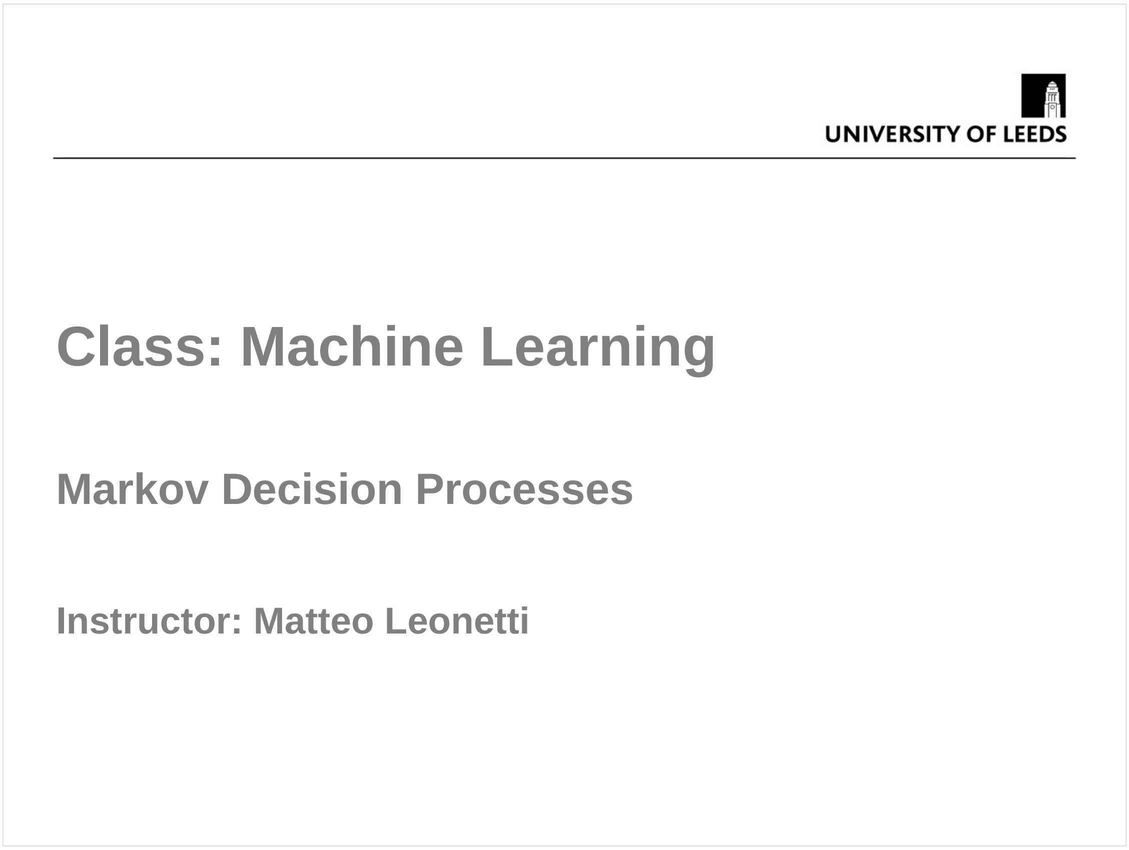
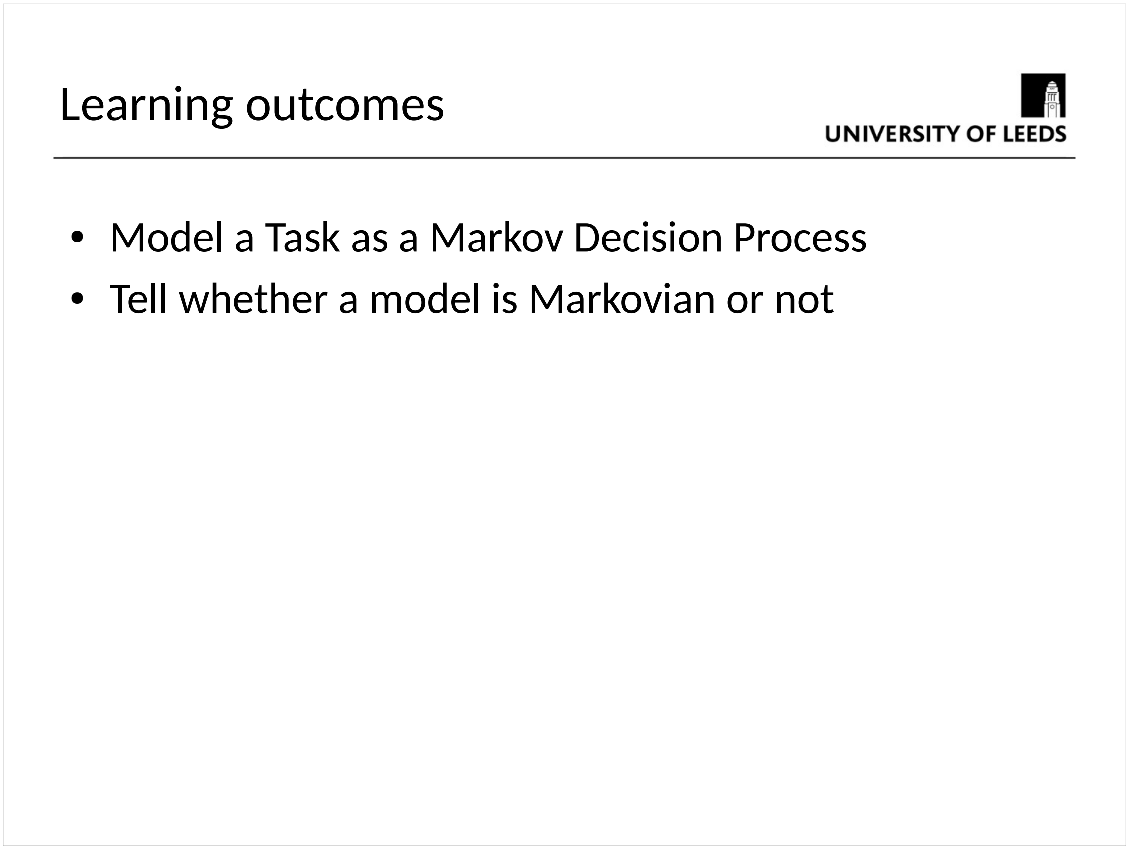


UNIVERSITY OF LEEDS

Class: Machine Learning

Markov Decision Processes

Instructor: Matteo Leonetti

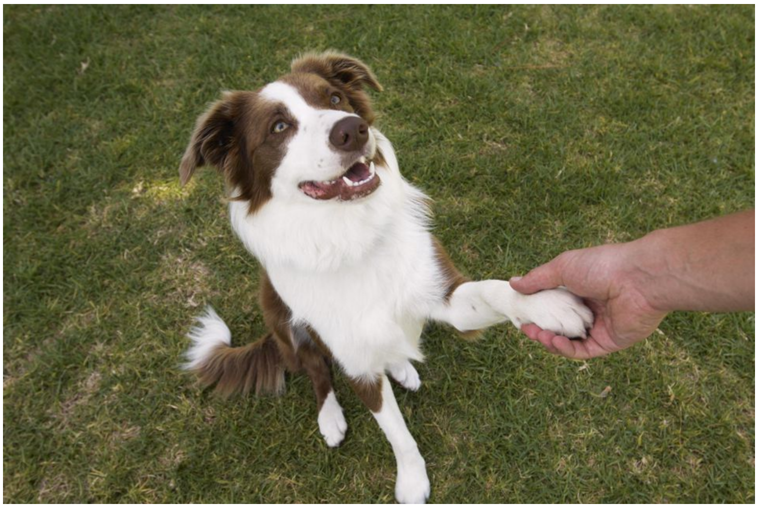


Learning outcomes

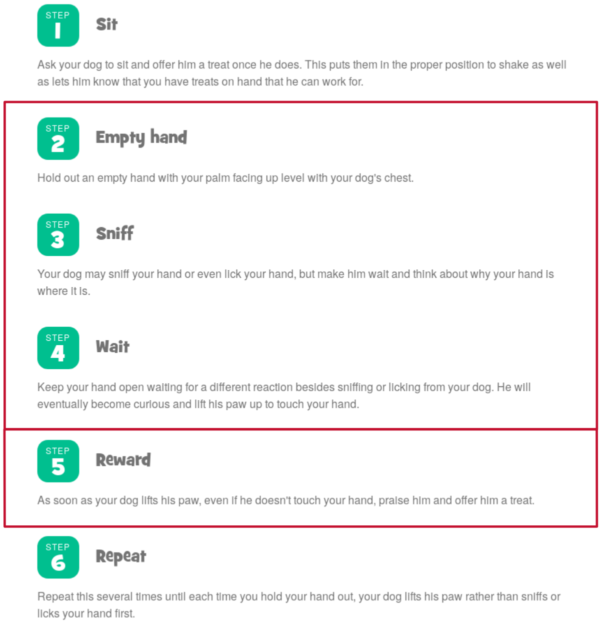
- Model a Task as a Markov Decision Process
- Tell whether a model is Markovian or not

Reinforcement Learning

UNIVERSITY OF LEEDS



Reinforcement Learning



STEP 1 **Sit**

Ask your dog to sit and offer him a treat once he does. This puts them in the proper position to shake as well as lets him know that you have treats on hand that he can work for.

STEP 2 **Empty hand**

Hold out an empty hand with your palm facing up level with your dog's chest.

STEP 3 **Sniff**

Your dog may sniff your hand or even lick your hand, but make him wait and think about why your hand is where it is.

STEP 4 **Wait**

Keep your hand open waiting for a different reaction besides sniffing or licking from your dog. He will eventually become curious and lift his paw up to touch your hand.

STEP 5 **Reward**

As soon as your dog lifts his paw, even if he doesn't touch your hand, praise him and offer him a treat.

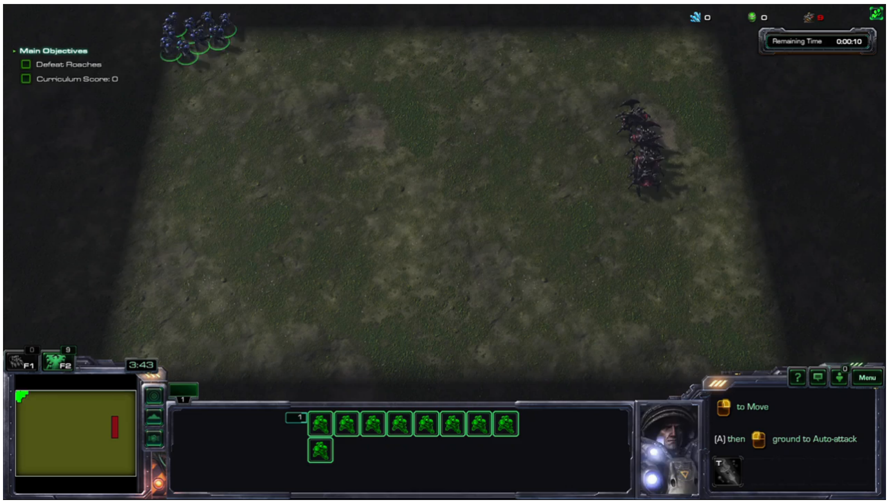
STEP 6 **Repeat**

Repeat this several times until each time you hold your hand out, your dog lifts his paw rather than sniffs or licks your hand first.

Exploration

Credit Assignment

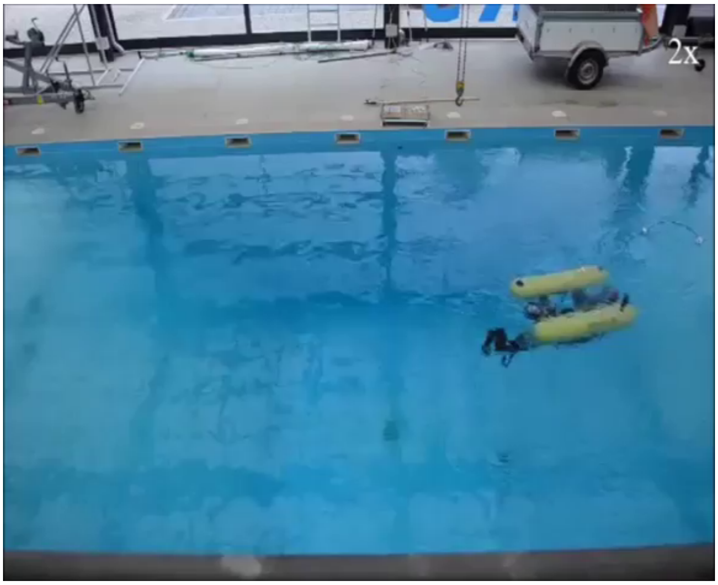
RL in StarCraft II



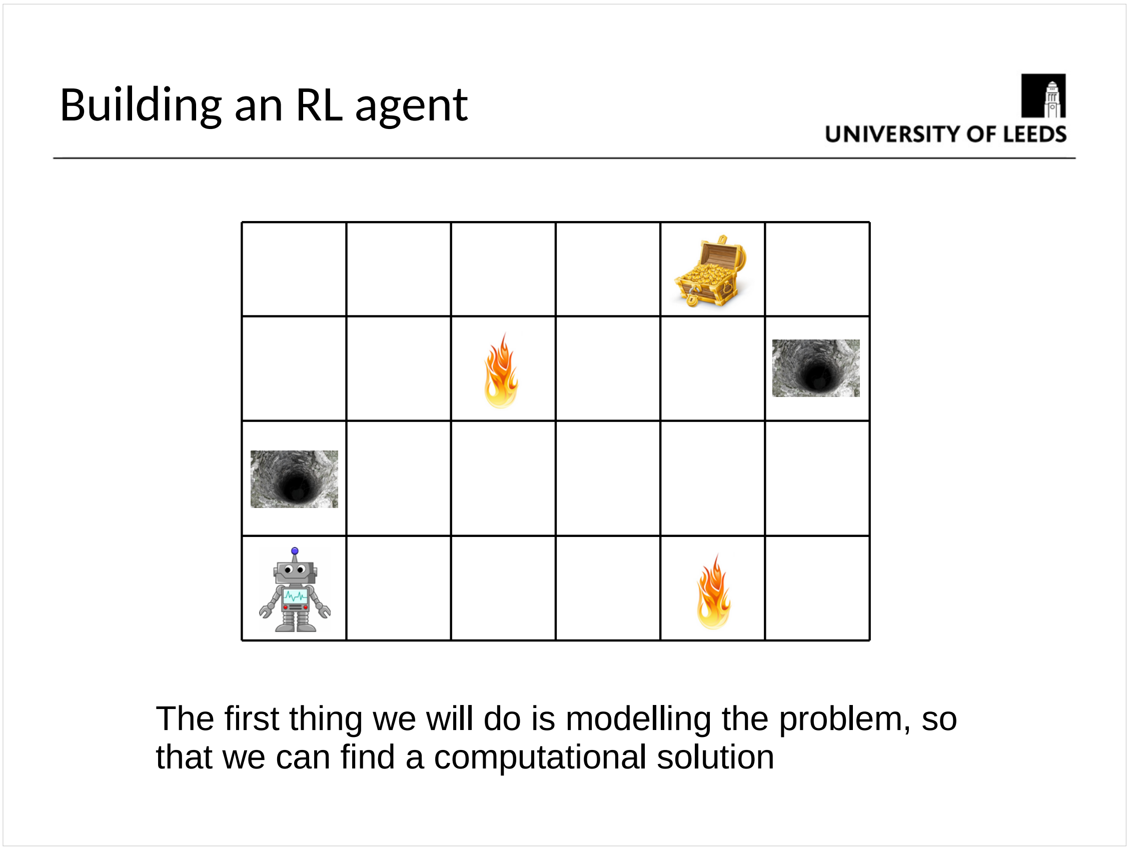
MSc Project by: Ryan Cross, Amir Faris, Daniel Spooner

Adaptation

UNIVERSITY OF LEEDS



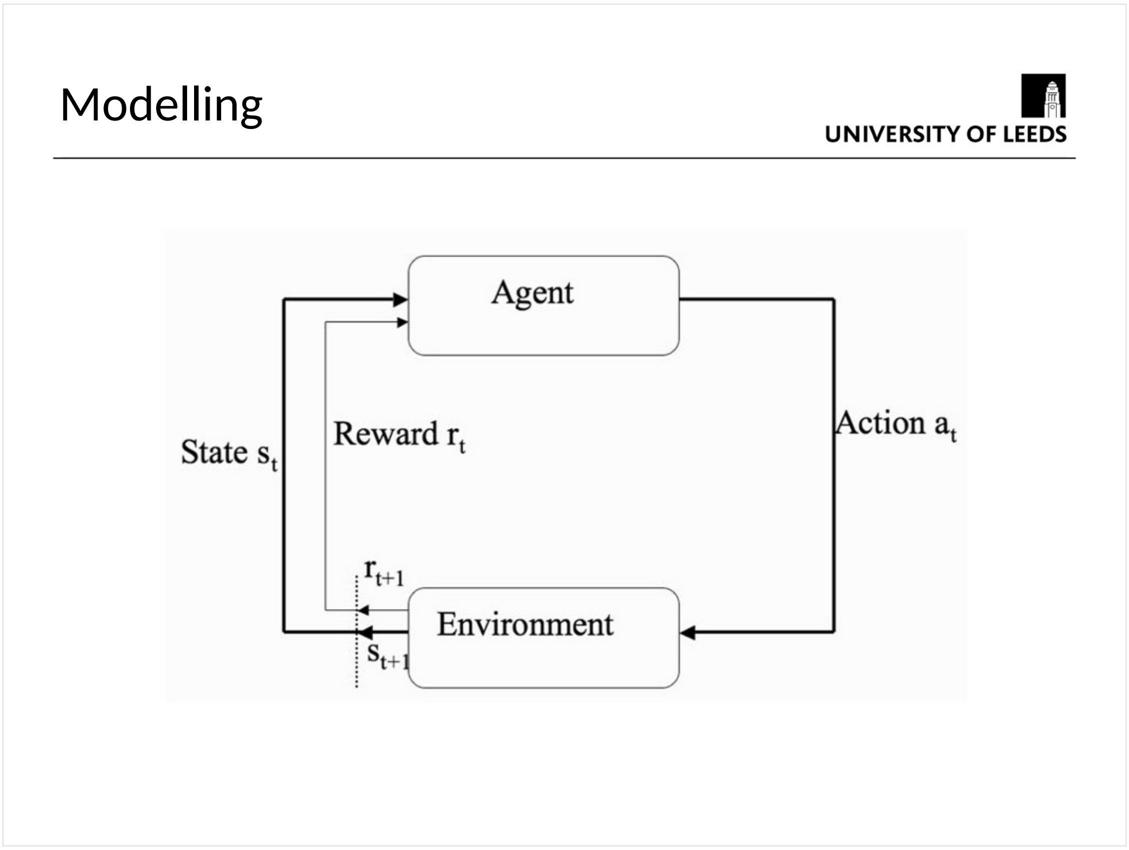
Joint work with: Seyed Reza Ahmadzadeh, Arnau Carrera, Marc Carreras, Petar Kormushev, Darwin G Caldwell



Building an RL agent

The first thing we will do is modelling the problem, so that we can find a computational solution

We are going to develop a mathematical model of the world in terms of *states*, which represent situations in which the agent can be, and *actions*, that is, things the agent can do to change the current situation.



Modelling

UNIVERSITY OF LEEDS

The agent is in a continuous loop with the environment. The environment provides the current state and the *reward*, a signal representing how well the agent is doing. Based on this information, the agent chooses the next action, and the cycle repeats.

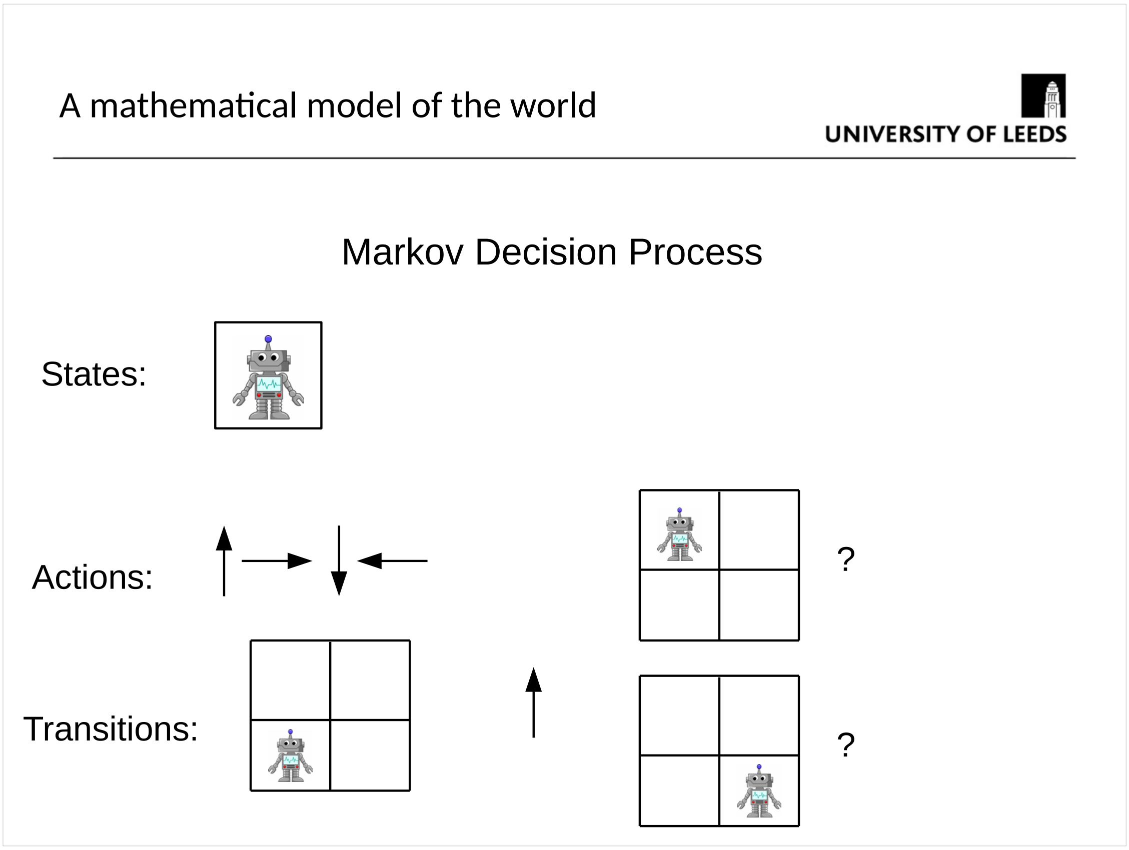
A different learning paradigm

Time matters: decisions at a certain point, affect decisions at later points → concept of *state*.

It is not supervised learning, because the agent does not have samples of the correct answers, for instance the optimal action for certain states

It is not unsupervised learning, because the agent does receive some feedback on its actions.

Closest machine learning paradigm to what humans and animals do. Methods have been inspired by biological learning systems, and models of real systems have been inspired by reinforcement learning.



A mathematical model of the world

Markov Decision Process

States:

Actions: ↑ → ↓ ←

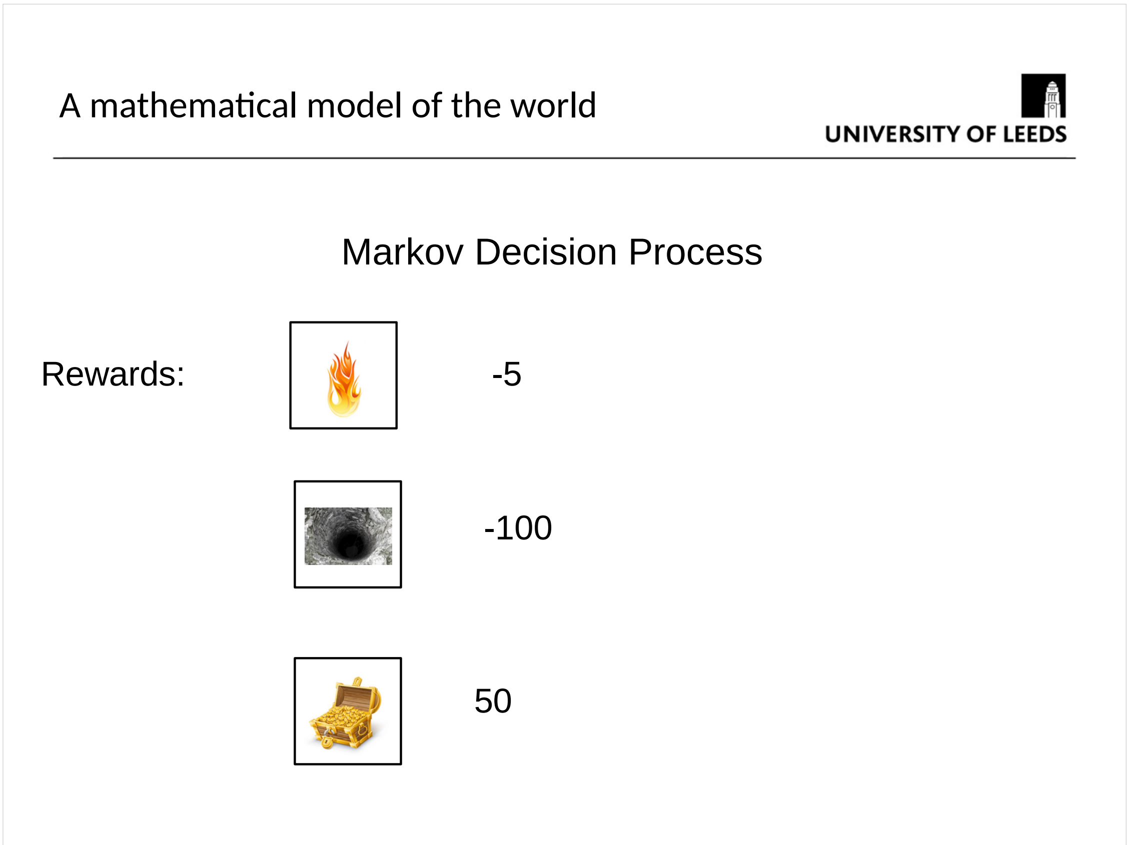
Transitions: ↑

?

?

A decision problem is represented in terms of the following sets and functions:

- S, a set of states. This represents the information on the situation the agent is in, and that can be used to make decisions.
-A, a set of actions. These are the capabilities of the agent: the things the agent can do. Actions change the environment, that is, they affect the state.
- T: a transition function. The transition function describes how the state changes as a consequence of the agent’s actions. For instance, if the agent is in the bottom left corner and executes the action *up*, at the next time step it will be in the top left corner and not anywhere else.



A mathematical model of the world

UNIVERSITY OF LEEDS

Markov Decision Process

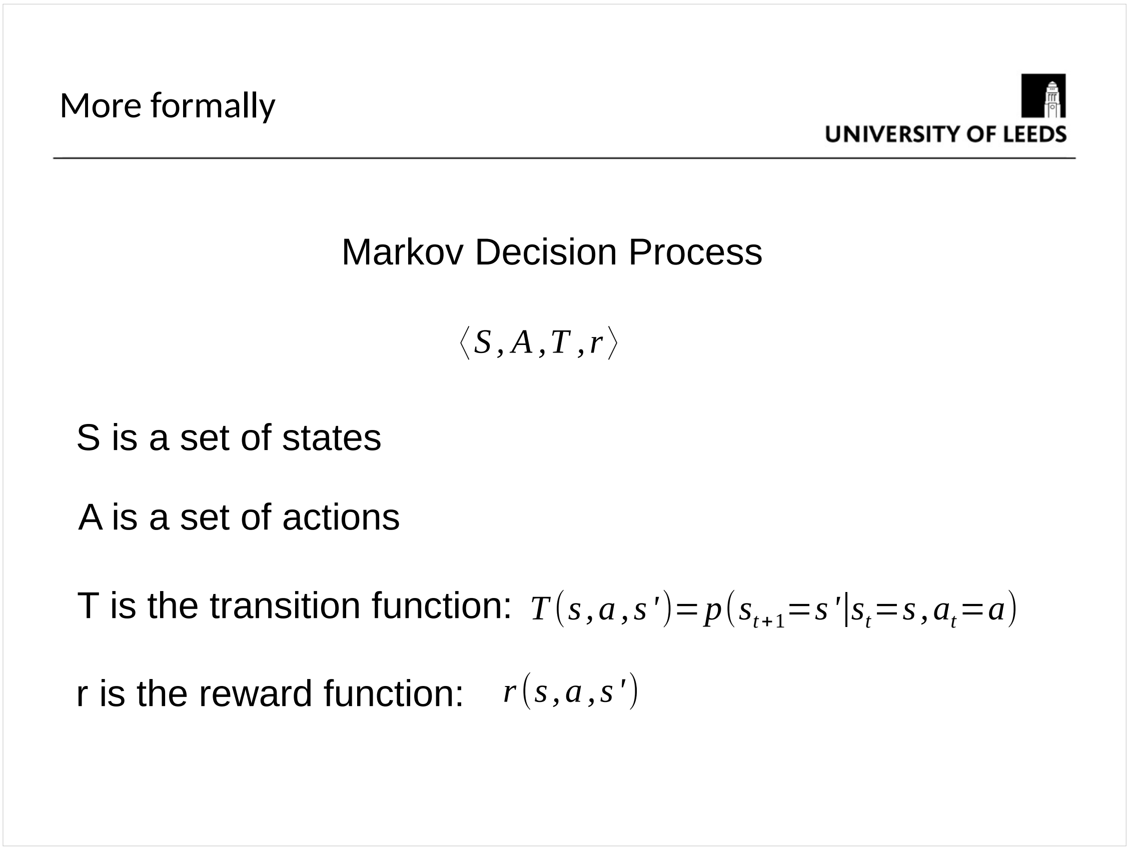
Rewards:

-5

-100

50

-r is the reward function. The rewards express in a quantitative way how desirable a certain transition between states is.



More formally

UNIVERSITY OF LEEDS

Markov Decision Process

$$\langle S, A, T, r \rangle$$

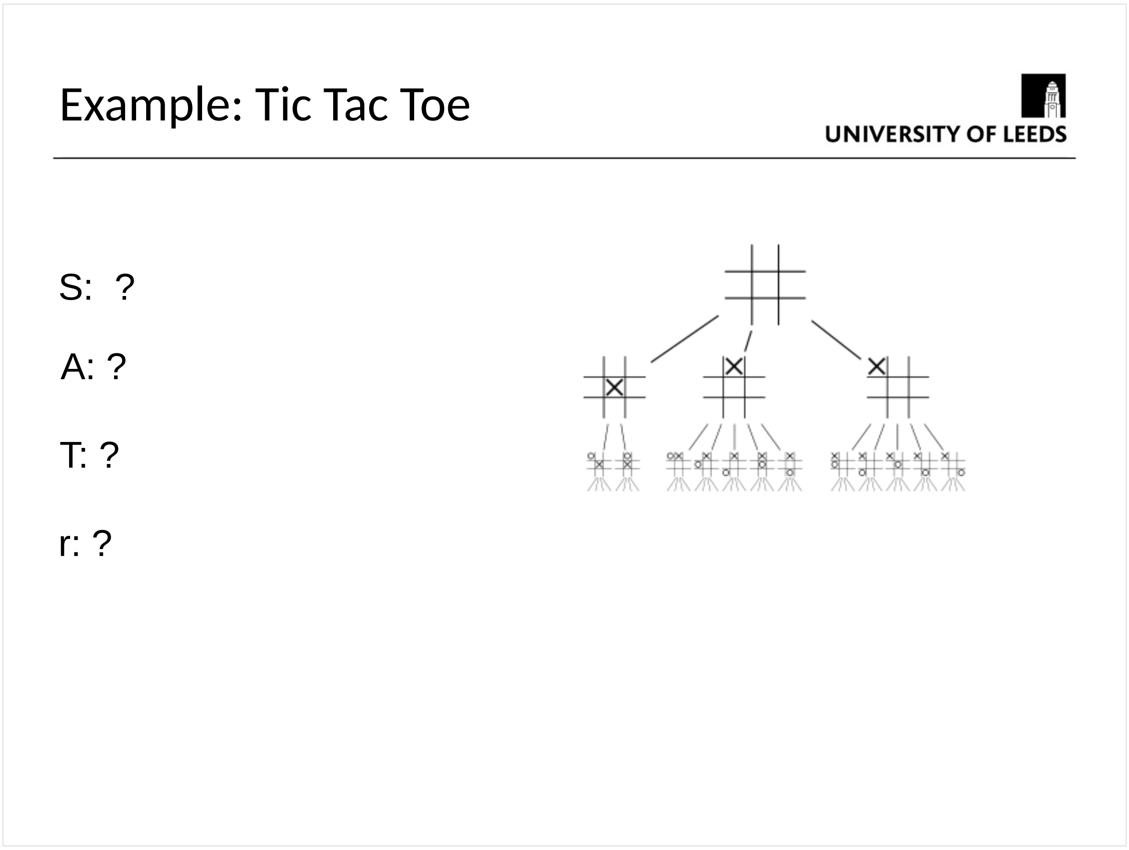
S is a set of states

A is a set of actions

T is the transition function: $T(s,a,s') = p(s_{t+1} = s' | s_t = s, a_t = a)$

r is the reward function: $r(s,a,s')$

These elements together form what is called a *Markov Decision Process*. This is a decision problem, and the agent is required to pick an action in each state with the goal of maximizing the long-term reward (this will be defined more precisely in a few slides).



Example: Tic Tac Toe

S: ?

A: ?

T: ?

r: ?

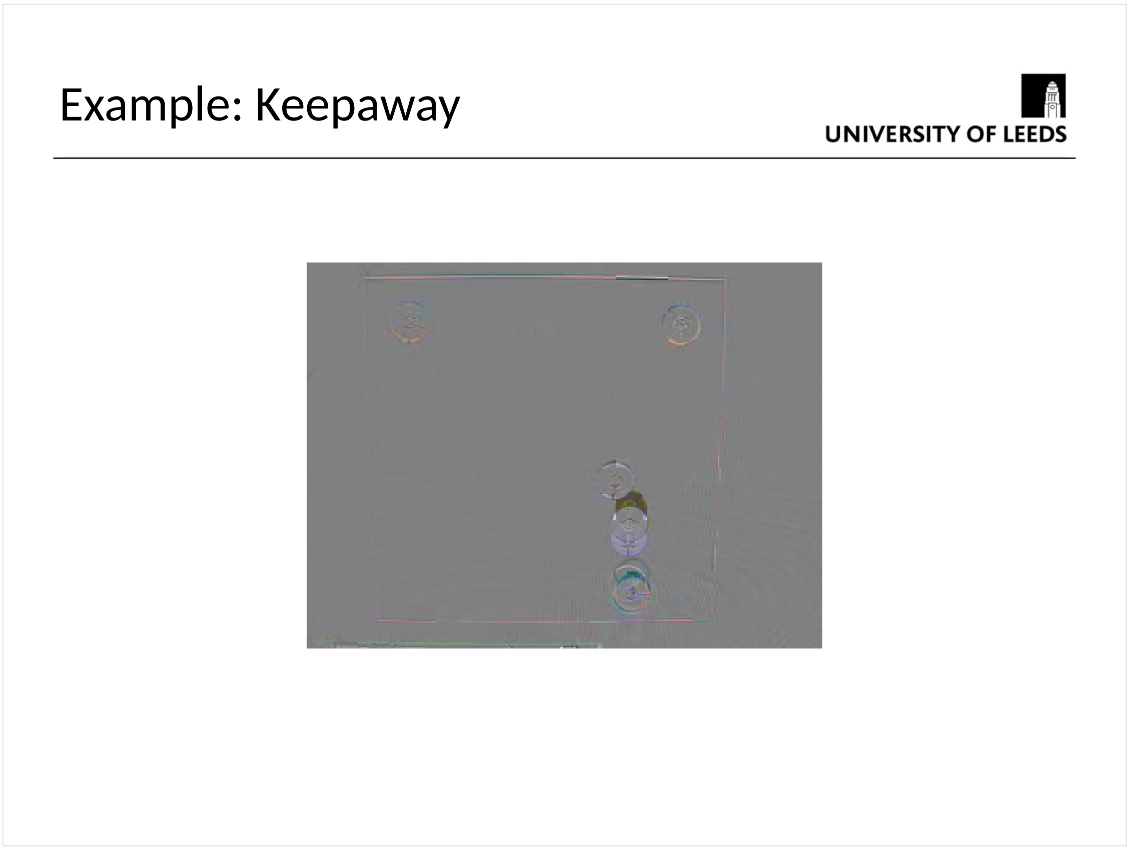
A simple game like Tic Tac Toe can be modelled as a Markov Decision Process (MDP) as follows:

S: is the set of all the possible positions of X and O on the board.

A the agent can place its symbol (let’s say we are playing the X) in each of the 9 positions of the board. So there are 9 corresponding actions.

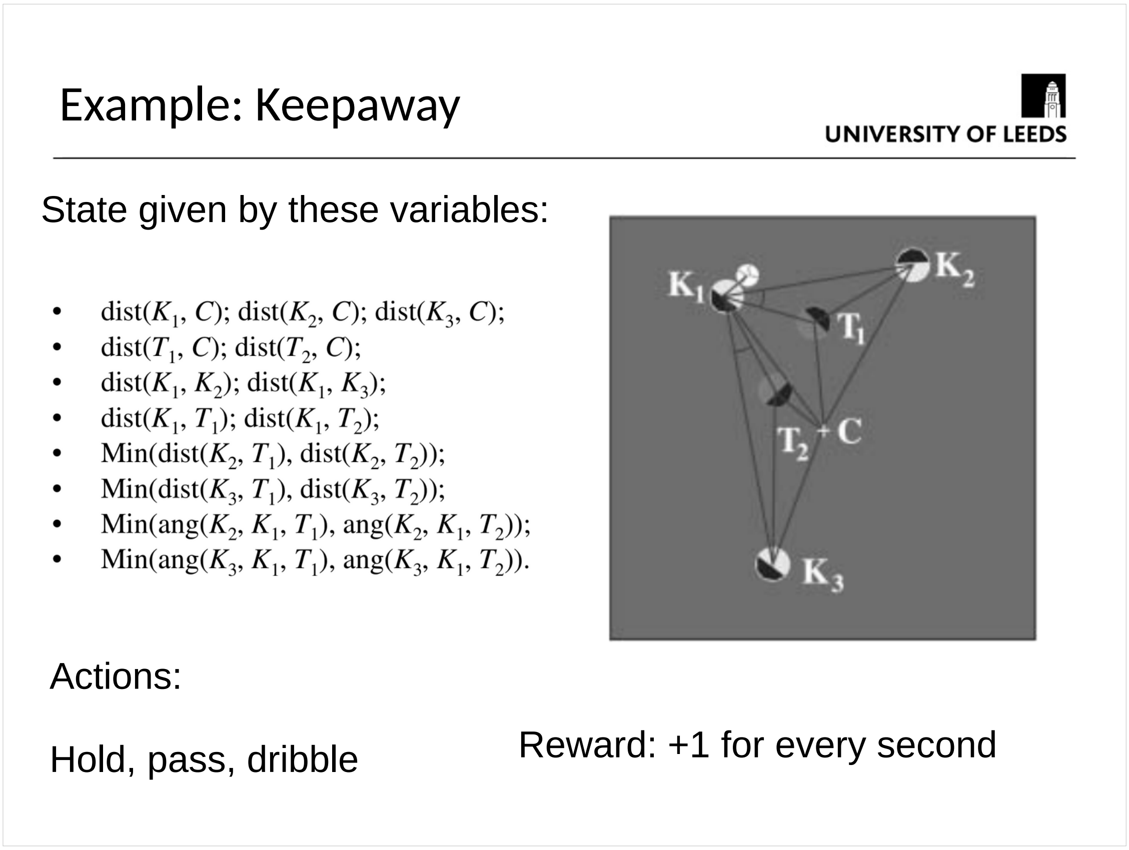
T: the transitions between decisions are probabilistic, and depend on the opponent. While placing an X in a particular location is a deterministic action (it always results in the X being added to that location) the next decision will take place after the opponent has chosen their action. Since we don’t know which action they will choose, we have to consider the effect of all their possible actions as possible next positions of the board, that is, next states.

R: We can assign a reward of 1 to a winning board, a reward of 0 to a tie, and a reward of -1 to a losing board.



Example: Keepaway

Keepaway is another example. This is a 2D simplification of soccer, in which a team of agents (the *keepers*) try to keep possession of the ball for as long as possible from the other team (the *takers*).



Example: Keepaway

UNIVERSITY OF LEEDS

State given by these variables:

- dist(K_1, C); dist(K_2, C); dist(K_3, C);
- dist(T_1, C); dist(T_2, C);
- dist(K_1, K_2); dist(K_1, K_3);
- dist(K_1, T_1); dist(K_1, T_2);
- Min(dist(K_2, T_1), dist(K_2, T_2));
- Min(dist(K_3, T_1), dist(K_3, T_2));
- Min(ang(K_2, K_1, T_1), ang(K_2, K_1, T_2));
- Min(ang(K_3, K_1, T_1), ang(K_3, K_1, T_2)).

Actions:

Hold, pass, dribble

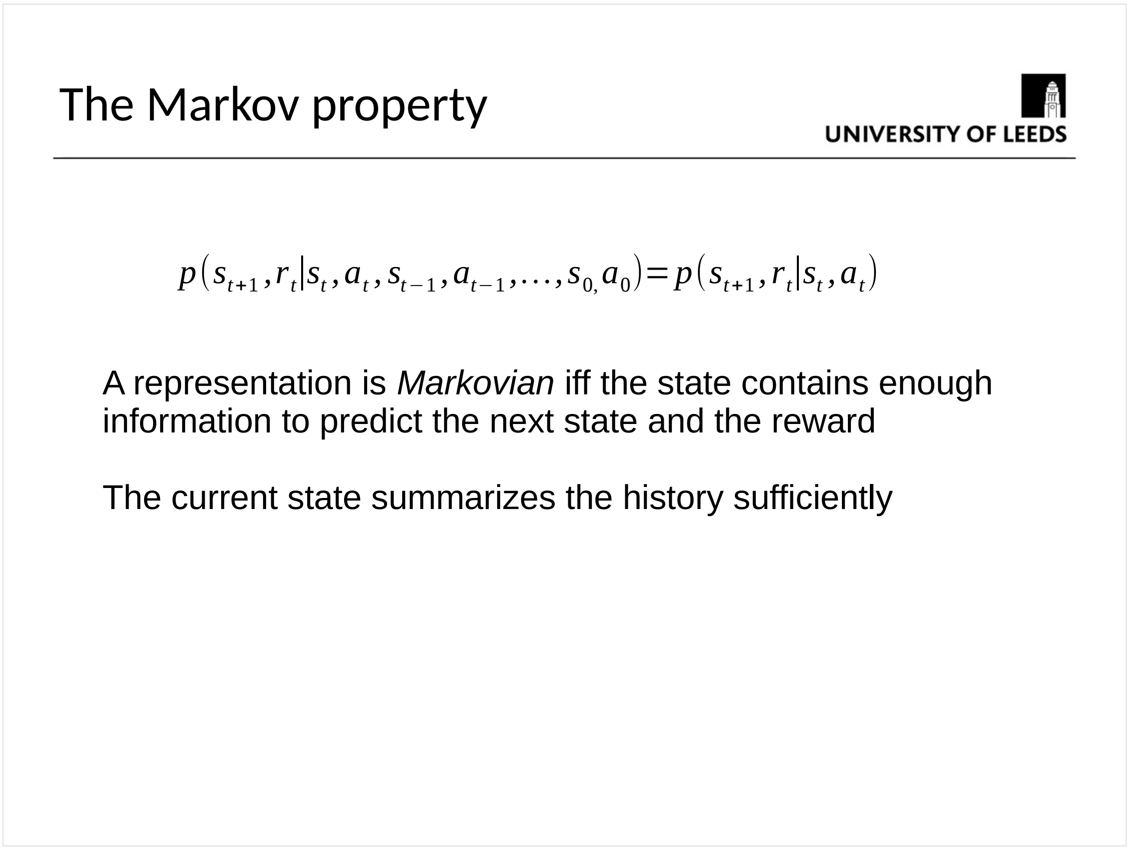
Reward: +1 for every second

This game has been proposed as a framework for reinforcement learning in

Peter Stone, Richard S. Sutton, and Gregory Kuhlmann. "Reinforcement learning for robocup soccer keepaway." Adaptive Behavior 13.3 (2005): 165-188.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.435.6595&rep=rep1&type=pdf

Where they also define a possible state representation, with several variables for the distances and angles between the keepers and their teammates (this is just an example to have an idea of what it means to define the states of an MDP, you don’t need to learn this particular one).



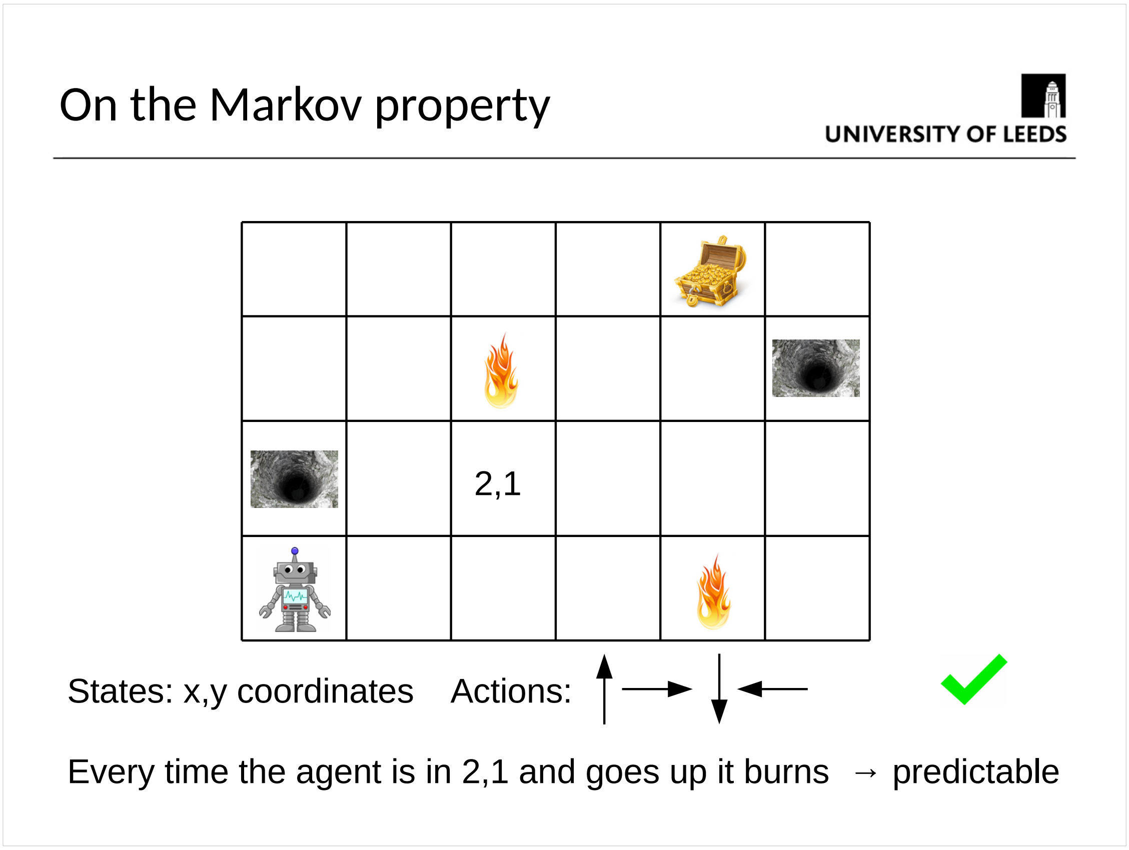
The Markov property

$$p(s_{t+1}, r_t | s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_{0,} a_0) = p(s_{t+1}, r_t | s_t, a_t)$$

A representation is *Markovian* iff the state contains enough information to predict the next state and the reward

The current state summarizes the history sufficiently

The M in MDP (Markov) means that the information contained in the states is enough determine the probability of the next state and reward. While in general you may need the full history to determine the current state, in Markovian systems the current state is enough. More details on this in the next slides.



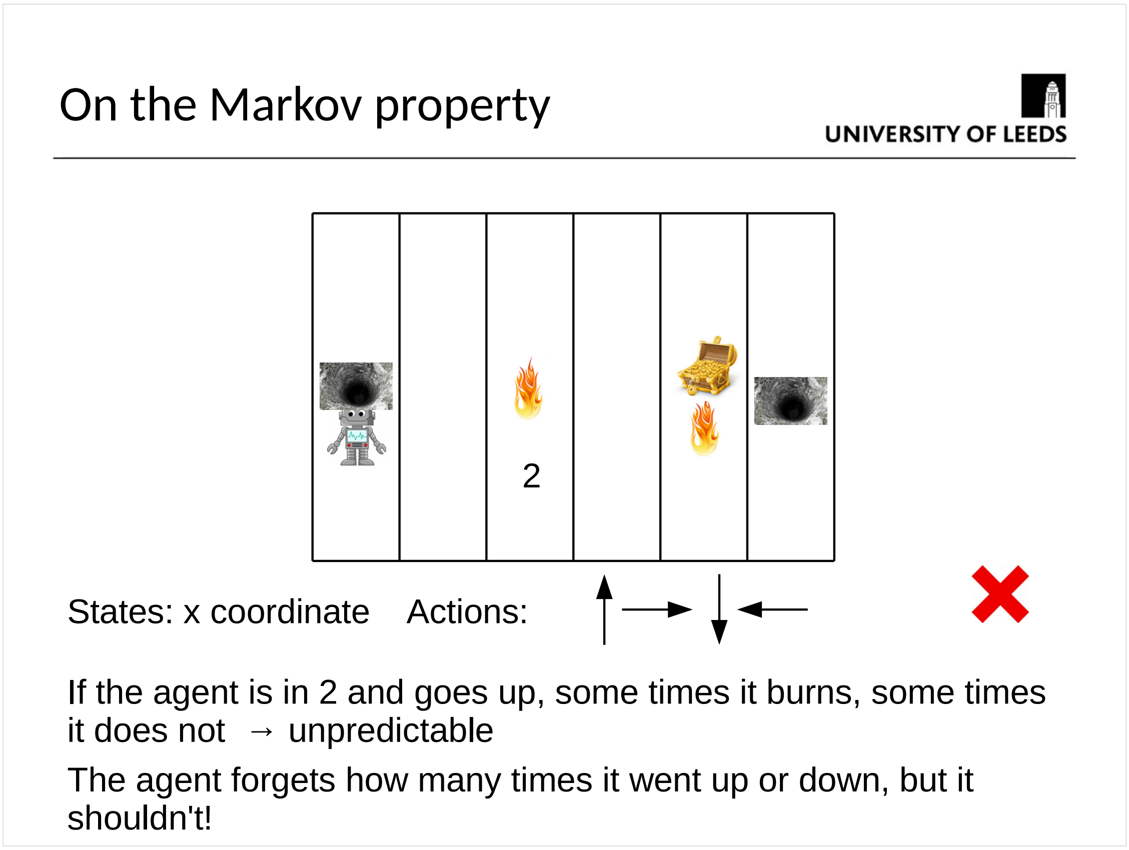
On the Markov property

2,1

States: x,y coordinates Actions: ↑ → ↓ ← ✓

Every time the agent is in 2,1 and goes up it burns → predictable

In this simple gridworld, the coordinates of the cell in which the agent is are enough to predict both the next state and the reward. For instance, if the agent is in 2,1 and goes up, we now that the next state is 2,2 and that it will end up in the fire.



On the Markov property

UNIVERSITY OF LEEDS

States: x coordinate Actions: ↑ → ↓ ← ✗

If the agent is in 2 and goes up, some times it burns, some times it does not → unpredictable

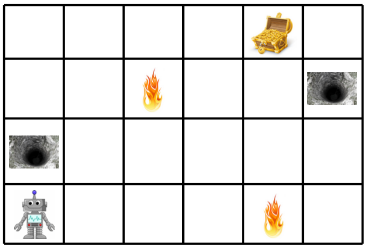
The agent forgets how many times it went up or down, but it shouldn't!

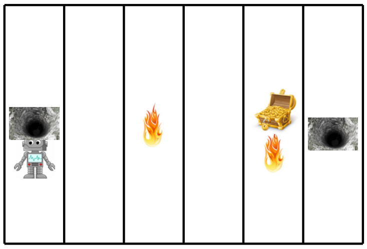
If we only consider the x coordinate as the state information, then we can’t predict the reward effectively anymore. When the agent is in 2 it can be anywhere on that column. When it executes the action up it might or might not get burned. With this little information it is impossible to learn an optimal policy, because the optimal decision depends on where on the column the agent actually is.

That is, without being able to discriminate between being above or below the fire, the agent cannot decide whether it is safe to move up.

Therefore, this representation that only has one coordinate in the state, is not Markovian.

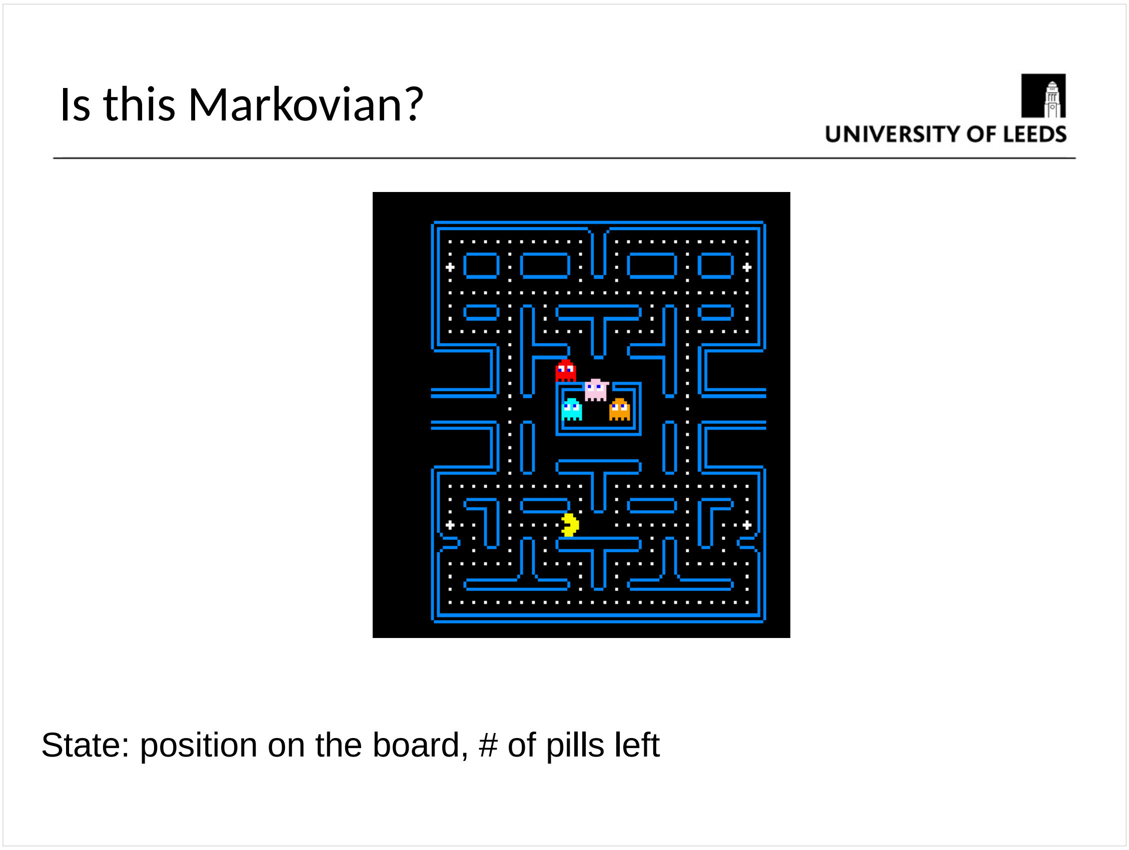
Observability





The Markov property is often about being able to observe all the necessary details OR remembering the necessary details observed in the past

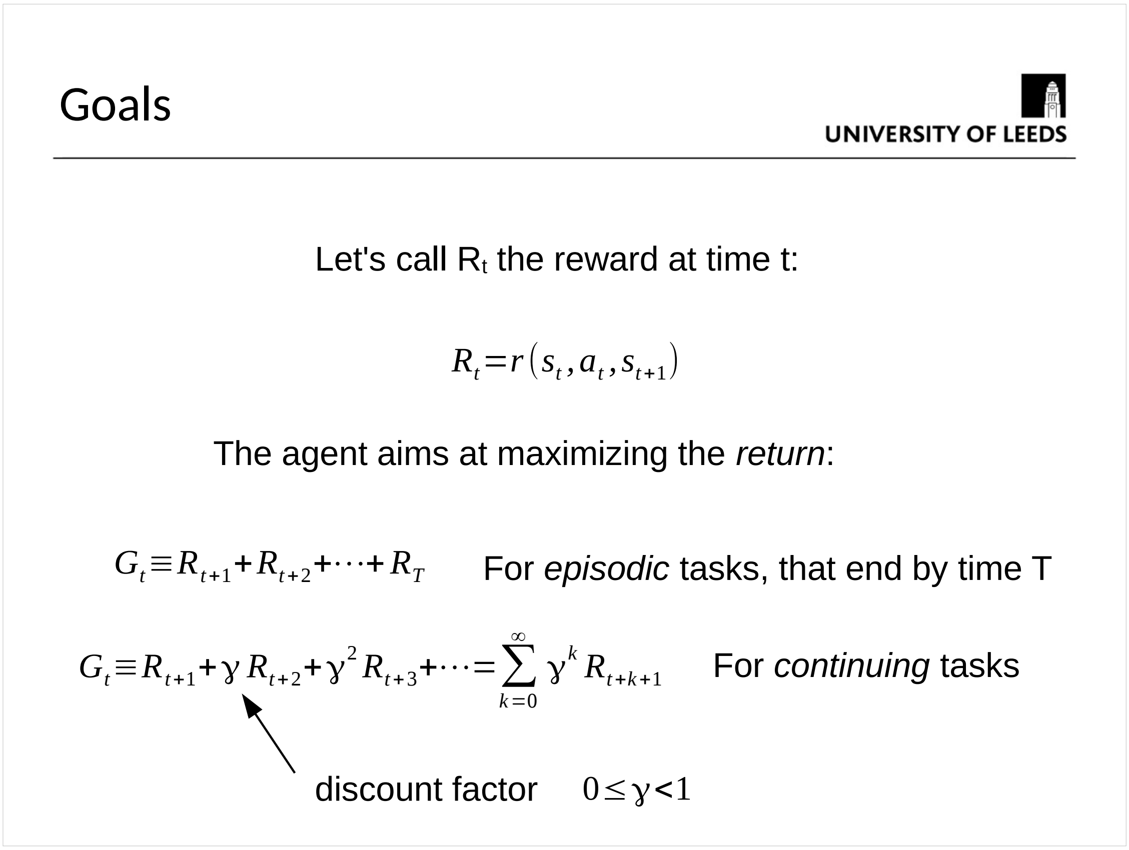
Other models known as Partially Observable Markov Decision Processes (POMDPs) capture partial observability, but we won't look at them in this class.



Is this Markovian?

State: position on the board, # of pills left

Would you need any more information to make optimal decisions in this game?



Goals

Let's call R_t the reward at time t:

$$R_t = r(s_t, a_t, s_{t+1})$$

The agent aims at maximizing the *return*:

$$G_t \equiv R_{t+1} + R_{t+2} + \cdots + R_T$$

For *episodic* tasks, that end by time T

$$G_t \equiv R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

For *continuing* tasks

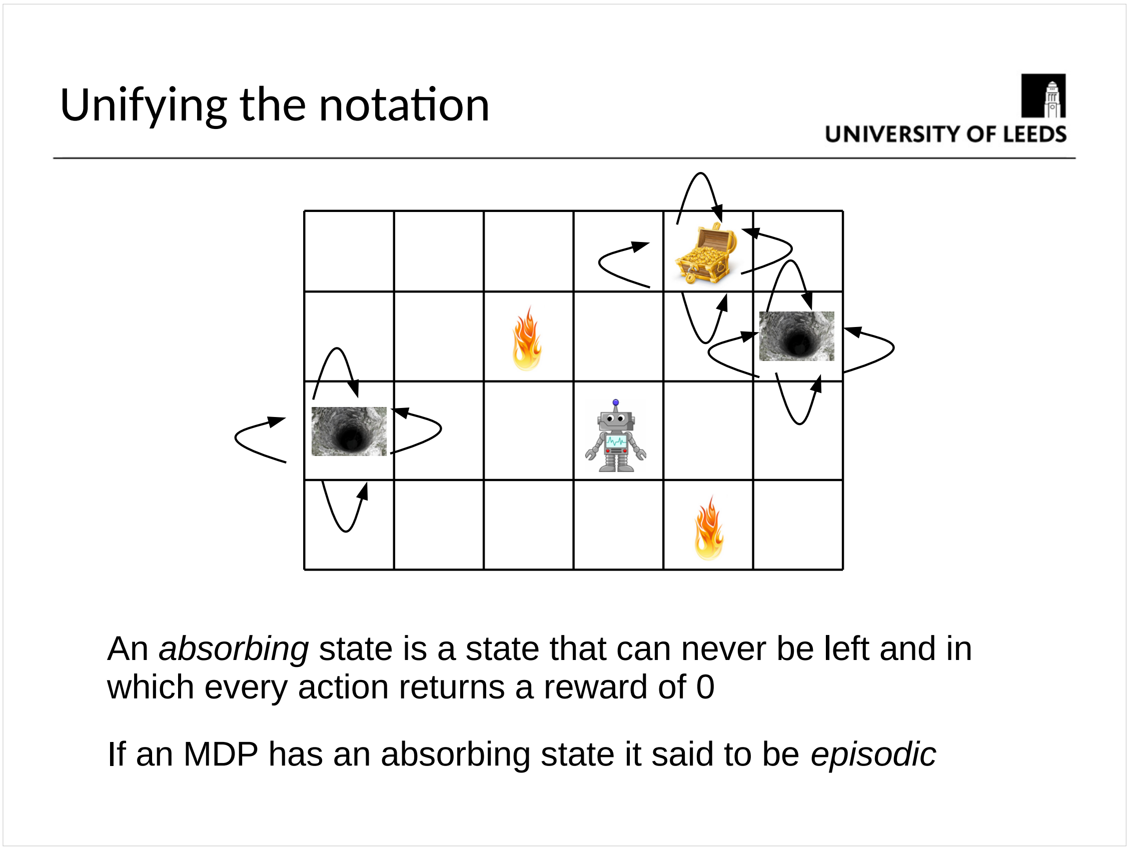
discount factor $0 \le \gamma < 1$

Now that we have thoroughly considered states and actions, let’s move to goals. What is the goal of the agent?

It is to obtain, from any given time step, the maximum *cumulative* reward, which is called the *return*.

If the task is episodic, which means it has an end, this sum is finite. However, if the task is continuing and the agent can potentially take actions forever, then the series of the returns can diverge. To make sure that it doesn’t happen, we apply a geometric series of *discounts* to the rewards.

The discount, γ, represents a preference towards earlier rewards versus later ones, because the later ones are discounted more. For instance, would you rather have 5 pounds now or five pounds in 2 years? And what about 10 pounds in 2 years? If you prefer 5 pounds now, you are discounting the future reward, so that, since it is further into the future, it has less value to you now. A sufficient discount will also make you prefer a lower reward now over a larger reward later.



Unifying the notation

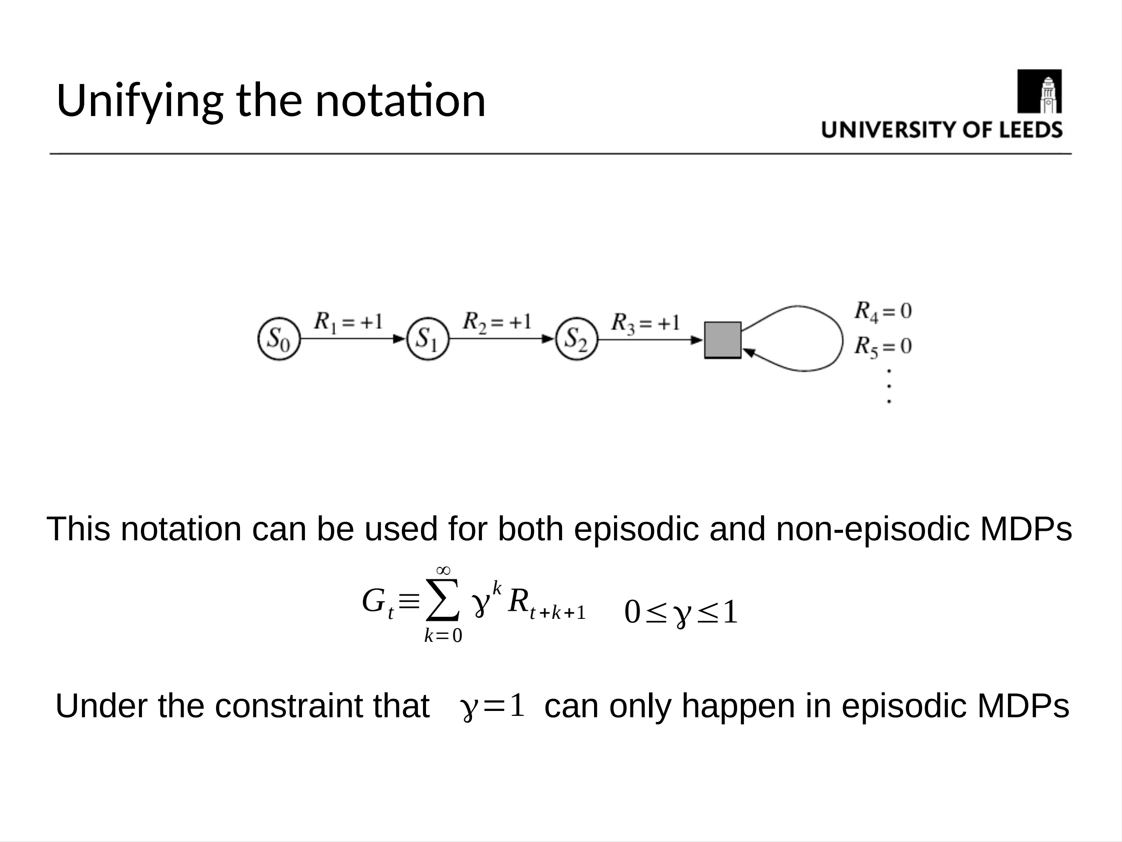
An *absorbing* state is a state that can never be left and in which every action returns a reward of 0

If an MDP has an absorbing state it said to be *episodic*

The two cases, episodic and continuing, can be unified by introducing special transitions in the states that would terminate an episode.

While in this grid world the episode ends when the agent finds the treasure or falls into a pit, we can create additional transitions so that regardless of the action taken by the agent in that state, it can never leave the state, and will forever receive a reward of 0.

This effectively turns an episodic task into a continuing one, because the agent will take actions forever (at least conceptually… then in practice you'll terminate the simulation!), but will be trapped in one of the terminal states with 0 additional reward.



Unifying the notation

This notation can be used for both episodic and non-episodic MDPs

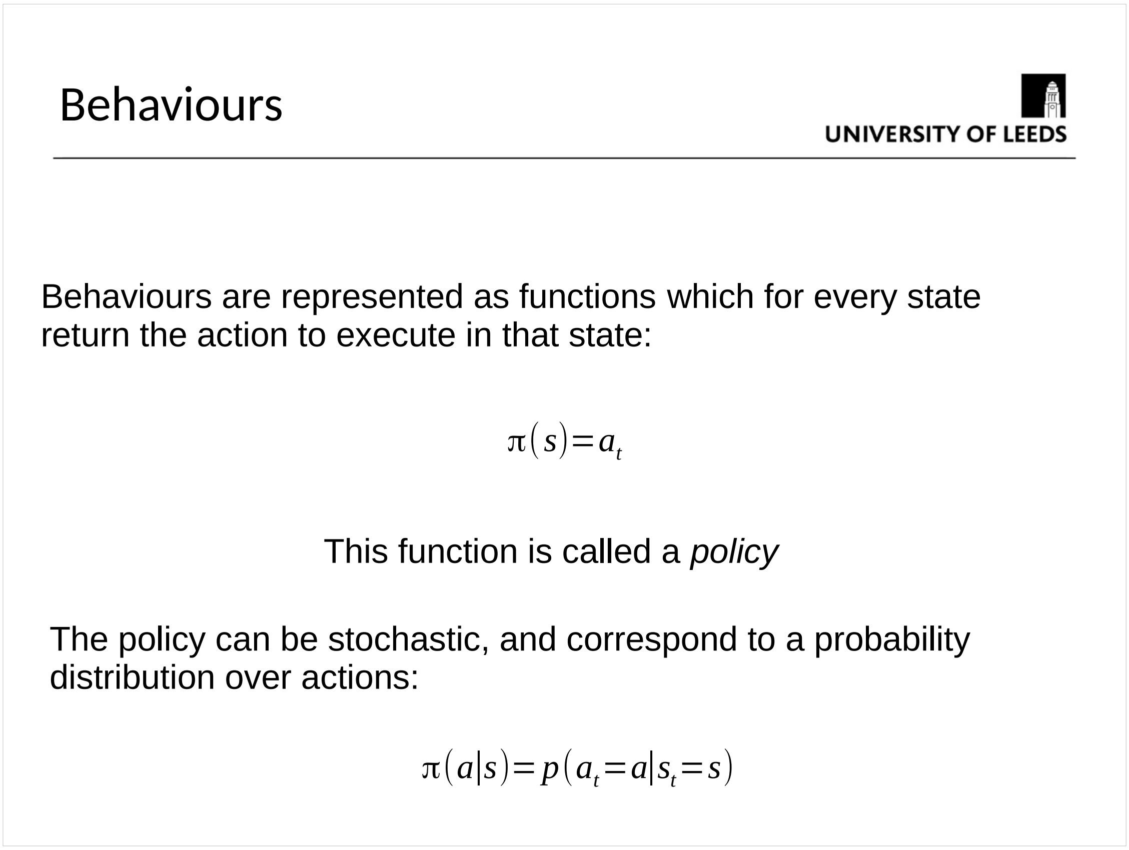
$$G_t \equiv \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \qquad 0 \leq \gamma \leq 1$$

Under the constraint that $\gamma = 1$ can only happen in episodic MDPs

Through absorbing states we turned episodic tasks into continuing tasks, and therefore can unify the notation for the return, using the return for continuing tasks in both cases.

If a task is really episodic, all the reward from some point on will be 0.

For this reason, γ can be 1 only if the task is episodic, since the series will have all 0s from some point and the sum will be finite. Otherwise it has to be < 1.



Behaviours

Behaviours are represented as functions which for every state return the action to execute in that state:

$$\pi(s)=a_t$$

This function is called a *policy*

The policy can be stochastic, and correspond to a probability distribution over actions:

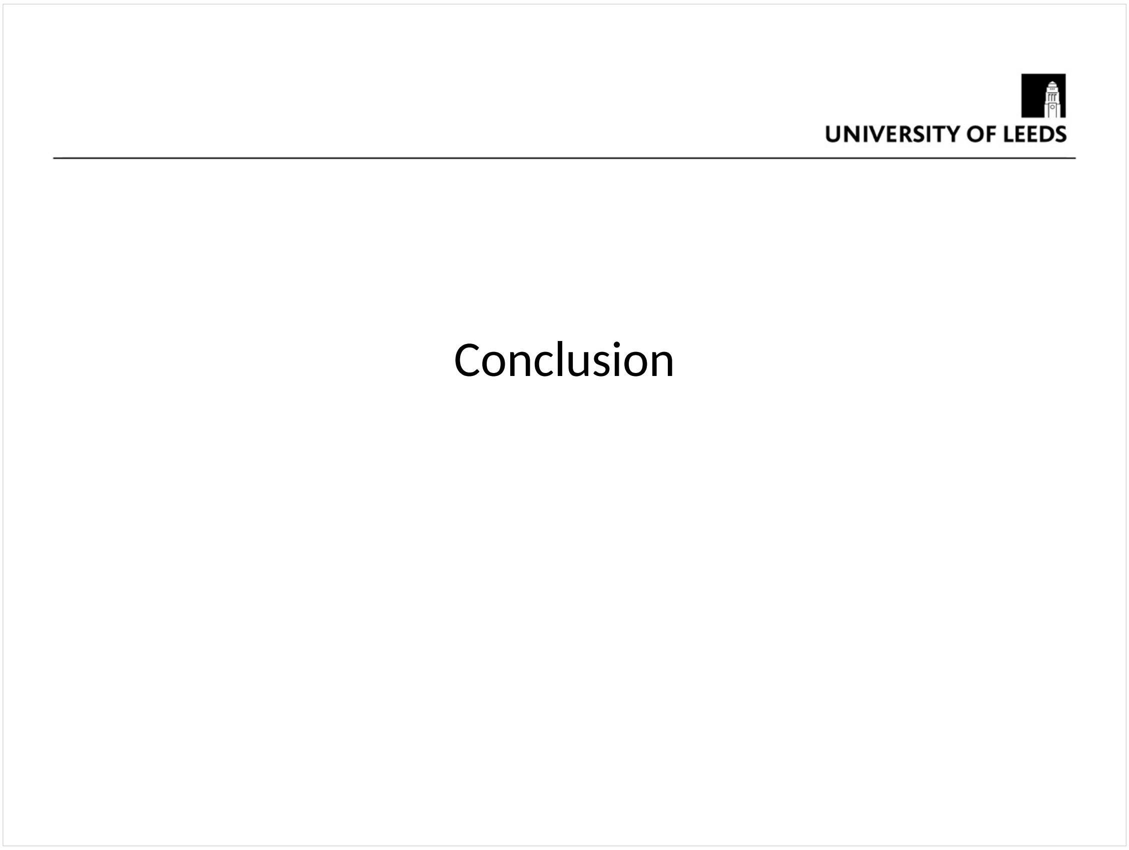
$$\pi(a|s)=p(a_t=a|s_t=s)$$

Behaviours are represented by functions called *policies*.

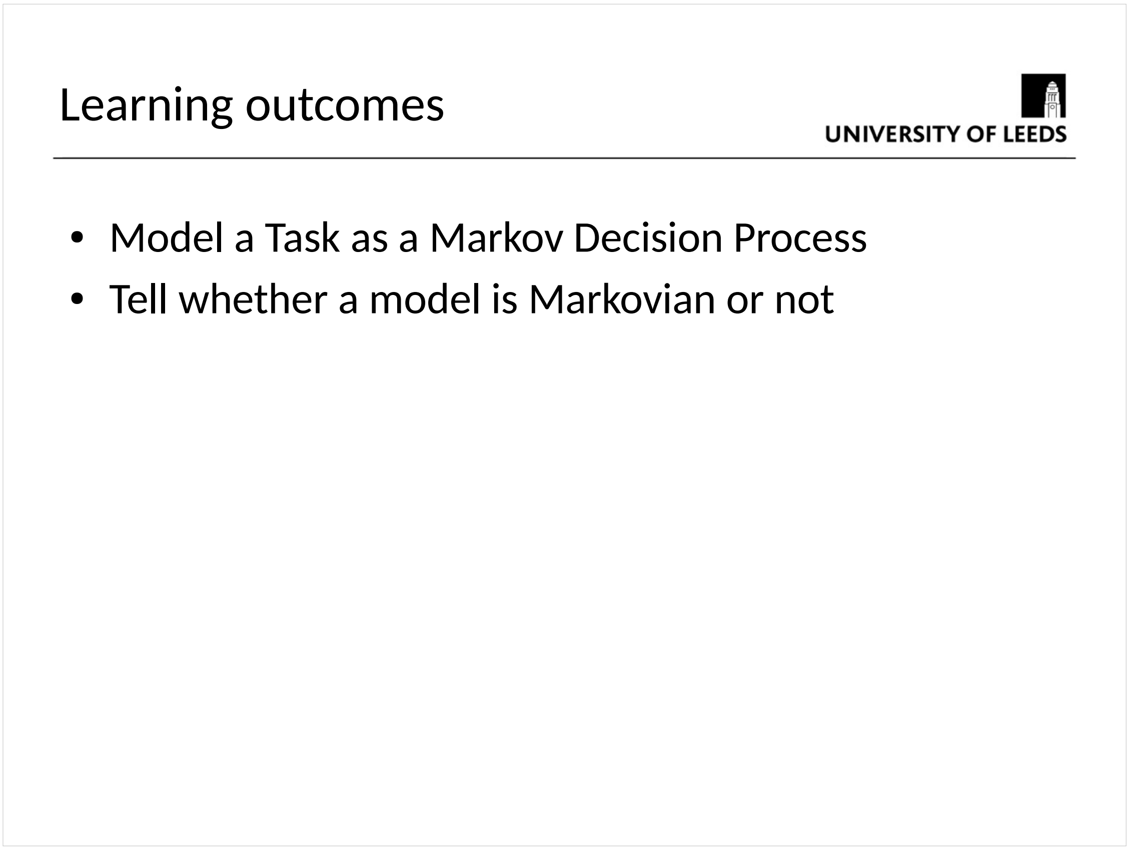
A policy returns the action to execute for each input state.

The returned action can be stochastic, in which case the policy is a probability distribution over actions.

For example, in the grid world domain, we could have a policy that in any state goes up with probability 0.5, and right with probability 0.5. Or any other number, as long as they sum to 1.



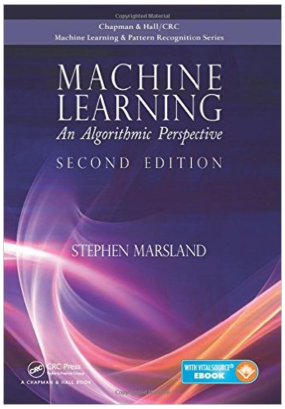
Conclusion



Learning outcomes

UNIVERSITY OF LEEDS

- Model a Task as a Markov Decision Process
- Tell whether a model is Markovian or not


Copyrighted Material
Chapman & Hall/CRC
Machine Learning & Pattern Recognition Series
MACHINE LEARNING
An Algorithmic Perspective
SECOND EDITION
STEPHEN MARSLAND
CRC Press
A CHAPMAN & HALL BOOK
WITH VITALSOURCE® EBOOK

Chapter 11