TEESSIDE UNIVERSITY
SCHOOL OF COMPUTING

ICA SPECIFICATION

Module Title: **Artificial Intelligence Applications**	**Module Leader**: S. Lynch
	Module Code: VIS3071-N
Assignment Title: **AIA 2016-17**	**Final Deadline Dates: TBC** *** see deadline notes**
	Deadline Time: 23.59
	Submission Method: online & presentation

Online Submission Notes:

- Please follow carefully the instructions given on the Assignment Specification
- When an extension has been granted, a fully completed and signed Extension form must be submitted to the Student Advice & Support Centre as soon as it had been signed.

FULL DETAILS OF THE ASSIGNMENT ARE ATTACHED
INCLUDING MARKING & GRADING CRITERIA

AI Applications Assessment

This document specifies the assessment for the AI Applications module.

key points...

- students will work in groups of 3 (or 2 if unavoidable due to tutorial sizes);
- this assessment specifies papers organised into 3 sections. Each group will complete 3 papers – 1 paper from each section.

paper types and deadlines

See later sections describing paper types and topic areas in detail.

There are 3 deadline/feedback stages for each paper. See module calendar for deadlines; see *marking-phases* document for details of the requirements at each stage.

Rough guide to deadlines...

paper	1. NetLogo	2. intelligence engines	3. language processing
approximate deadlines	weeks 3,5,6	weeks 7,10,11	weeks 11,13,14

papers

Papers describe fully developed ideas (and experiments) which have been thoroughly tested. The results, evaluation, analysis & conclusion are complete with reasoned evidence to defend them.

	pages
title, authors, abstract	½
intro/background	1
your work	1 ½
results & evaluation	2
analysis & conclusion	1

notes...

- the page count does not include diagrams
- abstract should be 100-150 words
- papers should cite a minimum of 5 works
- papers must be formatted according to the author instructions and paper templates provided
- topics described for papers present a number of examples/targets for investigation. Each group will be approved a different target/example to consider so groups **must** discuss the specifics of each paper with their tutor before starting the work;

peer marks...

- groups will distribute 13 peer assessment points for each paper, individual marks will be calculated from the group mark & peer assessment points as explained under the relevant section at www.agent-domain.org;
- an individual's overall mark will be a combination of their marks from each of the papers (marks from all papers will have the same weighting);

Paper 1- NetLogo agent modelling – pack/hive behaviour

The requirement for this work is to build a NetLogo model to examine the behaviour of interacting agents in medium-large scale aggregations. The model should define an abstraction or simulation of a system described in some academic literature (from biology, ecology, social studies, etc). Suitable examples include pack/hive/swarm formations, social structures, etc. Most experiments will involve contrasting types of utility gain resulting from differing behaviours (affecting collecting/consuming resources for example).

overview

Models could specify one or more breed of agent that forage for resources in their environment. Collection of resources (given different behaviour experiments below) provide a utility payoff to the hive/swarm and/or the individual.

Agents communicate with other agents in their group. Communication may be symbolic or non-symbolic, using pheromone trails, broadcasting, phone-a-friend, etc.

Models should contrast at least two different strategies for driving the behaviour of agents – see below for possible strategies to contrast.

sample behaviour experiments

There is a choice of experiments to be run in this exercise. In all cases research groups must have their experiments approved (before they start building code) with the contact person for this work. All models should contrast (at least) two different strategies. Examples...

1. sustainable levels of altruism in different types of social organisations;
2. factors effecting spread of social norms/memes (recycling, fashion, etc);
3. gaining utility from a resource requiring cooperation, eg:
    - requiring a minimum number of individuals to gain utility (consume any of the resource);
    - requiring (2 or more) cooperating individuals to accept/establish/negotiate different roles (cutter/carrier/etc);
4. consider utility payoffs based on collectivism, individualism and altruism;
5. two different competing groups that exist in conflict;
6. foraging behaviours based on smaller social groups (meerkats, etc).

You may also mix-in other factors...

1. resource must be returned to the hive to gain utility;
2. different activities (searching/carrying) have differing costs, etc;
3. communication strategies: pheromone trails, broadcasting, phone-a-friend, etc.

You may suggest additional strategies but these must be discussed and approved before you start the work.

Paper 2 – intelligence engines

You are required to demonstrate that you can construct a cross-platform AI application with an intelligence engine developed in Clojure and a reactive agent-based front-end in Netlogo. Your paper will describe its design and provide a critical appraisal of its capabilities and its extensibility. Your critique will reflect back on your design & the interaction between subsystems, highlighting the strengths and weaknesses of your approach.

Your practical work will contain a back-end control system (based on planning or search or rule application) and an agent based, visual front end. Look at the SHRDLU program used in lectures and read the examples below.

The front end will be built using Netlogo (or exceptionally Slick2D but if you use Slick2D you will be expected to develop a Clojure wrapper). The control system will be written in Clojure (or exceptionally some other dialect of Lisp). You are not required to build the control system from scratch, you may use or adapt code provided. You are expected to specify your own operators, rules, LMGs, etc.

The emphasis of this work is to...
(i) demonstrate your ability to develop an "intelligent engine" – so it will not be acceptable to build a simple legal move generator for the Clojure breadth-search for example;
(ii) successfully develop simple interoperability between your engine and another subsystem (preferably one based on a different software paradigm see next point...);
(iii) successfully interoperate between a symbolic subsystem using a high-level representational framework (facts, beliefs, rules, etc) and a lower level (reactive, non-symbolic?) subsystem.

Within these guidelines, you may either...
(i) build a fully developed or non-trivial control system or...
(ii) compare & contrast 2 (or more) different prototyped approaches to control system construction.

Note: tutors will be available to advise at all stages re: the nature of the work, design decisions and paper structure.

Paper 2 – intelligence engines : planning

The purpose of this work is to investigate planning, examining the ways in which plans may be specified & processed and comparing the performance of different approaches to plan formation.

example work

1. compare performance and useability of the Clojure *ops-search* and *planner* (see www.agent-domain.org);
2. compare performance of hierarchical plans against flat (non-hierarchical) plans;
3. analyse the performance of different approaches to the construction of belief systems.

typical practical work

1. identify a suitable domain for planning (check this with tutors before continuing with the work);
2. identify suitable operators for the domain. These should be appropriately generic (make sensible use of matching) and with hierarchical operators they should be organised into 2-3 levels of abstraction (the hierarchy levels);
3. extend, wrap or layer operator inference engines to process your operators;
4. develop the Netlogo front end & the communication control between subsystems (see agent-domain for a Clojure/Netlogo comms link);
5. test your work with suitable start an goal states (at least 3 different tests) – profile these tests for state-space size/growth and processing time. Note the testing & profiling are an important part of the paper.

Paper 2 – intelligence engines : other

possible investigations

1. develop a non-trivial rule-based inference engine (in Clojure) to derive high-level strategic behaviours & goals for low-level reactive agents existing in a Netlogo front-end. Contrast different strategies of rule-interpreter or deliberation;
2. implement a moving target search (D* for example) and, using a test-case environment, compare its performance with other types of search;
3. using a suitable example (discuss this with tutors)... implement at least 3 search algorithms (with at least 2 of them using cost information). Compare their operation with 2-3 problem sizes.
4. develop a minimax algorithm in Clojure to drive behaviour in Netlogo agents and thereby allow performance analysis to compare minimax with reactive behaviour.

Note: You may modify existing engines (see some on agent-domain.org for example) but you must have prior approval for this.

Paper 3- Language Processing

The requirement for this work is to produce a *technical paper* explaining how some aspect of semantically-based language processing can be achieved in the context of a system which aims to infer appropriate meanings for non-trivial sentences.

Paper 3 will normally relate to the work produced for paper 2, describing how a (constrained) natural language interface can be constructed for it. The additional practical work will *normally* integrate a language processing subsystem into the earlier work which will include elements of morphology, sentence structure transformation, parsing and semantic analysis.

Tutors will consider requests to develop the language interface as a stand-alone (non-integrated) module. Requests must be agreed at least 1 week before the pitch phase and will only be approved if the language requirement is such that it requires case frame and/or verb-form resolution due to the semantic complexity of the language input.

In rare circumstances (typically only where work for paper 2 has not been developed sufficiently or has exhibited other problems) tutors may proposed other language processing work.

The nature of this work will be discussed in detail in lectures and examples of the types of processing involved can be found in the sample SHRDLU system.

Assessment criteria

Each of the assessment tasks are different and have varying submission requirements (outlined in the specification of each task). All tasks require you to submit some kind of program code or specification, all require you to provide some discussion and/or critique.

deliverables

The general documentation for deliverables and success criteria (below) applies to this work. Pitches and reviews will be marked according to the description for these stages in the *marking-phases* document.

Papers will be marked according to the following guidelines...

authorship (20%)
- template used correctly
- statement of aims
- structure & narrative
- references
- wording & grammar

results (20%)
- style of presentation
- completeness
- clarity

conclusion & critique (20%)
- tied to aims & objectives
- evidenced
- appropriate
- wrap-up/summary

model (20%)

paperwork (20%)
- feedback & planning
- peer marking

The following table is intended to give you further guidance on the different submission requirements. Note that the table is not intended to provide a checklist and should be used to supplement the requirements given in the task specifications.

requirement	**Pass (40-59)**	**Good (60-69)**	**Excellent (70+)**
problem introduction	demonstrated understanding of main issues & objectives;	good understanding of the problem; appreciation of main issues & objectives;	full (but concise) explanation of the problem domain and the objectives of the task; clear insight into main challenges & compromises
code	code implements a reasonable solution; it is of acceptable quality & is adequately explained;	code meets all major requirements; it is of good quality & is well commented and/or explained; the code is possible to extend & maintain;	code meets all necessary requirements & performs consistently as intended; it is of good quality & elegantly specified; code is well commented and explained; code is easy to extend & maintain;
results	results adequate to highlight main behaviour; presentation reasonably clear;	results clearly show main behaviour; presentation is clean & adequate to support further discussion;	well presented results which show expected behaviour and highlight any other interesting features; presentation is well suited to further analysis;
discussion and/or critique	demonstrated undertanding of the problem and solution (see note below); some suggestions about alternative solutions;	a critical refection and/or evaluation of the problem and solution (see note below); suggestions for improvements and alternative solutions;	a thorough reflection on the nature of the problem and the validity and/or efficacy of the solution provided (see note below); sensible suggestions for improvements and alternative solutions;

Note: some of the tasks are *investigative*, ie: the purpose is to examine a particular aspect to a problem or solution strategy. This means that a well engineered solution may not always yield good results. A strong critical appraisal will take an unbiased view so may (correctly) report negatively on the suitability of the implemented strategy.