COMP3702: 2016 Summary

[**UQAttic**](http://uqattic.net)

**Get more out of your study time. Join UQAttic's revision chat.**

[**Other exam papers**](https://drive.google.com/drive/folders/0Bzh9QxUQUKpDZlIxNlRQdXhEYUU)

Please **contribute** to these documents.

If you're looking for an effective way to familiarise yourself with the course material, you can't go past collaborating with fellow students. We have laboured to put these up, and so at the very least point out where you think we are wrong!

You'll get more out of the course, you'll do better in the exam, and other students will benefit from your input as well.

To get editing permissions, simply go to the [chatroom](http://uqattic.net) and provide us with your Google Account address.

**Style.**

Type answers in blue beneath each question.

If you're unsure of your answer, highlight your answer text then hit Ctrl+Alt+M to create a comment beside the text. Once you're satisfied with the answer, click the "Resolve" button on the comment.

If you want some extra explanation from someone else on their answer, highlight the other person's answer and repeat the procedure above.

**Communicate.**

Head over to [uqattic.net](http://uqattic.net/) and click "Chat Now!". You'll find a chatroom full of students just like you. Talk about a revision document (like this one) or swap prep tips. If you have your own IRC client, point it to irc.uqattic.net, port 6667, channel #attic.

# Table of Contents

[**Table of Contents**](#_j9sdbh4e9c3l) **1**

[**Agent**](#_a3un9pnqeecn) **3**

[Agent Components](#_5ix9fypg3epa) 3

[A: Action Space](#_is48hjx2o42w) 3

[O: Percept Space](#_k02zitpvq0i8) 3

[E: Environment](#_k9c86r8lswh7) 3

[T: World Dynamics / Transition Space](#_6pe5rl6o6xrw) 4

[Z: Perception Function](#_6htn12qvyygq) 4

[S: State Space](#_usyog87jpcgz) 4

[U: Utility Function](#_gvqdcgp86abj) 4

[Agent Problem](#_fahnyiwaq0bz) 4

[**Deterministic Search**](#_icmach2rvopy) **5**

[In Discrete Space](#_qe2tvv7wn03a) 5

[Properties](#_qybq15blwgfo) 5

[Uninformed (Blind) Search](#_sz2arsxeed8m) 5

[Breadth First Search](#_a9tv3vbqot8) 5

[Bi-directional Breadth First Search](#_j24fsoczi5gt) 5

[Depth First Search](#_620aeusavwr2) 6

[Iterative Deepening Depth First Search](#_uzbr7nkvu8w6) 6

[Uniform Cost Search (a.k.a. Dijkstra's Algorithm)](#_axlr2jqhk3ka) 6

[Informed Search](#_3vwm1lu748gk) 6

[A\* Search](#_wy2l79knr336) 6

[Admissable Heuristics](#_p9cbihe4dci3) 6

[Consistent Heuristics](#_9k1lmqy5lz5x) 6

[In Continuous Space](#_31kncksej2gl) 7

[**Logic, Representation, Validity and Satisfiability**](#_27lbjkx5li1b) **8**

[**AND-OR**](#_o5b3nu1ci8lu) **9**

[**Min-Max**](#_74xjf1xsx7nn) **10**

[**Utility Theory**](#_fb24yqm4bsek) **11**

[**Markov Decision Process**](#_y4xval12de3r) **12**

[**Partially observable Markov Decision Process**](#_szt2bp4udk0z) **13**

[**Decision Tree**](#_4d7se72d13u5) **14**

[**Bayes Net / Bayesian Networks**](#_cgiwef4e9rtf) **14**

[**Reinforcement Learning**](#_7wkmtgnzn1m3) **15**

# Agent

An agent is a computer program that:

* Gathers information about an environment
* Takes actions in that environment autonomously, based on that information

Course covers only rational agents; agents who make rational decisions

## Agent Components

An agent is built from the following components and exists in some World

### A: Action Space

The action space is the set of all actions the agent can perform, could be infinite.

The best action is the action which maximises a given performance metric or criteria, given by the Utility Function, defined below.

### O: Percept Space

The percept space is the set of all ‘things’ the agent can perceive in the World.

### E: Environment

The environment models the relationship between the agent and the World.

Includes the state space (S), defined below, as well as World Dynamics and a Perception Function.

An Environment has many classes:

* Discrete vs. Continuous
  + Discrete: State / Action / Percept spaces are discrete. E.g. board game
  + Continuous: State / Action / Percept spaces are continuous. E.g. movement in 3d space
* Deterministic vs. Stochastic / Non-deterministic
  + Deterministic: After performing an Action from a State the Agent always knows exactly what State it will end up in. World Dynamics are one-to-one.
  + Stochastic / Non-deterministic: After performing an Action from a State the agent can’t always be sure what State it will end up in.
* Fully Observable vs. Partially Observable
  + Fully Observable: The Agent knows the full state of the world and itself exactly. The percept function is a bijection.
  + Partially Observable: The Agent doesn’t know the full state of the world and/or itself exactly. The percept function is not a bijection.
* Static vs. Dynamic
  + Static: The environment as percepted is the same world actions are taken on (the world can’t change while the agent is ‘thinking’, e.g. board game)
  + Dynamic: The environment can change from how it was percepted during computation (e.g. self driving car)

### T: World Dynamics / Transition Space

A mapping of some starting State and applying some Action to some end State

### Z: Perception Function

The perception function is used when the environment is not fully observable.

### S: State Space

The set of all states of the agent and the environment. Only includes states which matter for the interaction between the agent (e.g. don’t include temperature if it has no effect, etc.)

### U: Utility Function

A function that takes a State (S) or a sequence of States and assigns it a value which indicates the desirability of being in that State or sequence of States with respect to the agent’s task.

## Agent Problem

An agent should find some mapping from sequences of percepts to some action it should take to maximise the utility function

# Deterministic Search

## In Discrete Space

### Properties

* Completeness: if a path exists, one will be found
* Optimality: if paths exist, the shortest / least cost path will be found
* Time / space complexity (Big O)

| Algorithm | Informed / Uninformed? | Complete? | Optimal? | Time Complexity | Space Complexity |
| --- | --- | --- | --- | --- | --- |
| Breadth First | Uninformed | Yes | Yes |  |  |
| Bi-directional Breadth First | Uninformed | Yes | No |  |  |
| Depth First | Uninformed | Yes | No |  |  |
| Iterative Deepening Depth First | Uninformed | Yes | Yes |  |  |
| Uniform Cost | Uninformed | Yes\* | Yes\* |  |  |
| A\* | Informed | Yes\* | Yes\* | Heuristic | Heuristic |

### Uninformed (Blind) Search

#### Breadth First Search

Search by selecting the next vertex to visit from a FIFO queue.

* Complete: yes (all vertices connected to the source are visited)
* Optimal: yes (vertices at lower 'depth' visited first, so the first time the goal is visited must be the shortest path)
* Time Complexity: O(bd), b = branching factor, d = depth of goal
* Space Complexity O(bd)

#### Bi-directional Breadth First Search

Perform BFS from the start vertex and from the goal vertex simultaneously, and 'meet in the middle'. Complexity is O(bd/2).

#### Depth First Search

Search by selecting the next vertex to visit from a FILO stack.

* Complete: yes (all vertices connected to the source are visited)
* Optimal: no (no guarantee that the first path found will be the shortest one)
* Time complexity: O(bm) b = branching factor, m = maximum depth
* Space complexity: O(bm) (much better than BFS)

#### Iterative Deepening Depth First Search

Perform depth-limited DFS in a loop, increasing depth each time. Similar to DFS, but is optimal. Time O(bd) and space O(bd).

#### Uniform Cost Search (a.k.a. Dijkstra's Algorithm)

Search by selecting the next vertex to visit from a priority queue with the best-known distance from the source as they key.

* Complete: yes, if all edges have positive weight
* Optimal: yes, if all edges have positive weight (lowest cost paths are always searched first)
* Time/Space complexity: O(b1+floor(C\*/ε))

### Informed Search

#### A\* Search

Similar to Uniform Cost Search, but makes use of a heuristic (approximation of the distance from a node to the goal). The priority queue keys are the distance from the initial plus the heuristic.

* Complete: yes, if all edges have positive weight
* Optimal: yes, if all edges have positive weight *and* the heuristic is admissible
* Complexity: depends on heuristic

#### Admissible Heuristics

A heuristic is admissible if it never overestimates the actual cost to the goal. If this is true, then A\* search is optimal. If it is not, then it may not produce the optimal solution (but it may be faster).

#### Consistent Heuristics

A heuristic is consistent (or monotone) if when the search expands a node, the path to that node is optimal - i.e. it will never expand a node where a lower cost unexplored path exists. Equivalently, the heuristic is 'non-decreasing'. A consistent heuristic is always admissible, but the opposite isn't true.

If the heuristic is consistent, then the search is optimal without ever revisiting nodes.

## In Continuous Space

For search in continuous state / action space, the space needs to first be discretized, either with uniform grid discretisation or Probabilistic Road Map (PRM).

4 components of the problem:

Sampling

Configuation Checking

Edge Valiation

Connection Strategy

# Logic, Representation, Validity and Satisfiability

Valid - all premise true and conclusion is true. *Always true -> cannot be false*

Satisfiability - a single set of values that makes the statement true. *Can be true / satisfied -> satisfiable*

## Resolution Refutation (Proof by Contradiction)

To prove something is true, add the inverse to the KB (knowledge base). Now resolve items in the KB to try and find a contradiction. eg. ( A Λ ㄱA )

## Useful rules:

https://www.youtube.com/watch?v=B9HQBY45a-Y

# AND-OR Tree

TODO

# Min-Max

TODO

# Utility Theory

TODO

# Markov Decision Process

S - State

T - Transition  
A - Action  
R - Reward

TODO representation and solution

# Partially observable Markov Decision Process

TODO and its relation to MDP

# Decision Tree

<http://www.saedsayad.com/decision_tree.htm>

<http://www.slideshare.net/marinasantini1/lecture-4-decision-trees-2-entropy-information-gain-gain-ratio-55241087> - SLIDE19 has entropy calculations

Decision Tree

Entropy

Information Gain

# 

# Bayes Net / Bayesian Networks

Bayesian Theorem

# Reinforcement Learning

TODO Representation and methods up to SARSA

Q-learning

https://studywolf.wordpress.com/2012/11/25/reinforcement-learning-q-learning-and-exploration/

SARSA

https://studywolf.wordpress.com/2013/07/01/reinforcement-learning-sarsa-vs-q-learning/