Department of Computer Science

Coursework Assessment Specification

Module Number: 08226 **Title:** **Artificial Intelligence**

Lecturer: MB

Coursework Assessment Number: 2 of 2

Title: Artificial Intelligence Assignment 2

Method of Working: INDIVIDUAL

Workload Guidance: Typically, you should expect to spend between 50 and 65 hours on this assessment

Date of publication: 9.00 am on Friday, 7th March, 2008

Date and time for submission: 9.30 am on Wednesday, 30th April, 2008

ONE copy of this assignment should be handed in via *Classroom Server Assignment Submission* no later than the time and date shown above, unless an extension has been authorised on a *Request for an Extension for an Assessment* form which is available from the Office or http://www.student-admin.hull.ac.uk/downloads/Mitcircs.doc. The extension form, once authorised by the lecturer concerned, should be attached to the assignment on submission (or given to the lecturer in the case of electronic submission).

BEFORE submission, each student must complete a departmental coursework cover sheet obtainable from the departmental student intranet (www-ug.dcs.hull.ac.uk or www-msc.dcs.hull.ac.uk). This assignment is being marked by *student name,* please ensure that you complete the correct cover sheet.

Notes:

Late penalties
For work submitted late the penalty is loss of **20% marks per day**. Work that is **5 or more days late** will automatically be graded as **FAIL**, and no re-submission will be allowed. For full details, see the Department's student handbook.

Use of Unfair Means
You are reminded of the University's Code of Practice on the Use of Unfair Means (http://www.student-admin.hull.ac.uk/downloads/code.doc) and that the work you submit for assessment should contain no section copied in whole or in part from any other source unless where explicitly acknowledged by means of proper citation.

Format of the Coursework Assessment Submission:
A *zip file* containing the deliverables outlined in the Coursework Assessment specification. The assignment will be marked by demonstrating submitted work and feedback given there and then.

ASSESSMENT STRATEGY AND ASSESSED COURSEWORK

This assignment is worth **25** % of the module marks.

The overall assessment strategy is designed to evaluate the student's achievement of the module learning outcomes, and is subdivided as follows:

ASSESSMENT STRATEGY AND LEARNING OUTCOMES

This assignment contributes towards the following module learning outcomes.

LO	Learning Outcome	Method of Assessment
1	*Conceptuualise simple cognitive tasks and processes in formal reasoning systems*	*Implementation.*
2	*Devise an AI based solution strategy for a formalized task or process.*	*Implementation*
3	*Design and write programs in a symbolic programming language*	*Implementation*

ASSESSMENT CRITERIA

Assessment Criteria	Contributes to Learning Outcome	Mark
Working commented code as in the detailed instructions contained in the body of the assignment	**1,23**	**100%**

Assignment details

Preparation:

- Chapter 9 of Rich, E. and Knight, *Artificial Intelligence*, MacGraw Hill, Second Edition, 1991 gives a selection of good preparatory material about frames and other structure in knowledge representation. ISBN:0070522634
- Chapter 9 & 10 of P.H. Winston, *Artificial Intelligence*, 3rd Edition, Addison-Wesley, 1993. ISBN 0201533774 this in another important introduction to the topic.
- Minsky, M. *A Framework for Representing Knowledge*, in P.H. Winston (ed.) The Psychology of Computer Vision, MacGraw-Hill:New York, 1975**.** ISBN-13: 978-0070710481 – this is the original presentation of the idea and is a much quoted and seminal paper.

ACW: 08226 Artificial Intelligence (Part 2).

In this assignment you are to design a frame system to implement a traditional knowledge representation system. This is a regular knowledge engineering technique in GOFAI.

A frame is a traditional way of storing related information about items in a knowledge base. Information can either be stored associated with a particular frame or it can be inherited from other frames. Frames are linked together in a class hierarchy (see lecture notes for more specific detail).

The following is the example we are going to use in this ACW. It denotes the class of *bands* (we'll treat solo artists as bands for the sake of this exercise). This class is subdivided into the two subclasses of *electric* and *acoustic* bands. The *electric* class is subdivided into two subclasses, namely *blues* and *punk*. Subdivisons of *blues* are *chicago* and *delta* with specific instances being (for *chicago)* muddy and elmore and (for delta) Robert. Thus in this ontology the classes are *electric, blues, chicago, delta, punk, british, new_york, acoustic, rock, soft, hard, folk, trad, commercial*, and the instances *muddy, elmore, robert, clash, ramones, nora, ry, richard, john, euan, eliza,* and *roger.*

band

- electric
  - blues
    - chicago
      - muddy
      - elmore
    - delta
      - robert
  - punk
    - british
      - theclash
    - new_york
      - ramones
- acoustic
  - rock
    - soft
      - nora
      - ry
    - hard
      - john
      - richard
  - folk
    - trad
      - ewan
      - eliza

commercial
roger

Each of the instances should have a particular property for example a song theclash – londoncalling or eliza – 'Worcester City'. (So a slot filler can either be an atom, term, or string).

The basic form of the data will be
frame(<name>,<type>,<list of contents>).

thus we have can have
frame(ewan,instance_of(trad),[songs(['Dirty Old Town'])]).
frame(folk,subclass_of(accoustic),[instruments([violin, fiddle, viola, melodeon, piano, tenor guitar])

1. Describe as frames, in the manner illustrated above, the taxonomy of bands. For each class or instance you must specify at least one specific slot for that frame. In the case of an artist (an instance) you can always quote a song. For a class again a property needs to be identified (e.g. blues is popular in us and uk, punk popular 1970s – note this is not a number). You will be marked on reasonable elaborations of the basic specification.
[10 marks]

2. Write a predicate call **fadd/2. fadd** should take the name of an object and the item to add. It will then need to modify the knowledge base to update the named frame (or produce an error message if no frame of this name exists) with the added detail. Finally it should list out the changed datastructure.
?- fadd(theclash,hero,joe_strummers).
Should print out
theclash frame has been added to. The frame now is:
frame(theclash,subclass_of(british),[hair(blue),[songs([londoncalling]),hero(joe_strummers)]).
yes [10 marks]

3. Write a predicate called **fchange/3** This predicate takes as its first argument a name of a frame and then as its second an item to change in the body of the frame and the third item its new value. If that item is not there to change then the frame should not be altered and an error message given back to the user. If the item is found then it should be changed accordingly and finally it should list out this changed data structure.
?- fchange(theclash,hero,mick_jones).
Should print out
theclash frame has been changed. The frame now is:
frame(theclash,subclass_of(british),[hair(blue),songs([londoncalling]),hero(mick_jones)]).
[10 marks]

4. Write a predicate called **fremove/2** which should remove an item from the body of the frame, or give an error message if this item is not found. Once again the modified datastructure should be printed, so
?- fremove(theclash,hero).
Should print out
theclash frame has had an item removed. The frame now is:
frame(theclash,subclass_of(british),[hair(blue),[songs([londoncalling])]]).

yes [10 marks]

5. Write a predicate **fdelete/1** that removes a leaf node only. It should refuse to delete a class. It should work as follows:
?- dfelete(nora).
The instance of nora has now been deleted. [10 marks]

6. Inheritance and loop detection.
An important part of frames is inheritance so we know that theclash has songs because theclash is an instance of a british, which is a subclass of punk, which is a subclass of a band and within the body of the band frame is the fact has(songs). Implement a predicate **fask** which asks if something is true about an instance or class object and if necessary performs inheritance to check super classes. Extra marks will be given if your fask checks that you haven't already checked this class due circular definitions in your knowledge base. You will do this via a loop detector which checks that each act of inheritance is unique and that steps are not being repeated. To do this you will need to record the chain of inheritance as you build it up and that your next step is not already on this list. Thus when I type
?- fask(theclash,has(songs)).
yes
Extra marks are available if you also write out the inheritance path. This you can get from your inheritance recording mechanism you developed above.
?- fask(theclash,has(songs)).
theclash instance_of british subclass_of punk subclass_of band which has(songs)
[30 marks]
7. Expand your system to include type checking. This will allow us to introduce a new keyword type/2 which should be either a Prolog type or a type of your own defining e.g
colour(red).
colour(blue).
colour(gold).
so my definition of tweetie would be
frame(theclash,instance_of(british)),[hair(blue),type(hair,colour))]).

Now if I change this attribute, because it has the **type** extra part we need to check that this item is actually a colour. Thus if I change the colour of theclash's hair to red I should get
?- fchange(theclash,hair,red).
Should print out

theclash frame has been changed. The frame now is:
frame(theclash,subclass_of(british),[hair(red),type(hair,colour))]).
but
?- fchange(theclash,hair,northampton).
Error: northampton is not a known colour. frame has NOT been updated. [20 marks]