness of the practice material provided in the course packet (based on 102 responses).

book [67]. Dr. Quilt chose the text because she found it to be "readable" for the target audience and the authors used notational standards that were consistent with the lecture slides and practice problems in the course packet. However, the lectures were not based on the textbook (classroom observations). Given this observation, a multiple-choice question investigated how useful the textbook was to the students. The converted 4-point Likert scale equated 3 with "extensive use" of the textbook, and zero with "no use" of the textbook. The average student response of 0.617 suggests that students were not using the book. A follow-up open-response item allowed students to described their personal use of the textbook. Common responses were that they had been told by previous students that the book was useless.

> "people taken this class said no need for the book" (survey 1: student 69)

> "Haven't needed it yet. I've been told by people who have take the coarse that it's a waste of money. I haven't bought it yet." (survey 1: student 15)

A few students reported that the textbook was too expensive.

> "This is my second try at this course and teacher. I believe I failed the class

> because I didn’t do the homeworks. I saw the textbook (didn’t look helpful), the price (far too much), the teacher doesn’t use the book, and the lack of use from other students(they thought it was a total waste). So, I didn’t buy it last year, and I didn’t buy this semester.” (survey 1: student 2)

> “I had forgotten that we had a textbook (there is nothing worse than having to buy a $60 book and never using it because the same info is in the lectures)” (survey 1: student 62)

> “I’m taking 6 classes this semester, I’ve already spent around $6,000 for tuition, and I don’t have a lot of money available for books.” (survey 1: student 4)

Some students expressed frustration with the book content.

> “I [...] find the textbook to be somewhat useful. For some things, it gives a more detailed explanation than the packet. However, for some of the examples, it reads like stereo instructions. The packet seems to present much more useful & understandable examples than the book.” (survey 1: student 45)

> “it is hopeless.....just too theoretical....it feels like reading a research paper..instead of a text book for undergrads..” (survey 1: student 53)

Others stated that they had not needed to reference the book up to this point in the course.

> “I haven’t had a need to consult the textbook yet. If I had more time, I might read the chapters, but time is at a premium.” (survey 1: student 82)

> “I think until now I could find whatever I needed in the course packet, so haven’t really felt the need to refer to the textbook so often.”
> (survey 1: student 71)

Dr. Quilt reported in the interviews that the textbook is required rather than optional because having a textbook seems to make students feel more secure. However, with the apparent low use of the textbook, it might be time to re-examine this requirement. Perhaps the book should be listed as optional for the course if its inclusion is for student comfort.

Discussion sessions are held weekly by a single teaching assistant in CS 341. The sessions meet every Wednesday evening for two hours, but are not a required part of the course. To investigate whether this added learning opportunity was useful to students, students were asked to record how many discussion sessions they had attended. At the time of the first survey, students could have attended up to three discussion sessions. Out of the 103 students who took this survey, only 28 reported attending even one discussion session. A possible explanation for this low attendance at discussion sessions could be that students did not view the material as challenging at this point in the semester. Thus, as the material becomes more difficult, participation in discussion sessions may increase.

The last part of the first survey investigated in more depth what the students thought of the course. When asked what they found to be confusing, most students agreed that out-of-class work expectations were unclear.

> “definitions and informal homework expectations” (survey 1: student 48)

> “I don’t know when to do informal homework. It would be nice to have a suggested schedule for the informal stuff.” (survey 1: student 30)

> “Not sure what the first project’s gonna be about. Dr. Quilt hasn’t really told us much about it. I’m kinda nervous because I really hope I can do well in this class.” (survey 1: student 72)

Some students indicated that they were confused about a specific topic.

> “I think the nondeterministic finite state machine part and some parts covered after that are kind of unclear to me.” (survey 1: student 71)

> "Everything. In class, I understand about 75% of the lecture. But when I leave, only 15% stays with me, and I'm completely lost. It seems easier when the teacher is there, but when I'm on my own, its a lot harder."
> (survey 1: student 57)

To address the concern about out-of-class expectations, a simple aid would be to include a general outline at the front of the course packet. The outline could detail all the homework, both required and voluntary, and locate the homeworks along a timeline. The concern regarding confusion on specific topics is more difficult to address. Chapter 5 suggests possible teaching strategies to prevent students from becoming "lost" outside of lecture.

The next open-response item asked students to expand on what they currently liked and disliked in the course. As a whole, and sometimes individually, students seemed to be torn on whether they liked or disliked the current course topics.

> "I think the finite state machine part was really good. And also the parts she is covering now like intersections and stuff. I am also curious to learn about turning machines." (survey 1: student 71)

> "[I like] the mathematics coupled with language" (survey 1: student 51)

> "[I dislike] the material that is covered, it seems useless and boring"
> (survey 1: student 42)

> "It's boring material, and the hour just seems to drag on"
> (survey 1: student 23)

A majority of students indicated that they liked the current lecture environment.

> "Good structured class - I really liked the first 2 day overview to give us a wide-range perspective of everything we'll learn, reasons why we learn it, and how it all fits together." (survey 1: student 101)

> "I really like Professor Quilt. She speaks clearly and her explanations of the material make sense to me. I also just like theory. I'd rather learn theoretical computer science than go out and write a thousand lines of code."
> (survey 1: student 50)

At the same time, students were critical of the current lecture environment.

> "I don't like the way the lecture seems to sometimes bounce from one topic to another & then back again." (survey 1: student 45)

> "I don't like Dr. Quilt's style of presentation. She is like a streamed connection with no means for throttling. I find myself missing important topics because I'm still taking notes from/thinking about a previous comment. My beef isn't about lecture speed. She could cover the same amount of material and still be effective. It just seems to me that she doesn't take the time on important topics that students may likely need more time to absorb (eg. new notation). Take the extra time instead of rushing through and wasting time on some anecdote about currency. It bothers me to no end when a professor's solution is 'ask questions' during lecture. This isn't feasible with a group of more than 20 students. More often than not they have been teaching the same material over and over again for many years. They should have a general idea of how they should 'throttle' the lecture." (survey 1: student 21)

Eight-one students (79% of the respondents) expressed dissatisfaction with some aspect of the course. One of the final items on the first survey allowed students to recommend changes that would address their dislikes. Most of the recommendations centered around instructor activities in lecture.

> "Spend more time in class on harder examples that are similar to the ones on the exams. Most examples are simple so that we can understand but when

> exam time rolls around the problems will be much harder with less time to do." (survey 1: student 90)

> "It'd be great to see things, like finite state machines, other than on paper." (survey 1: student 55)

The theme of these comments seems to be addressing the type of examples used in lecture. This would suggest the accuracy of Dr. Quilt's assumption that students liked lectures in which she worked more examples. More time dedicated to working examples in lecture might allow the use of web-tools to illustrate concepts, since these types of examples take time to set up and execute. This approach could address the comment about examples via means other than paper and could also help students who are more visually oriented learners.

The next most popular recommendation focused on the course structure.

> "Make an alternate time for the Wednesday sections - I have a class that conflicts with them - perhaps the sections should be officially part of the class even if they are optional." (survey 1: student 30)

> "Make 50 minute long classes. Talk about uses of these topics in industry." (survey 1: student 7)

> "Have some non-graded pop quizzes just to let students check their performance from time to time to see if they are keeping up with the material or not." (survey 1: student 94)

The variety of comments demonstrate that students were not just passively putting up with the course but could readily conceive of realistic changes that would make the course operate better from their perspective.

4.3.2 Summary: First survey

The first survey establishes a sense of the characteristics and preferences for the student population enrolled in CS 341. Due to the majority of white and Asian male students in the course, the population in CS 341 is comparable to other CS courses at UT [113]. Almost all of the students reported carrying a full course load, and about half of the students reported working an average of 17-23 hours per week. Most of the students indicated that they were seniors and wanted to get a job in a Computer Science field after graduation. Fifteen percent of the respondents were repeating CS 341.

Students reported that they preferred to study alone, were motivated to do homework when they feared failing the course or when it was for a grade, and tended to leave assignments to the last minute. The students stated a preference for listening and doing homework when learning new material. Most respondents preferred courses to contain some whole-class discussion but to be mostly comprised of straight lecture, suggesting an acceptance of transmission-based learning environments.

At the time of the first survey, students indicated an appreciation for having the lecture slides printed out for them and that they found the slides useful. At the same time, having the slides may encourage a complacency about missing lecture as many students seemed to assume that the lecture material was contained wholly in the notes. The students reported that they were not using the textbook and considered it to be useless. Most students indicated that they were enjoying lecture (pace and content) but wished the lecture contained more examples. The students expressed a love/hate relationship with the material, stating that while they found the current topics interesting, they did not like having to learn them.

4.3.3 Second survey details

The second survey was posted online Friday, October 18, 2002. The students were told they had until Sunday, October 27th before the survey would be taken offline. The survey was actually taken offline on Monday, October 28, 2002. This survey was conducted approximately mid-semester in order to investigate how student perspectives had changed from the beginning of the semester. Out of 118 students, 101 completed the second survey, a response rate of 85%.

The second survey was structured to include questions about current understanding of topics in CS 341 and how participation in out-of-class activities had changed from the beginning of the semester. We also sought reactions to the first examination and first programming project. We ended this second survey by asking students to comment on their current perceptions of CS 341.

Student knowledge acquisition

The first examination had been graded and returned to students before the second survey was made available. On this examination, many students did not receive full marks on a problem that required them to construct a non-deterministic finite state machine. Grading of the examination problem did not reveal whether the confusion was because students had difficulty with the concept of non-determinism or the construction process. To investigate which factor students found difficult, we created a multiple-choice item for the survey that presented a non-deterministic finite state machine (see Figure 4.2) and asked students to determine whether the machine computed a specific function. The vast majority of students (94%) were able to answer this item correctly. This suggests that constructing non-deterministic functions, rather than recognizing them, was confusing to the students.

To determine whether students understood the more current lecture topics in the course, two multiple-choice items that tested students for their ability to recognize language hierarchies and define push-down automata. Only 62% of the students were able to correctly classify the given grammar into the correct language category, whereas 71% were able to correctly identify the function of a PDA. Push-down automata and grammar construction were more current lecture topics than non-deterministic finite state machines. Past topics

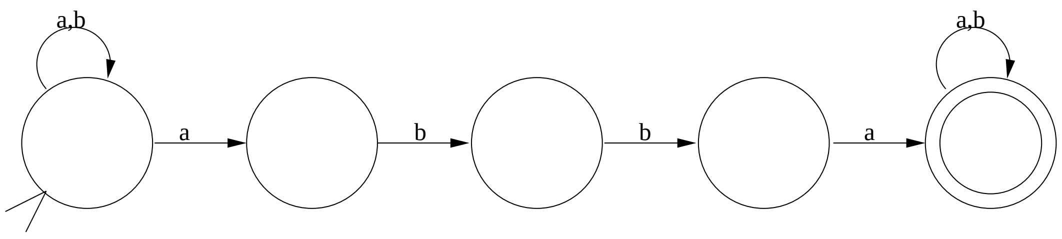


Figure 4.2: The non-deterministic finite state machine presented on the second survey to test recognition of non-deterministic functions.

being better understood is not surprising. In Automata Theory, new material builds on previous topics, thus the older concepts have had more rehearsal. A possible reason for incorrect responses on survey items related to the more current topics may be that students tend to procrastinate and wait to study until just before an examination (based on first survey data). Another contributor to the low rate of correct responses on current topic items may be that the current material was more difficult. To find out whether the students perceived the new material as harder, we asked them to rate the current pace and difficulty of the course through two multiple-choice items. On the converted 3-point Likert scale equating 2 with "too fast/hard" and zero with "too slow/easy," the average pace rating of 1.27 suggests general satisfaction with the current pace of the course. The average difficulty rating of 1.5 indicates that students felt the material was harder than at the start of the semester (when the comparable rating was 1.02).

Out-of-class learning

With the increase in topic difficulty and Dr. Quilt's repeated emphasis on the importance of practice for mastery of the materials in lecture, a logical expectation would be that more students would be attending discussion sessions and working practice problems. However, this was not the case. By the time of the second survey, only 26% of the students had attended even one discussion session, and slightly more than half (56%) admitted to

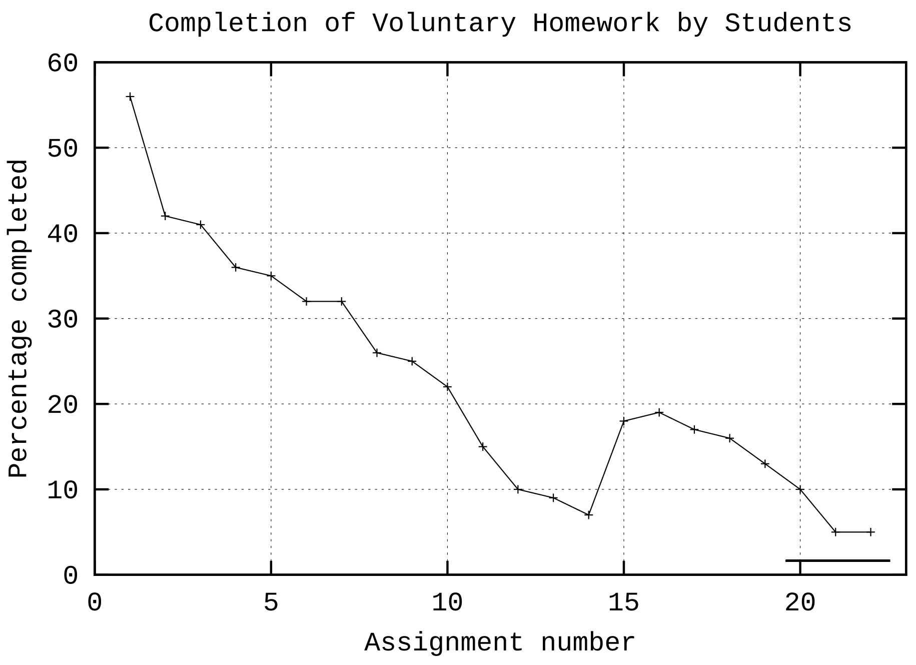


Figure 4.3: Percentage of student-reported completion on voluntary homeworks.

doing any of the voluntary homeworks. For each voluntary homework set, we asked students to report whether they had done all, part, or none of the problems. Of those students who reported doing the voluntary homework, most said they did not work even half of the problems. Figure 4.3 shows the dramatic drop-off in reported student completion of the voluntary homework. This would suggest that Dr. Quilt must revise the mechanisms she uses to encourage students to practice by working extra problems.

In contrast, when we asked whether the students had completed the required homework, which is turned in for a grade, 99% responded that they had done the required homework. When students admitted to not completing all the required homework, it was generally due to either missing lecture or poor time management.

"I have done some, but not all of the homeworks. This is probably because I have missed lecture a lot because the lecture is too slow and repetitive. I have a tendency to be bored if the teacher is not constantly pushing information at a fast pace." (survey 2: student 45)

"I don't remember to do them until it's already due." (survey 2: student 28)

The 99% reported completion rate suggests that students will complete homework that is graded, even when they count for only a small part of the overall course grade. In this offering of CS 341, students were allowed to work on homeworks collaboratively. We wanted to investigate whether students were taking advantage of this collaboration. A multiple-choice item asked how students were currently completing homeworks. The converted 4-point Likert scale equated 3 with "always alone" and zero with "never alone." The average value of 2.23 suggests that the preference for working alone from the first survey translated into preferred practice at the time of the second survey.

Responses on the second survey showed a slight increase in textbook usage compared to the first survey. The converted 4-point Likert scale equated 3 with "extensive use" and zero with "essentially no use." The average response was 0.69. This value increased from 0.61 on the first survey. On the other hand, the supplementary reading in the course packet received an average value of 1.56 (using the same 4-point Likert scale as the textbook survey item). This suggests that students considered the supplementary reading in the course packet the better reading resource. When asked if the lecture notes were helpful, 89% of the students responded affirmatively. On the first survey, 93% of the the students that the notes were helpful. This decrease may be due to the fact that the slides are intentionally incomplete. Students may have assumed completeness and subsequently not attended lecture, discovering later that they were missing important information.

At the time of the second survey, students had already taken and received their scores on the first midterm. We investigated how students had prepared for this examination, and used this information to establish a basis to determine whether preparation for this midterm would differ from later examinations. Most of the students reported studying alone (71%), some reported studying with at least one other student in the course (27%), and the rest (2%) admitted that they did not study at all. The preference to study alone is consistent with earlier findings.

For each of the two midterms and the final examination, Dr. Quilt set up examination reviews. The examination reviews were structured so that the review leader (a teaching assistant rather than Dr. Quilt) was available for four hours on the Sunday evening prior to the test. This scheduling requires a large commitment on the part of the teaching assistant. During this time, the review leader answered questions students had about the material on which they were to be assessed. To determine whether this extra learning opportunity was being utilized, we asked if students had attended the first examination review. Only 54% of the students responded affirmatively.

Reactions to the first examination

The second survey allowed us to ask about students' perceptions on the first examination. Sample examination problems taken from the first midterm can be found in Appendix D.1. This survey investigated two aspects of the first examination: students' perceptions about the fairness and overall experience in taking the midterm. When asked whether they found the first examination to be fair in the sense that all the material on the midterm was covered in lecture, 89% of the respondents said the examination was fair. In contrast, when asked if they thought the examination was a good evaluation of what they knew about the material from the first third of the semester, only 72% of the respondents agreed. When asked to sum up their feelings about the first midterm with a multiple-choice item, most students (59%) chose the "made some stupid mistakes" option. A follow-up open-response item allowed students to comment on what they would have changed about the midterm. Most of the students who responded to this survey item complained about a particular problem that contained a newly defined function called "nice." The problem asked students to use the newly defined function and prove properties about its application to regular languages.

> "the naughty and nice part was graded to harshly. it was also thrown in just to confuse people. We don't need any help failing the class."

(survey 2: student 24)

> “I thought the last question was unfair (7b I think). In most classes students aren’t taught material, they are taught how to learn material, for 7b to be fair, then questions with similar thinking patters (not same question or similar question, just one that the answer has to be constructed as such) should be asked on homeworks.” (survey 2: student 25)

Many students would have removed the “nice” problem from the examination if they had been given the chance. An interesting fact was that the “nice” function was defined during lecture on the day of the examination. Students who attended class should have already been familiar with this examination problem (observation notes). These suggestions imply that students wanted examination problems that could be solved without creativity, although one goal of CS 341 is to foster the ability to create unique solutions. Perhaps students were not aware of this learning objective.

Students also complained that the test focused too narrowly on proofs and did not sufficiently cover all the material from the first third of the course.

> “[I would have] included minimization and some of the other stuff we covered so the test would be more broad instead of just focusing on proving if a language is correct along with regular language problems”
> (survey 2: student 44)

> “I would have made the theory questions a little bit clearer and had a more build a machine to do this, what formula does this machine accept, simplify this regular expression, etc.” (survey 2: student 45)

> “I wish there were less proofs. I wish there were other stuff on there that we learned about eg. minimizing states, equivalence class, etc. There was so much we learned, and only about 30% was covered on the exam.”
> (survey 2: student 56)

These comments suggest that students wanted more operational type questions. Dr. Quilt acknowledged that students were better at solving these type of problems in the second interview.

Many students reported being upset about the algorithm used to assign credit on the examination.

> "The grading algorithm was ridiculous. Just by a stupid error, you would loose half of the points, even though it was completely clear that you understood the subject material. Also on the proofs, you would loose all the points if you were proving instead of disproving it. on those problems, i believe your justification should get you some points." (survey 2: student 3)

> "The grading policy. One person got a perfect score but the vast majority of the class was in the 65%-70% range. Some sort of curve should be implemented so the majority of the class doesn't fail because of one bright student."
> (survey 2: student 99)

This last comment shows the student's confusion on being shown a histogram instead of a grade on the midterm. Figure 4.1 is an example histogram that students were shown. Dr. Quilt did not assign any letter grades until the end of the semester when she did in fact use a grading curve. Apparently this fact was not understood by student 99.

Reactions to the first programming project

In addition to the first midterm, at the time of the second survey students had also completed their first programming project: to simulate a non-deterministic finite state machine. Appendix E.1 includes a full description of the project. The project counted as 7% of the final course grade. Almost all of the students taking this survey (95%) reported that they had completed the first programming project. When asked if the project had taught them anything useful, 74% of the students said yes. When asked specifically about what they learned, most respondents reported that they better understood non-determinism after completing the project.

> "It helped me understand the algorithm for running through a NDFSM, for sure." (survey 2: student 80)

> "I learned how to see how a fsm was implemented and it helped me understand non deterministic machines better." (survey 2: student 93)

Other students reported incidental learning.

> "i knew nothing about perl, but i was convinced to learn it for this project. it does so much, and now i am glad that i got some experience in programming with it." (survey 2: student 77)

> "Learned how to program a parser. I spent about 18 hours programming a robust parser and 30 minutes programming the finite state machine simulator." (survey 2: student 99)

A few respondents stated that they found the programming project to be a waste of time.

> "I did not learn a lot from the first programming assignment because the algorithm was given in the book and we just had to type it out."
> (survey 2: student 84)

> "Very little, just forced me to commit my knowledge into an implementation. Really more time consuming than anything else." (survey 2: student 63)

Perceptions of CS 341

In her initial interview, Dr. Quilt admitted that attendance had always proven to be a big problem when she has taught CS 341. Dr. Quilt reported that after about two weeks, many students generally stop coming to class. The classroom observations confirmed this trend of a decrease in student attendance. The second survey asked the students to comment on whether they were attending lecture. An astonishing majority (93%) responded that they attended lecture. However, the responses they provided to explain why they do or do not attend made clear that the students interpreted "attendance" to mean "mostly" instead of "consistently":

> "I didn't attend class last week because I was stressing out over CS 372 project, but I usually attend class regularly when there's no 'special' reasons like above." (survey 2: student 73)
>
> "I have missed 3 lectures. I was sick on each occasion." (survey 2: student 23)
>
> "If I have something do or a test i skip most of my classes that day studying for it" (survey 2: student 65)

Reasons students listed for attending lecture included understanding the material and clarifying the lecture notes in the course packet.

> "Lecture is absolutely essential to understanding. The notes alone are not sufficient. Neither is the text book. I learn more quickly by coming to lecture than the comparable amount of time I would have to spend outside of class. In other words, coming to lecture actually saves me time." (survey 2: student 29)
>
> "I attend lecture because if I try to go through the notes by myself, I don't understand it, but when I follow in class, I understand it." (survey 2: student 83)
>
> "I think it is essential to attend lecture since Dr. Quilt shows us examples, which we might not find in the notes." (survey 2: student 57)

In fact 93% of the students reported that they found the examples used in lecture to be helpful for their understanding of the material. When asked if they were participating in lecture, many students reported that they were participating in some way: About 33% reported asking questions during lecture, 41% reported answering an instructor-posed question, 10% reported posing a problem to be discussed in class, 15% reported correcting a mistake in an example, and 10% said they had participated in other ways.

Among the 7% of the students who admitted skipping lecture, many claimed that they learned the material better on their own.

> "I find the lectures boring, and I learn more by doing than by listening."
> (survey 2: student 3)

> "Sometimes Dr. Quilt explains material very fast and I just end up copying notes that she writes up on the board. So I try to copy, learn, and comprehend the material all at once. I get bogged down and get discouraged, especially when there are brighter students asking thought provoking questions and I still have no idea what anyone is talking about. So I usually stay home and try to teach myself the material. Mathematical logic was never one of my strong points." (survey 2: student 51)

To determine the courses on which students spent most of their time and why, we asked them to explain which classes demanded most of their out-of-class time. Eighty-one students named another Computer Science course as taking most of their time. The reason given by almost all of these students was that the other CS course included a great deal of required homework.

> "CS 352 [Computer Systems Architecture] because there is a lot of reading and there is a required homework due every week. Also, we have now started a big project that is worth a large portion of our grade." (survey 2: student 49)

> "CS371P [Object-oriented Programming], just because I have some heavy projects. Even though I love CS341 stuff, I don't have enough time left to study it." (survey 2: student 69)

The twenty-nine students who named CS 341 as taking most of their time cited the amount of assigned homework, both voluntary and required, as the reason.

> "CS 341. This has the most homework. And no, I don't like having to take this class." (survey 2: student 85)

> "CS341 because it is a more complicated subject. I mean the material is easy to understand, but coming up with a solution is not easy. Sometimes the solution to a problem is tricky. You either see it right away or you don't."
> (survey 2: student 89)

This last comment acknowledges the difficulty of solving abstract problems. Many students appear to view the process of solving abstract problems as non-algorithmic or "tricky".

On the first survey, one survey item asked students to anticipate the grade they would earn in CS 341. All students anticipated passing the course at that time. To follow up on this, the second survey asked students to comment on whether they were worried about their grade in the course, and more specifically whether they were worried about passing. Both survey items were necessary due to the fact that some students are concerned about not receiving an A or B, which is different from worrying about passing the course. More than half (68%) of the students reported they were worried about their final grade, and 34% reported that they were now worried about passing the course.

One aspect that may have contributed to students' concerns about passing the course was its reputation. An open-response item on the second survey asked students to comment on what they had heard about the course previously. Most of the respondents stated that they had head from friends or other students that the course was difficult, or that Dr. Quilt was the most difficult professor.

> "I heard from friends that this was a weed out course and that Dr. Quilt only passes a certain percentage of her class. This was verified by pickaprof.com."
> (survey 2: student 83)

> "other people - hard and suicide-inducing" (survey 2: student 94)

> "Most of my friends and a couple of people whom I don't know grimaced whenever I mentioned the words 'Dr. Quilt'. So I guess they thought it was very difficult class." (survey 2: student 8)

> "I heard it was the hardest CS class I would ever take." (survey 2: student 81)

About 7% of the students responded that they had heard something positive about the course.

> "I heard from a friend that this class was really tough. I especially heard that Quilt was a little difficult, but that I would learn a lot in her class."
> (survey 2: student 6)

> "A friend told me although Quilt had a reputation as being hard he didn't think she was too bad. He said I would probably do well b/c she has coding projects whereas supposedly the other prof doesn't. He thinks I'm a good coder."
> (survey 2: student 29)

The combined factors of having heard that the course was difficult and the lack of concrete grades on examinations and other required assignments make it understandable that the students were becoming worried about their progress in the course.

When describing what they did not like about the course, students gave a wide variety of answers. Many students explained that they wanted more lecture time devoted to complex examples.

> "The teacher jumps over examples in class and says that she doesn't think we need to do them, but there's no alternate way to find out how to do these examples (well we can ask TAs or the teacher in office hours but what if i can't make them a lot)." (survey 2: student 5)

> "Maybe that some examples don't get solved entirely during lecture, so getting a complete solution would be great." (survey 2: student 7)

Some students made clear that they did not agree with the degree program requirement of CS 341 or were altogether apathetic about the course.

> "I ABSOLUTELY HATE IT!! I find this class worthless, I will likely never use the material ever again, and I don't see why the university has to make this class a core required class!! (This class fails people about to graduate!)" (survey 2: student 32)

> "<try and be positive>...I like that it is in the morning." (survey 2: student 23)

Others students disliked specific attributes about the course or the course materials.

> "I personally feel that the pre-created slides work against me understanding the material. The professor is not required to edit themselves to only writing down the important parts of the lecture, Thus allowing the students to get a better idea of the important aspects of the material, and also giving them time to take notes as they have to write it down too. But everybody seems to do it." (survey 2: student 50)

> "The class notes are good. However they are compressed in such a way that sometimes impt. stuff is place at the bottom of the page.Better layout of notes will be great, where every major point is denoted by a new page or bold print." (survey 2: student 19)

To help track patterns of student confusion over the semester, the second survey contained an open-response item that asked students to explain what, at this point in the semester, was confusing to them. No student reported trouble with earlier concepts. The majority of respondents reported trouble with constructing proofs.

> “I find the proofs hard to do.” (survey 2: student 3)

> “The expected amount of detail in proofs. Many times I’ll explain something in great detail when I only need to say a few words.” (survey 2: student 8)

Our review of the results on the first examination showed that many students were having trouble with proof construction. In response to a yes/no survey item asking whether a lecture on proof construction would have been helpful, 70% said yes.

Another area where students reported trouble was in understanding the particulars of the push-down automata formalism.

> “Constructing PDAs is a little difficult for me now, but I’m hoping the practice homework should help.” (survey 2: student 49)

> “Context-free grammars and translating these into pdas, which I also have difficulty understanding.” (survey 2: student 39)

Similar to the first survey responses, students continued to comment on the difficulty of abstract problem solving.

> “just some problems that are tricky. i get everything once i see the answer, but it’s usually difficult for me to come up with the right answer myself.”
> (survey 2: student 53)

> “None of the material really confusing. The only problem I have is that there is no set procedure to go use to evaluate a problem. Most of it is intuition. I know all the material very well and still did not do so good on the test. This is because I do not have a keen sense of intuition. A little bit not fair, I think.”
> (survey 2: student 89)

Although the students complained about the current material being hard to understand, their descriptions of what they liked about the course indicated that even though the material was hard to grasp, the complexity was challenging and interesting.

> "I think the problems and material are really interesting. I enjoy puzzle-like problems." (survey 2: student 6)

> "I find the material interesting and intellectually stimulating."
> (survey 2: student 46)

Students also indicated an appreciation for the extra learning opportunities Dr. Quilt had in place.

> "Lots of support in the class with problem sessions and office hours and all."
> (survey 2: student 19)

> "The staff's help outside of class." (survey 2: student 53)

Some students liked the fact that they could see that the material would be immediately useful in other CS courses.

> "I can actually use this material in other classes, and the examples explain a lot" (survey 2: student 26)

> "I have seen the theory material come up in other classes as well as FSM's"
> (survey 2: student 45)

Similar to the responses on the first survey, students continued to express a love/hate relationship with the topics in CS 341.

4.3.4 Summary: Second survey

The results from the second survey showed that in the middle of the course, students were having difficulty absorbing the current materials at the current class pace. Students expressed difficulty in understanding the current lecture topics. Additionally students did not perform well on three survey items that were based current material. However, the results also showed that students were generally satisfied with the rate at which the material was being taught at this point in the course, although they admitted that the material had become more difficult in the time since the first survey. With the student reports of increased difficulty, a logical conclusion would be that students would be availing themselves of out-of-class learning opportunities. However, only slightly better than half of the students reported doing voluntary homework and approximately one-fourth of the students were attending the Wednesday night discussion sessions. In contrast, nearly all the students reported doing the homework that was to be turned in for grading. Nearly all students reported completing the first programming project; most students also reported that they thought the project was worthwhile. There was a slight increase in the reported use of the textbook between the first and second survey, most probably due to completing required homework and examination preparation. Student responses indicated that the supplemental reading portion of the course packet was a more valued reading source.

Most students indicated that they were satisfied with their performance on the first midterm and that they felt the examination was fair. For those who were not satisfied, reasons listed for being dissatisfied included their own stupid mistakes, one problem in particular was unduly tricky, or that the grading was too harsh. Responses on this survey revealed some frustration with the use of histogram grade reporting instead of A, B, C designations.

Students reported that for the most part they were still attending lecture, and that they found lectures particularly helpful when examples were worked in class. The trend of a love/hate relationship with the material continued, as did an apparent frustration with abstract problem solving. The abstract problem-solving frustration was exacerbated by students' complaints about the difficulty they experienced in understanding and creating proofs. These observations, combined with the larger required work-loads in other CS courses, seemed to contribute to the concern that more than half of the students expressed about their final grade in the course. At this point in the semester, 34% of the students expressed concern about passing the course. We hypothesize that the increasing difficulty and abstractness of the material, in addition to the course's reputation as a weed-out course, contributed to the students' fear of failing CS 341.

4.3.5 Third survey details

The third survey was posted online Monday, December 2, 2002. The students were given until Monday, December 9th to complete it. The survey was actually taken offline on Tuesday, December 10, 2002. This survey was conducted during the last week of classes before the final examination. Of the 118 students enrolled in the two sections of CS 341, 95 students completed this third and final survey, yielding an 80% response rate. The drop-off in student participation was most likely due to normal end-of-semester activities such as studying for final examinations, completing end-of-semester projects, and preparing for graduation and winter break. Students may become extremely selective about doing extraneous work at the end of the semester, which could cause them to view completing a survey as a low priority. The third survey content can be found in Appendix G.3.

The initial part of the third survey examined students ability to correctly answer survey items involving current course topics and how students were engaged in out-of-class learning. We then investigated student reactions to the second midterm and the second programming project. The last part of the survey investigated continuing perceptions of CS 341 and how well students felt their prerequisite theory courses had prepared them for this course.

Student learning acquisition

Each of the three surveys included items designed to test specific concepts in CS 341. The purpose of these items was to determine whether students were keeping up with topics presented in the course at the time of each survey. For this last survey, Dr. Quilt assisted with the design of three items with increasing difficulty. The purpose was to try to produce problems that would indicate student performance in the overall course. The intention was to design the items so that:

- Students who would received a high final course grade in CS 341 would accurately answered all three survey items.
- Students earning a moderate final course grade would accurately answer the first two items, but not the last (hardest) one.
- Students who would receive a low final course grade would probably only answer the first (easiest) correctly but not the other two items.

The first (easiest) item tested knowledge of the language hierarchy using the Turing machine formalism. More than half of the students (68%) were able to answer this survey item correctly. The second item was meant to test the undecidability result over Turing machines. This survey item was directly related to an example presented in class, however, only 32% of the students were able to make the comparison accurately. The last item presented a faulty reduction proof and asked the students to chose among three multiple-choice answers what conclusion could be drawn from the proof. This last problem was especially tricky in that it used a common student mistake to create the erroneous argument. Although students had explicitly been warned about the mistake in lecture and the course packet contained a slide describing the pitfall of creating an inaccurate reduction proof, only 23% of the students were able to recognize that no conclusions could be drawn from the proof given in the item description. Because there were three options, with random guessing, we would have expected 33.3% of the students to answer this last item correctly. The Student Performance Typing Section shows that correct responses on these survey items did indeed reflect end-of-course performance.

Out-of-class learning

During the three instructor interviews, Dr. Quilt had expressed on several occasions during the three interviews that the only method that is effective in learning this material is practice. On a survey item that asked if the students were doing the voluntary homework, only 29% responded affirmatively. This would suggest that Dr. Quilt's method for mastering the material was not being used by many students at this point in the course. Figure 4.3 depicts which voluntary homeworks did get attention from the students. The main pattern is that as the material becomes more difficult, the students spent less time on voluntary homework problems. This drop-off in student completion may have been due to increased work in other courses; however, if Dr. Quilt's message about practice had been received and accepted by the majority of her students, this is opposite from the trend we would expect.

For the students who did continue to work on the voluntary homeworks, the reasons they gave for continuing this work were consistent with Dr. Quilt's statements about mastering the material in CS 341.

> "they help a lot! i'd say they are one of the most important resources for doing well in the class. the solutions at the back really help." (survey 3: student 84)

> "I need the examples for practicing and becoming more comfortable with the problems." (survey 3: student 41)

Some students acknowledged the importance of practice, but because of various reasons, were not doing the voluntary homework consistently. Instead these students expressed the goal of using the voluntary homework problems as practice for the final examination.

> "A lot of other work to do. I will try to do them before the final."
> (survey 3: student 25)

> "I haven't had time to start studying for Automata yet. I'm taking 18hrs...5 cs classes. Won't start studying for a while." (survey 3: student 12)

The overwhelming reason students gave for not doing the voluntary homework was interfering demands due to other course-work that was not voluntary.

> "I had other school work to complete from other classes that were for a grade. So, I never went out of my way to complete those voluntary assignments." (survey 3: student 2)

> "i have other courses with a fair amount of work due in them as well that i must complete that is not voluntary" (survey 3: student 17)

Students also expressed that doing the voluntary homework was not an effective method for learning the material.

> "Doing the regular homeworks seems to be enough OR I do not seem to be able to gain any additional understanding from the voluntary homeworks to help me with the regular homeworks." (survey 3: student 26)

> "I felt/feel like giving up after the 2nd exam. I was very upset about the nature of the exam which lead to severe decrease in interest. I no longer wanted to attend lecture or do homework assignments because in the back of my mind I felt like it didn't matter anymore. The ambiguity of Dr. Quilt's current grading scheme only serves to compound this feeling." (survey 3: student 19)

The last response may be due to the histogram grade reporting method that Dr. Quilt employed throughout the semester.

The responses made clear that for many students, the voluntary homework was not an effective learning tool. We were curious as to whether students had taken advantage of the other provided learning resources. At the time of the third survey, the number of students who reported they were attending Wednesday evening discussion sessions had increased to 57%. The number of students reporting that they had been reading either the textbook (17%) or the supplementary reading in the course notes (30%) was still very low.

However, these results support the finding from the second survey that students were using the supplementary reading in the course packet more than the textbook. When asked if the lecture notes were helpful in studying or in completing assignments for the course, 77% of the students still indicated that the notes were useful. This value had decreased from the second survey in which 89% of the students reported the notes as helpful. Student responses to a multiple-choice survey item about studying at the end of the course indicated a continued preference for studying alone. The converted 3-point Likert scale equated 2 with studying "with two or more" students and zero with studying "alone"; the average value was 0.63.

There was an increase in the number of students who reported attending the examination reviews: 54% on second survey compared to 71% on the third survey. Students' written reactions to the review session for the second examination were mixed. Most students who responded reported that they gained information from attending, and preferred the teaching assistant's lecture style to that of Dr. Quilt's.

> "He's a very clear lecturer and gives good examples. Much slower talker that Quilt." (survey 3: student 12)

> "I liked that we went over a variety of problems, including the format that we're supposed to use for the exams. Things I found difficult to understand in class were easier to understand here." (survey 3: student 35)

Some students were not impressed with the format of the review, or reported that they liked the review but that it did not help them study for the second examination.

> "I disliked how it was across sections of the class. Some students in other classes were clearly not as familiar with material as our class. Also, (the TA) did some very VERY basic examples. I was hoping to work through more thought-provoking problems that were more likely to be on the test as opposed

to basic problems that were done in the notes."
(survey 3: student 44)

"It seemed like most of the people asking questions did not attend class..."
(survey 3: student 22)

"I liked the fact that I was very involved with the Lecturer at the review session. I was on my toes so to speak. I felt that the theory of 341 was excellently presented, although a few more 'exam' type of questions would have helped. Overall, I felt that it was a great review." (survey 3: student 34)

The first comments made by student 44 and student 22 refer to the fact that the examination reviews were open to all students currently enrolled in CS 341 regardless of the instructor. These comments indicate that it might be a preferable for each professor to create their own review sessions and then perhaps to have one in common.

Reactions to the second examination

The second midterm covered the formalism of push-down automata, context-free grammars, normal forms, and more complex proofs for showing language placement in the Chomsky Hierarchy. The material built upon the formalisms taught during the first third of the semester. When asked if the second examination had been fair in the sense that all material covered on the midterm had been presented in lecture, only 62% of the respondents agreed. Less than half (43%) of the students stated that the second examination was a good evaluation of what they had learned over the middle third of the semester. This suggests that a majority of the students felt the examination did a poor job of assessing their knowledge. When asked to describe their feelings about the second examination, many students (37%) stated that they had found several problems to be difficult. Other students complained about not having enough time to finish the examination.

"knew most questions did not have the time to finish" (survey 3: student 91)

> “i felt rushed, so i made stupid mistakes.” (survey 3: student 48)

There was a dramatic drop in the “fairness” rating for the two midterms (89% judged the first midterm to be fair, whereas only 62% rated the second midterm to be fair). This decrease might be attributed to student reports on the increased difficulty of examination problems and their frustration with the length of the second midterm.

> “I wish the problems on the exam were more closely related to the homework and practice problems. It doesn’t really help if I am able to do all the practice problems, but then the problems are so much harder that I don’t understand.” (survey 3: student 46)

> “After taking the exam I truly wonder what Dr. Quilt’s motives were. I have never been so upset. I felt prepared for the exam and felt I knew the material. I think many of the questions required reasoning skills outside the realm of the course material. That in itself isn’t a problem but when combined with the following: 1.) Too long 2.) Interdependency between steps/questions it is a recipe for disaster. We should have been warned ahead of time that most of use probably wouldn’t be able to finish the exam. The only advice was and I quote “This exam is really hard so just do the best you can.” The exam should have consisted of straightforward questions that concisely targeted different points in the material.” (survey 3: student 19)

> “I didn’t think that I had enough time to solve as many problems as I knew how to solve. I would’ve finished the test given another half an hour.” (survey 3: student 44)

> “I think some of the questions were much harder than problems we were exposed to in class. I think the test was too long. I don’t know why teachers feel that they must make students rush through a test. As a student that prepared

> very well for the test by pouring over examples and working problems, I knew how to do the problems but I didn't have time. I don't think it should be a test on who can work the problems the fastest and that is the way it turned out. My opinion because I have struggled with this a lot in the past is that there is not as much correlation between how fast students can work problems and how much they know the material. Of course there is some, but not as much as many think." (survey 3: student 41)

It is possible that the perception of the first examination as relatively easy may have set up inappropriate expectations among the students. We discuss this further in Chapter 5.

Although a majority of the students stated that they attended lecture (77%), observed attendance numbers indicate that this percentage is optimistic. The classroom observations showed a slight increase in attendance after each examination was returned, but then attendance would steadily drop off. When asked why they did or did not attend lecture at this point in the semester, students who indicated that they were attending said that they did so in order to better understand the material or out of fear of failing the course.

> "Teacher better explains the concepts with plenty of examples."
> (survey 3: student 15)

> "If there is no attendance, you might as well kiss your grade good bye. This material is hard to learn on your own. There are also to many tricks that aren't described well" (survey 3: student 21)

One reason students gave for not going to class included conflicts due to other courses.

> "Most is covered in notes, and I have had to spend a lot of times on multiple projects from several CS courses." (survey 3: student 13)

> "Laziness, weather, work in other classes" (survey 3: student 50)

Some students stated that attending lecture was not helpful in learning the material.

> “concepts are easier understood with someone else, and there’s something about the rhythm of her speech that never fails to put me to sleep regardless of how hard i try” (survey 3: student 88)

> “The lectures just don’t seem very helpful in learning the material.”
> (survey 3: student 7)

Both attending and non-attending students commented that the pace of the class had been so fast that they found it difficult to understand and absorb the new material during lecture.

> “I think lectures haven’t helped me since I think she goes through the material too fast. Reading the material myself and asking fellow students is helpful.”
> (survey 3: student 8)

> “she is moving so fast now, that even with attending lecture i’m still struggling. i cant afford not to miss a lecture now.” (survey 3: student 11)

> “The course moves too fast and I can not keep up with it.”
> (survey 3: student 46)

On a multiple-choice item designed to investigate the current class pace, the average value was 1.56 (a converted 3-point Likert scale equated 2 with “too fast” and zero with “too slow”). At the same time students rated the course as more difficult than in the past two surveys. The converted 3-point Likert scale equated 2 with “too hard” and zero with “too easy”; the average value was 1.73. In the first survey the equivalent average difficulty rating had been 1.02 and on the second survey the average rating was 1.5. This trend suggests that Dr. Quilt’s goal of having more time to present the material at the end of the course is not only well intentioned but crucial.

Of the students who stated that they were attending lecture (77%), they continued to report little to no involvement during class. Student responses indicated that their participation was primarily to answer instructor-posed questions (38%) or ask their own questions in lecture (37%). Other reported participation activities included correcting in-class examples (20%), posing a problem to be worked in lecture (16%), or “other” activities (22%). These participation numbers contrast sharply with the low number of students who spoke in lecture during classroom observations.

Reactions to the second programming project

The second programming project had been completed by the time of this final survey. In the second programming project, the students were to use Unix-based tools, lex and yacc, to construct a parser. The details of this project are contained in Appendix E.2. The survey asked students whether they had completed the second programming project and whether they felt that they learned anything from the assignment. Almost all of the students (98%) responded that they had completed the second programming project; however, only 68% stated that they had learned anything from the exercise. The students who thought the parser project was a good learning exercise stated that they gained a more comprehensive idea about how parsers work.

> “Yes, absolutely. It was great. It gave me an excellent insight into how parses and work and ALSO how to understand how the computer works with languages.” (survey 3: student 26)

> “Not anything about grammars, but it is interesting to look at a parsing program.” (survey 3: student 22)

Students also reported having a better understanding of the lex and yacc tools.

> “I learned a lot about Lex and Yacc, and some more about parsing.”
> (survey 3: student 25)

> “I feel I became more familiar with Linux and I have been exposed to lex and yacc.” (survey 3: student 20)

Other students stated that they appreciated learning more about how compilers work.

> "The project sort of revealed the secret of how compilers do their magic. It was interesting to see how simple and elegant." (survey 3: student 6)

> "I understand how compilers are built more, and understand a little more about Context-Free Grammars." (survey 3: student 23)

In the Compilers course at UT, lex and yacc are heavily utilized. Some students who thought the second programming project did not help them learn stated that the project scope was too simple and the instructions were too detailed.

> "*crickets chirping* it was more oriented around using certain software packages than around parsing." (survey 3: student 30)

> "Not sure, I just followed the directions" (survey 3: student 50)

> "I couldn't learn much from something so vague and simple. All I did was follow the directions. Building a parser in another language would have been more helpful – a lot of the people in the lab just droned on about 'what's this for? i just want to get through this. its stupid' etc." (survey 3: student 59)

Others thought the project was a waste of time.

> "Nothing. I took Compilers. The assignment was mostly a waste of time. Assignment 1 was much better." (survey 3: student 12)

> "I honestly don't think I'll ever use lex or yacc. The entire programming assignment seemed to be focused on these utilities which I found not very relevant to my future endeavors." (survey 3: student 82)

Perceptions of CS 341

When asked about the course material on the final survey, 73% of the students said they found the examples used in class to be helpful to their understanding of the Turing machine formalism. In a follow-up open-response survey item, we asked students to state what was currently confusing in the class. Almost every topic presented after the second midterm was mentioned by at least one student. In fact, many students admitted that they were uncomfortable with all of the new topics.

> "The stuff at the end of the class. Seems like there isn't a real way to figure out the problems." (survey 3: student 24)

> "Turing machines and recursion. the previous subjects i could understand fairly well. i'm a bit lost now." (survey 3: student 32)

> "Everything. I can't express that enough. Reduction, mostly though. We didn't spend a lot of time on it in class." (survey 3: student 35)

This level of discomfort with the more recent lecture topics helps explain the students' concern about the final examination, which would cover the material from the last third of the semester. On a multiple-choice item, the most common student response was that they were somewhat worried about how they would do on the final. (In analyzing the student responses to this survey item, we could not convert the multiple-choice answers into a Likert scale. The provided multiple-choice options were not phrased in a way that allowed for a consistent scale.) When asked to describe the strategy they would use to succeed in passing the final, the vast majority said they planned to spend a lot of time reviewing the course materials.

> "Studying. Lots of it. Inordinate amounts of time spent on sample problems and review of the materials. And when I get tired of that, I'll study some more. And attending the review session." (survey 3: student 71)

> “I’m going to study a lot and get tutoring.” (survey 3: student 82)

Most of the student responses expressed some variation of the view that “more time is better,” whether the activity was working problems or reading. Many students stated that they would spend as much time as they could afford on preparations for the final examination. Other students said they planned to focus on specific areas of the course material in their preparation for the final.

> “Go over keys of how to prove or disprove the different kinds of languages.” (survey 3: student 83)

> “I believe it will be about understanding reduction and distinguishing within the class of languages (chomsky hierarchy)” (survey 3: student 3)

Many students also responded that they planned to attend the final examination review.

> “I plan on studying everything available to us and attending the review session.” (survey 3: student 5)

> “i plan on reworking test I and test II which i haven’t looked at in a while to brush up / recall problems i might still have with old material. i then plan on reworking all of the homeworks. i then plan on working on TA’s material from the final review / supplementary homeworks in my trouble areas as time permits.” (survey 3: student 17)

When asked to articulate what they currently liked and disliked about the course, students once again reported a love/hate relationship with the material.

> “i enjoy Turing machines, even though i find them hard to do.” (survey 3: student 11)

> “The power of Turing machines. The concepts are very hard for me to grasp and apply.” (survey 3: student 15)

> "I like Turing machines...I don't like the wrap-up, I think much less time should have been spent before exam 1 and much more after exam 2."
> (survey 3: student 22)

In fact, a majority of the respondents suggested that, due to the complexity of the later material, it would have been preferable to spend more time on the harder topics while spending less time on the easier beginning formalisms.

> "Allowing more time on Turing machines" (survey 3: student 29)

> "I would teach the easier stuff faster and leave a little bit more time to digest the tougher stuff." (survey 3: student 44)

When commenting on fixing problems with CS 341, students were specifically told that any "changes" they suggested could not include getting rid of the course as a requirement for graduation. We reasoned that this constraint would lead to better and more realistic suggestions, although some students still questioned the requirement of CS 341 for the degree. One suggestion students had for improving the course was to adopt a better textbook.

> "About the only thing that can, and should, be changed to improve the class is a definitive textbook. From my understanding, there really aren't very many good texts out there. Maybe I'd write one, I don't know. Lily has provided a good body of supplemental material, but there is a need for additional, non-contradictory material." (survey 3: student 71)

> "pick a better textbook, include complete examples in the course notes (instead of incomplete ones)" (survey 3: student 65)

Another suggestion was to stress the real world applicability of course topics.

> "I would add more information on the material's applications in the real world."
> (survey 3: student 6)

> "Would try to relate it more to real-world applications." (survey 3: student 33)

Some students suggested putting in place a more straight-forward or improved grading strategy.

> "I STRONGLY feel that any instructor of a course like this with such a high failure rate should give the students more feedback on where they stand. I have performed above average on every assignment and test, yet I still don't know if I will get a C since 40% fail, and I was always under the impression that somewhere between 30%-70% was 'average' and got a C. So, maybe I can only get a D out of this? Is that fair? All quilt will tell me is that 'i'm probably doing ok'. that doesn't really mean much when my graduating is on the line. there seems to be some disrespect with that statement in that i am voicing a legitimately deep concern of mine, and she is not addressing it. i would like to know at least an approximate 'zone' i am in (eg. 77-79% $\longrightarrow$ C). Because 40% fail, it makes any guessing on my part about how well i am doing, being a slightly above average student in the class, completely hopeless which gives me a lot of anxiety about the course that shouldn't be necessary. very disappointing and frustrating that no one (quilt) will tell me."
> (survey 3: student 17)

> "I'd make the tests worth less and the homeworks worth more to the final grade. I would also let the students retake the test at least once because that would give students the motivation to keep trying, and it also means the students would still be learning this stuff (most likely better), which is the most important aspect of being here." (survey 3: student 48)

The idea of reworking test problems, suggested by student 48, is an interesting idea. Dr. Bettie Gyers, a Computer Sciences instructor, uses this technique in her section of CS 336, one of the prerequisite theory courses for CS 341, and considers this approach to test-taking to be a great pedagogical tool (informal conversation with Computer Science faculty member). A few students suggested creating more time for class or to split CS 341 into two courses.

> "i'd make it cs 441 or 541 and either make it meet every day (like calculus) or give it a discussion section (back to my suggestion about advertising the actual time commitment on the course schedule)" (survey 3: student 30)

> "maybe split it up into 2 classes. basics for the first class (which is required for all cs majors) then have a second class for those that are interested in it." (survey 3: student 32)

The idea of allowing more time for the course (having the course as four hours instead of three) was also expressed by Dr. Quilt in the third interview.

Prerequisite preparation for CS 341

Because this was the final survey for these students, the survey included some items to investigate how students viewed their prerequisite preparation for and interest in the course. The first item was multiple-choice and asked whether students felt their prerequisite knowledge was sufficient for CS 341 without review. Sixteen percent of the students responding to this survey item stated they were well prepared for the assumed prerequisite topics. Half of the students stated that their prerequisite knowledge was adequate when coming into CS 341, and the remaining 34% felt their prerequisite knowledge was inadequate for the material in this course

Because interest plays a role in learning [21] we wanted to investigate how interest in Automata Theory had changed over the semester. Students were asked to answer two multiple-choice items to rate their interest in Automata Theory upon entering the course and at the time of the final survey. The converted 5-point Likert scale equated 2 with "very interested", 0 with "neutral", and –2 with "very disinterested." The average reported interest in Automata Theory at the beginning of the course was 0.02, whereas the average at the time of the final survey interest had risen to 0.12. The increase in interest, given that students were more worried about performance in the course at this time than on the previous two surveys, is encouraging.

4.3.6 Summary: Third survey

The most notable finding on this final survey was that students performed so poorly on survey items testing current Automata Theory material. Fewer than half of the students who completed the third survey chose correct answers to two of three multiple-choice items designed to test how well they understood the new course topics.

At the time of the final survey, students rated the course very difficult, lecture pacing too fast, and the lecture notes as less helpful than in the past. We also observed a drop in reported voluntary homework completion. Students stated they did not have time to do voluntary work for CS 341 due to the demands of other required course work and end-of-term projects. There were increases in reported discussion session attendance and participation in examination review sessions. This increase may be indicative of students' desire to see more examples worked in order to better understand the new material.

Although most students reported they had completed the second programming assignment, they did not find it as useful as the first programming project. Many students stated that they felt as if they had only followed directions and did not do any original coding. Other students stated that learning to use lex and yacc, Unix parsing tools, was worthwhile.

In open-response survey items about the second midterm, the overwhelming theme was disappointment and anger. Most students reported that they found the examination to be a poor assessment of what they had learned during the middle third of the semester. Students also remarked that they had found the examination to be extraordinarily difficult and long, and that they saw no good reason it should have been that way.

There was a decrease in reported lecture attendance perhaps due to the frustration with the second examination or to their stated perception that discussion sessions and examination reviews were more useful for learning. Many students reported that lecture was not helpful to them in learning the material. Among the students who stated that they attended lecture, they reported that example problems covered in lecture were useful for learning the concepts.

On this final survey, students were asked to think back over the semester and suggest changes for the course. Our definition of "change" explicitly forbade the suggestion of eliminating CS 341 as a required course for the CS degree. The majority of the students suggested spending more time on topics in the last third of the course by spending less time on topics covered in the first third. They indicated that the first third of the course was easy and could be covered at a faster rate. In this way, more time would be available at the end of the course for the more difficult topics. Other suggestions for improving the course included adopting a better textbook, splitting the course into two courses to encourage more in-depth coverage of the material, and including more examples of real-world applications related to the formalisms learned in Automata Theory.

4.4 Classroom observations

The focus of data collection during classroom observations was to understand the number of students who were attending lecture, how these students interacted with the professor during lecture, and of what the lecture consisted. We also tracked the assigned and voluntary homework to get a sense for what students were asked to do outside of class time. Dr. Quilt was teaching two section of CS 341, each of which met twice a week on Tuesdays and Thursdays. We label the early morning class Section 1 and the late morning class Section 2. In order to observe instruction on all topics and interactions in both sections, the observation schedule was to attend Section 1 every Tuesday and Section 2 every Thursday. Blank note paper was used to record events as they happened. When events that we were tracking occurred in lecture they were written down. This written record is referred to as *field notes*. The field notes were used to construct Excel spreadsheets, which allowed us to generate graphs and descriptive statistics to support our findings. This section explains the findings from our classroom observation data.

4.4.1 Attendance during lecture

At the beginning of each class, we counted the number of students in attendance. Due to late arrivals, we also counted the number of students present at the end of each class. There were a total of 118 students enrolled in the two sections that Dr. Quilt taught: 82 were enrolled in the 9:30-11:00am section, which we will call Section 1, and 36 were enrolled in the 11:00-12:30pm section, Section 2. Figures 4.4 and 4.5 show the attendance for each week of the semester. On three occasions during the semester, the researcher was unable to attend the scheduled CS 341 lecture. The attendance graphs were completed by averaging (to the nearest whole number) adjacent attendance numbers to fill in missing data.

After the second week of the semester, attendance in Section 1 dropped dramatically. The average number of students attending Section 1 over the entire semester was 45 (55%), slightly more than half the number of students who should have been attending. Attendance for Section 1 was especially low after the 6th week, with fewer than half of the students attending lecture. There was a slight increase in attendance during the 11th week when the second examination was handed back in class.

Although not as extreme as Section 1, a similar pattern of attendance was observed in Section 2. The average attendance in Section 2 for the semester was 26 (72% of the enrollment). After the 7th week, with the exception of week 11 when the second examination was returned, student attendance was below the semester average. The odd spike in attendance in Section 2 during week 4 was due to the fact that Dr. Quilt allowed students to attend either lecture time. Because the first examination was being handed back dur-

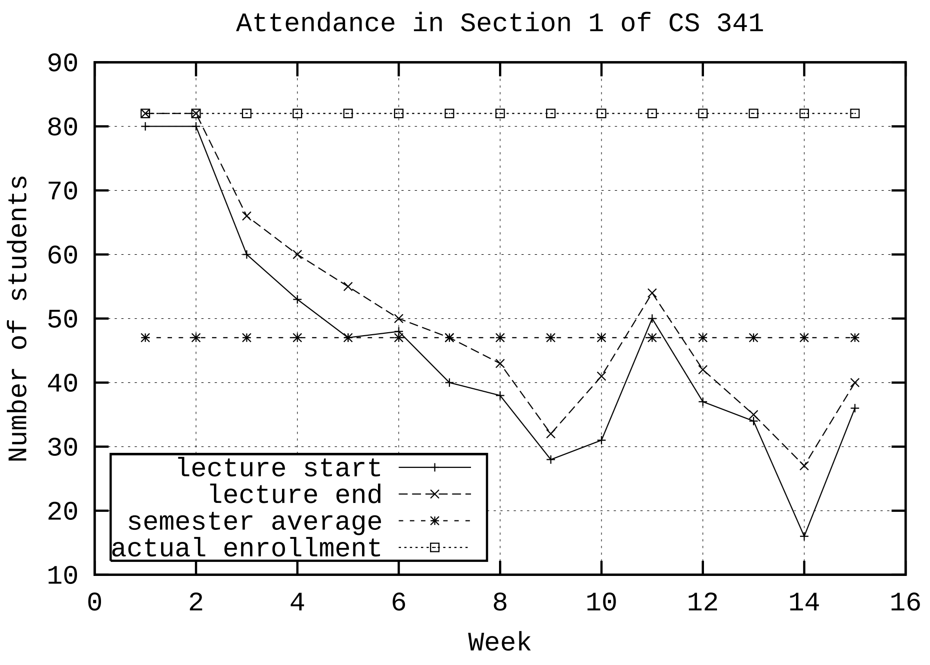


Figure 4.4: Section 1 attendance by week. (Missing data for week 4.)

ing class, many Section 1 students chose to attend Section 2. These attendance statistics contradict what students reported on the surveys concerning their attendance. However, as explained previously, students may have interpreted the question, “Do you attend lecture?” as “Do you attend lecture at all?” instead of “Do you attend lecture regularly?”, which is what we intended.

4.4.2 Student-instructor interactions

During lecture we recorded all student-instructor interactions. The interactions ranged from dynamic involvement, such as a student going to the front of the class to show a solution to a particular problem, to a student answering “yes” to an instructor-posed question. In Section 1, there was a dramatic increase in the number of student-instructor interactions around the time of and especially after the second examination. Some of the increase was probably because students were concerned about mastering the material in time for the final.

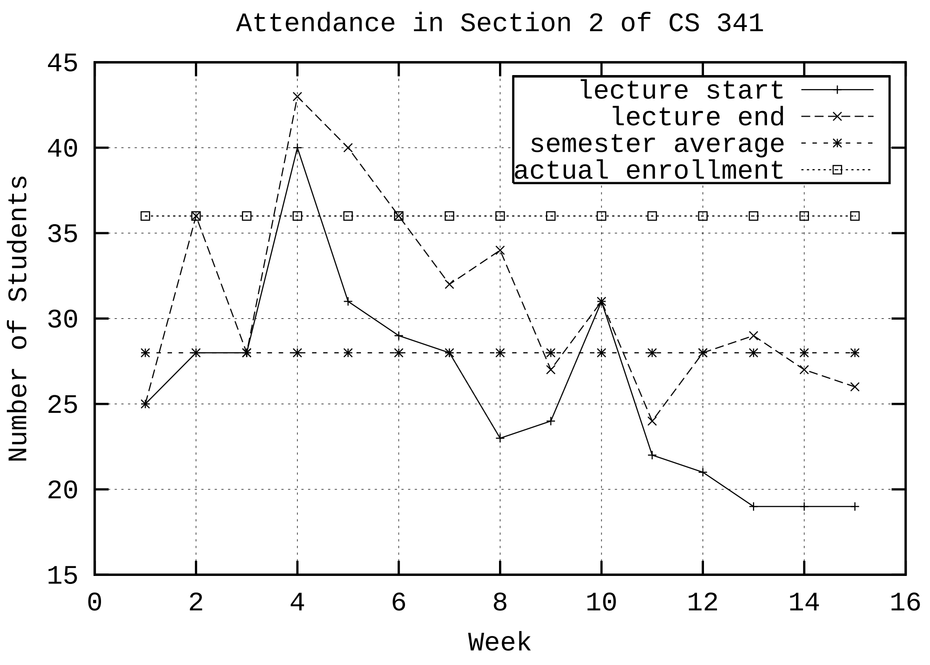


Figure 4.5: Section 2 attendance by week. (Missing data for weeks 6 and 14.)

We also observed that the few students who exhibited a clear understanding of the material began to speak out more. The graph in Figure 4.6 shows the number of interactions for Section 1.

In Section 2, no regular pattern of student-instructor interactions emerged. We observed no variations that could be related to topic changes or examinations, as shown in Figure 4.7. For the most part, a higher number of interactions occurred when Dr. Quilt worked example problems during class. The students would either murmur the next step in a solution, correct a mistake made by the instructor, or ask questions about a particular solution technique after the problem was solved. This increase in interaction supports the importance of using examples during lecture.

In her third interview, Dr. Quilt had stated that she felt Section 2 was not as interactive as Section 1. Although Figures 4.6 and 4.7 suggest that the two sections were both very active in participating verbally during lecture, Dr. Quilt's perceptions may have been

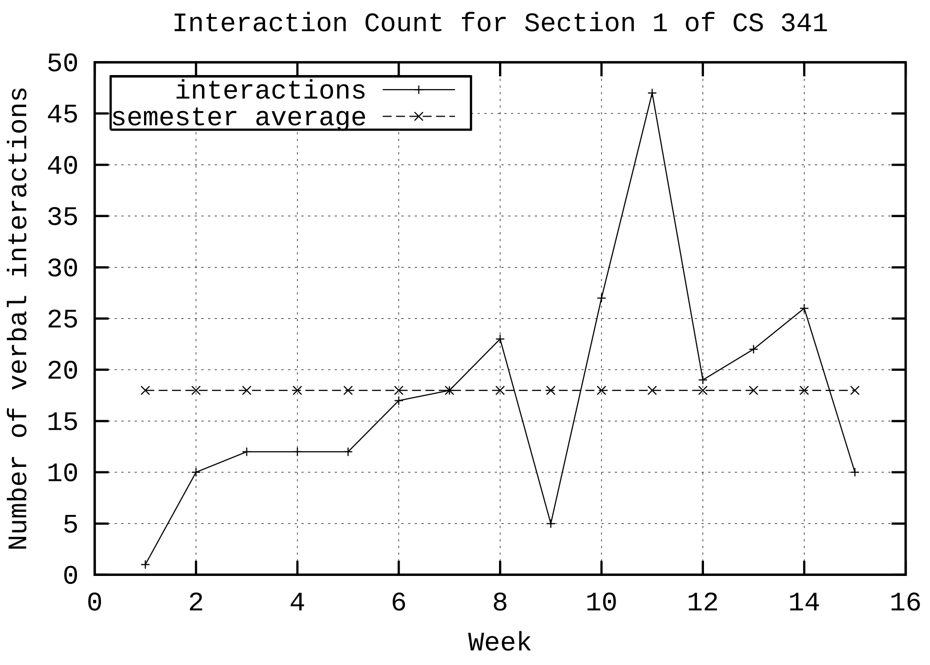


Figure 4.6: Number of student-instructor interactions in Section 1 by week.

was influenced by two factors. The first factor is that a greater number of dynamic interactions occurred during Section 1. Students asked more questions of a probing nature and commented more often on examples that they did not understand. Students in Section 1 also regularly posed problems for Dr. Quilt to work during class in order to help themselves understand her problem-solving strategies. Section 2 was much less dynamic. The interactions in Section 2 tended to be mumbled agreement to instructor-posed questions, or stating the next step in an algorithmic solution. Few students posed any questions of their own, or discussed any in-class examples at length.

4.4.3 Pace and coverage of lectures

To gauge the pace of the lectures, we tracked the number of slides used during each class meeting and the number of examples worked in each lecture. An example was defined as either the presentation of a problem that repeated a previous concept (redundant presenta-

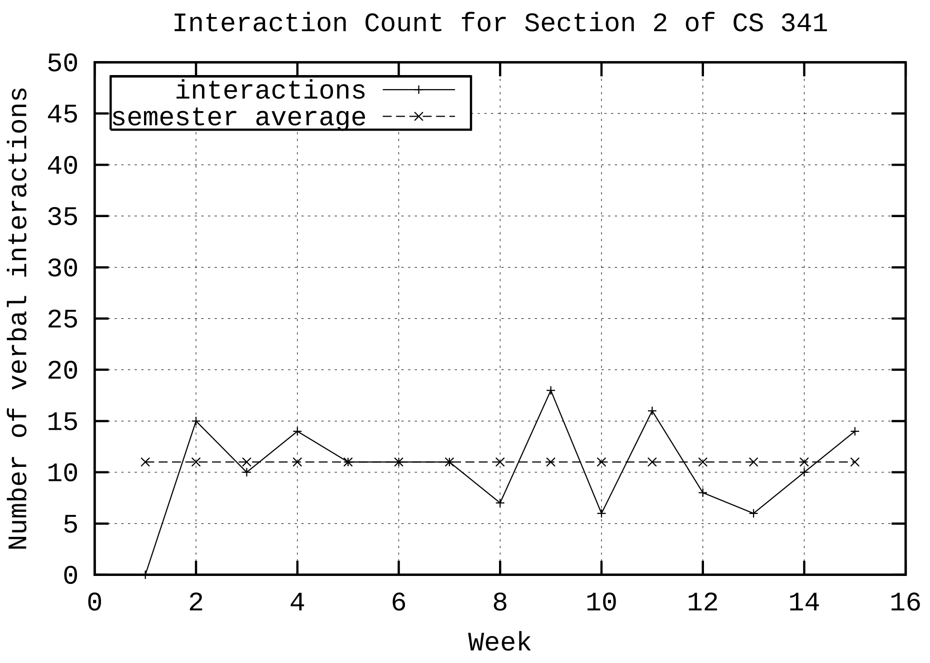


Figure 4.7: Number of student-instructor interactions in Section 2 by week.

tion) or the use of a pictorial representation to clarify concepts (such as a picture of a finite state machine along with a textual definition.) The graphs in Figures 4.8 and 4.9 show the data gathered on the number of slides and examples used in lecture each week. Since each section of CS 341 was observed once a week, the graphs include observations from both Sections 1 and 2. During weeks 4, 6, and 11 when the researcher missed a class meeting, the number of slides and examples were calculated by consulting the course packet slide copies and counting the number of slides and examples between observations of the preceding and subsequent classes.

We found no significant increase or decrease in the number of slides used or examples worked during the semester. Although the daily counts differed from week to week, the trend for the semester fluctuated around the mean. The only exception was before the second midterm, week 11, when Dr. Quilt decided to concentrate on working example problems in lecture to help students prepare for the second examination instead of progressing

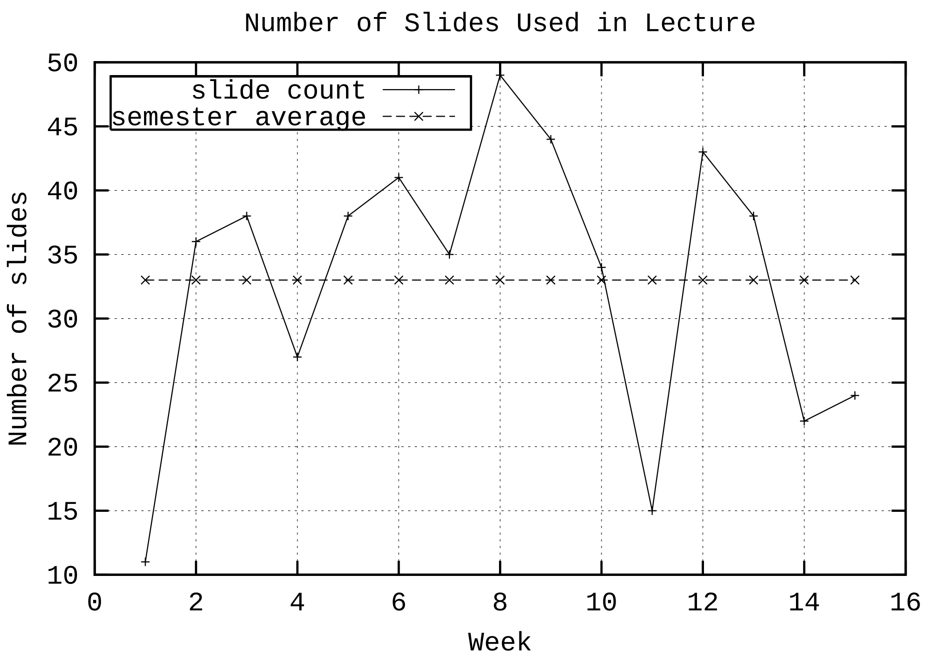


Figure 4.8: Lecture slide count by week.

through the course material.

Dr. Quilt used the slides to structure the lectures and to impart new topics and information efficiently. As described earlier, the semester was roughly divided into thirds. The focus during the first third of the semester was on finite state machines, the second third on push-down automata, and the last third on Turing machines. This partitioning of topics may not be an ideal way to organize the topics for CS 341. On the surveys, most students did not complain about the pace of the course being too fast until the last third of the semester. They also commented on the material being very difficult to grasp during the last third of the semester. Many students suggested that a viable solution would be to reduce the time spent on finite state machines in the beginning of the course, as they found the material was easy to learn and could be mastered more quickly than it had been presented. In Chapter 5, we discuss the logistics of creating more time for the material covered during the last third of the semester by decreasing the time spent on finite state machines.

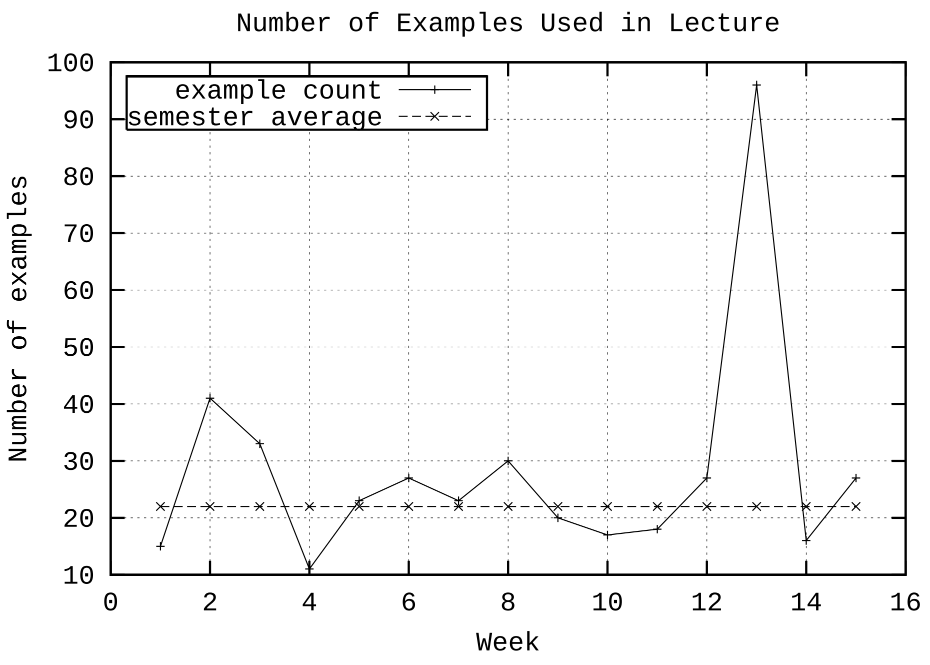


Figure 4.9: Lecture example count by week.

4.4.4 Voluntary workload

Dr. Quilt stated repeatedly that the material, given its abstract nature, cannot be mastered without practice. To this end, she provided the students with a large number of voluntary problems that were not collected or graded. In addition she assigned required homework and projects. In order to determine the amount of work that each student would do for CS 341 if they completed all voluntary and non-voluntary homework, we first counted the amount of voluntary work assigned each week. The voluntary homework consisted, on average, of reading 9 pages in the textbook, reading 6.2 pages in the supplementary reading portion of the course packet, and solving 24.8 problems per week. In a multi-part problem, we counted each part as an individual problem. Thus if the homework consisted of a single problem with 13 parts, we counted the number of problems as 13, not one. Figures 4.10 through 4.12 show the breakdown of suggested voluntary work per week.

The trend in the voluntary homework assignments is noteworthy. In the beginning of the semester, the voluntary workload was very large, as the material became more difficult the voluntary workload dropped. Based on Dr. Quilt's expressed belief that the material is best learned through practice, the more difficult material should have more practice available, not less.

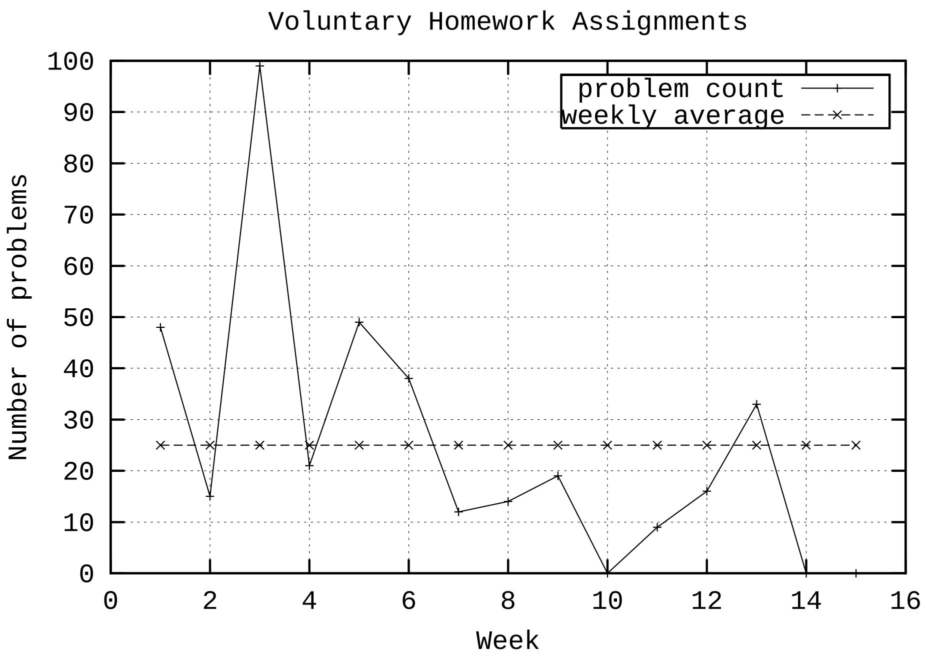


Figure 4.10: Number of voluntary problems assigned each week.

4.4.5 Required workload

Next we analyzed the amount of required homework each week. Each of the seven required homeworks had an average of 15 problems (using the same "parts" definition as with the voluntary homework). Three homeworks and the first programming project were assigned before the first midterm, two homeworks and the second programming project were assigned between the first and second midterms, and two homeworks were assigned between the second midterm and the final examination. The first required homework was over the

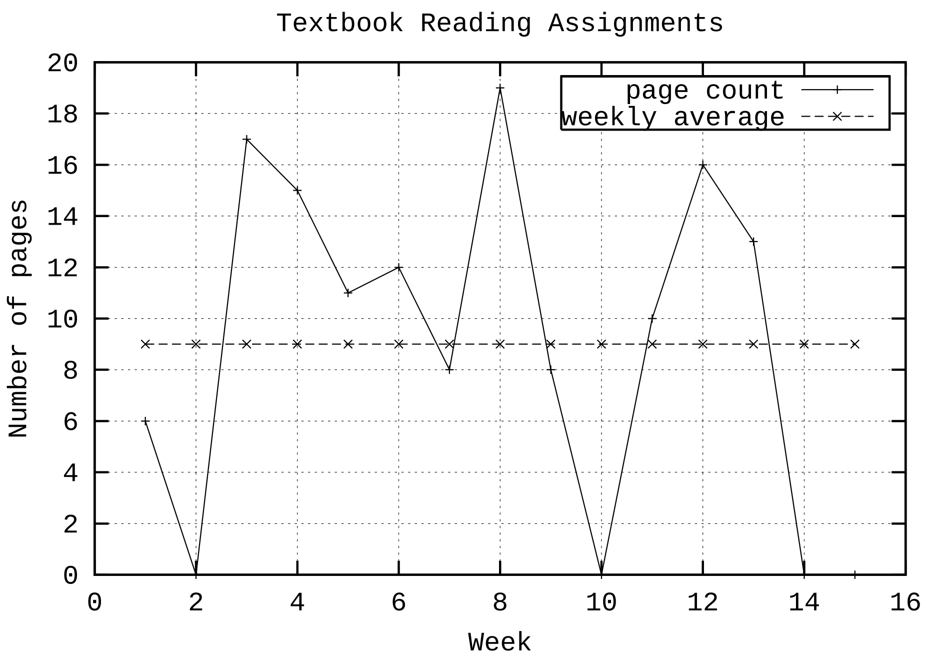


Figure 4.11: Number of pages in textbook assigned each week.

basic definitions used to construct the three formalisms in the course. The remaining six homeworks were divided equally among the three formalisms, with two homeworks dedicated to finite state machines, two on push-down automata, and two on Turing machines. Most homeworks were collected one week after they were assigned. On average, homework solutions were posted on the course website later during the day the assignment was due and the homework was physically returned two weeks after the due date. Posting the answers on the website helps compensate for the necessary lag time between homework turn-in and return. Immediate online availability of the solutions means that the students should still have their own solutions fresh in their minds, which would allow them to check their answers against the provided solutions before their memory fades.

Aside from the seven homeworks, two programming projects were required. Both programming projects were assigned so that the students had more than two weeks to work on them. For both projects the submitted programs were evaluated and returned to the

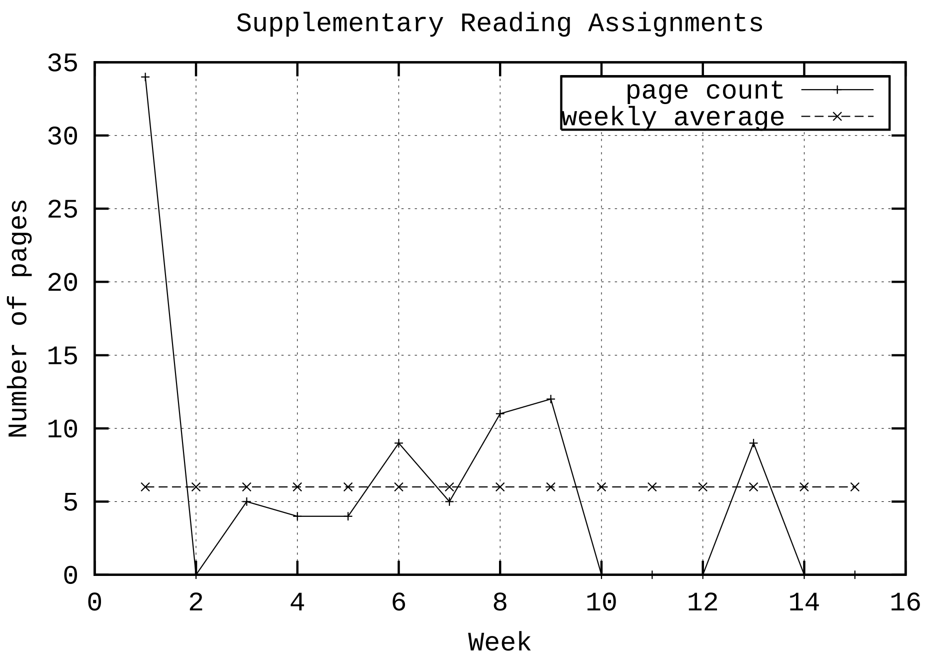


Figure 4.12: Number of pages in supplementary reading section of the course packet assigned each week.

students in 19 calendar days.

4.4.6 Examinations

The two midterms were held at night, outside of the regular class times, to provide a two-hour time block in which to take the examinations. The completed examinations were marked and returned within two days. Once the graded examinations had been returned in class, the answers and grading rubrics were posted on the course website. Making the grading rubrics available to the students in this way allowed them to better understand any partial credit they had earned, and also allowed them to evaluate whether the grading of any problem was erroneous or unfair.

4.4.7 Summary: Classroom observations

In the classroom observation findings, four issues were of particular importance:

- attendance was a problem,
- student-instructor interactions were more prolific when examples were used in lecture,
- the amount of voluntary work dropped off in proportion to the difficulty of the material, and
- posting solutions quickly was a useful strategy.

Attendance in both sections began to decrease after only two weeks. One reason for this is that students may have believed that having the lecture notes in the course packet equated to having the complete lecture. The lack of direct incentive to attend may have also contributed to this problem. In addition, the fact that CS 341 had a lower level of required work compared to other CS courses (student survey responses) may have compounded this situation. This combination of factors may have encouraged some of the students to spend the time they should have spent in CS 341 lectures on work for other courses. Such students may have had the ambition to use the notes to catch up with Automata Theory outside of lecture. If attendance at lectures is important, then this aspect of poor attendance must be addressed. We suggest some possible solutions in Chapter 5.

Both Dr. Quilt and the students agreed that the presenting examples during lectures was an effective learning tool. However in interviews, Dr. Quilt indicated that she was not convinced that students would be motivated to read outside of class and therefore felt compelled to cover in lecture every detail she expects students to know. This attitude left her very little time to spend on examples during lecture, which in turn reduced the opportunity for increased student participation. One solution may be to introduce different techniques for motivating students to read the relevant material before coming lecture. One idea might be to implement quick pop quizzes. Working more examples in lecture might also improve course attendance.

If learning through practice is necessary for learning material in Automata Theory, then as the material becomes more difficult there should be more practice problems available to the students. Dr. Quilt agreed but indicated during interviews that this would be hard to accomplish. As the material becomes more difficult, it also becomes more challenging to develop reasonably tractable problems to assign students (informal conversation). For the problems that are provided for practice on the difficult topics, a better approach may be to give incomplete solutions for some problems, since simply viewing a solution may create a false sense of understanding. When students are not provided with answers to problems, such as in the required homework, it is prudent to hand out solutions as soon as the homework has been turned in for grading. This allows students to begin to understand their mistakes immediately, rather than being forced to wait several weeks while the assignment is marked and recorded.

4.5 Material analysis

The two primary artifacts available to students while studying Automata Theory were the textbook and the course packet. An informal inspection of the textbook revealed that the book was efficient in introducing major concepts without being too verbose. The text suggested practice problems for each section, although no solutions were given. Unfortunately, the low utilization of the textbook reported on the student surveys made it an ineffective learning resource.

4.5.1 CS 341 course packet

The student surveys also revealed that students used the course packet more than the textbook. The students stated that for both understanding and review, the course packet was superior to the book. The course packet was organized into three sections: lecture slides, voluntary homework sets, and supplementary readings. By including lecture slides in the course packet, students were able to review for lecture if they chose, and during lecture could listen more and write less without feeling that they were missing critical information. Students' survey responses stated that they did indeed engage in these activities, which would not be possible without the inclusion of the slides in the course packet.

Students also appreciated having voluntary problem sets. Students used the voluntary problem sets to simply read the answers in order to understand the type and style of solutions that particular formalisms required, help in forming solutions to required homework problems, or to provide practice for examinations. By the end of the course, many students admitted that working these problems did help them learn the concepts presented in lecture. Without this resource, the students would not have had as rich a learning environment

Lecture slides

The pages in the first part of the course packet (lecture slides) had inconsistent alignment and spacing attributes. The ad-hoc layout may be because Dr. Quilt printed the slides from Microsoft Word files created by cutting and pasting the information from slides into several documents. The result was a document with poor readability. On first glance, it was unclear where the information for each slide began and ended. In addition, the titles and the text of the slides were confusing. Some students complained about the fact that the slides were incomplete, that some examples were only partially worked out on the slide. While the intent was that this type of example would be completed in lecture, this intention may have been unclear to the students. Lastly, although the lecture slides were composed into cohesive “sections” via labels on the bottom of the pages, the labels may have been useless for students who might be oblivious to whether the current topic was “regular expressions” or “finite state automata.” Each section of the lecture slides began with a list of voluntary work assignments. The current formatting made it unclear which parts were assignments and which parts were slide text. In Chapter 5, we discuss formatting suggestions and changes to the lecture slides to make them more useful to students.

Voluntary homework sets

The second part of the course packet consisted of voluntary homework sets and their solutions. Many times a new voluntary homework set starts did not start on a new page (sometimes the end of one problem set would be on the front of a page and the start of the next problem set would be on the back.) In the same way, solutions to the homework problems did not always start on a new page. This formatting made it confusing to see where a new voluntary homework set, or its solutions, began and ended. Students complained on the surveys that the solutions for the voluntary homework were often either incomplete or missing details. Another student complaint was that they found it easy to “see” the solution when reading the answer, but hard to produce a solution from scratch. Both of these issues can be addressed with some reworking of the solution sets for the practice problems. These changes are discussed in Chapter 5.

Supplementary reading

The last part in the course packet contained supplementary reading material. The current formatting had assigned reading assignments begin in the middle of a page, or begin on the back-side of a previous assignments. For instance, if Section 6 of a technical paper was assigned in the absence of Sections 1-5, Section 6 was not separated from the rest of the paper. Additionally Section 6 might not appear at the start of a new page. Again, this formatting choice made it confusing to find the beginning and end of assigned readings.

4.5.2 Summary: Material analysis

The course packet for CS 341 was well received by the students in Dr. Quilt’s sections of Automata Theory. The inclusion of the lecture notes in the course packet allowed students to pay more attention during class, helped students to structure note-taking (which enhanced studying of covered topics), and gave students the means to read ahead to prepare for lecture.

Based on survey responses, some students appreciated the voluntary homework problems. Practicing related problems and being able to read completed solutions provided students with valuable feedback on how to solve related problems. The availability of these problems may have contributed to the mastery of the material through repetition and practice. The repetition supported by the voluntary homework allowed students to refine their problem-solving strategies without the fear of negatively affecting their grade in the course.

When supplementary reading is included as part of a self-contained course packet, it may be prudent to make any textbook optional. Students' survey responses indicated that the supplementary reading portion of the course packet was used more frequently than the textbook. This usage may have been influenced by the fact that lectures were not based on the required textbook (classroom observations).

4.6 Student performance typing

Grades are play a large role in how students and instructors base their opinion of effective or ineffective learning in CS 341. In the instructor interviews, Dr. Quilt used examination performances when describing student learning. Similarly, student quotes from open-response survey items often emphasized grades as a motivating factor for doing work, or as a basis for concern when describing their learning experiences in CS 341. These factors motivated our analysis of attitudes and opinions held by students in different performance categories.

Students were given the opportunity to voluntarily sign a statement saying that their grades in CS 341 could be used as part of this study. Out of the 118 students enrolled in the two sections, 107 signed the release. Of the 107 students who gave permission, 77 had taken all three surveys. We ranked these 77 students according to their final weighted raw score: the numeric grade without the A, B, C, D, or F designation. We then grouped these students into four groups: those with a score of 80 or above (10 students), those with a score of 79-70 (21 students), those with a score of 69-60 (25 students), and those with a score of 59 or below (21 students). We labeled these four performance categories respectively as: ultra performers, high performers, average performers, and low performers respectively. The ranges for the four categories were chosen based on natural breaks in the scores around the numeric boundaries. The number of students in each performance group gave a normal distribution thus the naturally selected ranges were not modified. Note that these categories are not based on final course grades. Table 4.8 summarizes our four performance groups.

Raw course score	Label	Number of students in category
80 or above	ultra performers	10
70—79	high performers	21
60—69	average performers	25
59 or below	low performers	21

Table 4.8: Summary of performance categories used for student typing.

4.6.1 Population characteristics and academic background

Table 4.9 profiles the four performance groups based on demographic and academic background data from the first survey. The first characteristic of note is the average age of each group. As the performance category changes from ultra to low performers the average age increases, although this difference in age is not significant (post-hoc ANOVA produced $p > 0.14$). Another interesting trend is related to past academic performance. The students that repeated the course fell into lower performance groups. A significant indicator of which students would have a difficult time in the course was their past performance in the two required theoretical courses. Those students receiving a below a “B” average (3.0) in PHL 313K-*Logic, sets, and functions* or CS 336-*Analysis of programs* are more likely end up in a lower performance group in CS 341. Post-hoc ANOVA produced $p < 0.01$ for both course grades.

	ultra	high	avg	low
Ethnicity (in each performance category)				
Asian	50%	40%	46%	25%
White	50%	45%	50%	60%
Other	0%	15%	4%	15%
Average age (years)				
	20.7	21.2	22.5	22.5
Repeating CS 341	0%	5%	17%	35%
Past performance in theory (C=2.0,...,A=4.0)				
PHL 313K	3.78	3.40	3.04	2.8
CS 336	3.78	3.00	2.79	2.6
Average academic hours taken in Fall 2002				
CS	8.5	7.7	8.1	8.4
Total	13.1	13.5	13.0	12.5
Working outside school				
% students	60%	45%	46%	55%
avg hours/week	15-19	14-19	15-20	23-29

Table 4.9: Demographics and academic background of students by performance category.

4.6.2 Performance on and reactions to the examinations

Grades in CS 341 were largely determined by examinations (73% of the course grade). There were two midterms and one final examination. Although the coverage of each examination was not designed to be cumulative, the theory covered in each examination did build upon material presented earlier in the semester. Thus, good performance on an examination could be interpreted as mastery of all material presented up to the time of the that assessment. Table 4.10 describes the average survey responses from each performance group regarding each examination. Based on Table 4.10 the main trend is that the top two (ultra and high) and the lower two (average and low) performance groups tended to have similar views on each examination. The students had expressed dissatisfaction with the second midterm, labeling it as too difficult and too long. The students also rated the second examination as a poor evaluation method for what they had learned.

	ultra	high	avg	low
First examination				
Was fair (0=no, 1=yes)	1.0	1.0	0.8	0.8
Was good assessment (0=no, 1=yes)	1.0	0.8	0.8	0.6
General thoughts on performance (0=did great, ..., 3=hard time on several problems)	1.0	0.8	1.3	1.2
Second examination				
Was fair (0=no, 1=yes)	0.9	0.7	0.5	0.6
Was good assessment (0=no, 1=yes)	0.6	0.5	0.3	0.4
General thoughts on performance (0=did great, ..., 3=hard time on several problems)	1.8	1.7	2.3	2.6
Final examination				
Expectations for final (0=confident, ..., 3=very worried)	1.4	1.7	2.0	2.5

Table 4.10: Survey-reported perceptions on examinations by performance category.

Because grades are used as an indicator of successful learning by students, we investigated whether students could accurately assess themselves during the semester before all examinations and homeworks were graded. The average values in Table 4.11 indicate that students did have a sense of their performance category, albeit the average anticipated grades were all at least a “B” (3.0). By mid-semester, students’ perceptions about their performance were more accurate due to the results from the first examination and half of the graded homeworks. Even the ultra performers were worried about their final performance in the course. However, they were not worried about passing the course, which requires earning at least a “C” (2.0).

	ultra	high	avg	low
Beginning of semester				
Expected grade (0=F,...,4=A)	3.78	3.35	3.08	3.00
Anticipate problems (0=no, 0.5=maybe, 1=yes)	0.4	0.6	0.7	0.6
Middle of semester				
Worried about grade (0=no, 1=yes)	0.7	0.5	0.7	0.9
Worried about passing (0=no, 1=yes)	0.0	0.2	0.3	0.7

Table 4.11: Perceptions on course grade by performance category.

4.6.3 Student learning acquisition and class pace perceptions

Each of the three surveys included three or four items designed to test how well students understood current lecture topics. The performance on these survey items turned out to be a good tool for differentiating among students from the four performance groups, as shown in

Table 4.12. As the performance group moves from ultra to low, accuracy in answering the survey items decreases. For all four performance categories there was a drop in accuracy for the topics presented during the last third of the semester.

In the previous surveys, students had indicated that the lecture pace was too fast when the more difficult, end-of-semester topics were being presented. In Table 4.13 the average responses for pacing and course difficulty ratings are broken down by performance group. The ultra performers did not seem bothered by the pace of class, but the low performers had started to perceive the pace as too fast at the time of the second survey. By the time of the final survey, high and average performers indicated that the course was moving too quickly in lecture.

	ultra	high	avg	low
First survey				
Equivalence relation item	100%	95%	83%	80%
Converse item	78%	90%	83%	80%
Cardinality item	100%	90%	88%	95%
Hierarchy item	89%	75%	88%	85%
Second survey				
NFSM item	100%	95%	96%	95%
Grammar item	78%	60%	58%	65%
PDA item	100%	85%	63%	70%
Third survey				
Lang. class item	90%	76%	70%	48%
TM item	40%	33%	26%	29%
Reduction item	20%	19%	26%	22%

Table 4.12: Performance on survey items related to course material by performance category. Table reports the percentage of students answering each item correctly.

	ultra	high	avg	low
Pace of lecture				
(2=fast, 1=just right, 0=slow)				
Beginning of semester	1.3	1.2	1.2	1.3
Middle of semester	1.2	1.2	1.3	1.6
End of semester	1.3	1.6	1.7	1.6
Difficulty rating of course				
(2=hard, 1=just right, 0=easy)				
Middle of semester	1.0	1.5	1.7	1.7
End of semester	1.3	1.7	1.9	1.9

Table 4.13: Students' views of pacing and course difficulty over the semester by performance category.

4.6.4 Out-of-class learning

Students in CS 341 were given many voluntary homework assignments over the semester. The voluntary assignments included reading the textbook, reading supplementary articles included in the course packet, and working homework sets in the course packet. As Table 4.14 shows, reading either the textbook or supplementary reading material in the course packet remained relatively constant within each performance group with a slight increase for the high, average, and low categories at the end of the semester. Table 4.15 shows the average percentages of completion for each performance group with respect to each voluntary homework set. (Voluntary homework set #15 was assigned immediately after the first examination was returned to the students.) When looking for trends in the completion of voluntary homework sets reported by each performance group, on several sets the low performers reported working more problems than the average and high performers. This would suggest that simply working the problems was not helping those students.

	ultra	high	avg	low
Textbook use during the semester				
(use frequently=4,..., do not use=0)				
Beginning	0.8	0.5	0.8	1.2
Middle	0.4	0.7	1.1	1.1
End	0.2	0.5	0.8	1.1
Supplemental reading use during semester				
(use frequently=4,..., do not use=0)				
Middle	2.0	1.9	2.3	1.7
End	0.8	1.2	1.3	1.1

Table 4.14: Completion of voluntary reading assignments by performance category

	ultra	high	avg	low
Percentage completion on voluntary homework sets				
#1	55%	43%	33%	48%
#2	39%	28%	25%	35%
#3	39%	28%	29%	30%
#4	33%	25%	26%	25%
#5	33%	23%	26%	25%
#6	28%	21%	27%	20%
#7	33%	20%	23%	20%
#8	22%	13%	15%	24%
#9	28%	15%	17%	18%
#10	17%	13%	8%	16%
#11	11%	3%	11%	8%
#12	6%	0%	13%	0%
#13	6%	3%	10%	0%
#14	6%	0%	8%	0%
#15	11%	13%	23%	15%
#16	11%	13%	23%	18%
#17	11%	13%	21%	18%
#18	11%	15%	17%	20%
#19	11%	10%	15%	18%
#20	6%	10%	13%	10%
#21	0%	5%	10%	5%
#22	0%	5%	8%	5%

Table 4.15: Completion of voluntary homework sets by performance category.

In addition to voluntary homework, Dr. Quilt scheduled other activities designed to help students understand the material in CS 341. The first two lectures, dubbed the "big picture lectures," were designed to give students a schema for the theory that would built over the semester. Additionally, weekly discussion sessions and examination reviews were offered to facilitate student questions outside of lecture and to expose students to more example problems. General student responses indicate that they appreciated these three learning structures.

We wanted to investigate whether specific performance groups found any of these built-in structures particularly helpful. Table 4.16 details student perceptions of the big picture lectures, as well as whether students reported attending the discussion sessions and examination reviews. The high performance category tended to take advantage of these alternate learning functions more than the other groups.

	ultra	high	avg	low
"Big picture" lectures				
(very helpful=3,..., not helpful=0)				
	2.0	2.2	1.9	1.6
Average reported discussion session attendance (0-2)				
Beginning	0.5	0.7	0.5	0.2
Middle	0.5	0.4	0.3	0.2
End	0.6	1.0	0.5	0.3
Reported examination review attendance				
First	60%	65%	63%	45%
Second	78%	90%	75%	60%

Table 4.16: Students' views of course-specific learning structures by performance category.

Based on learning and study preferences summarized in Table 4.17, we had expected the low performers might take advantage of the discussion sessions in CS 341 Low performers reported a preference for small group interactions, which would suggest that these students should have found the discussion sessions to be appealing. Instead, the high performers were more likely to attend the discussion sessions. The ultra performers expressed preferences for learning by doing homework, for studying alone, and an acceptance of straight lecture. These factors may explain their lack of attendance at the discussion sessions and examination reviews.

	ultra	high	avg	low
Like to learn by (percentage responding positively)				
Reading	88%	70%	50%	70%
Listening	77%	85%	88%	90%
Homework	100%	80%	58%	60%
Coding	56%	65%	58%	50%
Preferred class type				
(0=small group interactions,...,straight lecture=2)				
	1.4	1.2	1.5	0.9
Preferred mode of study				
(alone=0, with one other=1, with two or more=2)				
Beginning Preference	0.0	0.4	0.4	0.3
First midterm	0.4	0.8	0.6	0.3
Second midterm	0.4	0.8	0.7	0.6

Table 4.17: Students' learning preferences by performance category.

4.6.5 Student involvement in lecture

Table 4.18 details student participation preferences and reported activities during lecture. Ultra performers reported perfect attendance during the semester. This supports their preference for straight-lecture courses. The high and low groups had the highest averages for asking and answering questions in lecture. However, the more dynamic activities of posing a problem or correcting an in-class example are clearly related to performance group rather than participation preferences.

	ultra	high	avg	low
Beginning of semester				
Like to participate in lecture? (2=regularly,...,not at all=0)				
	0.4	0.8	0.5	0.7
Middle of semester				
Attendance	100%	95%	87%	90%
Asked question	33%	40%	17%	42%
Answered question	56%	55%	25%	44%
Posed a problem	33%	10%	9%	6%
Correct example	33%	21%	9%	0%
End of semester				
Attendance	100%	90%	67%	80%
Asked question	44%	30%	21%	50%
Answered question	56%	40%	17%	47%
Posed a problem	22%	25%	9%	16%
Correct example	33%	15%	17%	16%

Table 4.18: Participation patterns by performance category.

Better performance groups interacting in a more dynamic manner may be due to personality or to self-efficacy judgments. The idea that self-efficacy may be responsible for the more dynamic types of interaction during lecture is supported by reported interest ratings in Automata Theory, detailed in Table 4.19. It is reasonable that students who view Automata Theory as useless and have no interest in the course might view themselves as incompetent in learning the material of CS 341. If this hypothesis is true, it would be worthwhile to focus on making the material of Automata Theory more relevant to the average and low performers, perhaps by incorporating more real-world uses for the topics being studied.

	ultra	high	avg	low
Interest in material to start (very interested=4,..,very disinterested=0)	3.0	2.0	1.6	2.1
Interest at end of course (much more interested=4,...,much less interested=0)	2.4	2.3	2.1	1.6
Perceived usefulness of material at end of course (very useful=2, somewhat useful=1, not useful=0)	1.3	0.9	0.7	0.5

Table 4.19: Students' reported interest in subject matter by performance category.

4.6.6 Summary: Student performance typing

In this section we examined differences in student perceptions by grouping students into four performance categories: ultra performers, high performers, average performers, and low performers. These categories were based on end-of-course numeric scores rather than on final course grades. The analysis revealed several early indicators for success in the course, including age, and grades in prerequisite theory courses. The average age increased as performance decreased. Additionally students who reported good grades in prerequisite theory courses performed better in Automata Theory.

Students who expressed a high interest in Automata Theory tended to attend lecture and were more disposed toward dynamic interactions during class. Those students who did not express an interest in the material being taught in Automata Theory were less likely to engage in correcting in-class examples or posing problems to be worked in class. The ultra and high performers reported more frequent engagement in these activities, suggesting that these students were more confident in their ability to understand the lecture material and thus took a more active learning role during class.

Students in the low performance category reported preferences for lectures that are more interactive as well as for working in smaller group settings. However, these students did not seem to be taking advantage of discussion sessions, which were styled to facilitate this type of learning.

The survey items tailored to test students' understanding of current course material seemed to be useful indicators of performance category. Each survey contained three or four items designed to test students' understanding of current lecture topics. On most items there was a clear demarcation between performance groups, with the ultra performers demonstrating the most understanding. The observed relationship between performance on these survey items and students' end-of-course scores can be used by an instructor for formative evaluation of learning during the semester.

4.7 Answers to Research Questions

In this section, we revisit our original research questions. We explain the answers and detail the supporting data.

How was the course created? Were any standard curricular goals used? From the first interview with Dr. Quilt we found that the course has a "field acceptable" curriculum that was not modified for CS 341. Although each instructor brings their own learning objectives to the course, no standard curricular goals had been established for CS 341 except to keep the structure of the course in alignment with the accepted sequencing of information.

What does the instructor see as the goal(s) of this course? From instructor interviews, particularly the third interview, Dr. Quilt expressed several student learning goals that included students working and becoming comfortable with formal notation, and a 15 week exposure to the formalisms presented in CS 341. Dr. Quilt stated that her main responsibilities in the course were to cover all the material she wanted students to know during lecture, and to motivate students to work on the out-of-lecture assignments.

What types of questions does the instructor ask the class during lecture? Dr. Quilt mainly asked rhetorical questions. Using Bloom's taxonomy [20] the majority of the questions were categorized as "Knowledge" or "Comprehension", the lowest two classifications of question types. In the second interview Dr. Quilt states that the reason she uses questions is to let students know if they have adequately prepared and understood current lecture material.

How do the students view learning the material? During all three surveys, students expressed a love/hate relationship with the material taught in CS 341. Students stated that they disliked the abstractness of the material, but liked solving problems. Some respondents even saw an immediate use in other classes for the theory being developed over the semester.

What do the students think is missing from the course? Student survey responses suggested that students would like more time to be dedicated to the material covered in the last third of the semester. Students also stated that they appreciated the presentation of example problems in lecture and would like more and harder examples to be used.

What do the students think is good about the course? Survey responses indicated that students appreciated the two initial "overview" lectures, having the lecture slides on which to take notes during lecture, and the out-of-class learning opportunities. On the third survey we also noted an increased appreciation for the examination reviews.

What motivates students to do homework? Although some students stated on the surveys that they did homework because they liked the material in CS 341, the majority of students did homework because they directly attributed doing the homework to obtaining a better course grade. This trend is supported by student reports on completion of required versus voluntary homeworks. Almost all students reported completing the homework that was graded, however, as the semester progressed, fewer and fewer students reported working on the voluntary assignments.

Are students attending and benefiting from discussion sessions? Students who reported attending discussion sessions found them to be useful in learning the material for CS 341. Many students did not attend discussion sessions until late in the semester. This may be because the material in CS 341 builds during the course.

Are students coming to class? According to the student surveys, students reported attending class. However, classroom observations did not support this claim. After the second week of lectures, attendance began to drop and never improved. From the student performance typing, we found that ultra performers reported perfect attendance on average. This would suggest that attendance in CS 341 is critical for student success.

What is the pace of class? From classroom observations we found no sharp increase or decrease in the number of lecture slides used in lecture over the semester. Student survey responses indicated an overall acceptance of class pace up until the last third of the semester. The low performers rated lecture pacing as too fast during the middle of the semester, suggesting that these students were not absorbing the material during lecture during the last half of the course.

What drives student involvement in lecture? From classroom observations we found that students were more likely to interact during class when example problems were worked during lecture. From the student surveys, we saw that the majority of students reported interacting during lecture by answering instructor posed questions and by asking questions of their own. From the student performance typing, ultra and high performers were more likely to interact in a more dynamic fashion by correcting instructor examples and posing their own problems to be worked during class.

What examples are used in class? From classroom observations, every lecture involved some sort of example. Either a picture was provided following a textual definition, or a supplementary problem was posed to exemplify a specific point about the formalism currently under study. Student survey responses indicated that they preferred having example problems worked during lecture. They stated that observing the solution to problems similar to homework and examination questions was beneficial to learning the material in

CS 341.

In the next chapter we discuss the implications of our findings. We point out what pedagogical techniques and activities are beneficial to other Automata Theory instructors as well as suggestions based on what we would improve in CS 341.

Chapter 5

Discussion, Conclusions, and Implications

This chapter discusses how the findings in Chapter 4 can be used to create a better learning environment for Automata Theory. Although our study focused on a single class, because Automata Theory has a similar curriculum at other college institutions [7, 41, 66, 94] it is our intention that these findings will be helpful to all instructors teaching and students learning Automata Theory.

Although the focus of this chapter is the results from Chapter 4 and their implications for improving teaching practices in order to create a better learning environment for students, this information can also benefit students. A student of Automata Theory who reads this chapter would better understand what Dr. Quilt, and perhaps their own instructor, expects of them and the decisions behind teaching practices currently in use. By understanding decisions behind course practices, a student can gain insights into study habits and interaction patterns that can facilitate better performance.

The remainder of the chapter is organized as follows: we first present and discuss aspects of CS 341 that can become good practices for other instructors of Automata Theory. We then discuss the less successful aspects and propose improvements based on CS 341 findings, suggestions from the literature, and our own experience. We conclude the chapter by suggesting future research and summarizing the implications from our discussion.

5.1 Effective practices

This section highlights aspects of the course that were helpful to creating a good learning environment in CS 341. These exemplified teaching strategies and tools would benefit the teaching of any Automata Theory course.

5.1.1 Use of overview lectures

Often, when a student walks into a new course at the beginning of a college term, the instructor makes some introductory comments on logistics (books, grading policy, etc.), passes out a syllabus, and then jumps immediately into lecture. The syllabus that is passed out is designed to give the students an outline for what topics will be discussed in the semester. In Automata Theory this is insufficient. The topics in Automata Theory do not stand alone, but rather build upon one another. A student may miss this building aspect if they simply look at a timeline of topics presented in Automata Theory.

> "[Students] don't have a sense of the theory we are going to build."
> (first interview with Lily Quilt)

Dr. Quilt provides a valuable service for the students by using two lectures to provide an overview of the material in this class. These "big picture" lectures expose the students to how the theory of the course will build up over the semester and to relationships among topics on the timeline. In this way students can form an effective mental schema for understanding the ideas presented during the remaining lectures [78]. In the first survey, students indicated an appreciation for the overview lectures. The ultra and high performance groups reported the overview lecture more helpful than the average and low groups.

5.1.2 Use of example problems in lecture

In addition to the big picture lectures, students appreciated having example problems solved during lecture. In all three surveys, students rated examples as an effective learning tool.

> "I think it is essential to attend lecture since Dr. Quilt shows us examples, which we might not find in the notes." (survey 2: student 57)

As the difficulty of the course increased, students actually called for more complex examples to be used in lecture instead of examples they considered straightforward.

> "Spend more time in class on harder examples that are similar to the ones on the exams. Most examples are simple so that we can understand but when exam time rolls around the problems will be much harder with less time to do." (survey 1: student 90)

In light of these student responses, Dr. Quilt's inclusion of specific problems to be exemplified in lecture is a good teaching mechanism.

5.1.3 Use of solution explanations in lecture

A drawback to examples worked in lecture is that the instructor comes prepared with a fully worked out solution. As novices, students do not understand the process of creating the solution. This lack of expertise makes steps in the solution process look like "tricks" (as the students call them) were used to solve the problem. Without understanding how to create the "trick" a solution may seem straightforward, when in fact it is the production and use of the trick that makes the solution easy.

For an example of when expertise leads to easy solutions, consider a proof using the Regular Language Pumping Lemma. The Regular Language Pumping Lemma is a proof method used to show that languages are not regular. Languages consist of strings. When a regular language is infinite, the lemma states that for all sufficiently long strings, there must be part of the individual string construction that can be repeated, or pumped, to generate an infinite number of strings in the language. In order to show that an infinite language is not regular, the Pumping Lemma is used in a proof by contradiction to show that the non-regular language is not pumpable and yet still infinite. Consider proving that $a^n b^n$ is a non-regular language. If the sample string chosen for the proof is $a^N b^N$, there is one case to consider. If the string chosen is instead $a^{N/2} b^{N/2}$, there are three cases to consider. It is the choice of the sample string to be used in the Pumping Lemma proof that dictates the number of cases to consider. Once the string is chosen, the solution process is algorithmic.

Students expressed this frustration in understanding solutions as early as the first survey. Students complained about concepts being clear during lecture but then feeling helpless when working on homework.

> "[...] In class, I understand about 75% of the lecture. But when I leave, only 15% stays with me, and I'm completely lost. It seems easier when the teacher is there, but when I'm on my own, it's a lot harder." (survey 1: student 57)

5.1.4 Use of questions in lecture

Dr. Quilt attempted to bridge the gap between expert and novice by asking rhetorical questions intended to elucidate the choices she had made in observing the example problem initially. Sometimes she would state, "In order to see how to do this, you need to understand that this problem can be rephrased in this way" (observation notes) or similar commentary. Such statements were designed to help the students understand how to create the solution, rather than just exposing them to the final solution. The "rephrasing" was generally translating a mathematical formula into an English description or manipulating formal notation into an equivalent statement to make the underlying concepts clearer. The ability to rephrase for clarity is a skill that students may not possess when first introduced to topics in Automata Theory. This technique is critical in teaching abstract material [105]. Improving upon instructor discourse in order to clarify the solution process is discussed later in the chapter.

5.1.5 Use of reasonable student learning objectives

The learning objectives that Dr. Quilt had for these students included familiarity and increased comfort with material taught in the course. She hoped students would gain two aspects of learning:

> “Number one the content of this material and I’ll go into that a bit more in detail here in a second, and two is just familiarity with the notation and thinking abstractly [...] I hope they got 15 weeks more sophisticated at doing that [...Then] there’s specific content [...] I want them to know about finite state machines and regular expressions and how to parse context-free languages and that it doesn’t take much to have a completely general-purpose computing engine and there are things that you can’t compute, and that problem reduction is a way to say things about various kinds of problems where you know something about some other problem.” (third interview with Lily Quilt)

Dr. Quilt had reasonable student-learning expectations. Learning objectives striving for topic mastery in Automata Theory may be too industrious since it takes repeated exposure to new topics in order to gain mastery [26]. Without drastically reducing the material to be taught in Automata Theory, repeated exposure to the later topics in the course is not possible in a single semester.

5.1.6 Use of material for assumed background knowledge

Dr. Quilt expects that students enrolled in CS 341 will be familiar with specific concepts of basic logic and discrete mathematics, but acknowledges that students are exposed to varying levels of these concepts before taking her course. She also knows that she will not have time to cover the prerequisite material during lecture. To address the needs of students who may lack some of this background, Dr. Quilt provides students with relevant reading and self-evaluation tools so students can do remedial work to catch up or know they should drop the course. In the course packet, the first supplementary reading section is a review of logic and discrete mathematics. She designed this section as a primer students could use to review or to learn basic skills necessary for understanding and completing Automata Theory. This review style might have been better received by the ultra performers who expressed preferences for learning through reading and doing homework. The other performance groups may have benefited from a more dynamic review.

By completing the self-assessed quiz, students could evaluate whether their understanding of the prerequisite material matched the instructor's expectations. This provision is a good practice. Many courses assume prerequisite knowledge. In a large institution, there can be wide variations in the content covered in differing sections of background courses. This variation is also a factor for students who transfer into a program from another institution. It is a useful service to provide students with a concrete way of assessing whether their learning meets the expectations of the course and, if not, a way to learn the prerequisite material on their own.

5.1.7 Inclusion of programming projects

The inclusion of some programming projects is a good practice for Automata Theory. Dr. Quilt included two such assignments: creating a simulator for non-deterministic finite state machines and creating a simple parsing program. Forging an explicit tie between class material and programming can help students to see the relevance of theory. For at least the first programming project, students reported that the experience helped their understanding of the concepts:

> "I learned how to see how a fsm [finite state machine] was implemented and it helped me understand non-deterministic machines better."
> (survey 2: student 93)

5.1.8 Use of course webpage

With the advent of convenient web access, it is common practice for instructors to post course material on a class web-page [22]. These materials can include the syllabus, outline of topics, and class announcements. Posting relevant course material can help keep students informed of course activities. The CS 341 website included solutions to required homework. In courses with paper-and-pencil homework, there can be a significant amount of time between when the work is turned in and when it is returned. By posting solutions online as soon as the homework is collected, students have a way to check their understanding immediately, before the memory of the assignment fades [95]. This is good practice for Automata Theory, especially in light of the fact that homeworks may be distributed to students after examinations.

5.1.9 Making grading algorithms known to students

Along with posting homework solutions immediately, Dr. Quilt posts examination solutions on the day the examination is handed back to students (generally two days after the examination is completed). In addition to the solutions, the grading rubric for the test is also posted. In this way students can check the accuracy of their solutions against the standard and understand why they lost full marks on problems they did incorrectly. This allows students to verify the accuracy of their examination score immediately. Students can also begin to understand the expectations of the instructor by understanding where most credit is awarded and thus develop better study habits for later examinations and homework. All courses that have web access can benefit from this example.

5.1.10 Use of student lecture notes

Web access is only one way in which computer usage can enhance teaching. With the widespread use of applications such as PowerPoint and Microsoft Word, all teaching material can be electronically stored and modified. This provides flexibility for updating the material and allows students to receive a copy of teaching materials that will be useful for learning. In CS 341, Dr. Quilt lectures from overhead projector transparencies or slides. These slides are created in Microsoft Word. One drawback to using slides is the pace of lecture can accelerate to the point that student note taking is impossible. To avoid students having to write and not listen, the slides are printed out and included in the student course packet. Dr. Quilt provides students with the slides in the hope that listening will take precedence over writing during lecture. The students appreciated having the slides. Most students did state that they listened more in lecture because of having the slides in their packet.

> "I like being able to listen and not have to worry about copying all the slides.
> It is nice to be able to write in my own notes on top of the slides."
> (survey 1: student 9)

The provision of lecture slides can provide an important support mechanism when listening is critical to understanding the course material, as it is in Automata Theory. With reduced note-taking activities in class, students are much more likely to become actively engaged in lecture. This leads to a more constructivist learning environment.

5.2 Creating a more effective learning environment

We acknowledge that all courses, no matter how effective, will always need improvement. Our goal is not to make CS 341 perfect, but rather to point out changes based on student responses, suggestions from the literature, and our own findings that might help improve student learning experiences in a general Automata Theory course.

5.2.1 Designing materials

While providing printed slides may be appreciated by students in a course that uses slides in lecture, this practice may also be one reason behind poor attendance. We saw this aspect of student thinking in CS 341 as early as the first survey. Students admitted to missing a few lectures because it was easier to catch up due to the information contained in the course packet.

> “[...] I don’t have to worry about accidentally missing a few details, as well as the fact that its easier to make up for missing a day of class”
> (survey 1: student 20)

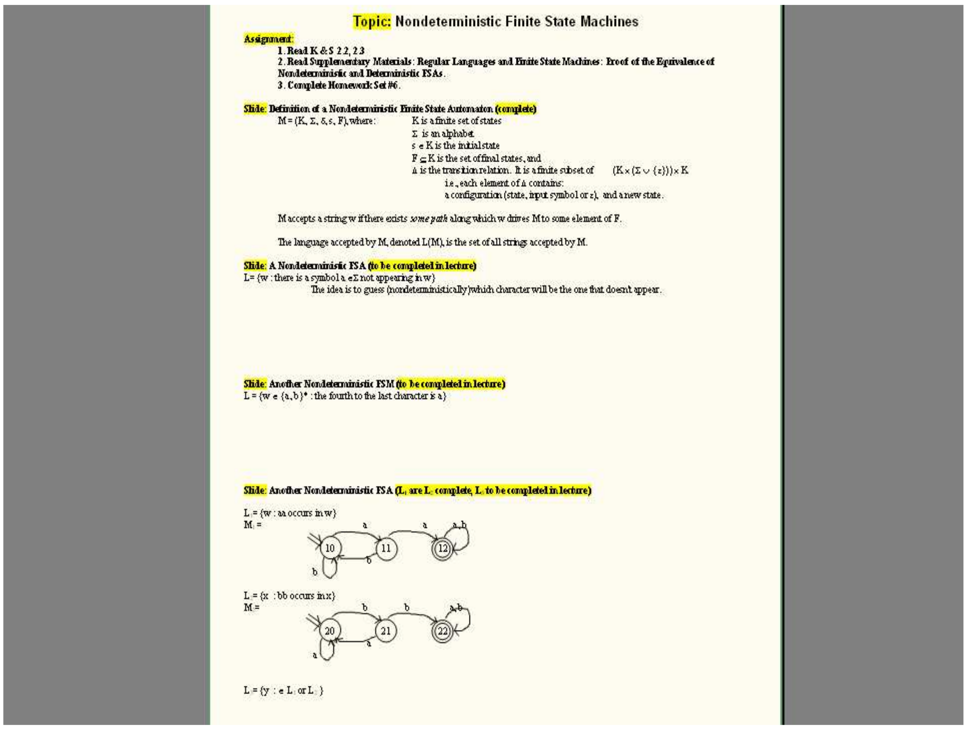
Perceptive students realized that the lecture slides were incomplete and only used as a learning tool in lecture.

> “They are incomplete. If you miss a lecture you may think that you have the material anyways, but it isn’t true, the slides can be missing vital parts.”
> (survey 1: student 37)

To make complete and incomplete information explicit to all students, slides and other course materials should be marked with explicit references for when lecture is critical for completion. For instance, a slide may contain a complete example to show a quick concept in lecture. This needs no modification. But if an example is only partially worked out, the student copy of the slide should denote “to be completed in lecture” so that a visual cue of both incompleteness and stressor for attendance is visible to the reader of the slide material, see Figure 5.1. Although this technique may appear to be coddling students, we feel that some students may need the extra information, as well as helping students not comfortable with English more secure about what needs to be completed during an English lecture. The ultra performers reported 100% attendance during the semester which would suggest that any hints that students have for attending would help performance in Automata Theory.

When Dr. Quilt introduced a new topic in CS 341, the students’ notes designated what reading and voluntary homework problems were related to the topic, see Figure 5.2. Dr. Quilt views these voluntary assignments as critical for learning the material.

> “CS students are not used to taking a theoretical class that has homework where



Topic: Nondeterministic Finite State Machines

Assignment:

1. Read K & S 2.2, 2.3

2. Read Supplementary Materials: Regular Languages and Finite State Machines: Proof of the Equivalence of Nondeterministic and Deterministic FSAs.

3. Complete Homework Set #6.

Slide: Definition of a Nondeterministic Finite State Automaton (complete)

M = (K, Σ, δ, s, F), where: K is a finite set of states
Σ is an alphabet
s ∈ K is the initial state
F ⊆ K is the set of final states, and
Δ is the transition relation. It is a finite subset of $(K \times (\Sigma \cup \{\varepsilon\})) \times K$
i.e., each element of Δ contains:
a configuration (state, input symbol or ε), and a new state.

M accepts a string w if there exists *some path* along which w drives M to some element of F.

The language accepted by M, denoted L(M), is the set of all strings accepted by M.

Slide: A Nondeterministic FSA (to be completed in lecture)

L= {w : there is a symbol $a_i \in \Sigma$ not appearing in w}

The idea is to guess (nondeterministically) which character will be the one that doesn't appear.

Slide: Another Nondeterministic FSM (to be completed in lecture)

L = {w ∈ {a, b}* : the fourth to the last character is a}

Slide: Another Nondeterministic FSA (L_1 are L_2 complete, L_3 to be completed in lecture)

L_1 = {w : aa occurs in w}

M_1 =

$$\begin{tikzcd} \arrow[r] & 10 \arrow[r, bend left, "a"] \arrow[loop below, "b"] & 11 \arrow[l, "b"] \arrow[r, bend left, "a"] & \boxed{12} \arrow[loop right, "{a,b}"] \end{tikzcd}$$

L_2 = {x : bb occurs in x}

M_2 =

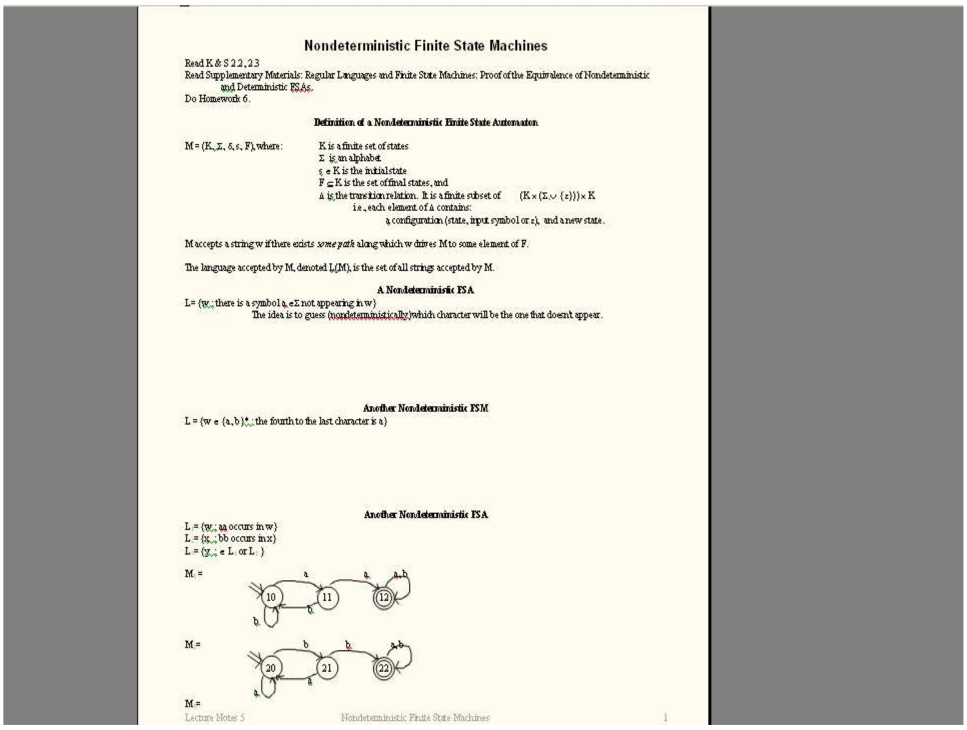
$$\begin{tikzcd} \arrow[r] & 20 \arrow[r, bend left, "b"] \arrow[loop below, "a"] & 21 \arrow[l, "a"] \arrow[r, bend left, "b"] & \boxed{22} \arrow[loop right, "{a,b}"] \end{tikzcd}$$

L_3 = {y : ∈ L_1 or L_2 }

Figure 5.1: Scan of a page from the lecture slide portion of the course packet after modification.

they sit around and stare at walls and solve problems. They are used to a lot of coding, they know how to do that. But this material is hard and you have to practice. I know no way to learn how to solve these problems short of practice. You just have to do it for hours a week, and I have to figure out how to get them to do it." (first interview with Lily Quilt)

If an instructor's view of learning agrees with Dr. Quilt's belief that practice is necessary for mastery, they must make sure the students understand the importance of doing voluntary homework problems. To help students understand that the suggestions for practice are necessary, they need to stand out on the page. Figure 5.1 shows suggested formatting changes based on the CS 341 course packet. We also suggest putting a preface, or quick



Nondeterministic Finite State Machines

Read K & S 2.2, 2.3
Read Supplementary Materials: Regular Languages and Finite State Machines: Proof of the Equivalence of Nondeterministic and Deterministic FSAs.
Do Homework 6.

Definition of a Nondeterministic Finite State Automaton

M = (K, Σ, Δ, s, F), where:
K is a finite set of states
Σ is an alphabet
s ∈ K is the initial state
F ⊆ K is the set of final states, and
Δ is the transition relation. It is a finite subset of (K × (Σ ∪ {ε})) × K
i.e., each element of Δ contains:
a configuration (state, input symbol or ε), and a new state.

M accepts a string w if there exists *some path* along which w drives M to some element of F.

The language accepted by M, denoted L(M), is the set of all strings accepted by M.

A Nondeterministic FSA

L= {w : there is a symbol a ∈Σ not appearing in w}
The idea is to guess (nondeterministically) which character will be the one that doesn't appear.

Another Nondeterministic FSM

L = {w ∈ {a, b}* : the fourth to the last character is a}

Another Nondeterministic FSA

L$_1$ = {w : aa occurs in w}
L$_2$ = {x : bb occurs in x}
L$_3$ = {y : ∈ L$_1$ or L$_2$ }

M$_1$ =

$$\begin{tikzcd}
\arrow[r] & 10 \arrow[r, bend left, "a"] \arrow[loop below, "b"] & 11 \arrow[l, bend left, "b"] \arrow[r, bend left, "a"] & 12 \arrow[loop right, "a,b"]
\end{tikzcd}$$

M$_2$ =

$$\begin{tikzcd}
\arrow[r] & 20 \arrow[r, bend left, "b"] \arrow[loop below, "a"] & 21 \arrow[l, bend left, "a"] \arrow[r, bend left, "b"] & 22 \arrow[loop right, "a,b"]
\end{tikzcd}$$

M$_3$ =

Lecture Notes 5   Nondeterministic Finite State Machines   1

Figure 5.2: Scan of a page from the lecture slide portion of the course packet before modification.

guide, in front of the course notes that would include a summary of the voluntary homework assignments to help students in organizing their study time. The students in CS 341 expressed confusion and frustration in not having a clear picture of what out-of-class work was expected of them.

> "I don't know when to do informal homework. It would be nice to have a suggested schedule for the informal stuff." (survey 1: student 30)

Instructors should keep usability and ease of reference, not just content, in mind when creating student learning material.

Aligning student learning methods with voluntary assignments

If practice for mastery is necessary for understanding the material in Automata Theory, students must have access to practice problems. These problems can be located in a textbook or instructor created materials. The course packet in CS 341 contained a large number of practice problems. Although many students did not take consistent advantage of the practice, the preference students gave toward learning through homework dovetails nicely with Dr. Quilt's belief that the best and most efficient way to learn the material in Automata Theory is through practice. The symbiosis between the instructor's belief in practice for mastery, and the students' preference for learning through homework suggests that as the material gets more difficult, more practice is available. The material analysis revealed that CS 341 does not accommodate this observation. As the material in the course increases in difficulty, the number of practice problems decreases. While creating good problems over difficult material is a challenging task for an instructor, it is necessary if learning through homework is to be accomplished.

Use of scaffolding in voluntary homework solutions

Many times, student materials either contain answers, or no commentary on solutions, to practice problems. Both of these strategies are insufficient for Automata Theory. The practice sets Dr. Quilt created have multiple problems on a single topic followed by complete solutions. The inclusion of solutions to every problem is based on Dr. Quilt's previous learning experiences. Frustrated as a learner when there was no way to check whether she had worked a problem correctly, Dr. Quilt vowed that she would not burden her students with the same insecurity.

> "When I was a student, taking math classes, I was sitting in my room at night on my bed doing my problems and it made me absolutely crazy that I could not check my answers [...] and I vowed that if I ever had anything to do with teaching that I was going to give people the ability to check their answers

> before they did five problems in a row wrong." (first interview with Lily Quilt)

Although a noble enterprise, the decision to provide solutions to every problem should be re-examined. This suggestion is buoyed by the fact that average and low performers tended to work more voluntary homework problems after the first examination and did not seem to receive a lot of help (with respect to performance) from the exercises.

Students complained about the elusiveness of the material in Automata Theory.

> "[...] i get everything once i see the answer, but it's usually difficult for me to come up with the right answer myself."
> (survey 2: student 53)

We believe this is what creates the love/hate relationship the students expressed through all three surveys over the semester. Solving problems is hard, but if the problems are challenging yet solvable, student interest in and enjoyment of solving problems is increased, which can increase students' willingness to learn the material [21].

> "I think the problems and material are really interesting. I enjoy puzzle-like problems." (survey 2: student 6)

> "i enjoy Turing machines, even though i find them hard to do."
> (survey 3: student 11)

The material in Automata Theory is already challenging, so instructors should focus on teaching students enough solution strategies to make problem solving possible. By providing the solutions to every problem, this goal is not realized. Although Dr. Quilt believes that the student will be responsible enough to work problems before viewing the solutions, we find from the student responses on how the voluntary homework is used that this is not the case.

> "they help a lot! i'd say they are one of the most important resources for doing well in the class. the solutions at the back really help." (survey 3: student 84)

More often than not, the students use the voluntary homework to study (viewing the solutions) or to do the required homework (using the solutions as scaffolds).

In order to prepare students to generate solutions from scratch, we suggest that problems should have a mix of three types of solutions: complete solutions, scaffolded solutions, and hint solutions. This step-wise reduction in help on solutions is not novel. Scaffolded and hint solutions can be observed in high school mathematics textbooks [10, 23, 71]. However, we have not observed this weaning method for Automata Theory material.

The complete solutions should be model solutions: correct in every minute detail. We also believe that it would be worthwhile to have these correct solutions trailed by a note addressing “here is what is acceptable on an examination” so that students are privy to what is absolutely correct and what is required for full marks on an examination [47]. Students in CS 341 expressed confusion on what was and was not acceptable on examinations, most specifically when constructing proofs.

> “The expected amount of detail in proofs. Many times I’ll explain something in great detail when I only need to say a few words.” (survey 2: student 8)

The scaffolded solutions should include enough of the solution to guide the student to a completely correct solution but not contain all the steps. For instance, a scaffolded proof may have steps missing, or a finite state machine graph may not have any transitions labeled. In this way the student is led down the right path but must struggle to make the solution complete.

In order to demonstrate scaffolded and hint solutions for homework problems, we have chosen the topic of the Regular Language Pumping Lemma defined earlier. We chose this topic due to students expressing difficulty in understanding the proof construction. See Figure 5.3 for a scaffolded solution showing that the language $L = \{a^n b^n\}$ is not regular. We chose the specific string $a^{\frac{N}{2}} b^{\frac{N}{2}}$ because it creates three cases to consider when constructing the proof.

Show that $L = \{a^n b^n\}$ is not a regular language by completing the partially worked out proof below.

Proof: We use the Pumping Lemma to show that $L = \{a^n b^n\}$ is not regular. Let N be the constant from the pumping lemma such that N$\leq \mid xy \mid$ and w=$a^{\frac{N}{2}} b^{\frac{N}{2}}$.
Case 1: y consists of only a's. Then when we pump y, the number of a's will not match the number of b's violating the condition that the number of a's and b's are equal. Contradiction.
Case 2: y consists of only b's. (finish argument)
Case 3: y is on the boundary of a's and b's. (finish argument)

Figure 5.3: An example of a scaffolded solution.

The third type of solution, hints, is the last step in preparing students to work problems completely on their own. The hint solution would state key points that are necessary in generating a completely correct proof but not why they are the key points or how to use them. In this way the intuition is given in the hint, but the solution based on the intuition must be formulated by the students. We modified the previous scaffolded solution (Figure 5.3) to be a hint solution in Figure 5.4. Although this idea of progressive hardness is not new, we believe it has a large role to play in teaching solution strategies for abstract concepts.

Show that $L = \{a^n b^n\}$ is not a regular language by constructing a proof by contradiction using the Pumping Lemma.

Hint: If you choose w=$a^{\frac{N}{2}} b^{\frac{N}{2}}$, your proof should have three cases.

Figure 5.4: An example of a hint solution.

5.2.2 Motivating good study habits

Practice problems are an effective learning tool only if they are used. Although the students in CS 341 continued to do some voluntary work as the semester went on, the amount of time spent learning in this manner was extremely low. The reasons given included, especially near the end of the semester, other required course work. On the other hand, almost all students continued to do the required homework in CS 341.

We feel this pattern of doing required but not voluntary work can be taken advantage of to compete for student time outside of lecture. Requiring more homework would require a larger percent of the course grade to be dedicated to homework for students to acknowledge the importance of doing the increased amount of work. With this comes the complication of cheating. Both Dr. Quilt and other students acknowledge that any activity done outside the classroom is open to cheating.

> "I understand why grades are weighted towards test more because of cheating when it comes to homework." (survey 1: student 27)

Although "cheating" may be viewed as form of collaboration, which is not discouraged in CS 341, we define "cheating" as a rote procedure for getting answers on work. For example, cheating would be copying code or homework answers from another student without helping in its creation. In Automata Theory, with the abstract nature of the material, simply seeing solutions does not facilitate understanding. Thus cheating should be discouraged.

Balancing cheating with assigning out-of-class work

In order to address this balance between individual assessment through grades and weighting homework as a larger portion of the course grade we suggest a *split passing* standard. Although we coined the term, the idea behind the split passing standard was created by an adjunct professor in the Computer Sciences department, Dr. Don Glenning. Dr. Glenning teaches Java, a programming language, and as such requires a large amount of programming in his course. Because programming is done in open labs, the propensity of students copying each other's code is high. Instead of fighting human nature, Dr. Glenning created the split passing standard which states that students must demonstrate competency in both the homework portion and the examination portion of the course in order to pass. For instance, if homework and examinations were weighted equally, and passing required a performance of 70% or better, a student would not pass if they had a perfect homework score and a 50% on examinations. Although using a split passing standard does not get rid of cheating, we believe it does emphasize to students that cheating will not help them pass, while at the same time allowing for collaboration.

Adjusting first examination to instill proper study habits

Motivating students to do their own homework is an important step in creating a good learning environment. Another step is to motivate viable study habits. In Automata Theory it is very easy for students to be lulled into complacency when studying finite state machines. The concepts are straightforward and most students expressed no difficulty with the concepts, nor did they have a hard time with many questions on the first examination. Unfortunately this may have done a disservice to the students.

Because the material in Automata Theory builds, the students should be exposed to study habits that will help them succeed on later examinations. By making the first examination easy, the students will not understand the level of study required to do well on later tests. We saw this clearly in CS 341. The students rated their performance higher and attended fewer discussion sessions and examination reviews earlier in the semester than in the last third of the semester. This sudden increase in extra learning activities was due in part to the perceived decrease in performance on the second examination. In order to avoid this late discovery of inefficient study habits, we would suggest making the beginning examination just as hard as the later examinations. For examinations over finite state machines, this can be done by incorporating more abstract problems, using rigorous mathematical notation, and including proofs. These problems will need to be mastered for material covering push-down automata and Turing machines. The goal of a harder first examination is to set expectations for the rest of the course. An examination with this learning objective should be weighted less than other, later examinations.

Provide incentive to do voluntary homework

Creating viable study habits early in the semester is important. Dr. Quilt and our student performance typing suggests that good study habits include:

- Working voluntary homework problems on a weekly basis.
- Reading related portions of the textbook and supplementary reading in the course packet before topics are introduced in lecture.
- Attending lecture.
- Actively engaging in problem solving during lecture.

In order to give importance to these activities the students need to see a direct advantage to doing the work. Some students may be mature enough to understand that the advantage will be in having more experience in creating solutions that will translate into an easier time and better performance on examinations. However, we think that most students are not this mature at the undergraduate level. Instead a more tangible advantage should be created. If, for instance, the instructor were to guarantee that a certain number of questions on every examination would be taken directly from the voluntary homework set, we think students would be very motivated to practice. In addition, the average and low performers who were actually working the voluntary homeworks after the first examination would see a direct benefit from these exercises. This combined with more required homework would definitely increase the amount of problems students work out of class.

5.2.3 Creating a better lecture environment

Motivating students to work on material out of class is only part of the big picture. The student performance typing shows that the better performers attended lecture more regularly than lower performers. We also saw a preference for learning by listening and for straight lecture with some discussion by students. This would suggest that for the population in CS 341, lecture is critical. We observed that many students stopped attending, especially in the last third of the semester when the material was the hardest. Those students who were still attending began to complain that the examples used in lecture were not complex enough to help them learn the material.

> "[...] Seems like there isn't a real way to figure out the problems."
> (survey 3: student 24)

We also observed that when challenging examples were used in class, more students interacted with the instructor creating the "discussion" that students preferred. Even Dr. Quilt stated that students liked having example problems worked in lecture. The reason Dr. Quilt gave for not incorporating more examples in class was her belief that students would not read outside of class.

> "I've taken the view that says if I need you to learn it I will cover it in class. I still need you to practice it, but there is no fact that I need you to know that I have not covered in class. I could go the other way and say, I'm expecting you to get the facts out of the reading and then we're just going to do examples in class. But I think that that would probably be a disaster, because I don't think they would read!" (second interview with Lily Quilt)

Without the necessary student motivation to prepare, Dr. Quilt felt obliged to go over every necessary detail in lecture (strong teacher-regulation form of instruction [111]). This left little lecture time for complex examples, much less the strategy for arriving at solutions.

Teaching abstract solution strategies

This neglect of the solution strategy is what students indicated as “tricky” when they talked about the topics in Automata Theory.

> “[...] I mean the material is easy to understand, but coming up with a solution is not easy. Sometimes the solution to a problem is tricky. You either see it right away or you don’t.” (survey 2: student 89)

Without a solution strategy, the answer to any problem is surprising, or simple once you see it but impossible to arrive at. The goal of Automata Theory is not to impress students with the elegance of expert solutions but to get students to be able to solve abstract problems [106]. This process begins by facilitating abstract thinking. Some students already possess this ability, but the majority of computer science students do not. Instead they have been trained to recognize patterns and break problems down into pieces so that an algorithmic solution can be applied. This may be why examples are so useful to these learners, it builds a viable pattern on which to solve other problems.

The question then becomes: how do you teach abstract thinking in a more algorithmic manner? One way is to exemplify the solution strategy used by the instructor. Instead of simply providing the solution to a problem in class, or stating the intuitive leap that makes the problem easy to solve, the students should be exposed to the iterative thought process that lead to the intuition that created the solution. We would suggest that at several times during the semester, and especially during the presentation of new topics, the instructor take the time to “reason” out the solution to an example problem. We believe that by showing the students the “algorithm” by which the solution was created, they will be more able to produce solutions on their own.

Creating more time lecture time for harder topics

Of course this type of example presentation requires more lecture time. Automata Theory has a lot of material to cover and it may not seem possible to create more time in order to present problem solving strategy during lecture. Students, in the third survey, suggested spending less time on finite state machines since they found this abstraction easier to understand than push down automata and Turing machines.

> "I would teach the easier stuff faster and leave a little bit more time to digest the tougher stuff." (survey 3: student 44)

Another possibility might be to move the introduction of the finite state machine abstraction into earlier, prerequisite courses [58, 60, 101]. If this view were accepted and the finite state machine abstraction were presented in prerequisite courses, the first third of CS 341 could be shortened to a review, providing more time for the difficult topics in Automata Theory. A third suggestion involves moving some of the lecture material to required homeworks. We think students would be able to learn these concepts out of class.

If students were given work on required homeworks to help offset lecture requirements, there should be a way for students to assess whether their out-of-class learning is sufficient. Several students suggested the use of quizzes to help assess their learning.

> "Have some non-graded pop quizzes just to let students check their performance from time to time to see if they are keeping up with the material or not." (survey 1: student 94)

The use of quizzes could also be effective motivator for attendance.

Using quizzes as a pro-active learning devices, we believe some lecture material could be moved to quizzes as well. By pro-active we mean that the quiz would be structured such that it would be given out at the end of lecture and the material on the quiz would address topics to be discussed in the next class. Using quizzes to get students to do the necessary reading and preparation for upcoming lectures would not only motivate students to attend lecture, in order to get credit for the quizzes, but would leave more lecture time for problem solving. By allowing the quizzes to be completed outside of class, there is no lecture overhead. The pro-active quizzes should be part of the course grade to encourage completion. To discourage cheating, the pro-active quizzes would be sequentially numbered (Xeroxing impossible) and would have the requirement of being hand written (each student must fill out the quiz by hand, eliminating the possibility of cutting-and-pasting answers from another student). This does not mean that students would not simply copy from one another, but in conjunction with a split passing standard, they would be discouraged to do so.

Another use of quizzes would be to facilitate understanding lecture solution strategies. Quizzes could be designed to introduce problems to be solved in lecture, so that students would have had to have previously thought about how they would solve the problem before seeing the instructor's solution strategy. Because these questions are designed to facilitate understanding, they should be graded by completion standards: if the student suggests a solution they receive full credit for the problem. By allowing the student to struggle before seeing the solution, the "tricks" would be less magical and more mechanical since the strategy demonstrated in lecture could be compared to the original strategy used by the student on the quiz question [91].

Retaining flexibility when teaching multiple sections of a course

By allowing more time for topic coverage in Automata Theory, we believe that individual lectures could become more customized to the students in class. In the first interview with Dr. Quilt, she expressed a reluctance to individualize sections of CS 341. One of the reasons that Dr. Quilt gave for keeping the coverage of topics identical in both sections was her policy in allowing students to attend any of her sections. If the two sections were allowed to set their own pace, then students would be relegated to a single section, and common examinations would not be possible. By using pro-active quizzes and more required homework, the focus of lecture would be devoted to introduction of new topics and exemplifying solution strategies to problems. As long as the introduction to new topics was kept identical between sections, there is no need to introduce identical examples in each section. By allowing for different problems, both type and number, each section of CS 341 could conform to the students needs. Although assessing student needs in a large class may not be as effective as with a smaller number of students, some customization should be possible. For instance, different example problems could be used for each lecture delivery with little modification of lecture formatting. If different examples were used in repetitive lectures, students attending multiple lectures would get more exposure to solution strategies without having to sit through identical problems.

5.2.4 Soliciting student feedback

In conducting the research on CS 341 we noticed that students were very anxious to fill out the surveys once they realized that their opinions would be used to create a better learning environment. In fact, after the second examination, we overhead one student say, “Where is the survey? I want to slam this test!” Students also commented on the last survey that they were happy their opinions had been heard.

> “Thank you for taking the time to evaluate our opinions on this course.”
> (survey 3: student 38)

We feel that the involvement of students through surveys may be a good teaching tool. By using anonymous surveys to gather feedback about learning and specific aspects of the course, an instructor could use the surveys as a formative evaluation of their teaching during the semester [39]. The current system in place at UT has students report their thoughts about the course at the end of the semester when it is too late to make changes. With the use of surveys as a formative teaching assessment, lecture can be dynamically modified throughout the semester to better suit student learning. We also saw evidence that asking topic related questions was a useful tool for anticipating student performance in the course. The survey feedback could be used to help the instructor create challenging, but reasonable examinations and homeworks for the specific students in their course. Additionally, surveys provide psychological advantages for students. Students can use the surveys to vent about the course, as well as have a sense of impacting their own learning environment. We feel these factors help in creating a positive learning environment.

5.3 Recommendations for Future Research

This section proposes follow-up and replication studies, explores the effectiveness of the methods used for this research, and suggests additional analysis of data gathered for our case study of CS 341.

5.3.1 Contribution of pedagogical positivism

The use of pedagogical positivism allows the explicit definition of how to recognize instructional excellence. It can be argued that constructivism defines best practice when most students learn. However, the current definition of constructivism does not define how to measure learning and does not define best practice explicitly. Constructivism at its root is simply an epistemological stance on learning and how factual knowledge is constructed.

Pedagogical positivism makes explicit the definitions of student learning and “best practice”. Student learning, under pedagogical positivism, is defined as the combination of successful completion of the course (performance) and individual reported experience during the learning process (perceptions of learning). Best practice is defined as the circumstance that leads to the learning in a majority of students learning. We do not contradict the implied definition of best practice in a constructivist environment, but rather make it explicit.

5.3.2 Discussion of research methodology

In this discussion we revisit and reflect on the three-part model used for this case study which was first presented in Chapter 3 Figure 3.1. The model was used to examine a college-level course from three aspects: instructor, student, and material. We discuss both positive and negative aspects of the model.

Advantages

An immediate advantage that the model provided was on the structuring of instructor interview questions and student survey items. All questions and items were designed to address directly or indirectly the points listed in each sphere. Similarly, the material analysis was structured to look at clarity, layout, and cost trade-offs. Although interviews and surveys were how we chose to gather data on the points listed in the respective spheres, these techniques were not dictated by the use of the model. This flexibility allows other researchers using our model to define data gathering techniques that better fit their course or learning environment. This flexibility in data gathering techniques makes the model applicable for researchers studying other learning environments, even if they are not college-level courses.

The Venn diagram was useful in understanding what information needed to be gathered directly or data that would fall out during analysis. In general, the points contained in the spheres themselves were gotten through direct methods (interviewing, surveys, or direct observation of the materials). The overlaps were understood through the combination of the directly gathered data. This supports the visual choice of the Venn diagram for the representation of our model. The choice of the Venn diagram suggested that learning was contained in the area with the most overlap, or center of the model. This choice of location for learning is important since it highlights the importance of learning as the indicator of a successful course.

Suggested changes

Overall, we found the model to be very useful in framing the data we were to gather. The independence of the three data sources made simultaneous collection possible. However, we found some minor weaknesses or misalignments with the model's configuration. We discuss those changes here.

In the model presented in Chapter 3, grades were located in the overlap between students and instructor. We had placed grades in the overlap because grades are based on the instructor's view of student work. However, course grades are obtained directly at the end of the semester from the students. To keep alignment with the fact that information gathered directly is located in a single sphere, grades should not have been located in an overlap.

Another drawback to our original model was the number of points to be addressed directly. Additionally, many of the points in the instructor and student spheres were redundant or were not very helpful in the final analysis. Based on what we found helpful in our analysis we suggest the following changes:

Instructor sphere

- "Experience" was not well defined. We suggest modifying the point to read "experience with topic and teaching".
- "Beliefs", "student objectives", and "previous learning experience" turned out to be most influential in how Dr. Quilt built and executed her course. We suggest moving these points to the top of the list in order to emphasize them.

Students sphere

- "Beliefs" was not well defined and was redundant. We suggest removing that particular idea.
- "Interest" was not well defined. We suggest rewording the point to read "interest in course topics".
- "Previous learning" did not seem to influence student survey responses. We suggest replacing this idea with "concurrent obligations" since other courses, family, and employment were influential and noticeable in student responses.
- We suggest removing "grades" from the overlap between instructor and student and adding "performance" to the student sphere.

- “Experience in the course” is redundant when both “performance” and “opinions about the course” are included. We suggest the removal of “experience in the course”.
- “Opinions about the course”, “performance”, and “concurrent obligations” turned out to be the most influential in student perceptions of CS 341. We suggest moving these points to the top of the list in order to emphasize them.

Materials sphere

- “Clarity” turned out to be the biggest factor in instructor and student use. We suggest moving this idea to the top of the list in order to emphasize it.

The changes discussed in this section are reflected in the modified Venn diagram presented in Figure 5.5.

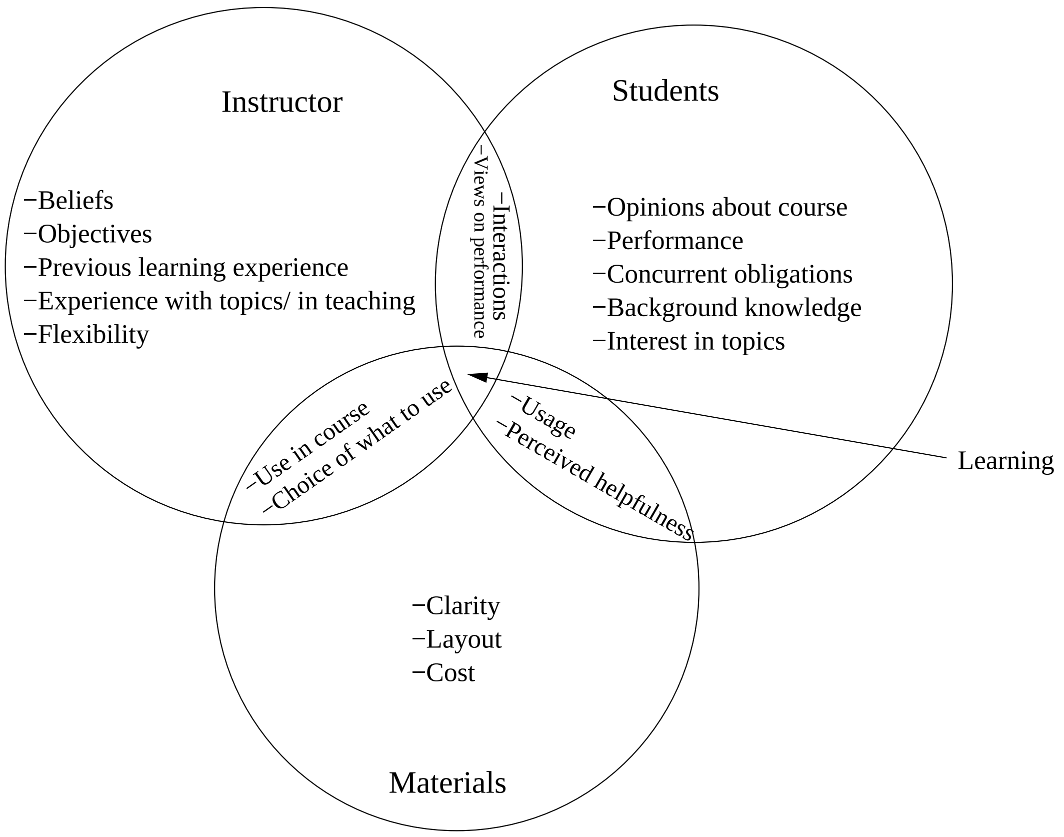


Figure 5.5: A modified version of the model first presented in Chapter 3 after the conclusion of the CS 341 case study.

5.3.3 Future work

The following nine follow-up studies are discussed:

1. Quantitative study examining the effectiveness of proposed changes based on this qualitative case study.
2. Quantitative study examining students' intellectual maturity.
3. Replication study with a less experienced instructor at UT.
4. Replication study with Dr. Quilt's class to determine the effects of the admissions filter that was instantiated Fall 2001.
5. Replication study at a different institution where Automata Theory is a required course.
6. Replication study at a different institution where Automata Theory is an elective course.
7. Effectiveness of pedagogical positivism as a theoretical basis for the study of college-level courses.
8. Effectiveness of using student performance typing analysis.
9. Other ways in which the data from our case study of CS 341 could be analyzed.

We look at each of these recommendations in turn.

Quantitative study examining the effectiveness of proposed changes based on this qualitative case study. By far the most straightforward future research, we plan to implement the suggestions described in this chapter in a specific section of CS 341, and run a quantitative comparison with an unmodified section to understand the viability of our recommendations. This is not only necessary to understand the validity of our suggestions, but to uphold our promise to the students. The rationale stated to the students in CS 341 for completing the surveys was the promise that their opinions would be used to create a better learning environment for later students. Carrying out such a study would keep our promise to the students.

Quantitative study examining students' intellectual maturity. Students who enter a Computer Science program may be intellectually prepared for the discipline. However, since Automata Theory topics diverge from the majority of material taught in other CS courses, it would be interesting to see if students are at an appropriate intellectual level for understanding Automata Theory [72]. It has been shown that incoming Computer Science students come into the discipline with insufficient logical skills [4]. A study of the intellectual level of students in Automata Theory would allow instructors to understand whether they should assume that all students in the course can learn the concepts, or whether there is a part of their class that cannot learn what is being taught.

Replication study with a less experienced instructor at UT. We chose to study Dr. Lily Quilt given her long-term experience in teaching Automata Theory. An interesting study would be to interview a less experienced instructor to see how perceptions on curriculum, learning objectives, and student expectations change with experience. By examining a class conducted by a newer instructor, we could also see which student perceptions persist, regardless of instructor experience, and which perceptions change.

Replication study with Dr. Quilt's class to determine the effects of the admissions filter that was instantiated Fall 2001. A new admissions filter was introduced in the Fall 2001 for undergraduate CS majors. All students wishing to pursue a CS degree must be admitted into the program after the completion of freshman-level classes. Prior to the admissions filter, any student accepted into UT could have chosen to study Computer Science. Because the admissions filter will affect the make-up of the senior population in Fall 2003, it would be interesting to repeat this study at UT to see if in fact the admissions filter is admitting students with a better ability for understanding Automata Theory as Dr. Quilt suspects. We would be able to use this additional data to make suggestions for institutions that employ an admissions filter as well as understand what student perceptions persist, filter or no.

Replication study at a different institution where Automata Theory is a required course A similar study conducted at another university would be useful in understanding how the student perceptions at UT generalize with respect to other institutions. Although many other schools provide a course on Automata Theory [7, 41, 66] with approximately the same material taught in CS 341, there may be differences in approach and progression through the topics. By conducting a study of this type we would be able to understand how UT's curriculum may differ and how valid our findings are to the larger population of Automata Theory courses.

Replication study at a different institution where Automata Theory is an elective course. Not all institutions require Automata Theory for a Computer Science degree. If Automata Theory is not a required course, our findings might not be valid. It would be interesting to see how students in an elective course differed from our sample where students were required to take the course.

Effectiveness of pedagogical positivism as a theoretical basis for the study of college-level courses. The use of pedagogical positivism allowed us to make concrete suggestions for the improvement of Automata Theory instruction. Without this theoretical basis, we would not have been able to generalize pedagogical suggestions. Another benefit of pedagogical positivism is that a single case study can create suggestions for future classes as long as the student population stays stable (the population continues to have the same requirements and similar experiences before entering the course).

Pedagogical positivism also requires the use of many “voices” in qualitative data. By using this theory, both instructor and student perceptions are critical as well as any material that is used in creating those perceptions. In using pedagogical positivism a researcher triangulates findings through the use of several data points in a very natural way. We are interested in finding out how valid pedagogical positivism is for studying other courses in CS and different disciplines.

Effectiveness of using student performance typing analysis. Our student performance typing led to the most concrete evidence in understanding learning outcomes. The performance groups displayed differing learning activities and preferences that would not have been discovered without this type of analysis. By analyzing the data in this manner we were also able to understand the usefulness of quiz questions as a formative instructional tool; students who did well on the quiz survey items tended to do well in the overall course. The strength of our findings from student performance typing suggests that other course studies could benefit from this concrete methodology. We would like to use this methodology in studying other courses in CS and different disciplines that base student performance on end-of-course grades.

Other ways in which the data from our case study of CS 341 could be analyzed. There was a large amount of data collected for this research. Our goal was to understand how instructor and student perceptions aligned and what aspects created a good learning environment for Automata Theory. This goal led us to look at all of the collected data, taking a broad view of the course. However, this is not the only way in which the data can be used.

Similar to the approach used in the Student Performance Typing section of Chapter 4, the data could be used to examine trends among specific groups of students in Automata Theory. The students were asked to report their gender on the first survey, therefore differences between men's and women's experiences in learning Automata Theory is possible. Analysis of the split data could reveal specific issues in Automata Theory that differ between genders. These results could be used to sculpt the course to provide better gender equity.

The data could also be used to investigate approaches for providing better equity for specific ethnic groups. Students were asked to self-describe their ethnic group on the first survey. This information could be used to partition the data in such a way that a group composed of minority ethnicities could be examined in comparison with majority ethnicities. It would be interesting to understand what issues were specific to ethnic groups and how those perceptions could be used to facilitate a better learning environment for Automata Theory or other prerequisite CS courses.

5.4 Conclusions

This section summarizes the practices we recommend for Automata Theory instruction. We present a condensed listing of suggestions, followed by a brief explanation of how each suggestion would facilitate teaching Automata Theory or creating a better learning environment. If the reader is interested in any specific suggestion described below, he or she can read the related sub-sections from the earlier parts of this chapter for a full discussion of the recommendation.

1. Begin an Automata Theory course with an overview of the theory that will be built over the semester. This will help students fit topics into a cohesive schema as new material is presented.

2. Work example problems during lecture. Computer Science students find concrete examples helpful in understanding abstract material.

3. Pose questions during lecture. Even rhetorical questions lend a structure within which

students can gauge whether their understanding of the material is appropriate at the current time in the course.

4. Set realistic learning objectives. Students may not become abstract thinkers during a single semester. Learning objectives should be set to focus on familiarity with formalisms and rigorous mathematical notations, not material mastery.

5. Provide students with a way to review prerequisite knowledge necessary for Automata Theory. In this way students can have a solid basis for understanding whether their prerequisite knowledge is appropriate for the course. If prerequisite knowledge is assumed, have a method by which students can measure their prerequisite knowledge before falling behind in the course. This assessment can be done through self-graded quizzes or should be part of the initial part of the course.

6. Include programming projects as part of the required coursework. Programming is a concrete exercise. Computer Science students have extensive exposure to this type of assignment and are thus comfortable doing this type of work. Programming can also illuminate the usefulness of theory in practice.

7. Use a course website to post solutions to homeworks and examinations. As most of the homework in Automata Theory is completed using pencil and paper rather than a computer, quick feedback is necessary so that students can check their work in a timely manner before their memories of the assignments or examinations fade.

8. If slides are used as a presentation mechanism during lecture, include the slides in student materials. This allows students to focus more attention on lecture content and less attention on taking notes during lecture. When students have copies of the lecture slides, they can better prepare for upcoming lectures and review of past topics. Slides should be marked clearly to convey to students which slides contain complete information and which parts will be completed in lecture.

9. Motivate good examination preparation early in the semester. Because the theory in Automata Theory builds, all examinations are cumulative in nature. If the first examination, which covers less challenging concepts from the first part of the semester is too simple, students may not realize that studying for the second or later examinations requires more work.

10. Provide practice homework problems. Practice with concepts and problem-solving skills in Automata Theory helps in ease and speed of completing examination questions. The practice problems should be structured in a scaffolded manner so that students are eventually weaned from seeing complete solutions. Additionally, if short-hand notations are acceptable on examinations, sample answers should include solutions that show the proper or acceptable use of the notation.

11. Use a split-passing standard which requires students to pass both homework and examinations to pass the course. Such a standard facilitates collaboration on homeworks and also discourages cheating.

12. Allow more time for difficult concepts. Finite state machines can be covered relatively quickly in order to allow adequate time for student understanding of the later concepts. Finite state machines can be covered more quickly by putting the majority of the relevant concepts on required out-of-class work.

13. Use quizzes. The use of quizzes encourages lecture attendance, allows formative assessment of teaching pace and student learning acquisition, and can be used to pro-actively prepare students for lecture.

14. Use student surveys. Students appreciate having the instructor hear their concerns and thoughts about the course. The positive psychological outcome of being involved in the learning process can help create a better learning environment.

In using these fourteen suggestions, instructors of Automata Theory will be able to create an environment better suited for student learning and performance.

Appendix A

Glossary of Terms

To aid readers from the two areas of Education and Computer Science, this glossary splits terminology and acronyms of the two disciplines into separate sections.

A.1 Computer Science terms

The definitions for this section have been taken from or inspired by either the Free On-Line Dictionary Of Computing (FOLDOC) website found at
`http://www.nightflight.com/foldoc/`
or Harvey Mudd College's CS 132 lecture materials (CS 132)found at
`http://www.cs.hmc.edu/courses/2001/spring/cs132/`
`slides/cs132-lect3.pdf`. FOLDOC is a collection of computing jargon for non-computing experts. We felt their definitions would be appropriate for this appendix, however all the terms we wished to include did not appear in FOLDOC. In searching for good definitions, the CS 132 site contained lecture slides that were designed for beginning Computer Science students. We felt the definitions were straightforward and appropriate for this appendix.

abstract machine — A procedure for executing a set of instructions in some formal language (e.g., mathematical notation, grammar, programming language), possibly also taking in input data and producing output. Such abstract machines are not intended to be constructed as hardware but are used in thought experiments about computability.

Chomsky Hierarchy — a classification used to group languages into distinct categories according to the computational power needed to accept or generate them. Each language has specific restrictions for membership strings. As the restrictions become less rigid, membership becomes harder to compute, and the language becomes more difficult to generate. From easy to hard: regular languages, context-free languages, recursive languages, recursively-enumerable languages, undecidable languages.

Chomsky normal form — a context-free grammar with a specific structure: each production rule has a single non-terminal on the left. The right-hand side of each production rule has either two non-terminals or a single terminal symbol.

concatenation — a function that is defined over strings or sets (or languages). The concatenation of two strings, a_1 and a_2, is a new string $a = a_1a_2$ where the two original pieces are adjacent to each other. The concatenation of two sets, R and S, creates a new set T where each member of T is the string concatenation of a member of R with a member of S.

context-free grammar — a grammar consisting of rules where the left-hand side must be a single non-terminal.

context-free language — language produced by a context-free grammar.

context-sensitive grammar — a grammar consisting of rules that can only change a single non-terminal.

context-sensitive languages — a language produced by a context-sensitive grammar. Also called recursively-enumerable languages.

decidable — a language L is decidable if every possible string created from the alphabet of L can partitioned into strings that are contained in L or not contained in L.

determinism — A property of a computation that has a single result.

deterministic — Describes an algorithm in which the correct next step depends only on the current state. This contrasts with an algorithm involving backtracking where at each point there may be several possible actions and no way to chose between them except by trying each one in turn and backtracking if the current choice fails.

DFSM — deterministic finite state machine.

event — a trigger for the next step of a computation to proceed.

finite state machine — An abstract machine consisting of a set of states (including the initial state), a set of input events, a set of output events, and a state transition function. The function takes the current state and an input event and returns the new set of output events and the next state. Some states may be designated as ”terminal states”. The finite state machine can also be viewed as a function which maps an ordered sequence of input events into a corresponding sequence of (sets of) output events.

FSM — (deterministic) finite state machine.

grammar — A formal definition of the syntactic structure of a language, normally given in terms of production rules (rules which produce strings) which specify the order of constituents and their sub-constituents in a sentence (a well-formed string in the language). Each rule has a symbol on the left-hand side naming a syntactic category (e.g. “noun-phrase” for a natural language grammar) and a right-hand side which is a sequence of zero or more symbols. Each symbol may be either a terminal symbol or a non-terminal symbol.

One rule is normally designated as the top-level rule, which gives the structure for a whole sentence.

A grammar can be used either to parse a sentence (see parser) or to generate one. Parsing assigns a terminal syntactic category to each input token and a non-terminal category to each appropriate group of tokens, up to the level of the whole sentence. Parsing is usually preceded by lexical analysis. Generation (a.k.a. creating strings) starts from the top-level rule and chooses one alternative production wherever there is a choice.

Kleene star — (Or “Kleene closure”, named after Stephen Kleene) The postfix “*” operator used in regular expressions, Extended Backus-Naur Form, and similar formalisms to specify a match for zero or more occurrences of the preceding expression. For example, the regular expression “be*t” would match the string “bt”, “bet”, “beet”,..., “beeeeet”, and so on.

language — A set of strings over a given alphabet. The alphabet is a set of characters (i.e., {a,b} or {1,2,3}) that defines the legal string characters.

Lex — A lexical analyzer generator for Unix and its input language. Lexical analysis is the first stage of processing a language. The stream of characters making up the source program or other input is read one at a time and grouped into lexemes (or “tokens”) - word-like pieces such as keywords, identifiers, literals and punctuation. The lexemes are then passed to the parser.

NDFSM — non-deterministic finite state machine.

NDPDA — non-deterministic push-down automata.

NFSM — non-deterministic finite state machine.

NPDA — non-deterministic push-down automata.

non-determinism — A property of a computation; indicated that the computation may have more than one result.

non-deterministic — Describes an algorithm in which there may be several possible actions and no criteria on which to choose between them except by arbitrarily choosing one.

non-terminal — A symbol is the left-hand side of at least one grammar rule.

normal forms — a type of grammar that has distinct guidelines for how production rules can be formed. For instance, the Chomsky normal form for context-free languages restricts the right-hand side to contain a single terminal or exactly two non-terminals.

PDA — push-down automaton or push-down automata, depending on context.

push-down automata — an abstract machine that combines the computational power of a finite state machine with a stack memory.

push-down automaton — singular of push-down automata.

recursive language — languages produced by a context-sensitive grammar that can be decided by a Turing machine, i.e. for every string a Turing machine can decide in finite time whether the string is in the language or not.

recursively-enumerable language — a language produced by a context-sensitive grammar.

regular expression — Any description of a pattern composed from combinations of symbols and the three operators: concatenation, union, and Kleene star.

The earliest form of regular expressions (and the term itself) were invented by mathematician Stephen Cole Kleene in the mid-1950s, as a notation to easily manipulate "regular sets" (regular languages), formal descriptions of the behavior of finite state machines, in regular algebra.

regular grammar — a grammar where the left-hand side of the rule must be a single non-terminal and the right-hand side is a sequence of terminals or a sequence of terminals followed by a single non-terminal.

regular language — language produced by a regular grammar or the evaluation of a regular expression.

semi-decidable — a language L is semi-decidable if every possible string created from the alphabet of L that is a member of L can be identified in finite time.

state — a computational singleton that has memory of the current step of a computation.

state transition — the necessary criteria for a state to progress to another state.

string — a sequence of characters. Examples are: “abba”, “5 + 7 = 42”, “it will rain this afternoon”. Sets of strings create a language.

terminal — a symbol in a grammar that corresponds to one “lexeme” - a part of the sentence with no internal syntactic structure (e.g., a number, an identifier, or an operator in a computer language).

TM — Turing machine.

token — a syntactically significant object such as an identifier or a number.

Turing machine — A hypothetical machine defined in 1935–1936 by Alan Turing and used for computability theory proofs. It consists of an infinitely long “tape” with symbols (chosen from some finite set) written at regular intervals. A pointer marks the current position and the machine is in one of a finite set of “internal states”. At each step the machine reads the symbol at the current position on the tape. For each combination of current state and symbol read, a program specifies the new state and either a symbol to write to the tape or a direction to move the pointer (left or right) or to halt.

In an alternative scheme, the machine writes a symbol to the tape *and* moves (a.k.a., write-and-move) at each step. This can be encoded as a write state followed by a move state for the write-or-move machine. If the write-and-move machine is also given a distance to move then it can emulate a write-or-move program by using states with a distance of zero. A further variation is whether halting is an action like writing or moving or whether it is a special state. Halting is the general signal that the Turing machine has finished computing.

undecidable — a language L is undecidable if there is no guarantee that members of L can be decided in finite time.

union — a function used for language definitions. Union takes two definitions and allows choice between the two for the creation of strings. If the language L is defined as "a^* union b^*" then strings of L would be of the form: $a, b, aa, bb, aaa, bbb, aaaa, bbbb, \ldots$.

Unix — An interactive time-sharing operating system invented in 1969 by Ken Thompson after Bell Labs left the Multics project.He said his original motivation for developing Unix was so he could play games on his scavenged PDP-7. Dennis Ritchie, the inventor of C, is considered a co-author of the system.

The turning point in Unix's history was when it was reimplemented almost entirely in C during 1972–1974, making it the first source-portable operating system. Unix subsequently underwent mutations and expansions at the hands of many different people, resulting in a uniquely flexible and developer-friendly environment.

By 1991, Unix had become the most widely used multi-user general-purpose operating system in the world.

Yacc — The LALR (Look Ahead Left Recursive) parser generator found on most Unix systems. An LALR parser generator is a program that takes a formal description of a grammar (e.g. in BNF) and outputs source code for a parser that recognizes valid strings obeying that grammar and performs associated actions.

A.2 Education terms

The definitions in this section have been taken from or inspired by Xrefer's free showcase website, located at `http://www.xrefer.com/search.jsp`.

constructivism — a cognitive theory stating that knowledge is actively constructed by the individual learner, thus teaching must address individual differences. This notion is generally attributed to Jean Piaget.

formative assessment — the use of assessment at an intermediate step in a teaching or learning activity in order to improve the intended outcome.

logical positivism — A philosophical movement that arose from the Vienna Circle in the 1920s. Influenced by Mach and Wittgenstein, it insisted that philosophy should be scientific, regarding it as an analytical (rather than a speculative) activity (see analytic philosophy), the purpose of which was clarification of thought. Any assertion claiming to be factual (i.e. excluding the axioms of logic or mathematics) has meaning only if its truth (or falsity) can be empirically tested. Metaphysical propositions and those of esthetics and religion are consequently meaningless, since it is impossible to say how they can be verified. A secondary goal of logical positivism was the analysis and unification of scientific terminology. After the Nazi invasion of Austria, members of the Circle emigrated to Britain and the USA, where the movement continued to be influential.

mixed-method — Term used to describe the mixing of qualitative and quantitative techniques in research.

pedagogical positivism — theoretical model used in this research. Instruction can be improved for a specific population (e.g., Computer Science undergraduate students at The University of Texas at Austin) of learners through examining teacher and student perceptions about teaching and learning in a course. See Chapter 3 for a more in-depth definition.

pedagogy — the science of teaching.

positivism — Philosophical system developed in the 19th century by A. Comte. Starts from the assumption that all knowledge is based on positive and observable facts, and therefore, directly or indirectly, on the findings of the physical sciences. Hence, in particular, a system that rejected metaphysics and other a priori speculation.

qualitative — a research approach concerned with meaning, rather than with measurement. The emphasis is on subjective understanding, communication, and empathy, rather than on prediction and control. A tenet of qualitative research is that there is no separate, unique, ’real’ world. Qualitative methods vary, and are generally based on empirical research, but there is some discussion over the extent to which the researcher should intervene, and much awareness of the way in which any research process will affect the subjects of the investigation.

quantitative — a research approach concerned with measurement rather than meaning. The emphasis is on prediction and control, and is a driven by the view that there exists a unique reality.

scaffolding — the use of teaching techniques to build a framework in which novice learners become more experienced. The expert (teacher) creates a situation in which a novice (student) can accomplish learning by creating tasks that gradually wean the novice from needing instructional help. The notion of scaffolding is attributed to Vygotsky [68].

social constructionism — Analysis of ‘knowledge’ or ’reality’ or both as contingent upon social relations, and as made out of continuing human practices, by processes such as reification, sedimentation, habitualization. Schutz’s phenomenology—the analysis of the structure of the common-sense world of everyday life—is an important influence, although current exponents draw on a variety of sources including hermeneutics, the later Wittgenstein's intersubjective theory of meaning, and the Marxist conception of praxis (which emphasizes how knowledge and politics are contingent upon work and economic relations). Social constructionists do not believe in the possibility of value-free foundations or sources of knowledge, nor do they conceptualize a clear objective-subjective distinction, or a clear distinction between 'knowledge' and 'reality'. The position, therefore, has profound implications for the practice and philosophy of science, and for political philosophy.

social constructionist belief — a belief founded upon a social-constructionism view of truth.

transmission-based instruction — an instructional method where the student is told the information to be learned. Straight lecture environments display this type of instruction.

Appendix B

Course Description

This Appendix details the information about the instance of CS 341 studied for this research. All information was copied from the course website.

B.1 Course Description

In this class, we will develop a single framework in which all kinds of computational problems can be defined and analyzed. The framework is based on the idea of a language, and problems are defined as the task of determining whether an input string is a member of some particular language. Thus this area is often called formal language theory. But a key idea is that all problems can be described as language recognition tasks so this framework shouldn't be thought of as limiting. Instead think of it as a simple mechanism by which problems that may initially appear very different can be compared and analyzed to see whether there can exist computational solutions to them at all, and, if there can, what power those solutions must possess. We will discuss computational models ranging from very simple (finite state machines) to pushdown automata (with the power to recognize context-free languages, including standard programming languages) to Turing Machines, which are powerful enough to solve any problem for which a solution exists, yet simple enough to describe that we can prove theorems about what they can and cannot do.

B.2 Class information

Uniques: 52065 TTh 11:00-12:30 PAI 3.14

52070 TTh 9:30-11:00 TAY 2.106

Textbook and other materials: The text book for this class is Theory of Computing: A Gentle Introduction, Efim Kinber and Carl Smith. Prentice-Hall, 2001.

Many students find that they want another book, both for alternative explanations and for problems. I recommend Introduction to the Theory of Computation, Michael Sipser. Brooks/Cole, 1996.

There is also a course packet that you should get at the Coop. The packet has three sections:

- Copies of the transparencies that will be used in class. You can take your class notes on these pages so that you will not have to spend each class period trying to copy my material.
- Homework problems with solutions. You should do the homework problems and check your solutions. Even though we won't collect these homework problems, you will have a really hard time in the class if you don't do the homeworks every week. If you can solve the homework problems, you should have no difficulty with the exams.
- Supplementary reading materials for some of the topics that will be covered in class.

If you look at the transparencies section, you will notice that at the beginning of each topic, there is a note telling you which homeworks to do and what (if any) supplementary materials you should read.

Evening Problem sessions There are several opportunities for instruction outside of class and office hours.

Wednesdays, except the weeks of the exams:	4:00 - 6:30	WRW 102
Sunday of weeks of the exams:	6:00 - 10:00 pm	TAY 2.106
Review for Final: Sunday December 8	6:00 - 10:00 pm	TAY 2.106

Homework The only way to learn the material in this class is to practice. It's like learning to play the piano. You can't learn much just by watching someone else. You actually have to do it yourself. You should plan to spend at least five hours every week working problems. The course packet contains a large number of homework problems. Most of them also have solutions so you can check your work as you go along. You will not be asked to turn in these problems.

In addition, about every two weeks there will be a homework assignment that you will be asked to turn in. You may work with other students on these homework assignments. But:

- Each student must turn in his own work with his own name on it. You may not turn in something that is word for word identical to someone else's. In other words, you may work together to figure out how to solve a problem, but each person must then write up the solution independently.
- You must indicate on your paper the names of the other people with whom you worked (whether they're in our class, some other class, took the class last year, or whatever).

Grading I think we all wish that we could have courses without grades. You hate worrying about grades. I hate having to assign grades. But grades are essential to insuring that your degree has the value it deserves. So we have to have a grading system and that system has to have two essential properties:

- It has to be fair to everyone in the class.

- It has to be a true measure of how much each student knows about the class material.

The system I will use in this class will assign grades as follows:

- Homeworks: 10%
- Midterm Exam 1: 20%
- Midterm Exam 2: 20%
- Final Exam: 33%
- Projects: 14%
- Survey participation: 3%. (If you respond to all three surveys, you will earn 3 points. Otherwise, you will earn none of these points.)

Evening Exams You can come to either the 5:00 exam or the 7:00 one, whichever you prefer.

	5:00 - 7:00	7:00 - 9:00
October 8	PAI 4.42	PAI 4.42
November 5	PAI 4.42	PAI 4.42

Useful Resources There is a very nice FSM simulator available from The University of Warwick. You may find that using it helps you a lot in getting the idea of how FSMs work.

Students with Disabilities Any student with a documented disability (physical or cognitive) who requires academic accommodations should contact the Services for Students with Disabilities area of the Office of the Dean of Students at 471-6259 (voice) or 471-4641 (TTY for users who are deaf or hard of hearing) as soon as possible to request an official letter outlining authorized accommodations.

Additional Class Policies You should read the CS Department Code of Conduct. The policies described there will be followed in this class.

Appendix C

Course topics outline

The original timeline of topics for CS 341 is presented here. This information was taken from the course website.

Week	Topics
Aug. 29	A Three Hour Tour Through Automata Theory
Sept. 3	A Three Hour Tour Through Automata Theory, continued
	Review basic techniques
	What is a Language?
Sept. 10	Regular Languages
	Finite State Machines
	Reminder **Thursday: Self-Evaluation Exam**

Sept. 17	Nondeterministic Finite State Machines
	Interpreters for Finite State Machines
	A Review of Equivalence Relations
	Equivalence of Regular Languages and FSMs
Sept. 24	Languages That Are and Are Not Regular
Oct. 1	State Minimization
	Review of Regular Languages and Finite State Machines
	Context-Free Grammars
	Reminder **Tuesday: Project 1 due**
Oct. 8	Parse Trees
	Pushdown Automata
	Reminder **Tuesday: Midterm 1 at 5:00 and at 7:00**
Oct. 15	Pushdown Automata and Context-Free Languages
	Grammars and Normal Forms
	Top Down Parsing
Oct. 22	Top Down Parsing
	Bottom Up Parsing, including lex and yacc
Oct. 29	Languages That Are and Are Not Context-Free
Nov. 5	Exam Review
	Turing Machines
	Reminder **Tuesday: Midterm 2 at 5:00 and at 7:00**

Nov. 12	Computing with Turing Machines
	Recursively Enumerable and Recursive Languages
Nov. 19	Turing Machine Extensions
	Grammars and Turing Machines
	Reminder **Tuesday: Project 2 due**
Nov. 26	Undecidability
	Reminder **Thursday: Thanksgiving**
Dec. 3	Undecidability
	Review

Appendix D

Sample Test and Homework Questions

This section contains sample test and homework questions pulled directly from examinations and homework assignments that were collected in CS 341.

D.1 Sample test questions

The two midterms were given during the semester at night and students had two hours to complete the examination. The final exam was held during the final examination period assigned by the university and students had three hours to complete the examination. Sample questions were chosen to show examples of both procedural and abstract thinking problems.

First midterm

1. Let $\Sigma = \{a, b, c\}$. $L = \{w \in \Sigma^* : \exists z \in \Sigma^* : w = zxz\}$
   Write a regular expression to define L, and construct a FSM that accepts L.
2. State whether $L = \{a^i b^j c^k : i \leq 2j \leq 3k\}$ is regular or not and prove your answer.

3. Consider any function $f(L_1) = L_2$, where L_1 and L_2 are both languages over the alphabet $\Sigma = \{0, 1\}$. A function is $nice$ iff whenever L_2 is regular L_1 is regular. State whether or not $f(L) = \{w : \exists x \in L \text{ and } w = x00\}$ is nice and prove your answer.

Second midterm

1. Let $L = a^n b^+ a^m : n \geq 0$ and $\exists k \geq 0, m = 2k + n$. Write a grammar that generates L, and construct a PDA that accepts $L\$$.
2. State whether $L = \{a^n b^{n*m} c^n : m, n \geq 0\}$ is regular, context-free but not regular, or neither. Prove your answer. Make sure, if you say that a language is context-free, that you show that it is not also regular.
3. Let Alt be a function that maps from any language L over some alphabet Σ to a new language L' as follows:

   $Alt(L) = \{x : \exists y, n\ y \in L, \bar{y} \bar{=} n, n > 0, y = a_1 ... a_n, \forall i \leq n\ a_i \in \Sigma$, and $x = a_1 a_3 a_5 ... a_k$, where k=(n if even(n) and n-1 otherwise)$\}$.

   Consider $L = a^n b^n$. Clearly describe $Alt(L)$.

Final exam

1. Determine whether $L = \{M : M$ is a TM, Σ of $L(M) = \{0, 1\}$ and $\forall w$, if w contains any instance of the character 1 then $w \notin L(M)\}$ is regular, not regular but context-free, not context-free but recursive, not recursive but recursively enumerable, not recursively enumerable. Prove that the language is in the appropriate language category by writing an appropriate grammar, exhibiting an appropriate recognizing machine, or using closure properties. Also prove that the language is not in the next most restricted category.

Note 1: if the language is a set of Turing machine descriptions and it is recursive, you do not need to prove that it is not context-free.

Note 2: if the language is not recursively enumerable then just prove that it is outside this language category (you have no way of proving what category it is in.)

2. Let $L = \{a^n b^{2^n+1}, n > 0\}$. Show a grammar that generates L. Just as with any code you write, comment your grammar so that it is clear what phases exist and what each one is doing.
3. Let R be a function from $2^{\{1,2\}^*}$ to $2^{\{1,2\}^*}$ defined as follows:
   $R(L) = \{w : \exists x \in L$ and $w = xx^R\}$. Are the regular languages closed under R? Prove your answer.

D.2 Sample homework questions

There were 7 required homeworks and 22 voluntary homeworks. Each of the seven required homeworks, we listed one representative problem. A comparable question from a related voluntary homework was chosen for comparison.

Required homeworks

1. Consider the statement: $\forall L_1, L_2,\ L_1 = L_2$ iff $L_1^* = L_2^*$. Is this statement true or false? Prove your answer.
2. Construct a non-deterministic finite state machine to accept
   $\{a^n b a^m : n, m \geq 0, n \equiv m (mod\, 3)\}$.
3. State whether or not $L = \{a^i b^j : 0 \leq i < j < 2000\}$ is regular or not and prove your answer.
4. Construct a context-free grammar that generates
   $L = \{xc^n : x \in \{a, b\}^*, \#_a(x) = n$ or $\#_b(x) = n\}$.

5. Build a PDA for the following language:

   $L = \{a^n b^m : m \geq n$ and $m - n$ is even $\}$.

6. Define a Turing Machine M that computes the function $f : \{a, b\}^* \longrightarrow N$, where $f(x) =$the unary encoding of $max(\#_a(x), \#_b(x))$. It's okay to describe M in English, but your description must be precise.

7. Consider the Post Correspondence Problem as described in the lecture notes. Express this problem as a language and who that it is recursively enumerable (i.e., show that the problem is semi-decidable). You do not need to prove that it is not also recursive.

Voluntary homeworks

1. (from HW #3) Is the following statement $(L_1L_2)^* = L_1^*L_2^*$ true or false for all languages L_1 and L_2?

2. (from HW #6) Draw a state diagram for a non-deterministic finite automata that accepts the language described by $(ba \cup b)^* \cup (bb \cup a)^*$.

3. (from HW #9) Show that $L = \{a^n b^j :| n - j |= 2\}$ is or is not a regular language.

4. (from HW #11) Consider the language

   $L - \{a^m b^{2n} c^{3n} d^p : p > m, andm, n \geq 1\}$. What is the shortest string in L? Write a context-free grammar to generate L.

5. (from HW #13) Construct a pushdown automata that accepts

   $L = \{w \in \{a, b\}^* : w$ has twice as many a's as b's $\}$.

6. (from HW #18) Give a Turing machine (in our abbreviated notation) that computes the function from strings in $\{a, b\}^*$ to strings in $\{a, b\}^* : f(w) = ww^R$.

7. (from HW #21) Consider the problem of determining whether M ever moves its head to the left when started with an input w, given a Turing machine M

and a string w. Is the previous problem solvable, or undecidable? Explain your answer carefully.

Appendix E

Programming project descriptions

There were two programming projects assigned during the semester. Each was due at 4pm on the due date. Late projects were marked off 20% if up to 24 hours late, 50% for up to 48 hours late. Projects were not accepted 48 hours after the due date.

The programming project descriptions were copied from the course website.

E.1 First programming project: A finite state machine simulator

In this project, you will build a simulator for nondeterministic finite state machines. (But remember that since every deterministic FSM is also a valid NDFSM, your simulator will also be able to handle deterministic FSMs.)

You will notice that the requirements outlined here are somewhat loose. That is intentional. I want you to have the opportunity not just to implement someone else's specifications (i.e., mine) but to think about what a reasonable set of specifications for this task would be. As you are designing your simulator, keep in mind the following:

If you were a user of your simulator, how would you want it to behave?

Your simulator will take as input:

(1) The definition of the machine that is to be simulated. The form for this definition is given below.
(2) One or more input strings. Your program will then simulate the behavior of the machine described in (1) on each of the input strings it is given.
(3) ... Another machine definition and its inputs, and so forth.

Thus a typical input sequence might be:
Definition of machine 1
Input 1
Input 2
Input 3
Definition of machine 2
Input 1
Input 2
End of session

You should use the following format for the definition of each machine whose behavior is to be simulated:
First line: a, b, c, d, ... (elements in the alphabet)
Second line: q1, q2, q3, ... (states in the set of states)
Third line: s (initial state)
Fourth line: f1, f2, f3, ...(all of final states)
Fifth line: q0, a, q1 (first element of the transition function (deterministic) or transition relation (nondeterministic))
Sixth line: q1, b, q2 (second element)
......(if you have more rules)

You may make whatever decisions you like about the details of this input format (use of commas, parentheses, etc.) as long as you document your decisions in your user documentation.

After entering a machine description, the user should be able to input a string (e.g. aabb) he wants to test. Your program should read the string and output the result: yes (if the string is accepted) or no (if the string is not accepted).

To get full credit on this assignment, it is important that you:

(1) Provide clear documentation on your program. You need to provide both user documentation (the details of what the inputs to the program look like), as well as internal documentation of your code. Make sure that your internal documentation starts with a brief description of the algorithms and data structures you are using.

(2) Make sure that your program handles both deterministic and nondeterministic FSMs. Be particularly careful of how you handle epsilon transitions in nondeterministic machines.

(3) Write a simulator that will not crash even if the input it receives is not legal (e.g., there is a transition that involves a state that is not part of the machine). Make reasonable decisions about what should happen given illegal input and document those decisions. At a minimum, you must output some sort of error message informing the user of the problem.

(4) Guarantee that your simulator performs correctly if the Dead state is not mentioned explicitly in the machine definition.

Note that program efficiency is not on this list. You may use any algorithm that works. You do not have to spend a lot of time looking for the most efficient possible one. But feel free to talk about whatever design tradeoffs you thought about in your program documentation.

You can use Pascal, C, C++, Java, Perl, Lisp, or Prolog to implement the simulator. If you want to use other languages, please tell us as soon as possible.

A set of standard test cases will be posted on the course homepage two days before the due date. Therefore, it is assumed that you should finish your program two days before the due date and use that two days to run the program with the test cases and print out everything you have to turn in.

What you should turn in:

1. A printout of the source code, including documentation,
2. A copy of the user documentation,
3. The hardcopy of the program output by running the test cases that will be posted on the class website. If there are additional test cases that you want to use to show off features of your program, include them also.

This project is to be done completely on your own. You should not show your code to anyone except the professor or the TAs, nor should you look at anyone else's code. If we find violations of this rule, they will be treated as cases of academic dishonesty.

E.2 Second programming assignment: Using lex and yacc

The purpose of this project is to use yacc/lex to experiment with the use of context-free grammars and parsers. You can do this project either by yourself or in a group of two people. The project is divided into two parts. If you choose to do it by yourself, you need only finish the first part. If you do it in a group of two people, you have to finish the two parts in order for both people to get full credit. Also, before you choose the two-person option, you should consider the fact that the second part requires you to write new C code for the actions associated with new grammar rules. So, if you choose this option, one of you needs to be a C programmer.

Because both lex and yacc are relatively complex tools, this project is structured as a set of specific exercises for you to do. You'll start with a simple yacc input example, run it, and then make modifications to it. Instead of turning in a program, you'll turn in a notebook with the results of your experiments with the two systems.

yacc is a Unix command that converts a context-free grammar specification into an LALR(1) parser. The core of yacc is its algorithm for creating the parsing tables that an LALR parser needs. yacc can cope with an ambiguous grammar; it exploits its own heuristic rules as well as user-supplied precedence rules to resolve ambiguities.

lex is a program that creates a lexical analyzer from a user supplied set of regular expressions and associated actions. The job of a lexical analyzer is to examine an input string and to convert it into a set of input tokens, where each token represents a syntactically significant object such as an identifier or a number. A lex-produced lexical analyzer and a yacc-produced parser can be combined to form the front end of a language interpreter. They work as follows:

For more information on lex and yacc, check the links on the class web site.

Both lex and yacc were originally created to run in a Unix environment. You should find them on all the department's Unix servers. However, other versions of both of them now exist as well. Flex and Bison are faster, gnu versions of lex and yacc (respectively). Flex++ and Bison++ use C++ instead of C as their execution environments. There are also Windows and Mac versions of lex and yacc. If you want to work in one of those environments, search the web to find information on the system you want.

There's also a system called JavaCC, which is similar to lex and yacc, but it generates top-down (as opposed to bottom-up) parsers and uses Java as the execution environment. You may use this if you like.

At the due date, you should turn in a document with one section for each numbered piece of the problem.

Make sure that you show your lex and yacc source code at each step where you make changes.

1. You will find lex and yacc input files for a small calculator program on the class web page.

   Also on the class web site Project page you'll find links to reference manuals for lex and yacc.

Run lex and yacc on those files. Compile the resulting yacc and lex outputs to create the executable a.out. If you have trouble doing this, retype the files. Sometimes .txt files created on one system are not interpreted correctly on a different one.

Run a.out on three examples of your own, and then on the following inputs and print out the results:

- 2-3+5
- (((100-10)-20)-(30+2))
- 2-20-13-4-20
- 2-(20-13)-4
- (20-(20 +(2 -(2 + (2 + (2- 2))))))
- 10 - ((((2-(4-2)-30)+2) - 10)-2)

2. Enhance the grammar that you give to yacc to handle ”*” and ”/”. You will also need to add these new symbols to your lex program. Rerun yacc and recompile a.out. Run it on three examples of your own, and then on the following inputs and print out the result:

   - 2*3+5
   - (((100*10)-20)-(30*2))
   - 2-20-13-4/2
   - 2-(20-13)-4/2
   - 20*(20 + 2 *2) + (2 + 2) * 2
   - 10 - ((((2*4-2-30)*3+2)*10)-2+3)

   Print out your lex and yacc files at this point and include them with your answers to this question. Label them clearly as with the step number.

What do you notice about the answers you are getting? Why are you getting those answers?

3. When yacc compiles a grammar into a parser, it produces a parse table that describes the state transitions that the parser will take. To see what these state transitions are, run yacc with the -v option and print the resulting file y.output.

4. Now you're going to trace the execution of the parser so you can see what's going on.

   (a)Again consider the next to the last expression of question 2:

   20*(20 + 2 *2) + (2 + 2) * 2

   Trace this one by hand and by using the debugging capability provided by yacc. For the hand trace, use the following format:

   Step

   Stack

   Input

   Action

   1

   $(0)

   4+5$

   shift

   2

   $(0)4(2)

   +5$

   reduce using expr -¿ NUMBER

3

$(0)expr(5)

+5$

shift

4

$(0)expr(5)+(3)

5$

shift

5

$(0)expr(5)+(3)5(2)

$

reduce using expr-¿NUMBER

6

$(0)expr(5)+(3)expr(5)

$

reduce using expr -¿ expr '+' expr

7

$(0)expr(5)

$

accept

Notes: In the stack column, each entry is shown twice. First, not in parenthesis, is the input token. Second, in parenthesis, is the state of the parser. You'll need to use the parse table that you printed in step 3 to see what these states are. So, for example, when we say that the stack contains

$(0)expr(5)+(3)

we're saying that the actual stack contains the states 0 5 3 and that they came from the inputs expr +.

$ represents the end of the input and the start symbol in the stack.

Some of the lines will get long. You may want to turn your paper into landscape mode.

Indicate on your trace where any shift/reduce or reduce/reduce conflicts occur.

Now trace the parsing of this same expression, this time using the runtime tracing facility that yacc offers. To invoke it, you need to turn on the debugging flag in the C declarations section of the grammar file. So insert the following code at the beginning of your .y file.

%{

#include <string.h> int yydebug = 1;

%}

Then, to make debugging mode actually happen at run time, you need to compile the yacc-generated .y file using the -t option, for example

> yacc -t expr.y

When you run the code, you will see debugging information similar to:

Starting parse Entering state 0 Reading a token: ? 1 Next token is 258 (NUMBER) Shifting token 258 (NUMBER), Entering state 1 Reducing via rule 2 (line 16), NUMBER $-$ $>$ expr state stack now 0 Entering state 3 Reading a token: Now at end of input. Reducing via rule 1 (line 13), expr $-$ $>$ Line 1 state stack now 0 Entering state 10 Now at end of input. Shifting token 0 ($), Entering state 11 Now at end of input.

Again, indicate where any conflicts have occurred.

(b) Trace the parsing of the last expression of question 2:

10 - ((((2*4-2-30)*3+2)*10)-2+3)

Just do this one using the automatic tracing facility. But you need to look at the trace and indicate where conflicts occurred and how yacc resolved them.

5. Test your calculator program using input such as the following to see how yacc processes the ambiguous grammar it was given:

   * 40-25-6-10-7

   * 40+4/2*2-10

   As in problem 4, show the parse steps, pointing out where the conflicts occur. You can do this by hand, or using yydebug.

6. yacc allows users to specify, in addition to a set of grammar rules, special rules for defining the precedence and associativity of operators. Specify the precedence and associativity of the operators '*', '/', '+', '-'. You should make sure that '*' and '/' have higher precedence than '+' and '-' , and all of them should be left associative. Test your new program and output the parse steps for the same inputs you used in step 5.

   Print out your lex and yacc files at this point and include them with your answers to this question. Label them clearly as with the step number.

7. Enter some invalid expressions, such as "4+5-", "5++4", and see what happens. Add error rules to allow users to re-enter a new expression once an error has been encountered. Print examples of what happens in these circumstances. Make sure that you show that your system can actually recover from an error. So, after being told that there's an error, if you enter a valid expression, it will process it correctly.

   In order to handle errors correctly, yacc must be able to clear its stack and to skip over input until it gets to a reasonable place to start over. You should first try to figure out how to do this on your own. There are a couple of different approaches that will work. There's a section in the yacc manual on how to handle errors. But it's not real clear, so if you're not succeeding, look at my solution.

Print out your lex and yacc files at this point and include them with your answers to this question. Label them clearly as with the step number.

8. Add the unary operator "-". This is of course the same symbol as the one we use for subtraction. Test this addition and show the output.

   Print out your lex and yacc files at this point and include them with your answers to this question. Label them clearly as with the step number.

9. Add to your grammar and to your lex rules the ability to accept two functions:

   * invert(expr), which converts a negative integer to positive and vice versa.

   * square(expr), which returns the square of the result of expr, that is, square(2*3) = 36.

   Test your new system on the following inputs:

   * invert(-5+4) * invert(invert(-5+4)) * square(5) * invert(-square(square(6*(4+5))))

   Output the parse steps of this last example. You can do it by hand or by using the tracing facility, but if you use the tracing facility, annotate it so it's clear that you know what it is doing. Are there any conflicts? If so, indicate them.

   Now test your program on several of your own inputs.

   Print out your lex and yacc files at this point and include them with your answers to this question. Label them clearly as with the step number.

10. Answer the following questions:

    (a) What is lexical analysis and what kind of rules does Lex use to describe it?

    (b) What is syntactic analysis and what kind of rules does Yacc use to describe it?

    (c) What is a token?

(d) What conflicts may occur in an ambiguous grammar?

(e) How does yacc resolve these ambiguous grammars?

(f) What are the possible actions yacc will take when it processes an input?

PART II

11. Enhance the program in Part I to implement the assignment statement:

    statement $\rightarrow$ id := Expr

    id $\rightarrow$ NAME

    where expr is an arithmetic expression as in Part I and id can be any valid identifier. Run the enhanced program and print its output.

12. Enhance both your lex and yacc programs to handle Boolean expressions. Assume the following grammar for Boolean expressions:

    BE $\rightarrow$ BE or BE

    BE $\rightarrow$ BE and BE

    BE $\rightarrow$ not BE

    BE $\rightarrow$ (BE)

    BE $\rightarrow$ term op term

    BE $\rightarrow$ true | false

    term $\rightarrow$ NUMBER | id

    op $\rightarrow$ $<|<=|=|!=|>|>=$

    Illustrate the operation of this new capability by running your program on the following inputs:

    * true and false or true

    * true and (false or true)

* true or (3>5) and (4!=5)

* ((3>4) or (4>2)) and true

* (true and (false or (true and (false or (3<=3)))))

13. Enhance your lex and yacc programs to handle the if-then-else statement, using the following grammar rules:

    S $\rightarrow$ if BE then S1

    S $\rightarrow$ if BE then S1 else S1

    S1 $\rightarrow$ expr | BE where expr is the expression defined in Part I.

    Try your program on the following input and output the parse steps in the format of step 4:

    * If a=3 then if b= 4 then 5 else 6+8

14. Answer the following questions:

    (a) Is your final grammar ambiguous?

    (b) What kinds of conflicts might occur in parsing this grammar?

    (c) If these conflicts occur, how does yacc resolve them?

    (d) Give two example inputs for each kind of conflict, and show the parse steps using the format in step 4, Part 1.

    (e) What is an alternative way to handle conflicts?

Appendix F

Instructor Interview Questions

The first interview was conducted on August 21, 2002; the second on November 4, 2002; and the third on December 19, 2002.

Questions were generally asked in the order shown below. When the flow of conversation brought up the answer to a question before it was asked, the researcher re-stated the question followed by, "I think we covered this." The interviewee could then either agree or elaborate.

F.1 First interview questions

1. Why was LIN 340 reassigned to the Computer Sciences department?

2. How did you create the course and choose what material to use?

3. How did the course change over the year(s) you have taught it? What is the rationale for the changes you made?

4. How did you learn the material that you teach? Did you have any problems with learning the material yourself? If yes, is this reflected in the curriculum?

5. What prerequisite knowledge do you expect students to have? What is the reason behind having students have this prerequisite knowledge? How will you deal with students that are weak (i.e., do not have the prerequisite knowledge)?
6. What areas of the course do you expect students to have trouble with? Is your answer based on past observations and/or grades? Have you changed the course to better help students in these areas? Have these changes helped?
7. What dictates the time spent on each topic during the semester?
8. How easy is it for you to change class materials, like the textbook?
9. How aligned are the other sections of CS 341 in terms of curriculum, pace, and materials?
10. What types of students do well in your class?
11. How do you help students that are having trouble?
12. What advice do you give students that are struggling?
13. Do you rely heavily on TAs? What are the responsibilities of the TAs in your class?
14. Do students utilize your office hours? Describe a typical student interaction during your office hours.
15. Do you provide more contact time with students aside from lecture and office hours, for instance study sessions before examinations?
16. Have you thought of alternative formats, beside lecture, for teaching this class? What is your rationale behind using just a lecture format?
17. What types of assignments do you give students and what is the reason these assignments are given? What are the students expected to gain from doing the homework exercises?

18. How are tests created for your class? What is the purpose of examinations, what are you checking for?

19. What is the grading policy in your class? How are grades determined? Are grades negotiable?

20. How do you end each semester? Do you have time to reflect after the course? Do you have post-class meetings with you TAs?

F.2 Second interview questions

1. Have you made any changes to the syllabus? What were the reasons behind the changes?

2. How is the current pace of class? What you expected?

3. How do you feel students are doing in the class this far?

4. Have you had any students attend office hours? Describe a typical discussion.

5. What advice are you giving students at this point?

6. What do you feel students are/are not learning well? Based on what evidence?

7. What was your reaction to the first exam? How did you create the questions for the first exam?

8. Do you feel like this is a weed-out course?

9. How have you been trying to involve students in lecture?

10. What types of questions do you ask students in lecture? What is your reasoning behind asking questions? How long do you typically wait before moving on after asking a question?

11. How closely aligned are the two sections in terms of time to cover the day's material, class participation, student understanding...?
12. What do you anticipate students will have the most problems with on the next exam? How are you going to try to help alleviate this problem?
13. I've noticed that some topics you cover in lecture are very high level and you tell the students that they do not need to know the details of what is being discussed. (e.g. normal forms, just need to know the theorem is right not how to prove it, etc.) What is your reason behind presenting these topics at all?

F.3 Third interview questions

1. What was your reaction to the second exam? How did student performance on the second exam match your expectations?
2. What was your reaction to the topics covered in class since the second exam? Do you feel you had enough time to cover the material? Do you feel the students understood the material?
3. Were the students in this semester different than other semesters? If so, how?
4. Did you receive any verbal feedback about the class from students? If so, what did they say?
5. What is your reaction to the final exam? How did student performance on the final match your expectations?
6. What do you hope students walked away from this class knowing? What do you think they actually learned?
7. Were you happy with the performance of your TAs this semester?

8. What grading algorithm are you going to use to determine final grades in the course?
9. If you could change anything about the course what would it be?
10. Are you going to change anything about the course? Based on what? Describe the changes.
11. Any last thoughts?

Appendix G

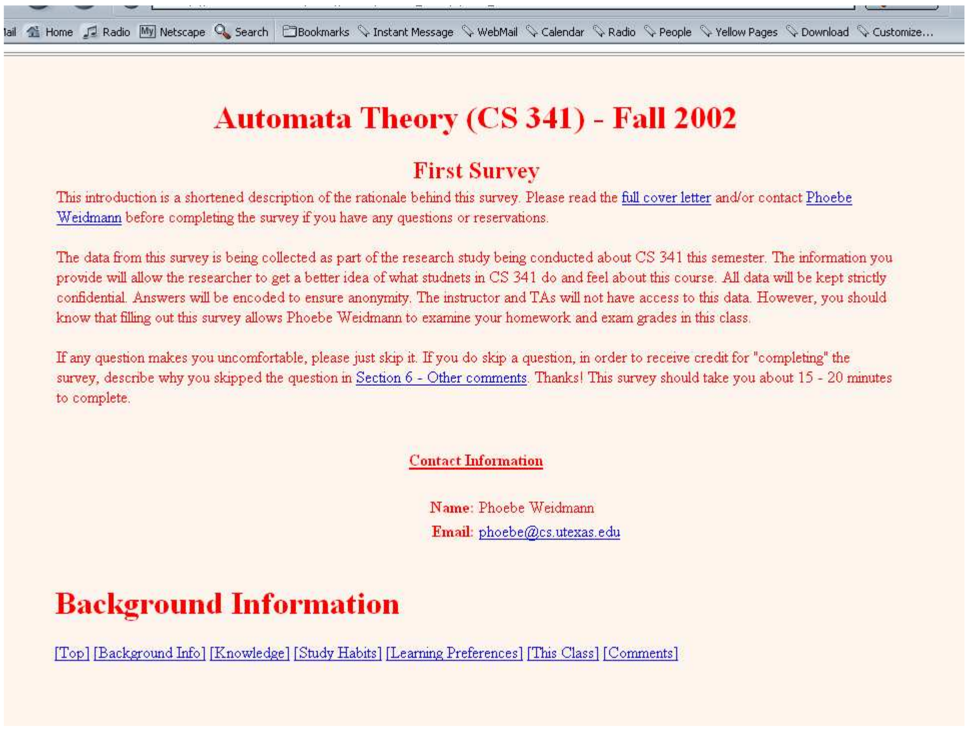
Student survey content

Each survey started with a brief statement describing highlights of the full cover letter. Figure G.1 provides a sample of the introduction formatting on each survey.

The first survey was posted online Friday, September 13, 2002 and the students were told they had until Saturday, September 21st to complete it. The survey was actually taken offline on Monday, September 23, 2002. This survey was conducted directly after the 12th class day so as to avoid adds and drops changing the student population in the class.

The second survey was posted online Friday, October 18, 2002. The students were told they had until Sunday, October 27th to complete it. The survey was actually taken offline on Monday, October 28, 2002. This survey was conducted at approximately mid-semester.

The third survey was posted online Monday, December 2, 2002. The students were told they had until Monday, December 9th to complete it. The survey was actually taken offline on Tuesday, December 10, 2002. This survey was conducted during the last week of classes.


Mail Home Radio Netscape Search Bookmarks Instant Message WebMail Calendar Radio People Yellow Pages Download Customize...

Automata Theory (CS 341) - Fall 2002

First Survey

This introduction is a shortened description of the rationale behind this survey. Please read the full cover letter and/or contact Phoebe Weidmann before completing the survey if you have any questions or reservations.

The data from this survey is being collected as part of the research study being conducted about CS 341 this semester. The information you provide will allow the researcher to get a better idea of what studnets in CS 341 do and feel about this course. All data will be kept strictly confidential. Answers will be encoded to ensure anonymity. The instructor and TAs will not have access to this data. However, you should know that filling out this survey allows Phoebe Weidmann to examine your homework and exam grades in this class.

If any question makes you uncomfortable, please just skip it. If you do skip a question, in order to receive credit for "completing" the survey, describe why you skipped the question in Section 6 - Other comments. Thanks! This survey should take you about 15 - 20 minutes to complete.

Contact Information

Name: Phoebe Weidmann
Email: phoebe@cs.utexas.edu

Background Information

[Top] [Background Info] [Knowledge] [Study Habits] [Learning Preferences] [This Class] [Comments]

Figure G.1: Formatting of online survey introduction.

G.1 First survey

The first survey (currently inactive) is available online at the following location:

`http://www.cs.utexas.edu/users/phoebe/student_surveys/done_first.html`

Background Information

1.1 Name:

1.2 Gender (male/female):

1.3 Race (for understanding student population makeup):

1.4 Age:

1.5 Is this your first time taking CS 341? (y/n)

1.6 Why did you choose this section of CS 341?

1.7 When (e.g. Fall'02) and from whom did you take CS 336?

1.8 What was your final grade in CS 336 (A–F,Q)?

1.9 When (e.g. Fall'02) and from whom did you take PHL 313K?

1.10 What was your final grade in PHL 313K (A–F,Q)?

1.11 Which of the following do you plan to take before you graduate (already taken, yes, undecided, no)?

   a CS 353, Theory of Computation

   b CS 357, Algorithms

   c CS 375, Compilers

1.12 When do you plan to graduate (e.g. Fall'02)?

1.13 What classes are you taking this semester (please list them)?

1.14 How many hours per week do you work at a job (if no job, put zero)?

1.15 What do you want to do when you graduate (e.g. go to graduate school, get a job at IBM, etc...)?

1.16 How do you think the material covered in this course will be helpful to you after you graduate?

Background knowledge

2.1 Describe, in your own words, what an equivalence relation is.

2.2 What properties must be present in an equivalence relation (e.g. reflexive, anti-symmetric)?

2.3 If “p implies q” does “not(p) imply not(q)” (y/n)?

2.4 What is the cardinality of a set containing 145 elements?

2.5 If a PDA can accept a context-free language, can it accept a regular language(y/n)?

Study habits

3.1 Have you been doing the voluntary homework in the packet (y/n)?

3.2 If yes, how much time do you spend on it per day (in hours)?

3.3 What is your preferred style of studying in most classes (by myself, with others)?

3.4 When you do study with someone else (or if you would study with someone else) what do you get (think you would get) out of the interaction?

3.5 Do you anticipate having difficulty with the material in this class (yes, maybe, no)?

3.6 What motivates you to study (e.g. feeling like I have to do it, fear of a bad grade, I only do homework if it’s graded, etc...)?

3.7 When do you do homework (e.g. keep up with it on a regular basis, do it a day before it is due)?

Learning Preferences

4.1 Do you like to actively participate in classes you take (e.g. ask questions, speak up during whole class discussions)(regularly, occasionally, no)?

4.2 Have you participated in CS 341 so far (y/n)?

4.3 By which of the following methods do you prefer to learn (choose all that apply): reading, listening, doing written homework, coding, other–specify.

4.4 Which style of class do you prefer (straight lecture, whole-class discussion, mixture of lecture and whole-class discussion, small group interactions, other–specify)?

4.5 Do you like having the lecture slides printed out for you (y/n)?

4.6 Why do you like/dislike having lecture slides?

4.7 Do you take notes in class (y/n)?

4.8 If you answered “yes” to the previous question, please explain how often you take notes and what you write down.

4.9 How do you prefer having your grade for a course determined (mostly by exams, more weight given to homework, other–specify)?

4.10 Pretend for a moment that you are an instructor for this course and need to create a grading scheme. What would your grading scheme look like (e.g. what would homework, exams and projects be worth; would you have other graded material handed in; etc...)?

Class so far

5.1 What grade do you expect to earn in CS 341 (A–F,Q)?

5.2 The first two lectures of the semester were designed to present the “big picture”. How helpful were these for you (very, somewhat, very little, no help)?

5.3 The examples presented during lecture are intended to help you understand the material. How helpful have the examples been for you (very, somewhat, very little, no help)?

5.4 The pace of the course is defined by the amount of material covered during each lecture. How do you find the pace of the course (much too slow, somewhat slow, about right, somewhat fast, much too fast)?

5.5 What is your reaction to having copies of the slides as part of the course packet (very useful, somewhat useful, neutral, not useful, other–specify)?

5.6 What is your reaction to having extra problems with solutions available to help you as you practice (very useful, somewhat useful, neutral, not useful, other–specify)?

5.7 How many of the Wednesday night discussion sections have you attended (0–3)?

5.8 Do you use the textbook (extensively, sometimes, occasionally, no)?

5.9 Explain your response to the previous question about your use of the textbook.

5.10 Which of the following best describes how you feel about this class at this time (very easy, somewhat challenging, very challenging, very frustrating, other–specify)?

5.11 At this time, what about CS 341, if anything, is confusing to you (e.g. definition of prefix, expectations about informal homework, etc...)?

5.12 At this time, what do you like best about CS 341?

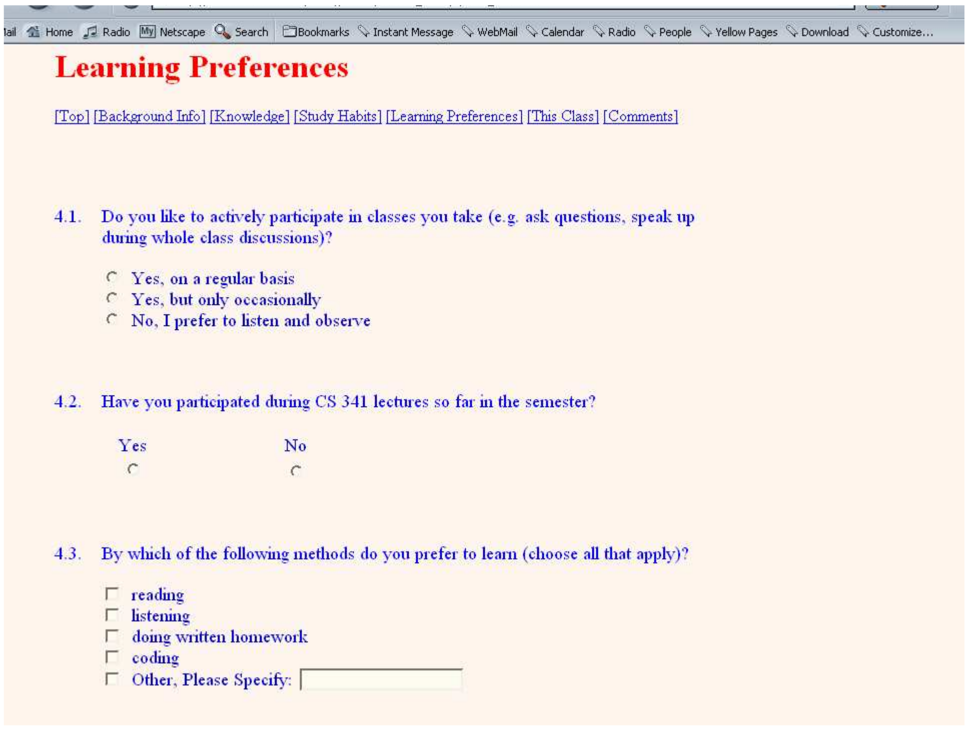
5.13 At this time, what do you like least about CS 341?

5.14 What recommendations do you have at this time for improving CS 341?

Other Comments

6.1 Add any other thoughts you would like to share with the researcher.

G.1.1 Samples of online formatting for first survey



Learning Preferences

[Top] [Background Info] [Knowledge] [Study Habits] [Learning Preferences] [This Class] [Comments]

4.1. Do you like to actively participate in classes you take (e.g. ask questions, speak up during whole class discussions)?

- Yes, on a regular basis
- Yes, but only occasionally
- No, I prefer to listen and observe

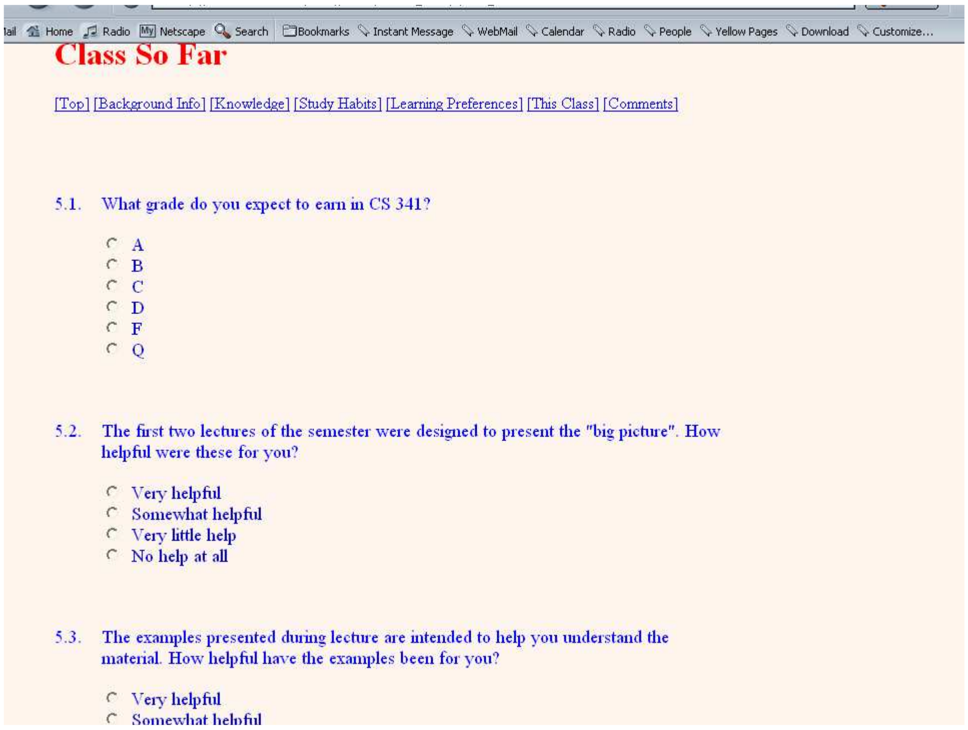
4.2. Have you participated during CS 341 lectures so far in the semester?

Yes	No
○	○

4.3. By which of the following methods do you prefer to learn (choose all that apply)?

- ☐ reading
- ☐ listening
- ☐ doing written homework
- ☐ coding
- ☐ Other, Please Specify:

Figure G.2: Sample one of online formatting for first survey.



Class So Far

[Top] [Background Info] [Knowledge] [Study Habits] [Learning Preferences] [This Class] [Comments]

5.1. What grade do you expect to earn in CS 341?

- A
- B
- C
- D
- F
- Q

5.2. The first two lectures of the semester were designed to present the "big picture". How helpful were these for you?

- Very helpful
- Somewhat helpful
- Very little help
- No help at all

5.3. The examples presented during lecture are intended to help you understand the material. How helpful have the examples been for you?

- Very helpful
- Somewhat helpful

Figure G.3: Sample two of online formatting for first survey.

G.2 Second survey

The second survey (currently inactive) is available online at the following location: `http://www.cs.utexas.edu/users/phoebe/student_surveys/done_second.html`

Personal Information

1.1 Please enter your name:

1.2 When did you start your degree program here at UT (e.g. Fall'02)?

1.3 With what degree are you going to graduate (B.A. CS, B.S. CS, Graduate degree in CS, other–specify)?

1.4 Did you complete the first survey (y/n)?

Knowledge concepts

2.1 Assuming Sigma is {a,b}, will the following machine $\mathcal{M}$ accept $\mathcal{L} = \{x : abba\ is\ a\ substring\ of\ x\}$?

Description of $\mathcal{M}$: Machine consists of five states in a single row labeled 1 - 5. State 1 has a self-loop labeled “a,b” and a transition to state 2 labeled “a”. State 2 has a transition to state 3 labeled “b”. State 3 has a transition to state 4 labeled “b”. State 4 has a transition to state 5 labeled “a”. State 5 has a self-loop labeled “a,b”. State 1 is the start state and state 5 is the only final state.

2.2 What type of language does the following grammar produce (regular, context-free but not regular, or neither)?

$$\mathcal{G} = \boxed{\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & aA \mid a \\ B & \rightarrow & Bb \mid b \end{array}}\ where \sum = \{a, b\}$$

2.3 Does the following PDA accept $\mathcal{L} = \{a^n b^n : n > 0\}$ (y/n)?

Description of PDA: Machine consists of two states in a single row labeled 1 and 2. State 1 has a self-loop labeled “a/X/” and a transition to state 2 labeled “b//”. State 2 has a self-loop labeled “b//X”. State 1 is the start state and state 2 is the only final state.

Studying

3.1 Have you done any of the required homeworks (y/n)?

3.2 If you have NOT done any of the required homeworks, please explain.

3.3 If you have done at least one of the required homeworks did you do them alone or with other students (always, usually, sometimes, never)?

   a alone

   b with one other student

   c with two or more other students

3.4 Did you study for the first exam alone or with other students (alone, with one other, with two or more, did not study)?

3.5 Have you done any of the voluntary homeworks (y/n)?

3.6 If you have done any of the voluntary homeworks, then which ones (all of it, part of it, did not do)?

   a #1

   b #2

   c #3

   d #4

   e #5

   f #6

   g #7

   h #8

   i #9

   j #10

   k #11

   l #12

   m #13

n #14

3.7 (Choices were: essentially everything, more than half, about half, less than half, essentially nothing)

  a Have you read any material in the textbook?

  b Have you read any material in the supplementary reading portion of your packet?

3.8 Are the notes helpful to you at this point in the course (y/n)?

3.9 Have you attended one of the Wednesday night discussion sections (y/n)?

3.10 Did you attend the first exam review (y/n)?

3/11 With your current class schedule, what class do you spend the most time on and why (e.g. CS 375 because it has a lot of required homework, CS 341 because I love to work out the problems...)?

Class so far

4.1 At this point in the course, do you attend lecture (y/n)?

4.2 Explain why you do or do not attend lecture.

4.3 Have you participated in lecture so far (y/n)?

  a asked questions

  b answered an instructor question

  c posed a problem

  d corrected an in-class example

  e other

4.4 Do you think the first exam was fair (in the sense that all material was covered in class) (y/n)?

4.5 Do you think the first exam was a good evaluation of what you know about finite state machines and regular grammars (y/n)?

4.6 If you could sum up your feelings about the first exam in a single statement, what would it be (good, made some stupid mistakes, had a very hard time with one question, had a very hard time with several questions, other–specify)?

4.7 What,if anything, would you have changed about the first exam?

4.8 Did you complete the first programming assignment (y/n)?

4.9 Do you feel that you learned anything from the first programming assignment (y/n)?

4.10 Describe what you learned from the first programming assignment.

4.11 Do you find the examples used in lecture to be helpful to your understanding of the material (y/n)?

4.12 How do you like the current pace of the class (amount of material covered in each lecture)(about right, too slow, too fast)?

4.13 Which of these statements best matches your view of CS 341 at this time (class is easy, class is "just right", class is hard)?

4.14 At this time what, if anything, is confusing to you?

4.15 At this time are you worried about your grade in this course (y/n)?

4.16 Are you worried about passing this course (y/n)?

4.17 Since the last survey (the past 5 weeks of class), what do you like about this class?

4.18 Since the last survey (the past 5 weeks of class), what do you not like about the class and how would you fix it?

Miscellaneous

5.1 What did you hear about this course before taking it? From what source did you hear it (e.g. friend, university web-page, etc.)?

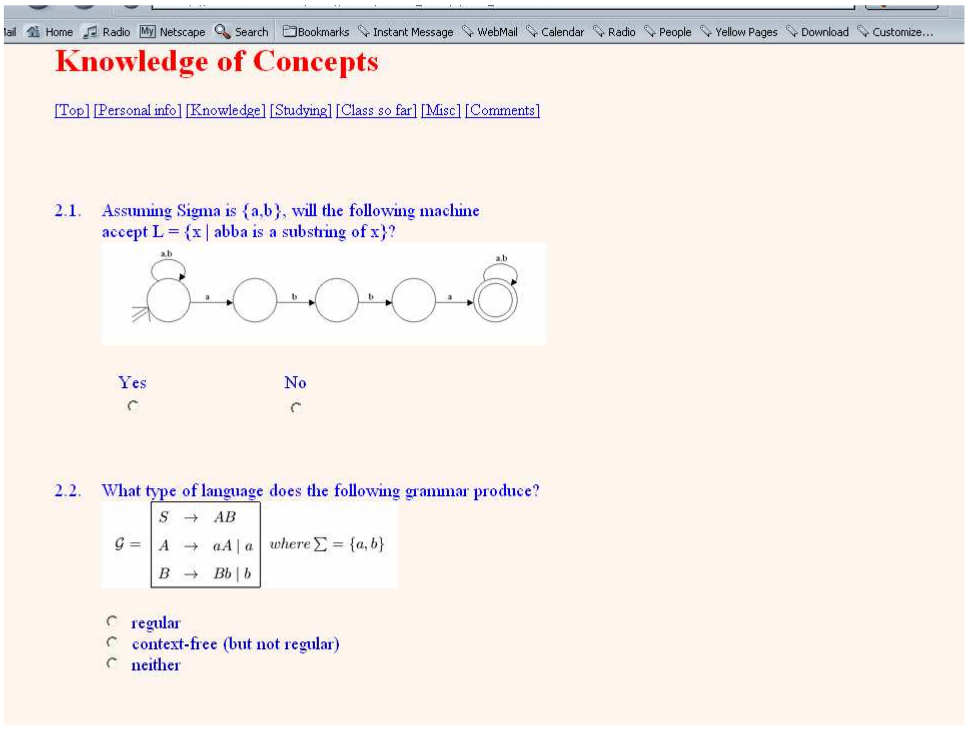
5.2 Would it have been helpful to you to have a lecture on how to create general proofs from scratch (y/n)?

5.3 Think about the material covered before the first exam. Looking back, would you have been comfortable being introduced to non-deterministic finite state machines before deterministic finite state machines (y/n)?

Other comments

6.1 Add any other thoughts you would like to share with the researcher at this time.

G.2.1 Samples of online formatting for second survey



Knowledge of Concepts

[Top] [Personal info] [Knowledge] [Studying] [Class so far] [Misc] [Comments]

2.1. Assuming Sigma is {a,b}, will the following machine accept L = {x | abba is a substring of x}?

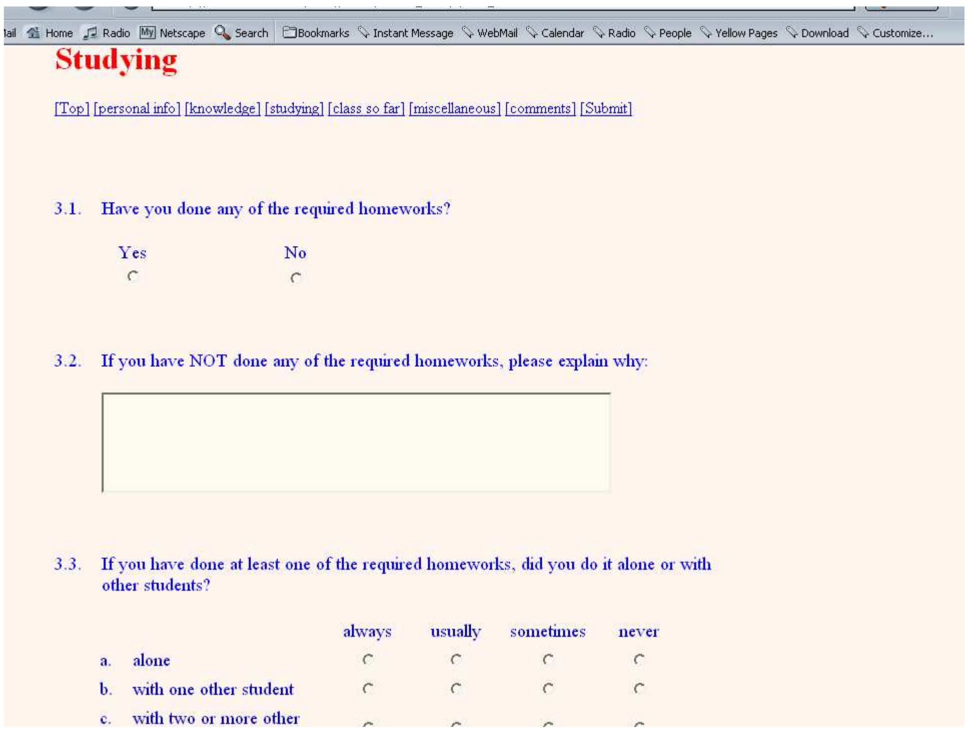
Yes No

2.2. What type of language does the following grammar produce?

$$\mathcal{G} = \boxed{\begin{array}{lcl} S & \rightarrow & AB \\ A & \rightarrow & aA \mid a \\ B & \rightarrow & Bb \mid b \end{array}} \quad where \sum = \{a, b\}$$

- regular
- context-free (but not regular)
- neither

Figure G.4: Sample one of online formatting for second survey.



Studying

[Top] [personal info] [knowledge] [studying] [class so far] [miscellaneous] [comments] [Submit]

3.1. Have you done any of the required homeworks?

Yes	No
○	○

3.2. If you have NOT done any of the required homeworks, please explain why:

3.3. If you have done at least one of the required homeworks, did you do it alone or with other students?

	always	usually	sometimes	never
a. alone	○	○	○	○
b. with one other student	○	○	○	○
c. with two or more other	○	○	○	○

Figure G.5: Sample two of online formatting for second survey.

G.3 Third survey

The third survey (currently inactive) is available online at the following location:
`http://www.cs.utexas.edu/users/phoebe/student_surveys/done_third.html`

Personal Information

1.1 Please enter your name:

1.2 Which of the earlier surveys did you complete (y/n)?

a first survey

b second survey

Knowledge of concepts

2.1 Consider a language L. Assume it is possible to construct a semi-decider Turing Machine M for L and a semi-decider Turing Machine M' for the compliment of L. What type of language is L (recursive, recursively enumerable but not recursive, not recursively enumerable, none of these choices)?

2.2 Consider a language L consisting of descriptions of exactly those Turing Machines that halt when given the input “aba”. (We do not care what the machines do on any other inputs.) Can a Turing Machine be build to decide L (y/n)?

2.3 Assume that H represents the halting problem language as described in class. I am curious about a language L, which may or may not be recursively enumerable. If I can construct an algorithm that uses H to solve every instance of L, what can I say about L (since H is recursively enumerable, so is L; if you have a solution for L, then you have a solution for H; none of the above)?

Studying

3.1 Have you done any of the voluntary homeworks since exam 2 (y/n)?

3.2 Which of these voluntary homeworks have you done (all of it, part of it, none of it)?

a #15

b #16

c #17

d #18

e #19

f #20

g #21

h #22

3.3 Please explain your reasons for doing or not doing the voluntary homeworks.

3.4 Since exam 2... (essentially everything, more than half, about half, less than half, essentially nothing)

a ...have you read any material in the textbook?

b ...have you read any material in the supplementary reading portion of your packet?

3.5 Are the notes from the course packet helpful to you at this point in the course y/n?

3.6 How many of the Wednesday night discussion sections have you attended since exam 2 (none, one, at least two)?

3.7 Did you study for the second exam alone or with other students (alone, with one other, with two or more, did not study)?

3.8 Did you attend the second exam review (y/n)?

3.9 If you attended the second exam review, what did you like or dislike about it?

Class so far

4.1 At this point in the course, do you attend lecture (y/n)?

4.2 Explain why you do or do not attend lecture.

4.3 Have you participated in lecture so far (y/n)?

a asked questions

b answered an instructor question

c posted a problem

d corrected an in-class example

e other

4.4 Do you think the second exam was fair (all material was covered in class) (y/n)?

4.5 Do you think the second exam was a good evaluation of what you know about PDAs and context-free grammars (y/n)?

4.6 If you could sum up your feelings about the second exam in a single statement, what would it be (good, made some stupid mistakes, had a very hard time with one question, had a very hard time with several questions, other–specify)?

4.7 What,if anything, would you have changed about the second exam?

4.8 Did you complete the second programming assignment (y/n)?

4.9 Do you feel that you learned anything from the second programming assignment (y/n)?

4.10 Describe what you learned from the second programming assignment.

4.11 Do you find the examples used in lecture to be helpful to your understanding of Turing machines (y/n)?

4.12 How do you like the current pace of the class in terms of the amount of material covered in each lecture (about right, too slow, too fast)?

4.13 Which of these statements best matches your view of CS 341 at this time (this class is easy, this class is "just right", this class is hard)?

4.14 At this time what, if anything, is confusing to you?

4.15 Which statement best describes your feelings about the performance you anticipate on the final exam (confident I will do well, fairly certain I will do OK, somewhat worried about how I will do, very worried about how I will do)?

4.16 Discuss your strategy for succeeding on the final exam.

4.17 Focusing on the class after exam 2, what do you like about this class?

4.18 Focusing on class after exam 2, what do you not like about this class?

4.19 How would you fix the aspects of the course that you dislike?

Wrap-up

5.1 How interested were you in theoretical Computer Science material when you started this course (very, somewhat, neutral, somewhat disinterested, very disinterested)?

5.2 Describe your interest in theoretical Computer Science material now compared to when you started this course (much more, a bit more, about the same, a bit less, much less).

5.3 How useful do you think the material covered in this class will be to you after graduation (very, somewhat, not at all, other–specify)?

5.4 How well do you feel the prerequisite theory courses prepared you for this course? In other words, how well did your background knowledge do in allowing you to understand the foundational material without heavy review (well prepared, adequate preparation, inadequate preparation–my fault, inadequate preparation–not my fault, other–specify)?

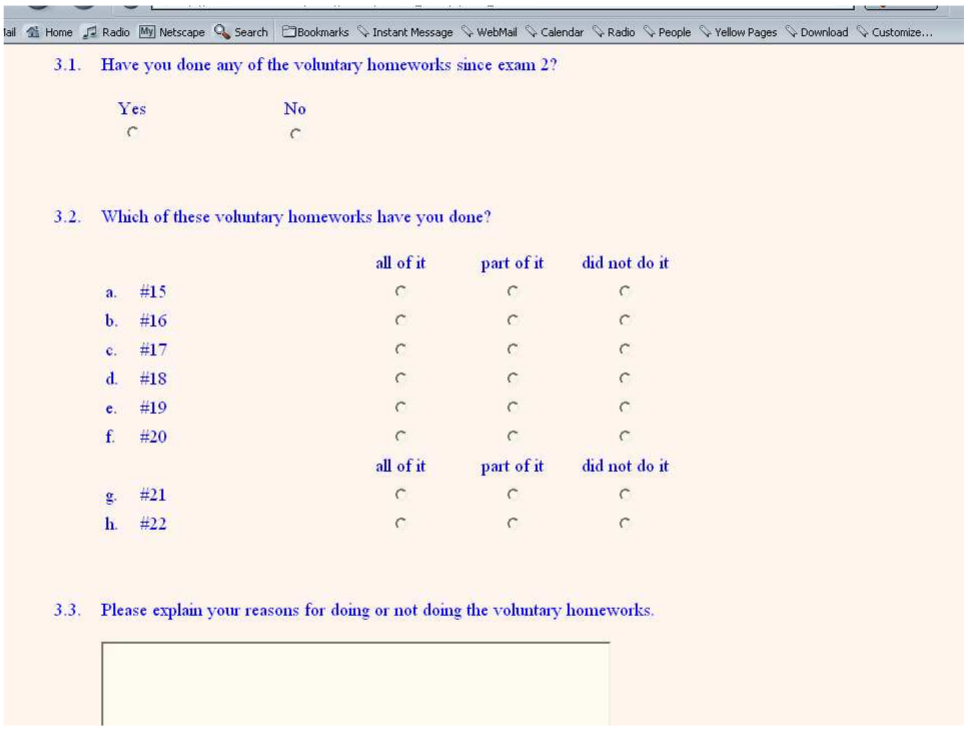
5.5 Pretend you are an instructor for this course. Describe how you would design your exams and explain your reasons for including or not including specific types of questions or content.

5.6 Pretend that you are in a position to change this course in any way except to get rid of it (so students must still take it, but anything else about it could be changed). How would you change the course?

Other comments

6.1 Add any other thoughts you would like to share with the researcher at this time.

G.3.1 Samples of online formatting for third survey



ail Home Radio Netscape Search Bookmarks Instant Message WebMail Calendar Radio People Yellow Pages Download Customize...

3.1. Have you done any of the voluntary homeworks since exam 2?

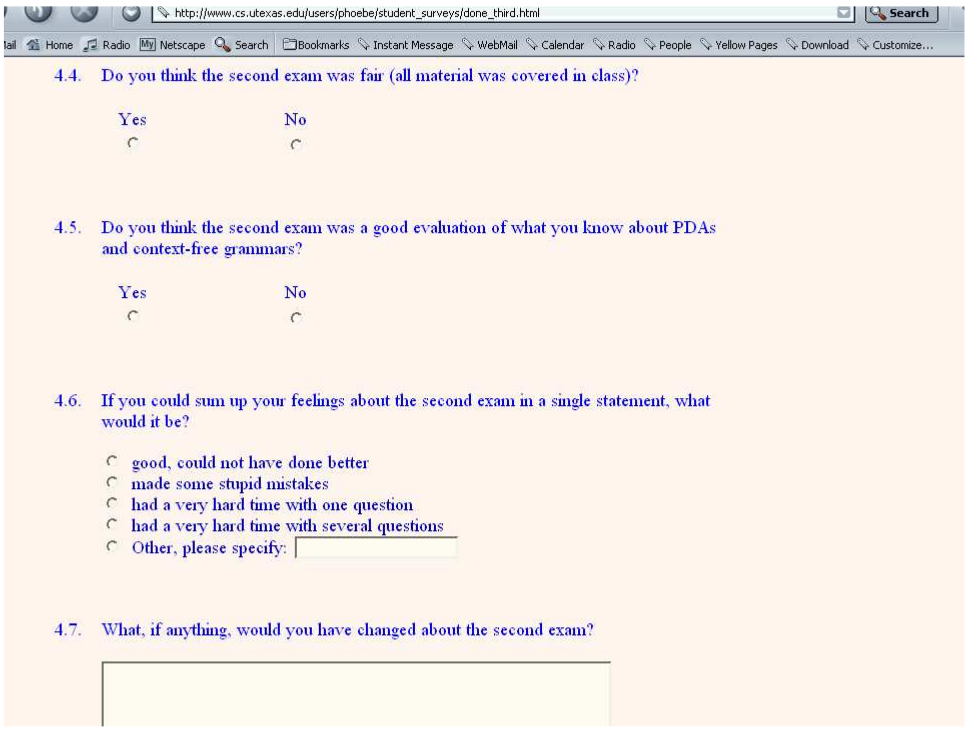
Yes ○ No ○

3.2. Which of these voluntary homeworks have you done?

		all of it	part of it	did not do it
a.	#15	○	○	○
b.	#16	○	○	○
c.	#17	○	○	○
d.	#18	○	○	○
e.	#19	○	○	○
f.	#20	○	○	○
		all of it	part of it	did not do it
g.	#21	○	○	○
h.	#22	○	○	○

3.3. Please explain your reasons for doing or not doing the voluntary homeworks.

Figure G.6: Sample one of online formatting for third survey.


http://www.cs.utexas.edu/users/phoebe/student_surveys/done_third.html
Search
Home Radio Netscape Search Bookmarks Instant Message WebMail Calendar Radio People Yellow Pages Download Customize...

4.4. Do you think the second exam was fair (all material was covered in class)?

Yes	No
○	○

4.5. Do you think the second exam was a good evaluation of what you know about PDAs and context-free grammars?

Yes	No
○	○

4.6. If you could sum up your feelings about the second exam in a single statement, what would it be?

- ○ good, could not have done better
- ○ made some stupid mistakes
- ○ had a very hard time with one question
- ○ had a very hard time with several questions
- ○ Other, please specify:

4.7. What, if anything, would you have changed about the second exam?

Figure G.7: Sample two of online formatting for third survey.

Appendix H

Sample data from classroom observations

Two samples of field notes are presented here. The field notes were originally hand-written and have been transcribed for this Appendix. Samples of how the field notes were used to create Excel spreadsheets are also provided by including screen shots of sample spreadsheets. The symbols “M” and “F” denote male and female students. These symbols do not track individuals, so two adjacent comments using “M” does not necessarily describe the same student.

H.1 Sample field notes

First sample Week 4, September 19, 2002, Section 2: 11-12:30pm in PAI 3.15
40 students at the start of class

Class started with discussion of whether people had started project 1, Lily gave some advice about the project and recapped what the class had gone over on Tuesday.

Homework 1 is being returned by sending the stack around the room during lecture

All but two students that I can see have their course packets out in front of them.

11:05	Review regular expressions slide
11:09	Student Shows Up (SSU)(M)
11:12	Lily asks how to union 2 machines - M student answers correctly. Lily comments "excellent" and says what student said.
11:14	Lily asks question about how to concatenate 2 machines - F student (front of class) answers, Lily nods and says what the student said
11:15	Lily asks question about how to make Kleene star machine - F in front murmurs and M at back (clearer voice) say answer.
11:19	M at front of class asks about epsilon transitions

...

Second sample Week 13, November 19, 2002, Section 1: 9:30-11am in TAY 2.106
34 students at start of class

9:40	Nondeterministic defn's slide
9:43	M asks if a function is still a function if it does more than just compute the function (adder that also multiplies)
9:45	Example of non-deterministic deciding slide

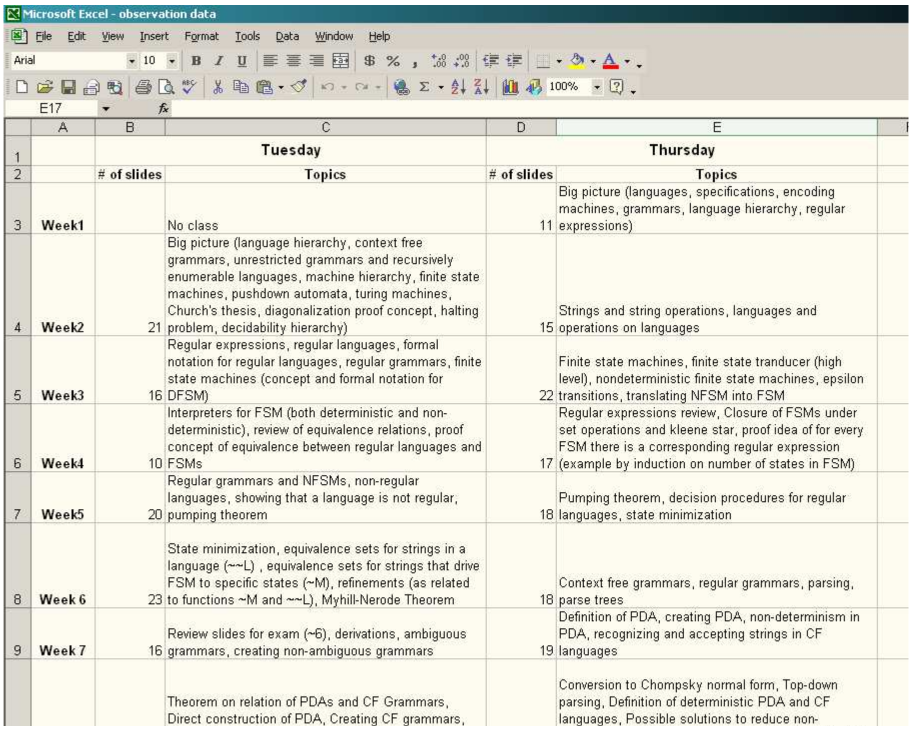
Personal note: What about deterministic FSM, PDA, TM...then non-deterministic FSM, PDA, TM??? Does this make sense?

9:46	SSU(F)
9:47	SSUM
9:48	SSU(M)
9:49	M answers “no” to whether you can “penny” simulate a non-deterministic TM in the same way we did with FSMs
9:50	M answers need a stack to try all computations of a TM computation tree

Personal note: still talking about “the construction non-det→det”...this is very confusing. Can this be made simpler?

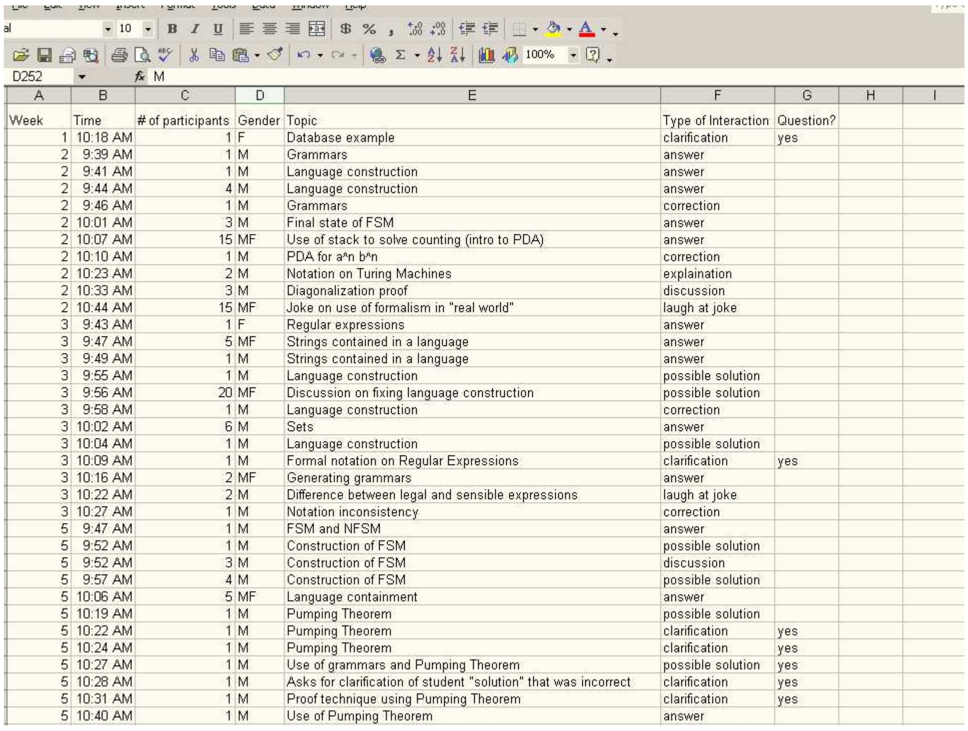
...

H.2 Sample of compiled spreadsheet information


Microsoft Excel - observation data
File Edit View Insert Format Tools Data Window Help
Arial 10
100%
E17

	A	B	C	D	E
1		Tuesday		Thursday	
2		# of slides	Topics	# of slides	Topics
3	**Week1**		No class	11	Big picture (languages, specifications, encoding machines, grammars, language hierarchy, regular expressions)
4	**Week2**	21	Big picture (language hierarchy, context free grammars, unrestricted grammars and recursively enumerable languages, machine hierarchy, finite state machines, pushdown automata, turing machines, Church's thesis, diagonalization proof concept, halting problem, decidability hierarchy)	15	Strings and string operations, languages and operations on languages
5	**Week3**	16	Regular expressions, regular languages, formal notation for regular languages, regular grammars, finite state machines (concept and formal notation for DFSM)	22	Finite state machines, finite state tranducer (high level), nondeterministic finite state machines, epsilon transitions, translating NFSM into FSM
6	**Week4**	10	Interpreters for FSM (both deterministic and non-deterministic), review of equivalence relations, proof concept of equivalence between regular languages and FSMs	17	Regular expressions review, Closure of FSMs under set operations and kleene star, proof idea of for every FSM there is a corresponding regular expression (example by induction on number of states in FSM)
7	**Week5**	20	Regular grammars and NFSMs, non-regular languages, showing that a language is not regular, pumping theorem	18	Pumping theorem, decision procedures for regular languages, state minimization
8	**Week 6**	23	State minimization, equivalence sets for strings in a language (~~L) , equivalence sets for strings that drive FSM to specific states (~M), refinements (as related to functions ~M and ~~L), Myhill-Nerode Theorem	18	Context free grammars, regular grammars, parsing, parse trees
9	**Week 7**	16	Review slides for exam (~6), derivations, ambiguous grammars, creating non-ambiguous grammars	19	Definition of PDA, creating PDA, non-determinism in PDA, recognizing and accepting strings in CF languages
			Theorem on relation of PDAs and CF Grammars, Direct construction of PDA, Creating CF grammars,		Conversion to Chompsky normal form, Top-down parsing, Definition of deterministic PDA and CF languages, Possible solutions to reduce non-

Figure H.1: Spreadsheet tracking slides covered in each lecture of CS 341.



D252 fx M

Week	Time	# of participants	Gender	Topic	Type of Interaction	Question?
1	10:18 AM	1	F	Database example	clarification	yes
2	9:39 AM	1	M	Grammars	answer	
2	9:41 AM	1	M	Language construction	answer	
2	9:44 AM	4	M	Language construction	answer	
2	9:46 AM	1	M	Grammars	correction	
2	10:01 AM	3	M	Final state of FSM	answer	
2	10:07 AM	15	MF	Use of stack to solve counting (intro to PDA)	answer	
2	10:10 AM	1	M	PDA for a^n b^n	correction	
2	10:23 AM	2	M	Notation on Turing Machines	explaination	
2	10:33 AM	3	M	Diagonalization proof	discussion	
2	10:44 AM	15	MF	Joke on use of formalism in "real world"	laugh at joke	
3	9:43 AM	1	F	Regular expressions	answer	
3	9:47 AM	5	MF	Strings contained in a language	answer	
3	9:49 AM	1	M	Strings contained in a language	answer	
3	9:55 AM	1	M	Language construction	possible solution	
3	9:56 AM	20	MF	Discussion on fixing language construction	possible solution	
3	9:58 AM	1	M	Language construction	correction	
3	10:02 AM	6	M	Sets	answer	
3	10:04 AM	1	M	Language construction	possible solution	
3	10:09 AM	1	M	Formal notation on Regular Expressions	clarification	yes
3	10:16 AM	2	MF	Generating grammars	answer	
3	10:22 AM	2	M	Difference between legal and sensible expressions	laugh at joke	
3	10:27 AM	1	M	Notation inconsistency	correction	
5	9:47 AM	1	M	FSM and NFSM	answer	
5	9:52 AM	1	M	Construction of FSM	possible solution	
5	9:52 AM	3	M	Construction of FSM	discussion	
5	9:57 AM	4	M	Construction of FSM	possible solution	
5	10:06 AM	5	MF	Language containment	answer	
5	10:19 AM	1	M	Pumping Theorem	possible solution	
5	10:22 AM	1	M	Pumping Theorem	clarification	yes
5	10:24 AM	1	M	Pumping Theorem	clarification	yes
5	10:27 AM	1	M	Use of grammars and Pumping Theorem	possible solution	yes
5	10:28 AM	1	M	Asks for clarification of student "solution" that was incorrect	clarification	yes
5	10:31 AM	1	M	Proof technique using Pumping Theorem	clarification	yes
5	10:40 AM	1	M	Use of Pumping Theorem	answer	

Figure H.2: Spreadsheet tracking student participation in first section of CS 341.

Appendix I

Post-hoc SIGCSE member survey

In this Appendix we detail the results from our SIGCSE member survey. The Special Interest Group in Computer Science Education (SIGCSE) is a group within the Association for Computing Machinery (ACM) dedicated to educational issues in Computer Science and related academic fields. Members of SIGCSE can join a list service (listserv), an email-based news group, to which current communications can be posted. It was this listserv that was source of participants for our post-hoc survey.

When examining the findings from this research, we lacked supporting evidence to the following questions:

(1) How prevalent is Automata Theory in other CS curricula?
(2) How pervasive is the requirement of Automata Theory for undergraduate degree completion?
(3) Are there any wide-spread student perceptions about Automata Theory being hard?

To answer these questions, we constructed a small email-based survey and posted the survey on the SIGCSE list service. The survey consisted of 5 questions:

(a) Does your institution offer an Automata Theory course?

(b) What is your department and institution?

(c) Approximately how many students take this course each academic year? At what level of their studies?

(d) Is this Automata Theory course a required part of the undergraduate curriculum?

(e) Describe any trends you have observed in student perceptions of the material in the course and their success rate in passing the course.

Respondents answered as many of the questions as they wished and emailed their responses back to us. The survey was posted on the SIGCSE listserv at 8:49pm Thursday, March 6, 2003. We continued to accept responses until Thursday, March 14, 2003.

We received 49 responses, but one was eliminated because there was a previous response from another instructor at the same institution. The email responses were entered into a Microsoft Excel spreadsheet where descriptive statistics could be run on the data. Respondents were located throughout the United States (including Hawaii) and internationally. In the 48 institutions (6% were non-US based), 92% reported offering an Automata Theory (or equivalently named) course and 56% reported that the course was required for graduation in Computer Science. The course was generally considered an upper-division class (71% stated the course consisted of junior and senior students). The average reported class size was 27 students. Trends for Automata Theory were as follows:

- 31% of the respondents said that nearly all students pass the course.
- 14% of the respondents stated that students have trouble passing the course, or the course has a lower pass rate than other equivalent coursework.
- 32% of the respondents stated that students find Automata Theory difficult.
- 38% of the respondents said that their students tend to question the utility of Automata Theory in Computer Science.

- 2% of the respondents stated that their students would probably not take the course if they had the choice.

Because these findings included schools that did not require Automata Theory for graduation, we next examined only those schools (27 of 48) that required students to take Automata Theory. In the 27 institutions (7% were non-US based), the average class size was 29 students, and 81% stated that the course was offered as an upper-division class. Trends for Automata Theory in this subset of institutions were as follows:

- 37% of the respondents said that nearly all students pass the course.
- 25% of the respondents stated that students have trouble passing the course, or the course has a lower pass rate than other equivalent coursework.
- 48% of the respondents stated that students find Automata Theory difficult.
- 48% of the respondents said that their students tend to question the utility of Automata Theory in Computer Science.
- 4% of the respondents stated that their students would probably not take the course if they had the choice.

The data suggests the following answers to our original questions:

How prevalent is Automata Theory in other CS curricula? Prevalent; 91% of the survey respondents reported having such a course at their institutions.

How pervasive is the requirement of Automata Theory for undergraduate degree completion? Relatively pervasive; the majority (56%) of the survey respondents reported that Automata Theory is required for the undergraduate Computer Science degree.

Are there any wide-spread student perceptions about Automata Theory being hard? Yes, respondents reported that students have trouble passing Automata Theory (14%)

or that students find the course difficult (32%). When examining only institutions that require Automata Theory, 25% of the respondents stated that their students have difficulty passing the course, and 48% stated that their students find Automata Theory difficult.

Bibliography

[1] ACM and IEEE-CS. *Computing Curricula 2001*, December 2001. `http://www.computer.org/education/cc2001/`.

[2] "Action Research Learning Project". Internet, last accessed March 5, 2003. Available at `http://alp.polyu.edu.hk/ar/ar_frm.html`.

[3] Aguirre Jr., Adalberto. The mass class as a learning community: Narrative, self-reflection, and understanding. *Radical Pedagogy*, 2001.

[4] Almstrum, Vicki L. *Limitations in the Understanding of Mathematical Logic by Novice Computer Science Students*. PhD thesis, The University of Texas at Austin, 1994.

[5] Anderson, James A. *Discrete Mathematics with Combinatorics*. Prentice Hall, Inc, 2001.

[6] Anyon, Jean. Social class and school knowledge. *Curriculum Inquiry*, 11(1):3–42, 1981.

[7] Atkins, David. "CIS 420/520 Automata Theory". Internet, last accessed March 5, 2003. Available at `http://www.cs.uoregon.edu/~datkins/archive/cis420_s02/`.

[8] Bagert, Donald J. A core course in computer theory: Design and implementation

issues. *ACM SIGCSE Bulletin*, 21(1):161–164, February 1989. Paper presented at the 20th technical symposium on computer science education.

[9] Bagert, Donald J., Daniel Cohen, Gary Ford, Donald K. Friesen, Daniel D. McCracken, and Derick Wood. The increasing role of computer theory in undergraduate curricula. *ACM SIGCSE Bulletin*, 20(1), February 1988. Summary of panel discussion presented at the 19th technical symposium on Computer science education.

[10] Bass, editor. *Geometry: Tools for a changing world*. Prentice Hall, Inc., 1998.

[11] "Bayesian logic - a whatis definition". Internet, last accessed March 5, 2003. Available at `http://whatis.techtarget.com/definition/0,,sid9_gci548993,00.html`.

[12] Bechtel, William. *Philosophy of Science: An overview for cognitive science*. Lawrence Erlbaum Associates, April 1988.

[13] Beecher, Jeff. Note-taking: what do we know about the benefits? ERIC Digest Number 12, January 1988.

[14] Ben-Ari, Mordechai. Constructivism in computer science education. *Computers in Mathematics and Science Teaching*, 20(1):45–73, 2001. `http://stwww.weizmann.ac.il/g-cs/benari/articles/cons.pdf`.

[15] Benzing, Cynthia and Paul Christ. A survey of teaching methods among economics faculty. *Journal of Economic Education*, 28(2):182–188, March 1997.

[16] Berque, Dave, David K. Johnson, and Larry Jovanovic. Teaching theory of computing using pen-based computers and an electronic whiteboard. In *6th Annual Conference on Innovation and Technology in Computer Science Education ITiCSE*. ACM press, June 2001.

[17] Berztiss, A. T. The why and how of discrete structures. *ACM SIGCSE Bulletin*, 7(1):22–25, September 1976. Paper presented at the 6th technical symposium on computer science education.

[18] Biglan, A. The characteristics of subject matter in different academic areas. *Journal of Applied Psychology*, 57:195–203, 1973.

[19] Bilska, Anna O., Kenneth H. Leider, Magdalena Procopiuc, Octavian Procopiuc, Susan H. Rodger, Jason R. Salemme, and Edwin Tsang. A collection of tools for making automata theory and formal languages come alive. *ACM SIGCSE Bulletin*, 29(1):15–19, March 1997. Paper presented at the 28th technical symposium on computer science education.

[20] Bloom, Benjamin S., editor. *Taxonomy of educational objectives: The classification of educational goals*. David McKay Publishing, New York, 1956. `http://www.coun.uvic.ca/learn/program/hndouts/bloom.html`.

[21] Boekaerts, Monique and Pietro Boscolo. Interest in learning, learning to be interested. *Learning and Instruction*, 12(4):375–382, August 2002.

[22] Brawner, Catherine E., Richard M. Felder, Rodney Allen, and Rebecca Brent. 1999-2000 SUCCEED faculty survey on teaching practices and perceptions of institutional attitudes toward teaching. Technical report, SUCCEED Engineering Education Coalition, December 2001. `http://www.succeednow.org/products/99faculty_survey.pdf`.

[23] Brown, editor. *Advanced Mathematics: Precalculus with Discrete Math*. Houghton Mifflin Co., 2nd edition, 1994.

[24] Bruce, Kim. Thoughts on computer science education. *ACM Computing Surveys*, 28A(4), December 1996.

[25] Bryan, Lynn A. Learning to teach elementary science: A case study of teacher beliefs about science teaching and learning. In *Annual meeting of the National Association for Research in Science Teaching*. NARST, April 1998.

[26] Burton, Leone. Undergraduate engineering education: teaching, learning, assessing—a symbiosis. *Engineering Science and Education*, 7(4):158–160, August 1998.

[27] Chalupa, Marilyn, Catherine Chen, and Thomas Charles. An analysis of college students' motivation and learning strategies in computer courses: A cognitive view. *Delta Pi Epsilon*, 43(4):185–199, September 2001.

[28] Chamillard, A. T. and Dolores Karolick. Using learning style data in an introductory computer science course. *ACM SIGCSE Bulletin*, 31(1):291–295, 1999. Paper presented at the 30th technical symposium on Computer science education.

[29] Chau, Y. S. and C. N. Winton. Undergraduate theory of computation: An approach using simulation tools. *SIGCSE Bulletin*, 20(1):78–82, February 1988.

[30] Chickering, Arthur W. and Zelda F. Gamson. Seven principles for good practice in undergraduate education. *AAHE Bulletin*, March 1987. A Publication of the American Association for Higher Education.

[31] Clarke, Edmond M., Jeannette M. Wing, et al. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4), December 1996.

[32] Cochran, Kathryn F. Research matters-to the science teacher. Technical Report 9702, NARST, January 1997. Available at `http://www.educ.sfu.ca/narstsite/research/pck.htm`.

[33] "ComputerUser.com High-Tech Dictionary". Internet, last accessed March 5, 2003. `http://www.computeruser.com/resources/dictionary/`.

[34] Craigen, Dan, Susan Gerhart, and Ted Ralston. Formal methods reality check: Industrial usage. *IEEE Transactions on Software Engineering*, 21(2), February 1995.

[35] Crotty, Michael. *The Foundations of Social Research: Meaning and Perspective in the Research Process*. SAGE Publications, 1998.

[36] "CS Degree Requirements at The University of Texas". Internet, last accessed March 5, 2003.
Available at `http://www.cs.utexas.edu/users/UTCS/undergradoffice/degrees/index.html`.

[37] Davidson, Jane. "Positivism and Context-Dependence". Internet, last modified July 21, 1999. Last accessed March 5, 2003. Available at `http://fac.cgu.edu/˜scrivenm/lectures/philos/CONTEXT2.HTM`.

[38] Denzin, Norman K. and Yvonna S. Lincoln, editors. *Handbook of Qualitative Research*. Sage Publications, 2nd edition, March 2000.

[39] Dodridge, M. Learning outcomes and their assessment in higher education. *Engineering Science and Education Journal*, 8(4):161–168, 1999.

[40] Doolittle, Peter E. "Constructivism and Online Education (Doolittle)". Internet, last accessed March 5, 2003. Paper is part of the 1999 Online Conference on Teaching Online in Higher Education available at `http://edpsychserver.ed.vt.edu/workshops/tohe1999/types.html`.

[41] Drabent, Wlodek. "TDDA89 Formal Languages and Automata Theory". Internet, last accessed March 5, 2003.
Available at `http://www.ida.liu.se/˜TDDA89/`.

[42] Drake, Corey, James P. Spillane, and Kimberly Hufferd-Ackles. Storied identities: teacher learning and subject-matter context. *Journal of Curriculum Studies*, 33(1):1–23, 2001.

[43] Erdle, S. and H.G. Murray. Interfaculty differences in classroom teaching behaviors and their relationship on student instructional ratings. *Research in Higher Education*, 24:115–127, 1986.

[44] Felder, Richard M. Reaching the second tier: Learning and teaching styles in college science education. *Journal of College Science Teaching*, 23(5):286–290, 1993. `http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Papers/Secondtier.htm`.

[45] Felder, Richard M. Any questions? *Chemical Engineering Education*, 28(3):174–175, 1994.

[46] Felder, Richard M. Matters of style. *ASEE Prism*, 6(4):18–23, December 1996. `http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Papers/LS-Prism.htm`.

[47] Felder, Richard M. and Rebecca Brent. Objectively speaking. *Chemical Engineering Education*, 31(3):178–179, 1997. `http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Columns/Objectives.html`.

[48] Felder, Richard M., G.N. Felder, and E.J. Dietz. The effects of personality type on engineering student performance and attitudes. *Journal of Engineering Education*, 91(1):3–17, 2002. `http://www2.ncsu.edu/unity/lockers/users/f/felder/public/Papers/longmbti.pdf`.

[49] Felder, Richard M., Armando Rugarcia, and James E. Stice. The future of engineering education: V. assessing teaching effectiveness and educational scholarship. *Chemical Engineering Education*, 34(3):198–207, 2000.

[50] Feldman, K.A. The superior college teacher from the student's view. *Research in Higher Education*, 5:243–288, 1976.

[51] Garcia-Barbosa, Tamara J. and John R. Mascanzine. Guidelines for college science teaching assistants. ERIC Digest number ED433193, 1998.

[52] Gonzalez, Dax. "UT Homework Service: Doing the Math (and Science)". Internet, last accessed March 13, 2003. Available at `http://www.utexas.edu/pda/pcc/txtell/homework.html`.

[53] Gramond, Eric and Susan H. Rodger. Using JFLAP to interact with theorems in automata theory. *ACM SIGCSE Bulletin*, 31(1):336–340, March 1999. Paper presented at the 30th technical symposium on computer science education.

[54] Hancock, Dawson R. Impact of verbal praise on college students' time spent on homework. *Journal of Educational Research*, 93(6):384, July 2000.

[55] Harrow, Keith. How to show something is not: proofs in formal language and computability theory. *ACM SIGCSE Bulletin*, 10(1):27–30, August 1978. Paper presented at the 9th technical symposium on computer science education.

[56] Hashweh, Maher Z. Effects of science teachers' epistemological beliefs in teaching. *Journal of Research in Science Teaching*, 33(1):47–63, January 1996.

[57] Hazzan, Orit. Reducing abstraction level when learning computability theory concepts. In *7th Annual Conference on Innovation and Technology in Computer Science Education ITiCSE*. ACM press, June 2002.

[58] Headington, Mark R. Introducing fintite automata in the first course. *ACM SIGCSE Bulletin*, 20(1):163–167, February 1988. Paper presented at the 19th technical symposium on computer science education.

[59] Hicks, C. The use of managed learning environments and automated assessment for supporting large-group teaching. *Engineering Science and Education Journal*, 11(5):193–198, October 2002.

[60] Hilfinger, Paul N., Mary Shaw, and William A. Wulf. Introducing "theory" in the second programming course. In *9th technical symposium on computer science education*, pages 55–58, 1978.

[61] Hodges, Andrew. "alan turing: a short biography - 3". Internet, last accessed March 10, 2003. Available at
`http://www.turing.org.uk/bio/part3.html`.

[62] "Homework Service Overview". Internet, last accessed March 5, 2003.
Available at `https://hw.utexas.edu/overview.html`.

[63] Hopcroft, John E., Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing, 2nd edition, November 2000.

[64] Hung, Ted and Susan H. Rodger. Increasing visualization and interaction in the automata theory course. *ACM SIGCSE Bulletin*, 32(1):6–10, March 2000. Paper presented at the 31st technical symposium on computer science education.

[65] "Jean Piaget Society". Internet, last updated February 20, 2003. Last accessed March 5, 2003.
Available at `http://www.piaget.org`.

[66] Kalpakis, Konstantinos. "CMSC 451, Section 101, Automata Theory and Formal

Languages". Internet, last accessed March 5, 2003. Available at `http://www.csee.umbc.edu/~kalpakis/Courses/451-fa99`.

[67] Kinber, Efim and Carl Smith. *Theory of Computing: A gentle introduction*. Prentice Hall, Upper Saddle River, NJ 07450, 2001.

[68] Kolar, Siobhan and Lisa D'Ambrosio. "Vygotsky Resources". Internet, last updated August 11, 2002. Last accessed March 5, 2003.
Available at `http://www.kolar.org/vygotsky`.

[69] Kolb, D.A. *The Modern American College*. Jossy-Bass, 1981.

[70] Kozen, Dexter C. *Automata and Computability*. Springer Verlag, April 1997.

[71] Larson, editor. *Heath Calculus with Analytic Geometry*. McDougal Littell & Co., 5 edition, 1994.

[72] Lawson, Anton E., Souheir Alkhoury, Russell Benford, Brian R. Clark, and Kathleen A. Falconer. What kinds of scientific concepts exist? Concept construction and intellectual development in college biology. *Journal of Research in Science Teaching*, 37(9):996–1018, 2000.

[73] Lederman, Norman G. Teachers' understanding of the nature of science and classroom practice: Factors that facilitate or impede the relationship. *Journal of Research in Science Teaching*, 36(8):916–929, August 1999.

[74] Lewis, Harry R. and Christos Papadimitriou. *Elements of the Theory of Computation*. Prentice Hall, 2nd edition, August 1997.

[75] Lin, Yi-Guang and Wilbert J. McKeachie. College student intrinsic and/or extrinsic motivation and learning. In *107th Annual Conference of the American Psychological Association*, August 1999.

[76] “LOGICAL POSITIVISM”. Internet, last accessed March 5, 2003. Available at `http://www.geocities.com/s011023/toms_files1/essays/positivisim.htm`.

[77] Margolis, Jane and Allan Fisher. *Unlocking the Clubhouse*. MIT Press, 1st edition, December 2001.

[78] Mayer, Richard E. The physchology of how novices learn computer programming. *ACM Computing Surveys*, 13(1), March 1981.

[79] Meduna, Alexander. *Automata and Languages: Theory and Applications*. Springer Verlag, August 2000.

[80] Page, Nicole A. “Key Theorists/Theories in Psychology–ROBERT GAGNE”. Internet, last updated October 17, 2001. Last accessed March 5, 2003. Available at `http://www.psy.pdx.edu/PsiCafe/KeyTheorists/Gagne.htm`.

[81] “Personality test based on Jung–Meyers-Briggs typology”. Internet, last accessed March 11, 2003. Available at `http://www.humanmetrics.com/cgi-win/JTypes1.htm`.

[82] Pohlmann, J.T. A description of effective college teaching in five disciplines as measured by student ratings. *Research in Higher Education*, 4:335–346, 1976.

[83] Pollio, Howard R. Any questions, please? *Teaching and Learning Issues*, (66), 1989.

[84] Pollio, Howard R. The two cultures of pedagogy: Teaching and learning in the natural sciences and the humanities. *Teaching and Learning Issues*, (75), 1996.

[85] Pollio, Howard R. and Hall P. Beck. When the tail wags the dog: perceptions of learning and grade orientation in, and by, contemporary college students and faculty. *Journal of Higher Education*, 71(1):84–102, January 2000.

[86] Polman, Joseph. *Designing Project-Based Science: Connecting learners through guided inquiry*. Ways of Knowing in Science Series. Teachers College Press, January 2000.

[87] Potosky, Denise. A field study of computer efficacy beliefs as an outcome of training: the role of computer playfulness, computer knowledge, and performance during training. *Computers in Human Behavior*, 18:241–255, 2002.

[88] Proulx, Viera and Richard Rasala. The future of computer science education. *ACM Computing Surveys*, 28(4es), December 1996.

[89] Ramakrishna, M.V. A learning by doing model for teaching advanced databases. In *Australian computing education conference*, pages 203–207. ACM Press, 2000.

[90] Raphelson, Alfred C. The use of slides in class: a demonstration of incidental learning. *Teaching of Psychology*, 14(2):103–105, April 1987.

[91] Renkl, A. Worked-out examples: instructional explanations support learning by self-explanations. *Learning and Instruction*, 12(5):529–556, October 2002.

[92] Renz, Peter. "The Legacy of R. L. Moore - FOCUS: The Moore Method – P.Renz". Internet, Last accessed March 19, 2003. Available at `http://www.discovery.utexas.edu/rlm/reference/FOCUS.html`.

[93] "ResearchWare/HyperRESEARCH Home Page". Internet, last accessed March 5, 2003.
Available at `http://www.researchware.com`.

[94] Rich, Elaine. "CS 341 Automata Theory". Internet, last accessed March 5, 2003. Available at
`http://www.cs.utexas.edu/users/ear/cs341/index.html`.

[95] Ritter, Michael E. and Karen A. Lemke. Addressing the ‘Seven principles for good practice in undergraduate education’ with internet-enhanced education. *Journal of Geography in Higher Education*, 24(1):100–109, March 2000.

[96] Rodger, Susan H. “JFLAP 4.0 Beta Version”. Internet, last updated January 16, 2003. Last accessed March 5, 2003.
Available at `http://www.cs.duke.edu/~rodger/tools/jflaptmp`.

[97] Rodger, Susan H. An interactive lecture approach to teaching computer science. *ACM SIGCSE Bulletin*, 27(1):278–282, March 1995. Paper presented at the 26th technical symposium on computer science education.

[98] Sadker, Myra and David Sadker. *Failing at Fairness: How our schools cheat girls*. Touchstone Books, March 1995.

[99] “Scaffolding”. Internet, last accessed March 5, 2003.
Available at `http://www.ncrel.org/sdrs/areas/issues/students/learning/lr1scaf.htm`.

[100] Seymour, Elaine and Nancy Hewitt. *Talking about Leaving: Why undergraduates leave the sciences*. Westview Press, November 2000.

[101] Shackelford, Russell L. and Richard J. LeBlanc. Integrating “depth first” and “breadth first” models of computing curricula. *ACM SIGCSE Bulletin*, 26(1), 1994.

[102] Simmons, Patricia E., Allen Emory, Tim Carter, Teresa Coker, Brian Finnegan, Denise Crockett, Lon Richardson, Robert Yager, John Craven, John Tillotson, Herbert Brunkhorst, Mark Twiest, Kazi Hossain, James Gallagher, Don Duggan-Haas, Joyce Parker, Fernando Cajas, Qasim Alshannag, Sheryl McGlamery, Jerry Krockover, Paul Adams, Barbara Spector, Tom LaPorta, Bob James, Kristin Rearden, and Kay Labuda. Beginning teachers: Beliefs and classroom actions. *Journal of Research in Science Teaching*, 36(8):930–954, August 1999.

[103] "SurveySuite". Internet, last accessed March 13, 2003. Available at `http://intercom.virginia.edu/cgi-bin/cgiwrap/intercom/SurveySuite/ss_index.pl`.

[104] Svinicki, M. D. *College Teaching: From theory to practice*, chapter Practical implications of cognitive theories. Number 45 in New Directions for teaching and learning. Jossey Bass, 1991.

[105] Tall, David. *Advanced Mathematical Thinking*, chapter 1, pages 3–21. Kluwer, 1991.

[106] Tall, David. Understanding the process of advanced mathematical thinking. An invited ICMI lecture at the International Congress of Mathematicians, August 1994.

[107] Toothman, Brian and Russell Shackelford. The effects of partially-individualized assignments on subsequent student performance. In *29th technical symposium on computer science education*, pages 287–291, 1998.

[108] Tremblay, Guy. An undergraduate course in formal methods: "description is our business". *ACM SIGCSE Bulletin*, 30(1):166–170, March 1998. Paper presented at the 29th technical symposium on computer science education.

[109] Tuckman, Bruce W. The relative effectiveness of incentive motivation and perscribed learning strategy in improving college students' course performance. *Journal of Experimental Education*, 64(3):197, March 1996.

[110] Van Blerkom, Malcolm L. Academic perseverance, class attendance, and performance in the college classroom. In *Proceedings of the 104th Annual Meeting of the American Psychological Association*, pages 9–13, August 1996.

[111] Vermunt, Jan D. and Nico Verloop. Congruence and friction between learning and teaching. *Learning and Instruction*, 9(3):257–280, June 1999.

[112] Wagenknecht, Christian and Daniel P. Friedman. Teaching nondeterministic and universal automata using scheme. *Computer Science Education*, 8(3):197–227, 1998.

[113] Weidmann, Phoebe K., La Vergne Lestermeringolo, Jane Ries, Elaine Rich, and Roger Priebe. Declining logical skills in the texas curriculum. Pilot study conducted to investigate indicators of which incoming Computer Science freshmen would benefit from taking a remedial logic course, April 2002.

[114] Weiss, Eileen Mary and Stephen Gary Weiss. New directions in teacher evaluation. ERIC Digest number ED429052, 1998.

[115] "xrefer". Internet, last accessed March 5, 2003.
Available at `http://www.xrefer.com/search.jsp`.

Vita

Phoebe Kay Weidmann was born on October 26, 1971 in Austin, TX to Fredrick William and Sarah Sue Weidmann. Her mother passed away when she was 7 years of age and her father remarried Cheryl Jean. When she was 15 years of age she went to live with her maternal aunt Evelyn McCarty and finished secondary school in Corpus Christi, TX.

Ms. Weidmann immediately went to The University of Texas at Austin where she earned a Bachelor of Science in both Mathematics and Computer Science in May 1994 and a Masters of Science in Computer Science in May 1996. Without pause she applied and was accepted into the PhD program in Computer Sciences where she struggled with an advisor change and a topic in systems that did not motivate her. In the summer of 2001 she made the switch to study Computer Science Education since her first love was teaching. She completed her PhD in Computer Science Education in May 2003.

During her studies in Computer Sciences Phoebe met and married Henry Pierluissi. They are still married and live happily with their four cats.

Permanent email address: phoebe@alumni.utexas.net

Permanent Address: 1510 Larkwood Dr.
Austin, TX 78723
USA

This dissertation was typeset with $\LaTeX\,2_\varepsilon$[1] by the author.

[1]$\LaTeX\,2_\varepsilon$ is an extension of $\LaTeX$. $\LaTeX$ is a collection of macros for $\TeX$. $\TeX$ is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin, and extended by Bert Kay and James A. Bednar.