Final Study Guide

Final Time and Place:

- Monday, May 14, 12-3pm
- Chandler-Ullmann 248

Format:

You can expect the following types of questions: true/false, short answer, and smaller versions of homework problems. Although you will have three hours to complete the final, it will only be about twice as long as the midterm. It will be closed book and closed notes. However, you may bring one 8 ½ x 11" "cheat sheet" with handwritten notes on both sides. All PDAs, portable audio players (e.g., iPods) and cell phones must be put away for the duration of the test, but you may use a basic, 4 function calculator. If you only have a programmable calculator, then you must clear its memory before the test, and at my request be able to prove to me that you have done so.

Coverage:

The test will be comprehensive, however approximately two-third of the questions will be on subjects covered since the midterm. In general, anything from the assigned reading or lecture could be on the test. In order to help you focus, I have provided a **partial list** of topics that you should know below. In some cases, I have explicitly listed topics that you do not need to know. In addition, you do not need to memorize the pseudo-code for any algorithm, but you should be able to apply the principles of the major algorithms to a problem as we have done in class and on the homework.

- Ch. 1 – Introduction
  - rationality
  - definitions of "artificial intelligence"
  - The Turing Test
  - **you do not need to know:**
    - dates and history
- Ch. 2 - Agents
  - PEAS descriptions
    - performance measure, environment, actuators, sensors
  - properties of task environments
    - fully observable vs. partially observable, deterministic vs. stochastic vs, strategic, episodic vs. sequential, static vs. dynamic, discrete vs. continuous, single agent vs. multiagent, known vs. unknown
  - agent architectures
    - simple reflex agents, goal-based agents, utility-based agents, learning agents
  - state representations
    - atomic, factored, structured
- Ch. 3 – Search
  - problem description
    - initial state, actions, transition model, goal test, path cost/step cost

  - tree search
    - expanding nodes, fringe
    - branching factor
  - graph search
    - explored set
  - uninformed search strategies
    - breadth-first, depth-first, uniform cost
    - similarities and differences / benefits and tradeoffs between strategies
    - evaluation criteria
      - completeness, optimality, time complexity, space complexity
  - best first search
    - evaluation function
  - informed search
    - heuristics
    - greedy best-first, A*
    - admissible heuristics
    - similarities and differences / benefits and tradeoffs between strategies
  - **you do not need to know:**
    - depth-limited, iterative deepening or bidirectional search
    - the exact *O*() for any strategy's time/space complexity *(but you should know relative complexity)*
    - details of proof that A* is optimal if h(n) is admissible
    - memory bounded heuristic search
    - learning heuristics from experience
- Ch. 5 - Game playing (Sect. 5.1-5.2, 5.4, 5.7-5.9)
  - two-player zero-sum game
  - problem description
    - initial state, actions, transition model, terminal test, utility function
  - minimax algorithm
  - optimal decision vs. imperfect real-time decisions
  - evaluation function, cutoff-test
  - **you do not need to know:**
    - alpha-beta pruning
    - forward pruning
    - details of any state-of-the-art game playing programs
- Ch. 8 – First-Order Logic
  - syntax and semantics
    - be able to translate English sentences into logic sentences
  - quantification
    - existential, universal
  - domain, model, interpretation
  - entailment
  - equality/inequality
    - making statements about quantity (e.g., exactly two brothers)
  - **you do not need to know:**
    - specific axioms from the domains given in class or the book

- Ch. 9 – Inference in First-Order Logic (Sect. 9.1-9.4)
  - entailment and correctness of inference *(also see Sect. 7.3, pp. 240-243)*
    - definition of entailment
    - sound, complete
  - substitution
    - apply substitutions, normal form
  - unification
    - most general unifier
  - forward-chaining
  - backward-chaining
    - pros / cons
    - diagramming inference process
  - negation as failure
  - **you do not need to know:**
    - inference rules, skolemization
    - constraint logic programming
- "Intro to Prolog Programming" Reading, Ch. 1
  - syntax
    - be able to write rules and facts in Prolog
    - translating to FOL and vice versa
  - backward-chaining, depth-first search
    - be able to find the answers to a goal given a simple Prolog program
  - negation as failure / closed world assumption
- Ch 10 – Planning (Sect. 10.1-10.3)
  - problem description
    - initial state, goal state, actions
  - The PDDL language
    - preconditions and effects
  - forward state-space search
    - applicable actions, result states
  - backward state-space search
    - relevant actions, predecessor states
  - planning graphs
    - levels: fluents and actions
    - persistence actions
    - mutual exclusion (mutex) links
      - between actions
        - inconsistent effects
        - interference
        - competing needs
      - between fluents
        - negation
        - inconsistent support
    - used as heuristics
      - max level, level sum, set-level

        - GraphPlan
            - building the graph
            - extracting the solution
    - **you do not need to know:**
        - the actions for any specific planning problem given in the book
        - proof of termination for GraphPlan
- Ch 12 – Knowledge Representation (Sect. 12.1-12.2, 12.5, 12.7-12.8)
    - categories
        - unary predicate vs. object representation
    - semantic networks
        - inheritance
        - compared to FOL
    - **you do not need to know:**
        - axioms for representing composition, measurements, etc.
        - description logic
        - Semantic Web
        - OWL
- Ch. 13 - Uncertainty
    - Boolean, discrete and continuous random variables
    - prior probability and conditional probability
    - full joint distribution, atomic events
        - calculate probability of an event from the full joint
    - independent variables
    - conditional independence
    - product rule, chain rule
    - Bayes Rule
- Ch. 14 - Bayesian Networks (Sect. 14.1-14.2, 14.4)
    - understand network structure
    - compute probability of an atomic event
    - compute $\mathbf{P}(X \mid \mathbf{e})$ by enumeration
    - **you do not need to know:**
        - variable elimination algorithm
        - clustering algorithms
- Ch. 15 – Probabilistic Reasoning Over Time (Sect. 15.1-15.3)
    - Markov assumption
    - first-order Markov process
    - stationary process
    - transition model and sensor model
    - representing a set of variables for a specific time period (e.g., $X_{a:b}$)
    - types of inference
        - filtering, prediction, smoothing, most likely explanation
    - **you do not need to know:**
        - the algorithms for any of the types of inference
        - simplified matrix algorithms
        - details of speech recognition or localization problem

- Ch. 16 - Making Simple Decisions (Sect. 16.1 – 16.3)
  - utility function
  - maximum expected utility
  - **you do not need to know:**
    - constraints on rational preferences
- Ch. 18 - Learning (Sect. 18.1-18.4, 18.6-18.9)
  - types of learning
    - supervised vs. reinforcement vs. unsupervised
  - supervised learning
    - hypothesis
      - goals: consistent, generalizes well
      - hypothesis space
    - training set vs. test set
    - positive vs. negative examples
  - decision trees
    - expressive power
    - learning
    - entropy, information gain
  - evaluating hypotheses
    - overfitting
    - learning curve
    - K-fold cross-validation
  - neural networks
    - activation functions
      - threshold, sigmoid
    - perceptron
      - linearly-separable functions
      - supervised learning method
        - learning rate, epoch, error
    - multi-layer feed-forward networks
      - be able to calculate output
      - what can be represented?
  - nearest neighbors
    - k-nearest neighbor algorithm
      - how it works
      - normalization of the dimensions
  - support vector machines
    - concepts (but not formulas for)
      - maximum margin separator
      - support vector
      - kernel trick
  - **you do not need to know:**
    - how to calculate the base 2 log (i.e., $\log_2$) -- if you need to compute this, I will provide a table
    - the back-propagation algorithm
    - linear regression, logistic regression

- recurrent networks
- k-d trees
- locality-sensitive hashing
- non-parametric regression