SIGCSE Committee Report

On the Implementation

of a

Discrete Mathematics Course

Bill Marion and **Doug Baldwin**
Committee Co-Chairs

2007 April

SIGCSE Committee Report

On the Implementation of a Discrete Mathematics Course

Preface

In late 2001, the SIGCSE Board announced a new program, known as the SIGCSE-Committee Initiative. This program is a means for SIGCSE members with like interests or common concerns to form a committee to provide a forum for ongoing, focused discussions on an issue of interest to the membership as a whole. New committees are proposed to the SIGCSE Board for approval. Upon receiving approval, a charge is written and facilitators announced. The committee's responsibility is then to develop a report that addresses the charge. Once the report is completed, the committee's work is done.

In early June of 2003, Bill Marion and Henry Walker found themselves discussing what a one-semester course in discrete mathematics that achieved the ACM/IEEE Computing Curricula 2001 goals for that subject might look like. (Both Bill and Henry served on the CC 2001 Task Force's Pedagogy Focus Group on Supporting Courses, of which the topic of discrete mathematics was a part.) Out of this conversation was born the first SIGCSE Committee Initiative committee, charged to provide a more detailed description of such a course than the description presented in the CC 2001 Report. What follows is the committee's report, which was three years in the making.

Bill Marion and Doug Baldwin
Committee Co-Chairs
2007 April

Committee Members

In addition to the chairs, the SIGCSE Committee on the Implementation of a Discrete Mathematics Course included the following people. We are extremely grateful to all for their assistance:

Dr. Iyad A. Ajwa, David Arnow, Reyyan Ayfer, Valerie Barr, Aaron Bloomfield, Steven Case, Gene Chase, Amitabh Chaudhary, Yi Chen, Mike Clancy, Jim Cohoon, Peter A Cooper, Jose Cordova, Rita D'Arcangelis, Adrienne Decker, Scot Drysdale, Harriet Fell, Jeff Forbes, Howard Francis, Meg Geroch, Peter Gibbons, Eric Gossett, John Grant, Cary Gray, Doug Hale, Judy Hall, John Hamer, David Hastings, David Hemmendinger, Peter Henderson, Linda Herndon, Lew Hitchner, V. Dwight House, John Impagliozzo, Aaron W. Keen, Charles Kelemen, Nancy Kinnersley, David Klappholz, Myungsook Klassen, Nobo Komagata, Joan Krone, Gerald Kruse, Jose E. Labra, Mark LeBlanc, Arnold Lebow, Robert Lechner, Lauren Lilly, Will Lloyd, Dana E. Madison, David E. Maharry, Hugh McGuire, Larry Muller, Kevin O'Gorman, Lynn Olson, Hemant Pendharkar, Gary Raduns, Steve Ratering, Roberta Sabin, Diane Schwartz, Angela Shiflet, Jane Sieberth, S. de Silva, Shai Simonson, Joe Sloan, Leen-Kiat Soh, Sarah Spence, Laszlo A. Szekely, Kathy Taylor, Susan Thompson, Nancy Tinkham, Henry M. Walker, John Werth, Tom Whaley, Pat Woodworth, J. F. Yao, and S. M. Yiu.

SIGCSE Committee Report

On the Implementation of a Discrete Mathematics Course

ABSTRACT

The committee was formed to develop a small number of models for a one-semester discrete mathematics course which meets the needs of computer science majors as articulated in the Joint ACM/IEEE Task Force Report on Computing Curricula 2001-Computer Science Volume (CC2001) [1]. What follows are descriptions of three such models with goals and objectives for each. Examples of syllabi, recommended textbooks, possible homework assignments, hands-on activities and nifty examples are also provided.

BACKGROUND

In the late 1990s a Joint ACM/IEEE Task Force was formed to revise the undergraduate computing curricula. The Task Force decided to tackle first the curriculum in computer science. At that time most undergraduate programs were following a mixture of models articulated in the 1978 and 1991 guidelines. The Task Force Report was published in December 2001. For the first time the study of discrete mathematics (discrete structures) was seen as foundational to the study of computer science and, therefore, listed as one of the core areas all computer science majors should learn.

In the first draft of the Report six topics in discrete mathematics were identified as the knowledge base for discrete structures—(DS1) functions, relations and sets, (DS2) basic logic, (DS3) proof techniques, (DS4) basics of counting, (DS5) graphs and trees, and (DS6) discrete probability. A Pedagogy Focus Group on Supporting Courses was formed and one of its charges was to build a model for incorporating the topics into some sort of coherent course structure. Keeping in mind that this course model was to be developed from the perspective of what computer science majors needed to know, the Group fairly quickly came to the conclusion that the discrete mathematics material should be taught over two semesters with examples and applications from computer science interwoven with the math topics [9]. The idea was that the applications would enhance the students' understanding of the mathematics being introduced, especially if there were connections made to what the students were learning in their first couple of computer science courses. Thus, the two-semester model was the one that was recommended and most fully developed.

The members of the Task Force agreed with this recommendation, but realized that at many colleges and universities it might not be practical to put in place a two-semester sequence which would be taken early in a student's four-year program. So, they included a one-semester model in the Report. Unfortunately, it wasn't as well-developed as the two-semester one. Hence, there was a recognition that, after the Report came out, a more focused effort would have to be devoted to building a coherent one semester model. That is the purpose of this report.

PROCESS

In June of 2003 the SIGCSE Board approved the formation of *The Committee on the Implementation of a Discrete Mathematics Course* with Bill Marion of Valparaiso University and Doug Baldwin of SUNY Geneseo as its co-facilitators. The charge given to the committee was "to work toward providing a few ($\leq$ 6) practical models for a one-semester course that will meet the basic needs of undergraduates in a computer science program. To be helpful, the Committee will work to provide the following for each model course:

* a detailed course outline which would include possible unifying themes for the course, ways in which to organize the course in terms of what material is covered, in what sequence, and possible textbooks

* detailed laboratory exercises to support possible laboratory sessions throughout the semester

* detailed homework assignments

* a listing of possible applications that might be covered during the semester, with the idea that each course offering would cover some, but probably not all, of the applications of any specific topic.

Models might include a version of the course that has College Algebra as a prerequisite, a second version that has Precalculus as prerequisite, and a third version that has one or two semesters of Calculus as prerequisite. In addition, models might take into consideration a course that is housed in a mathematics department as well as one in a computer science department. While the content of such courses might be similar, each model would consider the background and sophistication of the students in determining such factors as course pace and exercises. Together, such models could be expected to cover many of the practical contexts found at a wide range of schools." [8]

Shortly after the Board's approval a listserv was set up through ACM and a call went out to the SIGCSE community for those who were interested in the topic to join the committee. By August the committee membership reached sixty. After a month of fruitful discussion about what approaches to take to carry out the charge, the committee agreed on the following first steps.

One, rather than reinvent the wheel, we should find out what types of one-semester courses are already being offered. This might give us some sense of whether there are courses being taught which come close to CS115 [the one semester model in CC2001 Report]. And, two, distill any existing courses into courses of similar types.

By October an electronic survey of eleven relevant questions requiring relatively brief responses was ready to be distributed to computer science and mathematics faculty who teach courses in discrete mathematics that meet the needs of computer science majors. (See Appendix 1 for the survey questions.) The last two questions asked if the respondents would be willing to share with the committee their syllabi and/or receive a follow-up phone call.

The data we received provided us with a wealth of information to digest. The survey responses were then analyzed by Bill Marion and the following highlights were prepared for presentation at a SIGCSE 2004 Special Session [3].

We received 107 responses of which 83 were from departments at which only a one-semester discrete mathematics course was required of computer science majors. Of the 83, the breakdown in terms of type of institution was: 46 Liberal Arts Colleges, 27 Comprehensive Universities and 10 Research Universities.

In Liberal Arts Colleges the computer science major was slightly more likely to be housed in a joint Math/CS department whereas in the other two types the major was overwhelming found in a separate Computer Science department. Regardless, there was almost an even split on the issue of which department the course was taught in. Three other interesting findings were: (i) about half of the departments required their students to take the discrete mathematics course in their first year of study, (ii) a significant majority had set either college algebra or pre-calculus as a prerequisite rather than Calculus I or above and (iii) 16 had a CS I-type course as a prerequisite.

In addition, over 40 included a copy of their syllabus or provided a web address where their syllabus could be accessed.

At the same time as the survey data was being analyzed, the committee was asked by Doug Baldwin to determine what factors might distinguish one model for a semester course from another model. For example, commonalities and differences could be based on context within an institution, e.g., commonalities or differences in the level of preparation required of students, the department teaching the course, etc. Or, they could be based on the themes that motivate and structure the course, such as courses

motivated by algorithms versus courses motivated by mathematical logic versus courses motivated by cryptology. In the end, the Committee identified three dimensions along which models could be characterized: institutional context, themes, and pedagogies (e.g., lab-based vs lecture-based).

After getting feedback from some of the committee members and those who attended the Special Session we hosted at SIGCSE 2004, we (Bill and Doug) began to review a random selection of the syllabi we had received to see what commonalities and differences there might be among them. One of the things we noticed fairly quickly was that many of the CC2001 topics labeled as DS1 through DS3 were covered by almost all, but not in as much depth as in the recommendations, especially formal logic and proof techniques. There was quite a divergence on what material was covered in DS4, DS5 and DS6, though almost all courses spent some time on counting techniques. In many cases additional material (other than that recommended in CC2001) was introduced. We also discovered that our three dimensional space of models was far more complicated than the reality of the courses actually being taught, which all lay along a single "theme" dimension describing the extent to which courses were built around and motivated by the interests of computer science versus the interests of mathematics.

These observations led us to the view that it was not possible to develop a coherent package and cover all of the core topics. So, we decided to focus on building coherency into the models with the understanding that a number of core topics would have to be treated in other courses. Two models were built, one called a computer science-focused model and the other a math-focused model. With each model we noted which core topics were to be covered, how many hours were to be devoted to each and which would need to be covered elsewhere. In addition, we listed a number of textbooks that could be used if one were to adopt a specific model. In both models the topics in DS1 through DS4 formed the backbone of the course and the amount of time to be spent on logic and proof was strengthened. The major difference between the two was that in the computer science-focused model we interwove a number of computing applications and examples with the mathematics being taught.

In the Fall semester 2004 the committee received the draft we had developed and was asked to suggest modifications and/or identify additional models. Comments we received helped form the outline of the second Special Session, this time at SIGCSE 2005 [5]. At this session the two models were presented. Of the many comments, two led to significant revisions: one, include goals and learning objectives with each of the models and, two, revise the computer-science focused model so that some material in DS5 (graphs and trees) was covered.

The second of the two suggestions was fairly easy to implement, but the first seemed more elusive. Learning objectives were to a large extent fixed by the CC 2001 report—if our charge was to produce a course compatible with those guidelines, we could not adopt learning objectives substantially different from CC 2001's. In the end, we adopted their learning objectives verbatim. We referred the matter of articulating goals to the committee for its guidance, eventually developing the goals listed in the recommendations below. Also, it was now time to ask the committee's help in gathering materials and resources to support the two models, such as homework assignments, lab assignments and additional examples and applications that would enhance the students' understanding of the mathematics being introduced.

At the third and final Special Session (SIGCSE 2006) we presented the revised models and examples of some assignments and exercises [6]. The presentation prompted a discussion about adding a third model, one in which some of the material in DS6 (discrete probability) is included—leaving topics in graphs and trees for another course. Also, a call went out to the audience to provide us with some of their own examples, assignments and applications.

The recommendations are the result of this three-year collaborative effort.

RECOMMENDATIONS

The Committee proposed two recommendations. We summarize them as follows.

Recommendation 1
Three Models for Constructing a One-Semester Course in Discrete Mathematics
(A) Computer Science-Focused Model with Graphs and Trees
(B) Computer Science-Focused Model with Discrete Probability
(C) Mathematics-Focused Model

Recommendation 2
Homework Assignments, Laboratory Exercises, and Applications for each of DS1 through DS6

The contents of the three models for Recommendation 1 appear in Appendix 2A, 2B, and 2C, respectively.. The contents for Recommendation 2 appear in Appendix 3 and 4.

All of these models assume that a course is 40 to 45 hours of classroom instruction. In a semester system, this is typically realized as 14 to 15 weeks of 3 classroom hours each; in a quarter model it is typically realized as about 10 weeks of 4 classroom hours. Our models are designed to account for no more than 39 classroom hours, leaving time for exams, reviews, or small amounts of additional material chosen by the instructor. The models map these hours into semester weeks, but can easily be remapped for quarters.

SUMMARY

Three model courses have been presented that teach (most of) the CC 2001 discrete structures area in a single semester. As "model courses," the intention is not necessarily that anyone will teach these courses exactly as presented (although doing so is possible), but rather that the courses will serve as reference points for people designing or modifying their own courses.

In developing these models, the Committee realized that fitting the entire CC 2001 discrete structures area into a single semester is not feasible. This was both a problem and an opportunity for the Committee. On the one hand, it meant we could not produce models that covered the entire discrete structures area; every model would have to leave some topics to be covered in other courses. People designing one-semester courses from these models must be aware of this condition! However, not covering all topics in a single course also meant that each model could focus on certain aspects of discrete structures, even exceeding CC 2001 recommendations in some areas in order to thoroughly teach students to appreciate and use those elements of discrete mathematics. Differences between models reflect different choices of focus in this regard—primarily whether to focus on mathematical abstractions or on computing applications, and secondarily which topics to develop in the discrete structures course and which to develop in other courses.

The Committee hopes that these models will be helpful to the computer science and mathematics education communities at large. We encourage people to examine these models, use them, and report the results. Both Committee chairs welcome comments and feedback from the community.

Doug Baldwin
baldwin@geneseo.edu

Bill Marion
Bill.MarionJr@valpo.edu

REFERENCES

[1] ACM/IEEE Task Force Report on Computing Curricula 2001-Computer Science, Volume, December 2001. http://www.acm.org/education/curricula.html
[2] Baldwin, D., Henderson, P. Math Thinking Group. http://www.math-in-cs.org

[3] Baldwin D., Marion, B., Walker, H., Status Report on the SIGCSE Committee on the Implementation of a Discrete Mathematics Course. In *SIGCSE Technical Symposium on Computer Science Education*, pages 98-9, March 2004.
[4] Baldwin, D., Scragg, G., *Algorithms and Data Structures: The Science of Computing*, Charles River Media, Inc., 2004.
[5] Marion, B., Status Report on the SIGCSE Committee on the Implementation of a Discrete Mathematics Course. In *SIGCSE Technical Symposium on ComputerScience Education*, pages 194-5, February 2005.
[6] Marion, B., Final Oral Report on the SIGCSE Committee on the Implementation of a Discrete Mathematics Course. In *SIGCSE Technical Symposium on Computer Science Education*, pages 268-9, March 2006.
[7] Rosen, K. *Discrete Mathematics and Its Applications* (5th Ed.), McGraw-Hill Companies, Inc, 2003.
[8] Walker, H., SIGCSE Committee on the Implementation of a Discrete Mathematics Course. http://www.sigcse.org/topics/committees.shtml
[9] Walker, H., Report of the Supporting Courses Pedagogy Focus Group http://www.cs.grinnell.edu/~walker /curriculum/pedagogy-5.2.html

APPENDIX 1

SURVEY

Name of respondent:
Email address of respondent:
Name of college or university:

For each question, mark your response with X (to the right).

1. Type of College/University

Two-year
Liberal Arts
Comprehensive
Research
Other (Explain)

2. Type of Department

Joint Math and CS
Separate within Arts and Sciences
Separate within College of Engineering
Separate within College of Business
Other (Explain)

3. Type of Major

Computer Science
Computer Engineering
Other (Explain)

4. Discrete Mathematics (Discrete Structures) Course for CS majors is

Required
Elective
No Discrete Math Course
Other (Explain)

5. Prerequisite(s) for the Discrete Math course

College Algebra
Precalculus
Calculus I
CS I
Other (Explain)

6. Discrete Math Course for CS majors is taught by

Math Department
CS Department
Other (Explain)

7. Discrete Math Course is required or typically scheduled

Prior to beginning the CS major
During the first year of the CS major
During the second year of the CS major
After the second year of the CS major

8. Discrete Math Course is taught as a

One-semester course
Two-semester course sequence
One-quarter course
Two-quarter course
Other (Explain)

If your answer to question #8 was a "one-semester or one-quarter course", please answer these last three questions; otherwise, you are done.

9. Do you cover material related to

DS1
DS2
DS3
DS4
DS5
DS6
Other (List additional topics)

10. If you are willing to share your syllabus, either email it to me at Bill.Marion@valpo.edu or send me a hard copy at Bill Marion, Dept. of Math and CS, Valparaiso University, Valparaiso, IN 46383 or provide me with your web address

11. Would you be willing to receive a follow-up phone call from a member of the committee if we wish more information about your course? If so, please list your office phone number.

APPENDIX 2A

COMPUTER SCIENCE-FOCUSED MODEL WITH GRAPHS AND TREES

Goals

Introduce a formal system (propositional and predicate logic) on which mathematical reasoning is based.

Develop an understanding of how to read and construct valid mathematical arguments (proofs) and understand mathematical statements (theorems).

Develop the ability to see a problem from a mathematical perspective.

Introduce various problem-solving strategies, especially thinking algorithmically (both iterative and recursive).

Introduce important discrete data structures such as sets, relations, discrete functions, graphs and trees.

Motivate the need for mathematical structures and techniques by introducing computing applications.

Learning Objectives

(From the ACM/IEEE Joint Task Force on Computing Curricula, "Computing Curricula 2001: Computer Science," Dec. 2001, available at http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education /cc2001/cc2001.pdf)

DS1. Functions, relations, and sets

1 Explain with examples the basic terminology of functions, relations, and sets.
2 Perform the operations associated with sets, functions, and relations.
3 Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
4 Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.

DS2. Basic logic

1 Apply formal methods of symbolic propositional and predicate logic.
2 Describe how formal tools of symbolic logic are used to model algorithms and real-life situations.
3 Use formal logic proofs and logical reasoning to solve problems such as puzzles.
4 Describe the importance and limitations of predicate logic.

DS3. Proof techniques

1 Outline the basic structure of and give examples of each proof technique described in this unit.
2 Discuss which type of proof is best for a given problem.
3 Relate the ideas of mathematical induction to recursion and recursively defined structures.

4 Identify the differences among weak, strong and structural induction and give examples of the appropriate use of each.

DS4. Basics of counting

1 Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
2 State the definition of the Master theorem.
3 Solve a variety of basic recurrence equations.
4 Analyze a problem to create relevant recurrence equations or to identify important counting questions.

DS5. Graphs and trees

1 Illustrate by example the basic terminology of graph theory, and some of the properties and special cases of each.
2 Demonstrate different traversal methods for trees and graphs.
3 Model problems in computer science using graphs and trees.
4 Relate graphs and trees to data structures, algorithms, and counting.

Course Outline

DS2. Basic Logic (9 hours)
Week 1 – propositional logic, connectives, truth tables, tautology, contradiction, logical equivalences, modus ponens/modus tolens.
Week 2 – predicate logic, quantifiers, valid arguments.
Week 3 – possible applications: logic programming, proof by resolution, digital logic circuits.

DS3. Proof Techniques (12 hours)
Week 4 – nature of proof, direct and indirect proofs, proof by contradiction, counterexamples, existence and constructive proofs.
Week 5 – possible applications: number-theoretic proofs, RSA algorithm, Halting problem.
Week 6 – math induction (weak, strong, structural), well-ordering principle.
Week 7 – possible applications: standard searching and sorting algorithms, algorithm correctness arguments, recursive definitions, iterative and recursive algorithms.

DS1. Functions, Relations, Sets (6 hours)
Week 8 – sets, set theoretic proofs, functions; applications to asymptotic behavior, analysis of algorithms.
Week 9 – cardinality, relations; applications to computability, relational databases.

DS4. Basics of Counting (6 hours)
Week 10 – counting arguments (addition and multiplication principles), permutations and combinations, combinatorial arguments, pigeonhole principle; applications to analysis of algorithms.
Week 11 – recurrence relations, homogeneous and non-homogeneous linear recurrence relations with constant coefficients, Master Theorem; applications to analysis of algorithms.

DS5. Graphs and Trees (6 hours)
Week 12 – Directed and undirected graphs, traversal algorithms; applications of previous material on relations and counting to graphs (e.g., transitive closure) and analysis of graph algorithms.

Week 13 – Trees, spanning trees including greedy algorithms; applications of previous material on proofs and counting to analysis of trees and tree algorithms (e.g., structural induction involving trees, recurrence relations arising from trees).

Some Possible Texts

Epp, *Discrete Mathematics with Applications*

Gersting, *Mathematical Structures for Computer Science*

Hein, *Discrete Mathematics*

Johnsonbaugh, *Discrete Mathematics*

Maurer and Ralston, *Discrete Algorithmic Mathematics*

Rosen: *Discrete Mathematics and Its Applications*

Notes

This model exceeds CC2001's recommendations for time allotted to DS4 and DS5 to allow time to apply previous material to proofs and analyses related to graphs and trees. An instructor could, in principle, scale this back to make room for other material, but the Committee believes that the on-going application of proofs, logic, and similar material provides important reinforcement in a context that makes the subjects' relevance to computer science clearer.

This model leaves DS6, Discrete Probability, to be covered in a separate probability/statistics course.

APPENDIX 2B

COMPUTER SCIENCE-FOCUSED MODEL WITH DISCRETE PROBABILITY

Goals

Introduce a formal system (propositional and predicate logic) on which mathematical reasoning is based.

Develop an understanding of how to read and construct valid mathematical arguments (proofs) and understand mathematical statements (theorems).

Develop the ability to see a problem from a mathematical perspective.

Introduce various problem-solving strategies, especially thinking algorithmically (both iterative and recursive).

Introduce important discrete data structures such as sets, relations, and discrete functions.

Introduce discrete probability and expectation.

Motivate the need for mathematical structures and techniques by introducing computing applications.

Learning Objectives

(From the ACM/IEEE Joint Task Force on Computing Curricula, "Computing Curricula 2001: Computer Science," Dec. 2001, available at http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education /cc2001/cc2001.pdf)

DS1. Functions, relations, and sets

1 Explain with examples the basic terminology of functions, relations, and sets.
2 Perform the operations associated with sets, functions, and relations.
3 Relate practical examples to the appropriate set, function, or relation model, and interpret the associated operations and terminology in context.
4 Demonstrate basic counting principles, including uses of diagonalization and the pigeonhole principle.

DS2. Basic logic

1 Apply formal methods of symbolic propositional and predicate logic.
2 Describe how formal tools of symbolic logic are used to model algorithms and real-life situations.
3 Use formal logic proofs and logical reasoning to solve problems such as puzzles.
4 Describe the importance and limitations of predicate logic.

DS3. Proof techniques

1 Outline the basic structure of and give examples of each proof technique described in this unit.
2 Discuss which type of proof is best for a given problem.

3 Relate the ideas of mathematical induction to recursion and recursively defined structures.
4 Identify the differences among weak, strong and structural induction and give examples of the appropriate use of each.

DS4. Basics of counting

1 Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
2 State the definition of the Master theorem.
3 Solve a variety of basic recurrence equations.
4 Analyze a problem to create relevant recurrence equations or to identify important counting questions.

DS6. Discrete Probability

1 Calculate the probabilities of events and expectations of random variables for elementary problems such as games of chance.
2 Differentiate between dependent and independent events.
3 Apply the binomial theorem to independent events, and Bayes theorem to dependent events.
4 Apply the tools of probability to solve problems such as the Monte Carlo method, the average case analysis of algorithms, and hashing.

Course Outline

DS2. Basic Logic (9 hours)
Week 1 - propositional logic, connectives, truth tables, tautology, contradiction, logical equivalences, modus ponens/modus tolens
Week 2 – predicate logic, quantifiers, valid arguments
Week 3 – possible applications: logic programming, proof by resolution, digital logic circuits

DS3. Proof Techniques (12 hours)
Week 4 – nature of proof, direct and indirect proofs, proof by contradiction, counterexamples, existence and constructive proofs.
Week 5 – possible applications: number-theoretic proofs, RSA algorithm, Halting problem.
Week 6 – math induction (weak, strong, structural), well-ordering principle.
Week 7 – possible applications: standard searching and sorting algorithms, algorithm correctness arguments, recursive definitions, iterative and recursive algorithms.

DS1. Functions, Relations, Sets (6 hours)
Week 8 – sets, set theoretic proofs, functions; applications to asymptotic behavior, analysis of algorithms.
Week 9 – cardinality, relations; applications to computability, relational databases.

DS4. Basics of Counting (6 hours)
Week 10 – counting arguments (addition and multiplication principles), permutations and combinations, combinatorial arguments, pigeonhole principle; applications to analysis of algorithms.
Week 11 – recurrence relations, homogeneous and non-homogeneous linear recurrence relations with constant coefficients, Master Theorem; applications to analysis of algorithms.

DS6. Discrete Probability (6 hours)

Week 12 – Basic concepts of probability: probability space, independent and dependent events, random variables and expected values, the binomial theorem and Bayes theorem.

Week 13 – Possible applications: average execution time of algorithms, hashing, the Monte Carlo method or other randomized algorithms.

Some Possible texts

Epp, *Discrete Mathematics with Applications*

Gersting, *Mathematical Structures for Computer Science*

Hein, *Discrete Mathematics*

Johnsonbaugh, *Discrete Mathematics*

Maurer and Ralston, *Discrete Algorithmic Mathematics*

Rosen, *Discrete Mathematics and Its Applications*

Notes

This model exceeds CC2001 recommendations for time allotted to DS4 to allow time to apply previous material to reasoning about counting. An instructor could, in principle, scale this back to make room for other material, but the Committee believes that on-going application of proofs, logic, and similar material provides important reinforcement that will improve learning of both the subjects being applied and the ones to which they are applied.

This model leaves DS5, Graphs and Trees, to be covered in a separate algorithms/data structures course.

APPENDIX 2C

MATHEMATICS-FOCUSED MODEL

Goals

Introduce a formal system (propositional and predicate logic) on which mathematical reasoning is based.

Develop an understanding of how to read and construct valid mathematical arguments (proofs) and understand mathematical statements (theorems).

Develop the ability to see a problem from a mathematical perspective, and to use mathematical problem-solving strategies and tools to solve it.

Introduce important discrete data structures such as sets, relations and discrete functions.

Learning Objectives

(From the ACM/IEEE Joint Task Force on Computing Curricula, "Computing Curricula 2001: Computer Science," Dec. 2001, available at http://www.computer.org/portal/cms_docs_ieeecs/ieeecs/education /cc2001/cc2001.pdf)

DS1. Functions, relations, and sets

1 Explain with examples the basic terminology of functions, relations, and sets.
2 Perform the operations associated with sets, functions, and relations.
3 Use appropriate set, function, or relation models to analyze practical examples, and interpret the associated operations and terminology in context.
4 Apply basic counting principles, including uses of diagonalization and the pigeonhole principle.

DS2. Basic logic

1 Apply formal methods of symbolic propositional and predicate logic.
2 Describe how formal tools of symbolic logic are used to model algorithms and real-life situations.
3 Use formal logic proofs and logical reasoning to solve problems such as puzzles.
4 Describe the importance and limitations of predicate logic.

DS3. Proof techniques

1 Outline the basic structure of and give examples of each proof technique described in this unit.
2 Discuss which type of proof is best for a given problem.
3 Relate the ideas of mathematical induction to recursion and recursively defined structures.
4 Identify the differences among, weak, strong and structural induction and give examples of the appropriate use of each.

DS4. Basics of counting

1 Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
2 State the definition of the Master theorem.
3 Solve a variety of basic recurrence equations.
4 Use recurrence equations and/or counting techniques to analyze practical problems.

Course Outline

DS2. Basic Logic (9 hours)
Week 1 – propositional logic, connectives, truth tables.
Week 2 – normal forms, validity, modus ponens/modus tolens.
Week 3 – predicate logic, quantifiers, limitations.

DS3. Proof Techniques (12 hours)
Week 4 – transition from logic, basic notions, structure, direct proof.
Week 5 – counterexample, contraposition, contradiction.
Week 6 – induction, including strong induction.
Week 7 – (advanced/foundational induction) recursive definitions, well orderings.

DS1. Functions, Relations, Sets (9 hours)
Week 8 – sets, cardinality, the pigeonhole principle.
Week 9 – functions.
Week 10 – relations.

DS4. Basics of Counting (9 hours)
Week 11 – counting arguments, permutations & combinations.
Weeks 12-13 – recurrence relations.

Some Possible Texts

Epp, *Discrete Mathematics with Applications*

Johnsonbaugh, *Discrete Mathematics*

Notes

This model exceeds CC2001's recommendations for time allotted to DS1 and DS4 to allow students to really learn the material. An instructor could, in principle, scale these back to make room for other material, but doing so compromises the goal of producing a solid, 1-semester, math-oriented, discrete structures course that fits within a CC2001-compliant curriculum.

This model leaves DS5, Graphs and Trees to be covered in CS2 and/or Algorithms. While some of this material could be brought into this course by spending less time on DS1 and/or 4, it seems better for the math-focused model course to remain strong on fundamental mathematical concepts at the expense of requiring other courses to deal with topics that have a clearer connection to computer science. The computer science-focused models, in contrast, make this trade-off the other way, bringing DS5 or DS6 into the discrete structures course as a way to emphasize applications of the mathematics to data structures and algorithms.

This model also leaves DS6, Discrete Probability, to be covered in a separate probability/statistics course.

APPENDIX 3

Homework Assignments (Online)

The Committee has identified a set of exercises (over 140) from the 5th edition of Kenneth Rosen's text *Discrete Mathematics and Its Applications* [7]. These problems were chosen because they can be characterized as moderate to hard. Some of them are challenging enough to be assigned as group projects or take-home questions. Exercises similar to these can be found in the textbooks cited in the three models above. (Rosen's text was chosen because it is the one most familiar to one of the co-facilitators, Bill Marion, who has used various editions over time.)

Pending copyright clearance, the Committee's collection of homework problems will be posted on-line, at http://www.sigcse.org. Note that page numbers in the posted exercises refer to the first page in each set of exercises.

APPENDIX 4

Laboratory Exercises and Applications (Online)

The Committee's collection of examples that can be used as laboratory exercises for students to work on and/or applications designed to enhance the students' understanding of the mathematics topics they are learning can be found on-line, at http://www.sigcse.org/.

Many other examples can be found at the Math-Thinking website [2], http://www.math-in-cs.org, including a collection of Discrete Math Applets gathered together by Susanna Epp as a resource for use in discrete mathematics courses. (See the link under "Teaching Resources.")

Additional lab exercises developed as supplementary material for the text, *Algorithms and Data Structures: The Science of Computing*, by Baldwin and Scragg [4] can be found at http://www.charlesriver.com /algorithms. See, for example, the lab entitled "Recurrence Relations and Execution Time."