EECS 1028 3.00 Discrete Mathematics for Engineers
(Cross-listed with MATH1028 3.00)

Introduction to abstraction; use and development of precise formulations of mathematical ideas, in particular as they apply to engineering; introduction to propositional logic and application to switching circuits; sets, relations and functions; predicate logic and proof techniques; induction with applications to program correctness; basic counting techniques; graphs and trees with applications; automata and applications. Three-hour long weekly lectures and two hours of mandatory tutorials per week.

The detailed list of topics includes

1. Propositional logic, truth tables. **Applications**:
    - Building various switching circuits using OR, AND, NAND, etc., gates (cf. for example, from **Rosen** 9.3).
2. Sets (union, intersection, Cartesian product, power sets, etc.), cardinality of sets (finite and infinite), strings.
3. Functions and relations (domain, range, 1-1, into, onto, 1-1 correspondence, function composition, closures of relations, etc.) equivalence relations.
4. Predicate logic, properties of quantifiers (e.g., understanding the connection between $\forall$ and $\exists$ via $\neg$), proving statements with quantifiers.
5. Proof techniques including proof by contradiction, proof by cases, proving implications (assuming the antecedent and proving the conclusion).
6. Mathematical induction on natural numbers, and structural induction on recursively defined objects, such as formulae, trees. **Applications:**
    - Proving that simple (loop and recursive) programs behave as intended (this entails several case studies not just one example).
7. Basic counting, subsets of a set, binomial notation, binomial theorem; sum and product notation. **Applications:**
    - Estimating security of passwords (i.e., computing the number of words coming from a given alphabet, and interpreting this in terms of likelihood of guessing a password correctly).
8. Elementary graph theory, including trees and spanning trees. **Applications:**
    - Circuit analysis (finding the fundamental [independent] cycles of a circuit by finding the *spanning tree* of the underlying graph);
    - Storing and retrieving data efficiently (sort trees).
    - Huffman coding (cf. for example, **Rosen** p.701);
9. Finite state machines with and without output, as tables and as state diagrams. **Applications**:
    - Arguing (by induction) that a given machine behaves correctly according to a given specification.
    - Using automata to assess correctness of simple concurrent processes.
    - Using automata for text lexical analysis.

Learning Objectives:

By the end of the course, the students will be expected to be able to:

- Prove any propositional formula that is a tautology, using truth tables or syntactic proof techniques such as resolution.
- Show why a propositional formula is not a theorem (not a tautology).
- Build simple switching circuits using OR, NAND, AND, etc., gates.
- Manipulate expressions involving intersection, union and set-theoretic difference, and deduce whether different such expressions represent the same set.

- Express relations abstractly as a subset of a Cartesian product of sets, and construct a partition from an equivalence relation (and vice versa).
- Specify the domain and codomain of a given function, determine whether a function is 1-1 or onto (or both), and apply these concepts to infer whether composition of functions and inverse functions are well defined.
- Compute the cardinality of finite sets of objects constructed according to the various set-theoretic operations of union, intersection, Cartesian product and power set.
- Derive formulas of the cardinality of sets of objects depending on a parameter, such as the number of strings of length n from a given alphabet, or the number of ordered or unordered sequences of length n.
- Exploit the pigeonhole principle in cardinality arguments.
- Prove or disprove as the case may be simple formulas in quantified logic.
- Translate English mathematical statements into predicate logic formulas.
- Be able to correctly form the negations of “for some x A(x) is true” and “for all x A(x) is true”
- Prove simple mathematical statements by contradiction, by cases, or by assuming the antecedent.
- Prove by induction statements that depend on a natural number.
    - In particular: Prove the correctness of single loop programs and simple recursive programs.
- Prove statements about inductively defined objects by structural induction.
    - In particular: Prove the correctness of simple recursive programs; prove properties of well-formed formulas (e.g., equal number of left and right brackets).
- Be able to reason about graphs and (binary) trees and use them in several examples
    - To demonstrate an understanding of locating the fundamental cycles of an electrical circuit
    - Use a tree to efficiently store and retrieve information.
    - Be able to show simple properties of trees (examples: relation between number of nodes and edges; relation between number of nodes and height)
    - Use trees to find the Huffman codes of symbols of an alphabet that have probabilities of occurrence attached to them.
- Construct simple finite automata based on a specification and argue (typically by induction) that they behave exactly as specified.
- Construct automata that can recognize in a text its “arbitrary” words and its specific “keywords” the latter according to a given list (corresponding to the action of a “scanner” in a compiler, which finds numbers and identifier names in a computer program as well as keywords such as “if”, “then”, “else”, “begin”, “end”, etc.)
- Construct simple automata to assess the correctness of simple concurrent programs.

The grade weight distribution of the course components is as follows:

15% - Assignments (biweekly)
15% - Quizzes (following the assignments)
30% - Midterm (in-class)
40% - Final Exam (scheduled by the Registrar office)

Recommended Text: Discrete Mathematics and Its Applications, by Kenneth H. Rosen, ISBN: 0073383090; Publisher: McGraw-Hill.

Prerequisites: MHF4U and MCV4U
Course Credit Exclusions: EECS/MATH1019 3.00, MATH2320 3.00