Preparation resources

- Past exams: http://robotics.itee.uq.edu.au/~ai/doku.php/wiki/resources
- Tutorials: https://learn.uq.edu.au/webapps/blackboard/content/listContent.jsp?course_id=_124076_1&content_id=_4679265_1
- I should make sure to practice DPLL **lots** (see UQ theme park question from tutorial 6)

Known facts

- There **will** be a question on logic rules including De Morgan's laws and conversion to CNF.
- There **will** be a question on decision theory including Bayes' rule.
- There will **most likely** be a question on expected value of sampling information using Bayes' rule akin to tutorial 7 questions. Make sure you master tutorial 7.
- There will **most likely** be a question on using offline solvers with MDPs (i.e. value & policy iteration).
- There will be **no** continuous search or motion planning questions! 😁
- There is **no** need to memorise the DPLL algorithm -- only need to know how to use it.
- There is **no** need to study the utility of money.
- There is **no** need to learn how to compute solutions to MDPs **online** (i.e. RTDP or monte-carlo tree search)
- **There is no need to learn reinforcement learning!** 😁
- The exam has a total of 120 marks over 6 questions.
- The exam will have a total time of 120 minutes.
- The exam is closed book.
- The exam will be fairly similar to other previous exams. 😁

Rational agents

Q: What are the components that define a rational agent?

A:

- State space, S
- Action space, A
- Percept space, P
- Transition function, $T : S \times A \to S$
- Percept function, $Z : P \to S$
- Utility function, $U : S \to \mathbb{R}$

Q: Describe the difference between deterministic and non-deterministic.

A: Deterministic: the transition function is well-defined.
Non-deterministic: the transition function is ill-defined or does not exist.

Q: Describe the difference between fully observable and partially observable.

A: Fully observable: the percept space equals the state space ($P = S$).
Partially observable: the percept space is different to the state space.

Q: Describe the difference between static and dynamic.

A: Static: the environment does not change between the agent's actions.
Dynamic: the environment can change between the agent's actions.

Blind search

Q: What is the definition of blind search?

A: A search for a solution in a space where we do not have any additional information to base a "guess" on the cost of moving to the goal. In other words, there is no heuristic.

Q: Describe the general algorithm for graph search where the container type is unknown.

A:

- Let I be a node holding the initial state of the problem
- Add I to the container
- Loop
    - If the queue is empty, return failure
    - Remove a node from the container and call it t
    - Mark t as "explored"
    - If t holds the goal state, return success
    - For each node v that is a successor of t:
        - If v is not "explored" or in the container:
            - Add v to the container

Q: What kind of container does Breadth First Search use? Is it FIFO or FILO? What does the abbreviation stand for?

A: Queue, FIFO, first in first out.

Q: Describe the space and time complexity of BFS.

A:

- Time: $O(b^d)$ where
    - b is the branching factor and
    - d is the depth of the shallowest goal node
- Space: also $O(b^d)$

Q: Is BFS complete? Is it optimal?

A: Complete if the branching factor is finite. Optimal.

Q: What kind of container does depth first search use? Is it FIFO or FILO? What does the abbreviation stand for?

A: Stack, FILO, first in last out.

Q: Describe the space and time complexity of DFS.

A:

- Time: $O(b^m)$ where
  - b is the branching factor and
  - m is the maximum depth
- Space: $O(bm)$. Also acceptable: $O(m)$

Q: Is DFS complete? Is it optimal?

A: Complete if the branching factor is finite and there are no loops. Not optimal.

Q: Describe the general process for iterative deepening depth first search.

A: Repeatedly run DFS on deeper and deeper subtrees until the solution is found.

Q: Describe the space and time complexity of IDDFS.

A:

- Time: $O(b^d)$ where
  - b is the branching factor and
  - d is the depth of shallowest goal node
- Space: $O(bd)$

Q: Is IDDFS complete? Is it optimal?

A: Complete if the branching factor is finite. Optimal.

Q: What kind of container does UCS use? How are nodes in the container removed?

A: Priority queue, nodes with a lower value (i.e. lesser numbers) are removed first.

Q: What priority does each node get in UCS?

A: The number of steps away it is from the initial node.

Q: Describe the space and time complexity of UCS.

A:

Time and space:

$$O\left(b^{1+\text{floor}(C^*/\epsilon)}\right)$$

where

- b is the branching factor,
- C^* is the cost of the optimal solution,
- ϵ is the minimum cost of a step

Q: Is UCS complete? Is it optimal?.

A: Complete if the branching factor is finite and all edges have a positive cost. Optimal if all edges have a positive cost.

Informed search

Q: What is the definition of informed search?

A: A search for a solution in a space where we use additional information to base a "guess" on the cost of moving to the goal. In other words, we use a heuristic.

Q: What is the difference between UCS and Greedy Best First search?

A: The priority of a node is the value of the heuristic function at that node.
Also acceptable: the cost from the initial node is ignored.
Also acceptable: the priority of a node n is $h(n)$.

Q: What is the difference between Greedy Best First search and A*?

A: The priority of a node is the sum of the cost from the initial node plus the estimated cost to the goal node.
Also acceptable: the priority of a node n is $g(n) + h(n)$.

Q: What is the definition of an **admissible** heuristic?

A: A heuristic is admissible if it never overestimates the cost.

Q: What is the definition of a **consistent** heuristic?

A: The estimated cost of moving from node A to the goal is never more than the true cost of moving from A to B plus the estimated cost of moving from B to the goal. In other words:

$$h(A) \leq c(\overline{AB}) + h(B)$$

Q: What is the implication between admissibility and consistency of a heuristic?

A: Consistency implies admissibility but the converse is not true.

Logic

Q: What does it mean for a sentence to be **valid**?

A: The sentence is true for all possible interpretations. Also acceptable: the sentence is a logical tautology.

Q: What does it mean for a sentence to be **satisfiable**?

A: The sentence is true for at least one interpretation.

Q: What does it mean for a sentence to be **unsatisfiable**?

A: The sentence is false for all possible interpretations. Also acceptable: the sentence is a logical contradiction.

Q: What is the model of a sentence?

A: An interpretation that makes the sentence true.

Q: What does it mean for a sentence A to **entail** another sentence B? How is it denoted?

A: Either is acceptable:

- Every model of A is also a model of B.
- $A \to B$ is valid/always true.

It is denoted $A \models B$.

Q: What does it mean for an algorithm to be sound?

A: If it produces a result, the result is correct.

Q: What does it mean for an algorithm to be complete?

A: It will always produce a result.

Q: List all logical equivalence laws and their definitions.

A:

A	B	**Name**
$a \land b$	$b \land a$	Commutativity of $\land$
$a \lor b$	$b \lor a$	Commutativity of $\lor$
$(a \land b) \land c$	$a \land (b \land c)$	Associativity of $\land$
$(a \lor b) \lor c$	$a \lor (b \lor c)$	Associativity of $\lor$
$\neg(\neg a)$	a	Double-negation elimination
$a \to b$	$\neg b \to \neg a$	Contrapositive

A	B	**Name**
$a \rightarrow b$	$\neg a \vee b$	Implication elimination
$a \leftrightarrow b$	$(a \rightarrow b) \wedge (b \rightarrow a)$	Biconditional elimination
$\neg(a \wedge b)$	$\neg a \vee \neg b$	De Morgan's law
$\neg(a \vee b)$	$\neg a \wedge \neg b$	De Morgan's law
$a \wedge (b \vee c)$	$(a \wedge b) \vee (a \wedge c)$	Distributivity of $\wedge$ over $\vee$
$a \vee (b \wedge c)$	$(a \vee b) \wedge (a \vee c)$	Distributivity of $\vee$ over $\wedge$

Q: What is modus ponens? What does it translate to? What is your favourite example? How is it written with atoms a and b?

A:

- Suppose a implies b and a is true. Then b must be true.
- Mode that affirms.
- The sky is blue implies the water is blue. The sky is blue. Therefore the water is blue.
- $$\frac{\begin{array}{c} a \rightarrow b \\ a \end{array}}{a}$$

Q: What is modus tollens? What does it translate to? What is your favourite example?

A:

- Suppose a implies b and b is false. Then a must also be false.
- Mode that denies.
- The sky is blue implies the water is blue. The water is red. Therefore the sky cannot be blue.
- $$\frac{\begin{array}{c} a \rightarrow b \\ \neg b \end{array}}{\neg a}$$

Q: What is conjunction elimination? How is it written with atoms a and b?

A:

- Suppose a and b is true. Then a is true.
- $$\frac{a \wedge b}{a}$$

Q: How resolution is it written with atoms a, b, and c?

A:

$$\frac{\begin{array}{c} a \lor b \\ \neg a \lor c \end{array}}{b \lor c}$$

Q: What is model checking?

A: A way of determining whether a sentence is true for all possible interpretations.

Q: Was is the simplest approach to model checking?

A: Enumerate the models, i.e. all true/false combinations for the sentence and checking if the sentence is true in each case.

Q: What is a better approach to model checking? How does it work?

A: Theorem proving. Use logical equivalence laws and inference rules to make legal steps from the initial statement to the goal statement.

Q: What is a logical atom?

A: A single symbol with no logical connectives.

Q: What is a logical literal?

A: An atom or the negation of an atom.

Q: What is a logical clause?

A: A disjunction of literals.

Q: What does it mean to convert a sentence to conjunctive normal form?

A: We convert the sentence into a (either of these is acceptable):

- conjunction of disjunctions of atoms or the negation of atoms
- conjunction of disjunctions of literals
- conjunction of clauses

Q: What are the only logical connectives in a sentence that is in CNF?

A: And ($\land$), **or** ($\lor$), and **not** ($\neg$).

Q: Describe the steps to convert a sentence to CNF.

A:

1. Eliminate arrows using definitions

2. Drive in negations using De Morgan's laws
3. Distribute **or** ($\vee$) over **and** ($\wedge$)

Q: Describe the steps to the resolution refutation algorithm

A:

1. Convert all sentences to CNF
2. Negate the desired conclusion
3. Keep applying resolution until a contradiction arises

Q: What is a satisfiability problem or a constraint satisfaction problem?

A: A question of assigning values to each variable in a problem such that all the constraints are satisfied.

Q: Is the Davis-Putnam-Logeman-Loveland algorithm sound and complete?

A: Yes to both.

Q: Alternative to DPLL, name another method to solve satisfiability problems? Describe its steps. Is it sound and complete?

A: GSAT.

- Loop n times:
  - Randomly an assignment for all variables
  - Loop m times:
    - Flip the variable that results in the lowest number of unsatisfied clauses
    - If there are no unsatisfied clauses, return the assignment chosen

It is sound but not complete.

Q: What is a unit clause?

A: A clause that consists of only one unassigned literal.

Q: What is a pure variable within a sentence?

A: A variable which appears in the sentence as (either of these is acceptable):

- only ever positive or only ever negative
- always atomic or always negated

Decision theory

Q: What does the branching at each interleaving level of an AND-OR tree represent?

A: Branching at an AND level represents the agent's choices of action while branching at an OR level represents actions taken by the environment.

Q: What are the two types of nodes in an AND-OR tree? What level are these types found at?

A:

- State node found at each OR level
- Action node found at each AND level

Q: Describe the steps required to find a solution to an AND-OR tree.

A:

- Mark all the leaf nodes as "solved" if their state is a goal state
- Working up the tree:
  - label action nodes as "solved" if **all** its children are "solved"
  - label state nodes as "solved" if **at least one of** its children is "solved"
- If you reach the root, the solution is the sub-tree from the root where all nodes are "solved"

Q: What does the branching at each interleaving level of a minimax tree represent?

A: Branching at a MAX level represents the agent's choices of action while branching at a MIN level represents actions taken by the opponent.

Q: What is the thing each player of a minimax game are trying to minimise or maximise? What does it represent?

A: An evaluation function (also acceptable: a heuristic). Represents an estimate as to how favourable a game state is for the agent.

Q: Describe the steps of the minimax algorithm.

A:

- Compute the evaluation function at every leaf of the tree
- Working up the tree:
  - label MAX nodes as the maximum evaluation of its immediate children (i.e. successors)
  - label MIN nodes as the minimum evaluation of its immediate children

Q: Describe the steps of the alpha-beta pruning algorithm.

A:

- Maintain an upper and lower bound of the evaluation function at each node.
- Let α be the best already explored option along the path to the root for the **maximiser**
- Let β be the best already explored option along the path to the root for the **minimiser**
- Explore the game tree to depth h in a depth-first manner
- Back up α and β values wherever possible

- Prune branches that can't lead to changing the final decision

Q: What kind of environment is decision theory useful in?

A: A non-deterministic one.

Q: What are the components of a decision theory problem?

A:

- A set of **states** (also acceptable: **outcomes**)
- **Preference**: which state/outcome is preferred. Also acceptable: a **utility function**.
- **Lotteries**: a set of states/possible outcomes with their probability of occurring

Q: What is the formula for the expected utility given a decision theory problem with states A, B, and C?

A: (Probability of A occurring) $\times$ (utility of A) $+$ (probability of B occurring) $\times$ (utility of B) $+$ (probability of C occurring) $\times$ (utility of C), i.e.

$$P(A) \cdot U(A) + P(B) \cdot U(B) + P(C) \cdot U(C)$$

Q: How can we solve problems using the maximum expected utility?

A: Calculate the expected utility for each decision outcome. Make the decision that results in the highest expected utility (also acceptable: make the decision whose outcome maximises the expected utility).

Q: What does the branching at each interleaving level of a decision tree represent?

A: Branching at a **choice** level represents the agent's decisions while branching at a **chance** level represents the lottery that corresponds to the outcome of the decision made at its parent's node.

Q: What is Bayes' rule in a problem with outcomes A and B?

A:

$$P(B \mid A) = \frac{P(A \mid B) \cdot P(B)}{P(A)}$$

Q: When should Bayes' rule be applied in a problem with outcomes A and B?

A: When it is easier to know the probability of A given B rather than B given A (or vice-versa).

Q: MDPs model a problem where the non-determinism is what's known as **1st order Markovian**. What does that mean? And how would you represent that in probability notation?

A: The future state is only dependent on the current state and the action and does not take into account all the previous/past states.

$$P(s_{t+1} \mid s_t, a_t) = P(s_{t+1} \mid s_t, a_t, s_{t-1}, a_{t-1}, \cdots s_1, a_1, s_0)$$

Q: How do we define an MDP problem and how are the components denoted?

A:

- State space, S
- Action space, A
- Transition function, $T : S \times A \times S \to [0, 1] \subset \mathbb{R}$
- Reward function, $R : S \to \mathbb{R}$
  Also acceptable: $R : S \times A \to \mathbb{R}$
  Also acceptable: $R : S \times A \times S \to \mathbb{R}$

Q: What is the relationship between a transition function and the probability distribution in an MDP problem?

A: $T(s, a, s') = P(s' \mid s, a)$. Also acceptable: the probability of transitioning to state s' from state s via action a.

Q: Define the solution of an MDP problem and what it represents.

A: An **optimal policy** $\pi^* : S \to A$ which represents a strategy of choosing actions in states that will reap the greatest long-term reward when executed.

Q: In an infinite horizon MDP solver, what is the role of the discount factor? How is it denoted and what is its range?

A: The discount factor determines how much the agent values immediate reward versus long-term reward. Smaller discount equates to a more greedy agent. It is denoted γ (gamma) and is in the range $[0, 1]$.

Q: What is the Bellman equation for calculating the value of a state assuming the reward function is only dependent on the **current state**?

A:

$$V(s) = R(s) + \max_{a \in A} \left(\gamma \sum_{s' \in S} T(s, a, s') \cdot V(s') \right)$$

Q: What is the Bellman equation for calculating the value of a state assuming the reward function is only dependent on the **current state and the action**?

A:

$$V(s) = \max_{a \in A} \left(R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') \cdot V(s') \right)$$

Q: What is the Bellman equation for calculating the value of a state assuming the reward function is dependent on the **current state, the action, and the next state**?

A:

$$V(s) = \max_{a \in A} \left(\sum_{s' \in S} T(s, a, s') \left(R(s, a, s') + \gamma \cdot V(s') \right) \right)$$

Q: What is the difference between calculating the value and calculating the best action for a policy?

A: Instead of $\max$ use argmax in the calculation to return the best action.

Q: Describe the steps of the value iteration algorithm.

A:

- Initialise $V(s) := R(s)$ for all $s \in S$.
- Loop until $V(s)$ converges:
  - For all $s \in S$:
    - Update $V(s)$ with Bellman update

Q: How do we define an POMDP problem and how are the components denoted?

A:

- State space, S
- Action space, A
- Observation space, O
- Transition function, $T : S \times A \times S \to [0, 1] \subset \mathbb{R}$
- Observation function $Z : S \to O$
- Reward function, $R : S \to \mathbb{R}$
  Also acceptable: $R : S \times A \to \mathbb{R}$
  Also acceptable: $R : S \times A \times S \to \mathbb{R}$