Midterm Study Guide

Midterm Time and Place:
- Wednesday, March 2, 9:10-10am
- Packard 258 (our usual room)

Format:
The test will be held in class. You can expect the following types of questions: true/false, short answer, and smaller versions of homework problems. It will be closed book and closed notes. However, you may bring one 8 ½ x 11” “cheat sheet” with handwritten notes on one-side only. Also, all calculators, PDAs, and cell phones must be put away for the duration of the test.

Coverage:
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
  - agent architectures
    - simple reflex agents, goal-based agents, utility-based agents
  - you do not need to know:
    - learning agents

- Ch. 3 – Search (Sect. 3.1-3.5)
  - problem description
    - initial state, actions (successor function), goal test, path cost, step cost
  - tree search
    - expanding nodes, fringe
    - branching factor
  - 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
  o 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)
• Ch. 4 – Informed Search (Sect. 4.1-4.2)
  o best first search
  o evaluation function, heuristics
  o strategies
    • greedy search, A*
    • admissible heuristics
    • similarities and differences / benefits and tradeoffs between strategies
  o you do not need to know:
    • details of proof that A* is optimal if $h(n)$ is admissible
    • memory bounded heuristic search
    • learning heuristics from experience
• Ch. 6 - Game playing (Sect. 6.1-6.2, 6.4, 6.6-6.8)
  o two-player zero-sum game
  o problem description
    • initial state, actions (successor function), terminal test, utility function
  o minimax algorithm
  o optimal decision vs. imperfect real-time decisions
  o evaluation function, cutoff-test
  o you do not need to know:
    • alpha-beta pruning
• Ch. 7 – Logical Agents (Sect. 7.1-7.4)
  o knowledge-based agents
    • TELL, ASK
  o propositional logic
    • syntax and semantics
  o entailment, models, truth tables
  o valid, satisfiable, unsatisfiable
  o inference algorithms
    • criteria: sound, complete
  o model checking
  o you do not need to know:
    • details of the Wumpus world
• Ch. 8 – First-Order Logic (Sect. 8.1-8.5)
  o syntax and semantics
    • be able to translate English sentences into logic sentences
  o quantification
    • existential, universal
  o domain, model, interpretation
• Ch. 9 – Inference in First-Order Logic (Sect. 9.1-9.2, 9-4)
  o substitution, unification
    ▪ most general unifier
  o backward-chaining
    ▪ pros / cons
  o you do not need to know:
    ▪ inference rules, skolemization
    ▪ constraint logic programming
    ▪ negation as failure
• “Intro to Prolog Programming” Reading, Ch. 1
  o syntax
    ▪ be able to write rules and facts in Prolog
    ▪ translating to FOL and vice versa
  o backward-chaining, depth-first search
    ▪ be able to find the answers to a goal given a simple Prolog program