Modeling and Impact

Understand, guide, shape impact of computing on society/the world. Connect the role of Theory CS classes to other applications (in undergraduate CS curriculum and beyond). Model problems using appropriate mathematical concepts.

Model systems Model systems with tools from discrete mathematics and reason about implications of modelling choices. Explore applications in CS through multiple perspectives, including software, hardware, and theory.

- Selecting and representing appropriate data types and using notation conventions to clearly communicate choices (Week 1, Week 2)
- Connecting logical circuits and compound proposition and tracing to calcluate output values (Week 3)
- Determining the properties of positional number representations, including overflow and bit operations (Week 2, Week 3)
- Modeling data structures, such as linked lists or trees, as recursively defined data structures (Week 7)
- Modeling data with partitions and functions (Week 10)

Translate Translate between different representations to illustrate a concept.

- Translating between symbolic and English versions of statements using precise mathematical language (Week 5, Week 6, Week 7)
- Tracing algorithms specified in pseudocode (Week 2)
- Representing numbers using positional representations, including decimal, binary, hexadecimal, fixed-width representations, and 2s complement (Week 2)
- Translating between truth tables (tables of values) and compound propositions (Week 4)
- Drawing graph representations of relations and functions e.g. Hasse diagram and partition diagram (Week 10)

Communication

Clearly and unambiguously communicate computational ideas using appropriate formalism. Translate across levels of abstraction.

Use language conventions to improve arguments Use precise language to clearly present logical arguments and communicate with technical and non-technical colleagues.

- Distinguishing between tautologies, contradictions, and contingencies (Week 3)
- Determining whether a given binary relation is symmetric, antisymmetric, reflexive, and/or transitive (Week 9, Week 10)
- Determining whether a given function is one-to-one, onto, and/or invertible (Week 8)
- Determining whether a given binary relation is an equivalence relation and/or a partial order (Week 9, Week 10)
- Distinguishing between structural induction, mathematical induction, and strong induction (Week 7)
- Using appropriate signpost words to improve readability of proofs, including 'arbitrary' and 'assume' (Week 5, Week 6, Week 7)

Mathematical definitions and notation Use precise notation to encode meaning and present arguments concisely and clearly

- Precisely describing a set using appropriate notation e.g. roster method, set builder notation, and recursive definitions (Week 1, Week 2)
- Defining important sets of numbers, e.g. set of integers, set of rational numbers (Week 8, Week 9)
- Defining functions, predicates, and binary relations using multiple representations (Week 9, Week 10)
- Listing the truth tables of atomic boolean functions (and, or, xor, not, if, iff) (Week 3, Week 4)
- Using functions to compare cardinality of sets (Week 8)
- Classifying sets into: finite sets, countably infinite sets, uncountable sets (Week 8, Week 9)

Problem Solving Know, select and apply appropriate computing knowledge and problem-solving techniques. Reason about computation and systems. Use mathematical techniques to solve problems. Determine appropriate conceptual tools to apply to new situations. Know when tools do not apply and try different approaches. Critically analyze and evaluate candidate solutions.

Propositional and predicate logic Reason about the truth or falsity of complicated statements using Boolean connectives, quantifiers, and basic operations.

- Evaluating compound propositions (Week 3, Week 4)
- Judging logical equivalence of compound propositions using symbolic manipulation with known equivalences, including DeMorgan's Law (Week 3, Week 4, Week 5, Week 6, Week 7)
- Judging logical equivalence of compound propositions using truth tables (Week 3)
- Rewriting compound propositions using normal forms (Week 3)
- Judging whether a collection of propositions is consistent (Week 3)
- Writing the converse, contrapositive, and inverse of a given conditional statement (Week 4, Week 5, Week 6, Week 7)
- Determining what evidence is required to establish that a quantified statement is true or false (Week 4, Week 5, Week 6, Week 7, Week 8)
- Evaluating quantified statements about finite and infinite domains (Week 4, Week 5, Week 6, Week 7, Week 8)

Proof strategies and valid arguments Apply proof strategies, including direct proofs and proofs by contradiction, and determine whether a proposed argument is valid or not.

- Identifying the proof strategies used in a given proof (Week 5, Week 6, Week 7)
- Identifying which proof strategies are applicable to prove a given compound proposition based on its logical structure (Week 5, Week 6, Week 7)
- Carrying out a given proof strategy to prove a given statement (Week 5, Week 6, Week 7)
- Carrying out a universal generalization argument to prove that a universal statement is true (Week 5, Week 6, Week 7)
- Tracing and/or modifying a proof by contradiction (Week 8)
- Using proofs as knowledge discovery tools to decide whether a statement is true or false (Week 5, Week 6, Week 7, Week 8, Week 9, Week 10)

Recursion and Induction Work with recursively defined data and prove their properties using inductive reasoning

- Using a recursive definition to evaluate a function or determine membership in a set (Week 1, Week 2)
- Using mathematical induction to prove mathematical identities, inequalities, and other invariants (Week 6, Week 7)
- Using strong induction to prove mathematical identities, inequalities, and other invariants (Week 6, Week 7)
- Using structural induction to prove statements about recursively defined objects, e.g. strings and trees (Week 6, Week 7)

Modular Arithmetic Reason about modular arithmetic.

- Using the definitions of the div and mod operators on integers (Week 2, Week 9, Week 10)
- Using divisibility and primality predicates (Week 9, Week 10)
- Applying the definition of congruence modulo n and modular arithmetic (Week 9, Week 10)