1. Strong Methods (vs. Weak Methods)
   a. Adding Knowledge increases power of methods
      i. Evaluation functions of Chapter 2 are weak methods
      ii. Creating data that represents complex knowledge plus inference increases ability to search for goal state more efficiently

2. Types of Knowledge
   a. Standard CS “knowledge”
      i. Procedural (vs. Declarative)
         1. Procedural
            a. “How” to solve problem
               i. Steps to solve problem
               ii. Programming procedure
         2. Declarative
            a. “What” states the knowledge
            b. Equation is declarative
               i. \( d = rt \)
               ii. shows relationships
               iii. need algebra to solve this, “how”
      ii. Relational as in relational data bases – records of information containing attributes and values; combined in tables to represent relationships between data; SQL
      iii. Hierarchical – Relationships between classes of things; Object-oriented programming

b. AI “knowledge”
   i. Monotonic – truth value does not change with time
      1. Propositional Logic
         a. Propositions basic element, can be “true” or “false”
         b. Propositions connected by boolean operators (negation, “or”, “and”, “if-then”, “if and only if” to form “sentences” of meaning
Inference rules to prove new “sentence” correct given truth values of known facts and sentences

2. **Predicate Logic** (FOL: First-order Logic)
   a. Syntax
      i. Predicates represents attributes and relationships of variables
         1. Brown (x)
         2. Mother-of (x, y)
      ii. Boolean operators
      iii. Quantification (universal or existential) with variables distinguished from instances
   b. Semantics
      i. Predicates can be true or false
      ii. Inference rules to prove new “sentence” correct given truth values of known facts and sentences
         1. Modus ponens
         2. Inference Example
            a. All humans are mortal.
            b. Socrates is a human.
            c. Therefore, Socrates is mortal.
            \[ \forall x \text{ Human}(x) \Rightarrow \text{Mortal}(x) \]
            d. \[ \exists x \text{ Human}(x) \text{ where } x \text{ is bound to Socrates} \]
            \[ \therefore \exists x \text{ Mortal}(x) \text{ where } x \text{ is bound to Socrates} \]
   c. Proof by contradiction: Resolution Theorem Proving (RTP) can be automated (Robinson, 1960); used in Prolog programming language
   d. FOL is sound: can rely on truth value of derived sentences

3. **Frames**
   a. Common AI knowledge representation method
   b. Collection of “slots” that define the attributes of an object and its relationship to other objects
   c. Should correctly implement FOL to be “sound”

4. **Semantic Networks**
   a. Earliest AI knowledge representation method
   b. Network where nodes represent objects and arcs represent attributes including relationships to other objects
   c. Hierarchy represented by “isa”
   d. Part-of relationship represented by “has”
   e. Should correctly implement FOL to be “sound”

ii. **Other Logics used in AI**
   1. **Non-monotonic**
      a. truth value changes over time
      b. Monotonic logic requires that something is always true or always false
c. In FOL if you have something that is both true and false you can prove anything

2. **Bayesian**
   a. Truth value represented as a degree of certainty with a real value number between 0.0 (completely uncertain) and 1.0 (completely certain)
   b. Computed statistically as a conditional probability and prior probability

3. **Higher order logics** –
   a. Predicates that range over other predicates
   b. Example: “I know that the weather is rainy.”

3. **Knowledge Interchange Format (KIF)**
   a. Language to exchange knowledge between agents
   b. Based on FOL

4. **Knowledge Engineering**
   a. Process of extracting, discovering, analyzing and representing knowledge in an AI system
   b. **Ontology 101 (Noy & McGuinness paper)**
      i. *What is ontology?*
         1. Explicit formal specifications of the terms of domain and relationships
         2. Classes, attributes, relationships, instances, values
      ii. *Why create an ontology?*
         1. Generalization
         2. Reuse
         3. Sharing common knowledge with other programs
         4. Analysis of domain knowledge
      iii. *How does ontology differ from OO programming?*
         1. Structural properties vs. operational methods in OO
         2. Declarative vs. procedural knowledge
      iv. *How do you develop an ontology?*
         1. Define classes
         2. Arrange in taxonomy (hierarchy of relationships)
         3. Define attributes and values
         4. Fill in instances
      v. *Lessons learned from creating ontology*
         1. Many possible models!
         2. Iterative process
         3. Keep concepts close to actual physical objects and descriptions in natural language