

## CS 5541: Artificial Intelligence (4)

### Catalog Description:

Principles and programming methods of artificial intelligence. Advanced Lisp programming. Knowledge representation methods, state space search strategies, and use of logic for problem solving. Applications chosen from among expert systems, planning, natural language understanding, uncertainty reasoning, machine learning, and robotics.

**Textbook:** Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 2<sup>nd</sup> Ed. Prentice-Hall, 2003.

**References:** M. Felleisen, R. B. Findler, M. Flatt, and S. Krishnamurthi, *How to Design Programs*, MIT Press, 2002.

### Course Goals:

This course introduces the field of artificial intelligence (AI). Students learn about AI methods in machine problem solving (a) through search and reasoning, (b) through learning algorithms (e.g., neural networks), and (c) through algorithms dependent on particular instructors (e.g., robotics related techniques). Students also learn the AI programming language *Lisp*, and learn to think and write about the issues which dominate the field.

### Prerequisites by Course & Topic

CS 2511: Software Development – large scale software issues including management, design, modularity, and data abstraction.

MA 3355: Discrete Math – predicates and quantifiers, sets, relations, proof techniques, recursion and induction, combinatorial counting techniques, graph theory

### Major Topics Covered in the Course

- Lisp function and data definitions
- Knowledge representation, empiricist and rationalist approaches, and agent concepts
- State space search, informed and uninformed search techniques
- Iterative improvement techniques (e.g., genetic algorithms)
- Propositional logic and inference in logic
- Learning algorithms: decision trees, neural networks
- Case studies (e.g., Deep blue, Speech translation).

**Class/Laboratory Schedule:** Lecture: 3 hours per week, Laboratory: 1

### Laboratory Projects

- Introduction to Lisp (2)
- Simulated agents (2)
- Search in Lisp (2)
- Genetic Algorithms (2)
- Research Paper Reading and Summary (2)
- Neural networks (2)

### Course Contribution to Program Objectives and Outcomes:

1. Students learn mathematical techniques related to neural networks and information theory in decision trees. (*a*)
2. Student further enhance their knowledge of software design and engineering through programming assignments. (*b*)
3. Students increase their proficiency in algorithms and data structures by learning the intricacies of neural networks and genetic algorithms. They learn how to use search techniques to find solutions to formally specified goals. They also learn Lisp – an additional programming paradigm. (*c*)
4. Students apply knowledge gained in this course to AI problem solving methods, artificial neural networks, robotics and machine vision. (*d*)
5. Students continue to enhance their abilities to work independently on software design and implementation. (*e*)

### Estimate CSAB Category Content

	CORE	ADVANCED		CORE	ADVANCED
Data Structures			Computer Organization and Architecture		
Algorithms		2	Concept of Programming Languages		1
Software Design		1			

**Oral and Written Communications**

Every student is required to submit at least   1   written reports (not including exams, tests, quizzes, or commented programs) of typically   2-4   pages and to make        oral presentations of typically        minutes duration. Include only material that is graded for grammar, spelling, style, and so forth, as well as for technical content, completeness, and accuracy.

**Social and Ethical Issues****Theoretical Content**

The course presents theoretical material on the computational nature of intelligence. For example, the physical symbol system hypothesizes that intelligence stems from symbolic systems (e.g., computers). Theoretical material is also presented on logical systems (e.g., inference in propositional logic), and in learning (e.g., information theory used in decision trees and analysis of neural networks).

**Problem Analysis**

Students learn to analyze problem statements of traditional micro-world problems (e.g., the water jugs problem) into operators, start states, and goal states. Students also learn to analyze programming problem statements into software architectures that can be programmed in Lisp. Analysis problems are studied in building categorization systems that use decision trees and neural networks, and students should come away with skills in these areas.

**Solution Design**

The programming problems carried out in the lab portion of the class require the design of software solutions in Lisp. This course emphasizes the use of recursive functions, typical of a functional style of programming, and so the students learn to design in terms of this functional style.

**Coordinator/Prepared by:** C. Prince