What is the course about

- This course is about three traditionally central areas of the theory of computation: Automata, Computability, and Complexity
- Links to questions:
  - What are the fundamental capabilities and limitations of computers?
  - What makes some problems computationally hard and others easy?

Complexity, Computability, and Automata

- Complexity theory:
  - to classify problems as easy ones and hard ones.
  - i.e., the sorting problem is easy, while scheduling problem is much harder;
- Computability theory:
  - to classify solvable and not solvable problems
  - i.e., determining whether a mathematical statement is true or false

Complexity, Computability, and Automata (cont’d)

- Automata theory:
  - deals with the definitions and properties of mathematical models of computation;
  - allows practice with formal definitions of computation

Sets

- Sets: a group of objects (elements or members) represented as a unit
  - Infinite set: contains infinitely many elements;
  - Subset: set A is a subset of set B if all members of A are also members of B;
  - Proper subset: if A is a subset of B and not equal to B;
  - Empty set : a set with zero members;

Sets (cont'd)

- Intersection
- Union
- Complement
- Power set
- Cartesian product of *k* sets

Strings and Languages

- **Alphabet** – a nonempty finite set of symbols.
  - Notation: Σ .
  - Examples:
    - Binary alphabet {0,1}
    - English alphabet {a, b, c,….}
- **String over an alphabet** Σ - a finite sequence of symbols from that alphabet.
  - 00101 is a string over the binary alphabet.
  - dabd is a string over the English alphabet.

Strings and Languages (cont'd)

- **Empty string**: ε---the empty sequence with no symbols
- **Concatenation of strings**: Concatenation of two strings u.v ----- concatenate the symbols of u and v.
  - Notation: u.v
  - Examples:
    - 01.011 = 01011
    - ε.u = u.ε = u for every string u (identity property for concatenation)

Strings and Languages (cont'd)

- **Prefix** - u is a prefix of v if there is a w such that v = u.w
  - Examples:
    - ε is a prefix of 0 since 0 = ε.0
    - pen is a prefix of pencil since pencil = pen.cil
- **Suffix** - u is a suffix of v if there is a w such that v = w.u
  - Examples:
    - 0 is a suffix of 0 since 0 = ε.0
    - cil is a suffix of pencil since pencil=pen.cil

Strings and Languages (cont'd)

- **Substring** - u is a substring of v if there are x and y such that v = x.u.y.
  - Examples:
    - ver is a substring of the string *university* since university = uni.ver.sity
    - *a* is a substring of a since a = ε .a. ε

Strings and Languages (cont'd)

- **Language over alphabet** Σ - a set of all strings over Σ.
  - Notation: L.
  - Examples:
    - {0, 00, 01, 10, ...} is an infinite language over the binary alphabet.
    - {a, b, bd} is a finite language over the English alphabet.
- **Empty language** – an empty set with no strings. Notation: Φ.

Proof, theorem, lemma

- **Proof**:
  - a convincing logical argument that a statement is true;
- **Theorem**:
  - A mathematical statement proved true
- **Lemma** (a helping theorem):
  - A proved proposition

Proof by contradiction

- A common form of argument for proving a theorem.
- First assume that the theorem is false, then show that this assumption leads to an obviously false consequence, called contradiction.