Mathematical and Statistical Foundations (MSF)

Preamble

What is meant by mathematical foundations
A strong mathematical foundation remains a bedrock of computer science education, and mathematical and formal reasoning continue to play a significant role in computer science, in both theoretical and applied areas: developing algorithms, designing systems, modeling real-world phenomena, working with data. This Mathematical Foundations Knowledge Area – the successor to the ACM CS 2013 curriculum's "Discrete Structures" – seeks to identify the mathematical (inclusive of statistics) material that undergirds modern computer science. The bulk of the core mathematical topics of ACM CS 2013 remain; the change of name corresponds to a realization both that the broader name better describes some of the existing topics from 2013 and that some growing areas of computer science, such as artificial intelligence, machine learning, data science, and quantum computing, have continuous mathematics as their foundations too.

Methodology
The subcommittee's recommendations are informed by receiving input from the mathematical requirements in other Knowledge Areas (KAs), from the CS theory community, from various reports (example: Park City report on data science) and, critically, two surveys, one to faculty and one to industry practitioners. The first survey was issued to computer science faculty (with nearly 600 faculty responding) across a variety of institutional types and in various countries to obtain a snapshot of current practices in mathematical foundations and to solicit opinion on the importance of particular topics beyond the traditional discrete mathematics. The second survey was sent to industry employees (approximately 680 respondents) requesting their views on curricular topics and components.

Changes since CS 2013: While the traditional discrete math course remains a mainstay of computer science programs, what has changed is rising faculty concern about students' mathematical preparation and attitude coming into computer science. The most striking change from 2013, however, arises from considering the mathematical needs of the rapidly growing areas of artificial intelligence, machine learning, robotics, data science, and quantum computing. When survey's respondents, which included (self-identified) experts in these areas, rated the importance of a number of mathematical topics both for employment and graduate school, the top five broad content areas were: precalculus, calculus I, linear algebra, probability, and statistics.

Acknowledging some tensions
Faculty and students alike have strong opinions about how much and what math in CS. Generally, faculty, who themselves have strong theoretical training, are typically concerned about poor student preparation and motivation to learn math, while students complain about not seeing applications and wonder what any of the math has to do with the software jobs they

seek. Even amongst faculty, there is recurring debate on whether calculus should be required of computer science students, accompanied by legitimate concern about the impact of calculus failure rates on computer science students. Yet, at the same time, the discipline has itself undergone a significant mathematical change: machine learning, robotics, data science, and quantum computing all demand a different kind of math than what's typically covered in a standard discrete structures course. The combination of changing mathematical demands and inadequate student preparation or motivation, in an environment of enrollment-driven strain on resources, has become a key challenge for CS departments.

Summary of recommendations

- ***Standardize the prerequisites to discrete math***. The faculty survey shows that institutional variation in discrete-math prerequisites distributes nearly evenly across algebra, precalculus and calculus, suggesting differing approaches to the mathematical maturity sought. Requiring precalculus appears to be a reasonable compromise, so that students come in with some degree of comfort with symbolic math and functions.
- ***Include applications in math courses***. Studies show that students are motivated when they see applications. We recommend including minor programming assignments or demonstrations of applications to increase student motivation. While computer science departments may not be able to insert such applications into courses offered by other departments, it is possible to include applications of math in the computer science courses that are co-scheduled with mathematical requirements, and to engage with textbook publishers to provide such material.
- ***Apply available resources to enable student success***. The subcommittee recommends that institutions adopt remedial options to ensure sufficient preparation without lowering standards in discrete mathematics. Theory courses can be moved further back in the curriculum to accommodate freshmen-year remediation, for example. And, where possible, institutions can avail of online tutoring systems (such as ALEKS) alongside regular coursework.
- ***Expand core mathematical requirements to meet the rising demand in new growth areas of computer science***. What is clear, looking forward to the next decade, is that exciting high-growth areas of computer science require a strong background in linear algebra, probability and statistics (preferably calculus-based). Accordingly, we recommend including as much of this material into the standard curriculum as possible,
- ***Send a clear message to students about mathematics while accommodating their individual circumstances***. Faculty and institutions are often under pressure to help every student succeed, many of whom struggle with math. While pathways, including computer science-adjacent degrees or tracks, can be created to steer students past math requirements towards software-focused careers, faculty should be equally direct in explaining the importance of sufficient mathematical preparation for graduate school and for the very topical areas that excite students.
- ***Adapt to institutional mission and student context***. Faculty often advise our students with a version of "the more math you take, the better" and, for plenty of students, particularly those bound for graduate school, that advice is sound; for others, it is unhelpful or even off-putting. Yet, institutions and students differ, with positive consequences for society. Some of these differences arise from the diversity of systems

of higher education, within and across countries; some arise from varying goals for particular programs, for example adopting a more pre-professional curricular outlook or adopting a more foundational one. Liberal-arts colleges, for example, are severely constrained in how many technical courses they can require; yet, their student success validates their unique approach. Accordingly, we recommend that institutions creatively adapt these recommendations to their local context and strengths.

Core Hours

Knowledge Unit	CS Core	KA Core
Sets, Relations, and Functions	4	
Basic Logic	9	
Proof Techniques	10	1
Basics of Counting	5	
Graphs and Trees		1
Discrete Probability	6	2
Fundamentals of Calculus		TBD
Linear Algebra		TBD
Statistics		TBD
Total	**34**	**4**

Knowledge Units

MSF/PreCalculus

[Considered pre-requisites, not part of CS core]

Topics:

- Algebra: adding fractions, rules of exponents, solving linear or quadratic equations with one or two variables
- Functions: function notation, drawing and interpreting graphs of functions
- Exponentials and logarithms: a general familiarity with the functions and their graphs
- Geometry: distances between points, areas of common shapes
- Trigonometry: familiarity with basic trigonometric functions and the unit circle

MSF/Sets, Relations, and Functions

[4 CS Core hours]

Topics:

- Sets

    - Venn diagrams
    - Union, intersection, complement
    - Cartesian product
    - Power sets
    - Cardinality of finite sets
- Relations
    - Reflexivity, symmetry, transitivity
    - Equivalence relations, partial orders
- Functions
    - Surjections, injections, bijections
    - Inverses
    - Composition

Learning Outcomes:

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.

MSF/Basic Logic

[9 CS Core hours]

Topics:

- Propositional logic (cross-reference: Propositional logic is also reviewed in AI/Knowledge Based Reasoning)
- Logical connectives
- Truth tables
- Normal forms (conjunctive and disjunctive)
- Validity of well-formed formula
- Propositional inference rules (concepts of modus ponens and modus tollens)
- Predicate logic
    - Universal and existential quantification
- Limitations of propositional and predicate logic (e.g., expressiveness issues)

Learning Outcomes:

1. Convert logical statements from informal language to propositional and predicate logic expressions.
2. Apply formal methods of symbolic propositional and predicate logic, such as calculating validity of formulae and computing normal forms.
3. Use the rules of inference to construct proofs in propositional and predicate logic.
4. Describe how symbolic logic can be used to model real-life situations or applications, including those arising in computing contexts such as software analysis (e.g., program correctness), database queries, and algorithms.

5. Apply formal logic proofs and/or informal, but rigorous, logical reasoning to real problems, such as predicting the behavior of software or solving problems such as puzzles.
6. Describe the strengths and limitations of propositional and predicate logic.

MSF/Proof Techniques

[10 CS Core hours, 1 KA Core hour]

Topics:

[CS Core]

- Notions of implication, equivalence, converse, inverse, contrapositive, negation, and contradiction
- The structure of mathematical proofs
- Direct proofs
- Disproving by counterexample
- Proof by contradiction
- Induction over natural numbers
- Structural induction
- Weak and strong induction (i.e., First and Second Principle of Induction)
- Recursive mathematical definitions

[KA Core]

- Well orderings

Learning Outcomes:

[CS Core]

1. Identify the proof technique used in a given proof.
2. Outline the basic structure of each proof technique (direct proof, proof by contradiction, and induction) described in this unit.
3. Apply each of the proof techniques (direct proof, proof by contradiction, and induction) correctly in the construction of a sound argument.
4. Determine which type of proof is best for a given problem.
5. Explain the parallels between ideas of mathematical and/or structural induction to recursion and recursively defined structures.
6. Explain the relationship between weak and strong induction and give examples of the appropriate use of each.

[KA Core]

7. State the well-ordering principle and its relationship to mathematical induction.

MSF/Basics of Counting

[5 CS Core hours]

Topics:

- Counting arguments
    - Set cardinality and counting

    - Sum and product rule
    - Inclusion-exclusion principle
    - Arithmetic and geometric progressions
- The pigeonhole principle
- Permutations and combinations
    - Basic definitions
    - Pascal's identity
    - The binomial theorem
- Solving recurrence relations (cross-reference: AL/Basic Analysis)
    - An example of a simple recurrence relation, such as Fibonacci numbers
    - Other examples, showing a variety of solutions
- Basic modular arithmetic

Learning Outcomes:

1. Apply counting arguments, including sum and product rules, inclusion-exclusion principle and arithmetic/geometric progressions.
2. Apply the pigeonhole principle in the context of a formal proof.
3. Compute permutations and combinations of a set, and interpret the meaning in the context of the particular application.
4. Map real-world applications to appropriate counting formalisms, such as determining the number of ways to arrange people around a table, subject to constraints on the seating arrangement, or the number of ways to determine certain hands in cards (e.g., a full house).
5. Solve a variety of basic recurrence relations.
6. Analyze a problem to determine underlying recurrence relations.
7. Perform computations involving modular arithmetic.

MSF/Discrete Probability

[6 CS Core hours, 2 KA Core hour]

Topics:

[CS Core]

- Finite probability space, events
- Axioms of probability and probability measures
- Conditional probability, Bayes' theorem
- Independence
- Integer random variables (Bernoulli, binomial)
- Expectation, including Linearity of Expectation
-

[KA Core]

- Variance
- Conditional Independence

Learning Outcomes:

[CS Core]

1. Calculate probabilities of events and expectations of random variables for elementary problems such as games of chance.
2. Differentiate between dependent and independent events.
3. Identify a case of the binomial distribution and compute a probability using that distribution.
4. Apply Bayes theorem to determine conditional probabilities in a problem.
5. Apply the tools of probability to solve problems such as the average case analysis of algorithms or analyzing hashing.

[KA Core]

6. Compute the variance for a given probability distribution.
7. Explain how events that are independent can be conditionally dependent (and vice-versa). Identify real-world examples of such cases.

MSF/Graphs and Trees

(cross-reference: AL/Fundamental Data Structures and Algorithms, especially with relation to graph traversal strategies)

Topics:

- Trees
  - Properties
  - Traversal strategies
- Undirected graphs
- Directed graphs
- Weighted graphs
- Spanning trees/forests
- Graph isomorphism

Learning Outcomes:

[CS Core]

1. Illustrate by example the basic terminology of graph theory, and some of the properties and special cases of each type of graph/tree.
2. Demonstrate different traversal methods for trees and graphs, including pre, post, and in-order traversal of trees.
3. Model *a variety of* real-world problems in computer science using appropriate forms of graphs and trees, such as representing a network topology or the organization of a hierarchical file system.
4. Show how concepts from graphs and trees appear in data structures, algorithms, proof techniques (structural induction), and counting.

[KA Core]

5. Explain how to construct a spanning tree of a graph.
6. Determine if two graphs are isomorphic.

MSF/Calculus
[KA Core]
Topics:

- Sequences, series, limits
- Single-variable derivatives: definition, computation rules (chain rule etc), derivatives of important functions, applications
- Single-variable integration: definition, computation rules, integrals of important functions, fundamental theorem of calculus, definite vs indefinite, applications (including in probability)
- Parametric formulation, polar representation
- Taylor series
- Multivariate calculus: partial derivatives, gradient, chain-rule, vector valued functions, applications to optimization, convexity, global vs local minima
- ODEs: definition, Euler method, applications to simulation

MSF/Linear Algebra
[KA Core]
Topics:

- Matrices, matrix-vector equation, geometric interpretation, geometric transformations with matrices
- Solving equations, row-reduction
- Linear independence, span, basis
- Orthogonality, projection, least-squares, orthogonal bases
- Linear combinations of polynomials, Bezier curves
- Eigenvectors and eigenvalues
- Applications to computer science: PCA, SVD, page-rank, graphics

MSF/Statistics
[KA Core]
Topics:

- Basic definitions and concepts: populations, samples, measures of central tendency, variance
- Univariate data: point estimation, confidence intervals
- Multivariate data: estimation, correlation, regression
- Data transformation: dimension reduction, smoothing
- Statistical models and algorithms

Professional Dispositions

- **Meticulous:** Students must pay attention to detail when applying math to problems
- **Persistent:** Students need to be persistent to both learn and apply math.

Shared Concepts and Crosscutting Themes

Shared concepts:

- MSF/Graphs and Trees is shared with AL/Fundamental Data Structures and Algorithms

Course Packaging Suggestions

Every department faces constraints in delivering content which precludes merely requiring a long list of courses covering every single desired topic. These constraints include content-area ownership, faculty size, student preparation, and limits on the number of departmental courses a curriculum can require. We list below some options for mathematical foundations, combinations of which might best fit any particular institution:

- ***Traditional course offerings***. With this approach, a computer science department requires students to take math-department courses in the six broad mathematical areas listed above, for a total of 8 courses including Precalculus.
- ***A "Continuous Structures" analog of Discrete Structures***. Many computer science departments now offer courses that prepare students mathematically for AI and machine learning. Such courses can combine just enough calculus, optimization, linear algebra and probability; yet others may split linear algebra into its own course. These courses have the advantage of motivating students with computing applications, and including programming as pedagogy for mathematical concepts.
- ***Integration into application courses***. An application course such as machine learning can be spread across two courses, with the course sequence including the needed mathematical preparation taught just-in-time. This may have the advantage of mitigating turf issues and helping students see applications immediately after encountering math.
- ***Specific course adaptations***. For nearly a century, physics and engineering needs have driven the structure of calculus, linear algebra, and probability. Computer science departments can collaborate with their colleagues in math departments to restructure math-offered sections in these areas that are driven by computer science applications. For example, calculus could be reorganized along the lines described above, with all the computing needs fitted into two calculus courses, leaving later calculus for engineering and physics students.

Committee

Chair: Rahul Simha, George Washington University, Washington D.C., USA

Members:

- Richard L. Blumenthal, Regis University, Denver, CO, USA
- Marc Deisenroth, University College, London, UK
- Michael Goldweber, Xavier University, Cincinnati, OH, USA
- Cynthia Bailey Lee, Stanford University, Palo Alto, CA, USA
- David Liben-Nowell, Carleton College, Northfield, MN, USA

- Jodi Tims, Northeastern University, Boston, MA, USA