Proposition 2: *Discrete Math* should have a distinctly mathematical flavor

Discrete Math should not be entirely devoted to computer science topics. Far from it. Our CS/CIS students need to be trained in mathematics as a way of thought. They should have an appreciation for a variety of mathematical subjects, and become familiar with the use of a wide variety of mathematical notation and vocabulary. Computer science is fundamentally a *mathematical* science.

Proposition 3: *Discrete Math* can and does have a distinctly computer science flavor

There are many points of contact between the subject matter of *Discrete Math* (as taught from the current textbook) and computer science. Some of these are explicitly brought out by the book. Others are not, but they are there, nonetheless, to be revealed by the knowledgeable instructor. The points of impingement and overlap between computer science and *Discrete Math* as currently taught include the following:

Mathematical topic	**Relation to computer science**
regular expressions	essential to an understanding of formal language theory and compiler construction
congruence	the *mod* function, endemic to all computer languages
matrices	the computer language "array"
unary and binary operators	unary, binary, etc, operations of computer languages
associativity	in computer languages, the use of left/right/no associativity has a profound impact on semantics (*n.b.* the C language)
logical operators	these constitute the language of computer hardware
mathematical proof	the same mental gymnastics are used in the formal proof of program correctness
mathematical induction	can be used to "prove" programs
combinatorics and counting	useful for analyzing program complexity
recurrence relations	direct counterpart: recursive procedures
relations	the precedence relations of computer languages
relations as boolean matrices	a useful computer-oriented data representation for relations
relations as ordered pairs	how to represent them in a computer?
transitive closure	useful in the analysis of precedence relations
Warshall's algorithm	efficient algorithm for transitive closure
functions	the computer language *function* construct
factorial, floor, ceiling	essential to the study of computer languages and programming
computational complexity, big-O notation, analyzing a program to determine its computational complexity	these are primarily CS topics, but are intensively mathematical in nature
graph theory - Traveling Salesman Problem	how many ages of the universe would it take all the world's computers working together to solve the 50-city TSP by the method of exhaustion?
graph theory - TSP	programming a robot
graph coloring	direct applicability to certain aspects of computer code optimization
trees and tree searching: pre-order, post-order, in-order	ubiquitous in any study of programming and data structures excellent illustration of recursive procedures
labeled trees	these are the "abstract syntax trees" to be studied later in the computer languages and compiler courses
prefix and postfix notation	invented by a logician, but primarily a computer science topic
spanning trees	Kruskal's algorithm
boolean algebra	basis of computer circuitry
minterms of mathematical logic	used in circuit design, artificial intelligence
Karnaugh maps	distinctly CS-related - to reduce the number of circuit elements needed to provide a specified logical function