AIP lecture series – course outline

The lecture series is split into 12 units. Each unit accounts for approximately two weeks of study including lecture, tutorial & self study. Units are delivered roughly in the order presented but there is room for some flexibility according to the needs of students.

Lectures are mostly operated in the form of workshops with a high level of interaction between students & tutor to develop solutions to problems and examine the semantics of language constructs in Lisp. Lectures/workshops are supported by practical lab exercises and presentations. Presentations start after initial introductory sessions and take place in parallel to lectures.

Course work marks for students come from two elements of assessment...

1 presentations (40% of final mark)
2 programming assignment (60% of final mark)

Presentations are organised as follows...

- students are organised into groups of 3-4 people (groups of 5 are permitted but only in cases where it is not possible to divide students into groups of 3 & 4 because of student numbers);
- groups will each present 2-3 presentations during the module;
- each week 2 student groups per tutor session will tackle one presentation problem each and present their findings (preferably different groups will each do different problems);
- marks will be awarded for their analysis of the problem and the quality of their solution(s) – not for their presentation performance (we discourage using presentation aids like power-point);
- groups will distribute peer performance tokens to group members to reflect their individual contributions to the solution, there are 13 peer assessment tokens per presentation;
- individual marks for students are calculated from group marks and peer assessment points according to the presentation-mark-calculator found on the web site.

The programming assignment is an AI programming exercise involving search and/or planning. The programming assignment also involves group work. Details of the programming assignment will be provided part way through the module.

Lecture Series...

1. introduction to AI, the symbolic computation approach, Lisp & the Allegro IDE

2. search and legal move generation (LMG)
   the concept of search & LMG
   simple LMGs (numbers game, word transform)
   discussion of problems suited to search

3. (head) recursion
   using numbers, eg: factorial, sum-up-to
   functions using lists as data but returning single values, eg: member, sum-numbers
   functions using lists & returning lists, eg: increment-numbers

4. more (head) recursion
   recursing on nested lists/trees
   building an equals function
   building an equals/matcher function including wildcards

5. tail recursion
   using numbers, eg: factorial, sum-up-to
   functions using lists as data but returning single values
   reverse

6. mapping functions & lambda
   mapcar, remove-if, reduce
   others: count-if, find-if
   NB: forms like remove-if-not, find-if-not are formally deprecated but we occasionally use
   these forms if they help to simplify example code.

7. worked example #1 – query mechanism on tuples

8. worked example #2 – rule apply mechanism

9. worked example #3 – navigating a network

10. worked example #4 – applying operators

11. example AI application #1 (see notes below)

12. example AI application #2 (see notes below)

example AI applications

Towards the end of the module we investigate 2 or more AI applications that provide some additional context for studying AIP. These areas do not underpin the assessment so the choice is at the discretion (& perhaps influenced by the interest of) the lecturer. The following is a list of suitable application areas...

1 classic computer games
eg: chess, draughts, connect-4, etc
minimax, static evaluation, alpha-beta pruning

2 search with costs
best 1st, A+, A* search algorithms
example problems (travelling salesman, path planning in games etc)

3 constraint propagation
eg: solving Sudoku, Kakuro, timetabling, etc

4 learning by near-miss (Winston)

5 Netlogo modelling