Joint Institute SCNU - University of Aberdeen
Knowledge Representation (JC4002)
Group Assessment

The assessment is worth 25% of the overall marks for the course. Each item indicates the number of marks it is worth, clearly broken down in their specification. Students will provide their answers via MyAberdeen and receive feedback via MyAberdeen.

Learning outcomes associated with this assessment:

- Students will be able to identify the knowledge base (or lack thereof) of an AI application.
- Students will be able to evaluate the use of knowledge base in real-world scenarios.
- Students will be able to design, implement and evaluate knowledge-intensive AI systems.

Instructions to students:

- Your solution should be **one single ZIP file** containing your report, transcript, and all code files (Prolog files, PDDL domain file, PDDL problem files), that you should upload to MyAberdeen by the established time/deadline. **Do not email us your solution.**
- **Missing any of the files** will give you **0 mark**.
- This is a **group (of three)** assignment. If you share your solution with other groups or if you copy an existing solution, both groups will be investigated for collusion and plagiarism; the investigation can result in both groups getting 0 marks or being expelled from the course.
- Your zip file should be named "JC4002-[PROGRAMME]-Group-[GROUP_NO].zip". For instance, "JC4002-CS-Group-1.zip"[1], where PROGRAMME is your study programme [CS/AI/BMIS], and GROUP_NO is your group number.

About this assessment: You must **never share it** with anybody in or outside the course, even after you complete the course. *Please do not distribute or post solutions to any of the projects and notebooks.*

Academic Dishonesty: This is an advanced course, therefore we expect full professionalism and ethical conduct. You must work on this project **as a group**. You are free to discuss high-level design issues with other groups in your class, but every aspect of your actual formalisation/code/answer must be entirely your group own work. Furthermore, there can be no textual similarities in the reports generated by each group. Plagiarism, no matter the degree, will result in forfeiture of the entire grade of this assessment. Plagiarism is a serious issue and we take academic misconduct very seriously. Sophisticated *plagiarism detection* software will be used to check your code against other submissions in the class as well as resources available on the web for logical redundancy. Please **do not let us down and risk our trust**. If you do, we will pursue the strongest consequences available to us according to the guidelines provided by the university. For more information, see the Code of Practice on Student Discipline. **Use of automated services to generate submissions for assessment will be treated as academic misconduct and pursued under the University misconduct procedures.**

Late Submissions: You should apply for an extension by emailing your request (using your UoA email address) to uoa-ji-enquiries@abdn.ac.uk. You should also include any supporting evidence where possible e.g medical letter. Please familiarise yourself with the University's guidance on late submission.

[1] Do not put the quotation marks in your filename

Text-based Adventure in Prolog

1. Your assessment is to write a text adventure game in Prolog[2]. You have to use the SWI-Prolog interpreter for this assessment.

   You can copy the file "adventure-startercode.pl" (available on myAberdeen along with this PDF) and use it as a starting point. In this sample game there is one room, one object, and one direction you can go (north, but going in that direction takes you back to the same room). You can use or modify this code in any way you like to create your own game. Submitting the same file without any significant changes will merit 0 marks. You do not need to use this file as a starting point, but you must follow these requirements:

   (a) (10 marks) **Knowledge base facts** – Does your knowledge base include all the necessary facts to play the game? Are there any facts that are never being used in the game? Do you have dynamic facts and are they being managed correctly?

   (b) (20 marks) **Knowledge base rules** – Do you have rules for winning the game? And for losing the game? Do you have rules to manage the player inventory? Are you introducing randomness / uncertainty in the adversaries' decisions, environment or the game itself? Are you using arithmetic functions to manipulate your knowledge and the rules of the game?

PDDL-Driven Adversaries

2. In addition to the Prolog game, you will model (at least one) a simple intelligent adversary using PDDL and invoke an external planner from Prolog to control that adversary's behaviour. It is expected that the adversary(ies) has one default plan to be executed by your Prolog program. From the "adventure-startercode.pl", adversaries (for example) can be a *chaser* to chase and capture the player, or a *thief* to obtain the treasure before the player. You are strongly suggested to use the Pyperplan (STRIPS planner) and keep your PDDL simple (STRIPS level). If you decide to use different tools and more complex PDDL (e.g. temporal actions, numeric fluents, or conditional effects), specify them in details (e.g. tool name, tool version, PDDL version, capabilities included in your game) in your report. Missing this information will merit 0 marks for this section.

   You can copy the file "pyperplan_runner.pl" (available on myAberdeen along with this PDF), as well as the "domain.1.pddl", and "problem.1.pddl", and use them as a starting point to run pyperplan from Prolog. You must follow these requirements:

   (a) (15 marks) **Plan Executions** – How is the plan executed in the game? Have you implemented action validation for the plan to ensure that the suggested action is legal in the current world before applying it? Is there any strategy to tackle illegal actions or "no solution" problem?

   (b) (20 marks) **Plan Dynamics** – Does the plan evolve following the evolution of the game? Does the adversaries change their plan to ensure the player lose (e.g. starting as a thief and changing to a chaser once the player obtains the treasure)? Do you involve multiple adversaries? Do the adversaries work together?

Presenting Your Game

Along with the above requirements, you must fulfil the following requirements to achieve the highest mark:

(a) (5 marks) **Transcript** – See submission instructions for details.

(b) (15 marks) **Report** – See submission instructions for details.

[2] This is inspired by Prolog coursework from David Matuszek.

(c) (15 marks) **Oral presentation** – You will get a 10 minute slot during one of the last four sessions (i.e. the last two class and practical sessions). The allocation is random and you will not be told beforehand which session you are in, therefore, you have to attend both sessions. You will be evaluated on how well you make use of the allotted time, how well you explain the motivation behind your game, how well you explain the sophistication of your game's adversary, and how well you demonstrate a winning and losing run of your game. We may ask questions at any time during your presentation and how well you answer will also be evaluated.

Marking scheme: Your program should work, should satisfy the above requirements, and should be reasonably formatted. At most 35 marks will be awarded if the program does not run, provided that the group have presented orally.

You should have a **start/0 predicate** (similar to the one provided in the starting code) that we can use to start your game and print out what commands the player can use. Submissions that do not inform the player the commands that can be used at the start of the game will receive mark penalties.

We reserve the right for some subjective judgement. If we like your game, we may give you bonus points for it. If your game seems to lack creativity and interest, you may lose points.

Submission instructions: The following files should be included in your ZIP file (ZIP only, not RAR, 7z, or anything else):

- game_name.pl – source code. The game_name should be a short version of the name of your game.
- adversary_domain.pddl – domain PDDL file.
- adversary_problem.pddl (or several of it in the case of multiple adversaries) – problem PDDL file. If your game provides dynamic problem.pddl files (i.e. rewritten as the game evolves), you should provide the default problem.pddl file that is generated when the game starts.
- transcript.txt – a transcript of a sample run of your program. This is a walkthrough of the game, showing the commands entered and the computer's response to each command. You can get this by playing the game and copy/pasting the results. You have to show a successful run of your program (i.e., beating the game).
- report.pdf – game report containing:
  - the name of your game;
  - one paragraph describing what your game is about;
  - an explanation of your knowledge base and what was your reasoning for making the choices you did;
  - an inspiration game that you use (if you use any) and an explanation what differs between your game and your inspiration game from the gameplay perspective.
  - a list of technical details implemented in the game, including a brief description of what they are, and how they are implemented (e.g. complex inventory systems, random starting points, collaborative adversaries, evolving plans).
  - there is no specific template that you must follow, but here are the things we expect out of your report (if any of these instructions are not followed, you will get mark penalties):
    * at most 1500 words (add your word counter in the end of your report, references, figures, tables, and captions do not count);
    * text alignment justified;
    * no abstract;
    * section headers (numbers are optional but should be consistent if present);
    * We expect to see at least a figure and/or table to help explain your game;

  - references should be formatted following the ACM style (if you used any, but remember if you took inspiration from something and you do not cite it that is plagiarism).

Hints:

- There are many text adventure games available online (many even in Prolog) to draw inspiration from. Just be careful with plagiarism, only use them for inspiration, do not copy any of the code (remember you will have to explain your game and your code in your report).
- You could push this beyond the scope of the assessment by adding an interface, web server, learning component, etc. There are many tutorials for using these things in Prolog available in SWI-Prolog website and on internet forums. This is not required, but you will likely get bonus marks if you manage to pull it off successfully.
- Your game may have several win and loss conditions, just be careful they are not contradictory.
- The cut operator is your friend to make nice print statements and to make the flow of the game more natural.