COMP250: Artificial Intelligence

3: Planning

FALMOUTH UNIVERSITY

Game theory

Game theory

Game theory

- A branch of mathematics studying **decision making**

Game theory

- A branch of mathematics studying **decision making**
- A **game** is a system where one or more **players** choose **actions**; the combination of these choices lead to each agent receiving a **payoff**

Game theory

- A branch of mathematics studying **decision making**
- A **game** is a system where one or more **players** choose **actions**; the combination of these choices lead to each agent receiving a **payoff**
- Important applications in economics, ecology and social sciences as well as AI

The ~~Prisoner's~~ Student's Dilemma

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information
- Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information
- Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- If **both stay silent**, both receive a C grade

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information
- Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- If **both stay silent**, both receive a C grade
- If **Alice betrays Bob**, she receives an A whilst he gets expelled

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information
- Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- If **both stay silent**, both receive a C grade
- If **Alice betrays Bob**, she receives an A whilst he gets expelled
- If **Bob betrays Alice**, he receives an A whilst she gets expelled

The ~~Prisoner's~~ Student's Dilemma

- Two students, **Alice** and **Bob**, are suspected of copying from each other
- Each is offered a deal in exchange for information
- Each can choose to **betray** the other or stay **silent** — but they **cannot communicate** before deciding what to do
- If **both stay silent**, both receive a C grade
- If **Alice betrays Bob**, she receives an A whilst he gets expelled
- If **Bob betrays Alice**, he receives an A whilst she gets expelled
- If **both betray each other**, both get an F

Payoff matrix

Payoff matrix

	A silent	A betray
B silent	A: 50 B: 50	A: 70 B: -100
B betray	A: -100 B: 70	A: 0 B: 0

Payoff matrix

	A silent	A betray
B silent	A: 50 B: 50	A: 70 B: -100
B betray	A: -100 B: 70	A: 0 B: 0

Socrative `FALCOMPED`: what would you do?

Alice's thought process

Alice's thought process

- The best outcome overall is for both of us to stay silent

Alice's thought process

- The best outcome overall is for both of us to stay silent
- However if Bob stays silent, I can get a better mark by betraying him

Alice's thought process

- The best outcome overall is for both of us to stay silent
- However if Bob stays silent, I can get a better mark by betraying him
- And if Bob betrays me, I should betray him to avoid getting expelled

Alice's thought process

- The best outcome overall is for both of us to stay silent
- However if Bob stays silent, I can get a better mark by betraying him
- And if Bob betrays me, I should betray him to avoid getting expelled
- Therefore the rational choice is to betray

Alice's thought process

- The best outcome overall is for both of us to stay silent
- However if Bob stays silent, I can get a better mark by betraying him
- And if Bob betrays me, I should betray him to avoid getting expelled
- Therefore the rational choice is to betray

... and Bob's thought process is the same!

Nash equilibrium

Nash equilibrium

- Consider the situation where both have chosen to betray

Nash equilibrium

- Consider the situation where both have chosen to betray
- Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch

Nash equilibrium

- Consider the situation where both have chosen to betray
- Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch
- Such a situation is called a **Nash equilibrium**

Nash equilibrium

- Consider the situation where both have chosen to betray
- Neither person has anything to gain by switching to silence, assuming the other person doesn't also switch
- Such a situation is called a **Nash equilibrium**
- If all players are **rational** (in the sense of wanting to maximising payoff), they should converge upon a Nash equilibrium

Does every game have a Nash equilibrium?

Does every game have a Nash equilibrium?

	A rock	A paper	A scissors
B rock	A: 0 B: 0	A: +1 B: -1	A: -1 B: +1
B paper	A: -1 B: +1	A: 0 B: 0	A: +1 B: -1
B scissors	A: +1 B: -1	A: -1 B: +1	A: 0 B: 0

Does every game have a Nash equilibrium?

	A rock	A paper	A scissors
B rock	A: 0 B: 0	A: +1 B: -1	A: -1 B: +1
B paper	A: -1 B: +1	A: 0 B: 0	A: +1 B: -1
B scissors	A: +1 B: -1	A: -1 B: +1	A: 0 B: 0

Socrative `FALCOMPED`: what would you do?

Nash equilibrium for Rock-Paper-Scissors

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**
- E.g. if you always choose paper, I choose scissors

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**
- E.g. if you always choose paper, I choose scissors
- If we try to reason naïvely, we get stuck in a loop

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**
- E.g. if you always choose paper, I choose scissors
- If we try to reason naïvely, we get stuck in a loop
  - If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**
- E.g. if you always choose paper, I choose scissors
- If we try to reason naïvely, we get stuck in a loop
  - If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...
- The optimum strategy is to be **unpredictable**

Nash equilibrium for Rock-Paper-Scissors

- Committing to any choice of action can be **exploited**
- E.g. if you always choose paper, I choose scissors
- If we try to reason naïvely, we get stuck in a loop
  - If I choose paper, you'll choose scissors, so I should choose rock, but then you'll choose paper, so I'll choose scissors, so you'll choose rock, so I choose paper...
- The optimum strategy is to be **unpredictable**
- Choose rock with probability $\frac{1}{3}$, paper with probability $\frac{1}{3}$, scissors with probability $\frac{1}{3}$

Mixed strategies

Mixed strategies

- A **mixed strategy** assigns probabilities to actions and chooses one at random

Mixed strategies

- A **mixed strategy** assigns probabilities to actions and chooses one at random
- In contrast to a **pure** or **deterministic strategy**, which always chooses the same action

Mixed strategies

- A **mixed strategy** assigns probabilities to actions and chooses one at random
- In contrast to a **pure** or **deterministic strategy**, which always chooses the same action
- If we allow mixed strategies, **every game has at least one Nash equilibrium**

Guess $\frac{2}{3}$ of the average

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- Example:

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- Example:
  - If the guesses are 30, 40 and 80...

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- Example:
  - If the guesses are 30, 40 and 80...
  - ... then the mean is $\frac{30+40+80}{3} = 50$...

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- Example:
  - If the guesses are 30, 40 and 80...
  - ... then the mean is $\frac{30+40+80}{3} = 50$...
  - ... so the winning guess is 30, as this is closest to $\frac{2}{3} \times 50 = 33.333$

Guess $\frac{2}{3}$ of the average

- Everyone guesses a real number (decimals are allowed) between 0 and 100 inclusive
- The winner is the person who guesses closest to $\frac{2}{3}$ of the mean of all guesses
- Example:
  - If the guesses are 30, 40 and 80...
  - ... then the mean is $\frac{30+40+80}{3} = 50$...
  - ... so the winning guess is 30, as this is closest to $\frac{2}{3} \times 50 = 33.333$
- Socrative `FALCOMPED`: make your guesses!

The rational guess

The rational guess

- The average can't possibly be greater than 100

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666
- So no rational player would guess greater than 44.444

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666
- So no rational player would guess greater than 44.444
- Which means the average can't possibly be greater than 44.444

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666
- So no rational player would guess greater than 44.444
- Which means the average can't possibly be greater than 44.444
- So no rational player would guess greater than 29.629

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666
- So no rational player would guess greater than 44.444
- Which means the average can't possibly be greater than 44.444
- So no rational player would guess greater than 29.629
- ... and so on ad infinitum

The rational guess

- The average can't possibly be greater than 100
- So no rational player would guess a number greater than 66.666
- Which means the average can't possibly be greater than 66.666
- So no rational player would guess greater than 44.444
- Which means the average can't possibly be greater than 44.444
- So no rational player would guess greater than 29.629
- ... and so on ad infinitum
- So the only **rational** guess is 0, as every rational player should guess 0 and $\frac{2}{3}$ of 0 is 0

Rationality

Rationality

- Rationality is a useful assumption for mathematics and AI programmers

Rationality

- Rationality is a useful assumption for mathematics and AI programmers
- However it's important to remember that **humans aren't always rational**

FALMOUTH UNIVERSITY

Planning

Planning

Planning

- An **agent** in an **environment**

Planning

- An **agent** in an **environment**
- The environment has a **state**

Planning

- An **agent** in an **environment**
- The environment has a **state**
- The agent can perform **actions** to change the state

Planning

- An **agent** in an **environment**
- The environment has a **state**
- The agent can perform **actions** to change the state
- The agent wants to change the state so as to achieve a **goal**

Planning

- An **agent** in an **environment**
- The environment has a **state**
- The agent can perform **actions** to change the state
- The agent wants to change the state so as to achieve a **goal**
- Problem: find a sequence of actions that leads to the goal

STRIPS planning

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:
  - The **initial state** (a set of predicates which are true)

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:
  - The **initial state** (a set of predicates which are true)
  - The **goal state** (a set of predicates, specifying whether each should be true or false)

STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:
  - The **initial state** (a set of predicates which are true)
  - The **goal state** (a set of predicates, specifying whether each should be true or false)
  - The set of **actions**, each specifying:

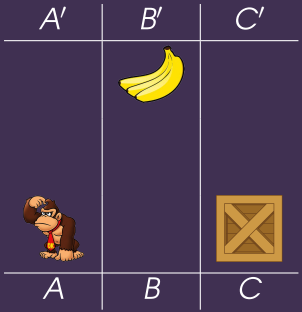
STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:
  - The **initial state** (a set of predicates which are true)
  - The **goal state** (a set of predicates, specifying whether each should be true or false)
  - The set of **actions**, each specifying:
    - Preconditions (a set of predicates which must be satisfied for this action to be possible)

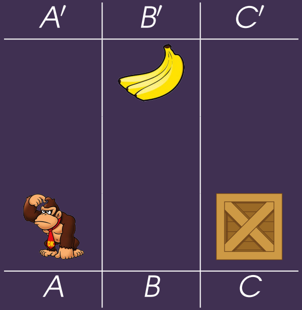
STRIPS planning

- **St**anford **R**esearch **I**nstitute **P**roblem **S**olver
- Describes the state of the environment by a set of **predicates** which are true
- Models a problem as:
  - The **initial state** (a set of predicates which are true)
  - The **goal state** (a set of predicates, specifying whether each should be true or false)
  - The set of **actions**, each specifying:
    - Preconditions (a set of predicates which must be satisfied for this action to be possible)
    - Postconditions (specifying what predicates are made true or false by this action)

STRIPS example



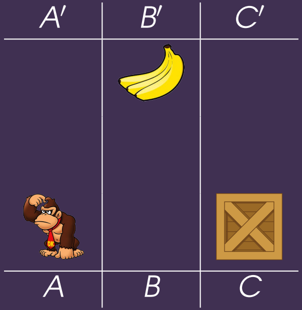
STRIPS example



Initial state:

```
At(A),
BoxAt(C),
BananasAt(B')
```

STRIPS example



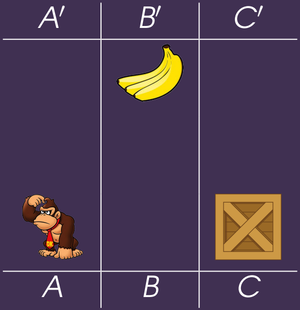
Initial state:

```
At(A),
BoxAt(C),
BananasAt(B′)
```

Goal:

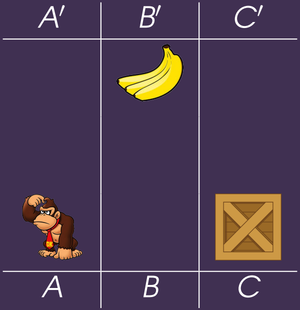
```
HasBananas
```

STRIPS example — Actions

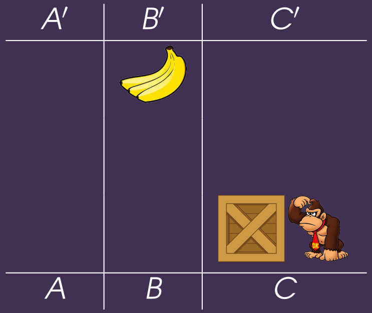



```
Move(x, y)
  Pre:  At(x)
  Post: !At(x), At(y)

ClimbUp(x)
  Pre:  At(x), BoxAt(x)
  Post: !At(x), At(x′)

ClimbDown(x′)
  Pre:  At(x′), BoxAt(x)
  Post: !At(x′), At(x)

PushBox(x, y)
  Pre:  At(x), BoxAt(x)
  Post: !At(x), At(y),
        !BoxAt(x), BoxAt(y)

TakeBananas(x)
  Pre:  At(x), BananasAt(x)
  Post: !BananasAt(x), HasBananas
```

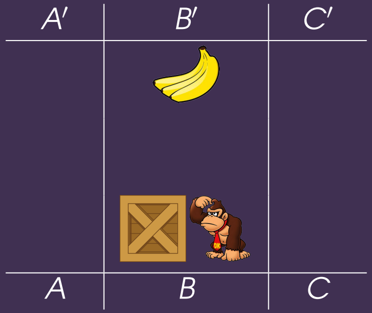
STRIPS example — Solution



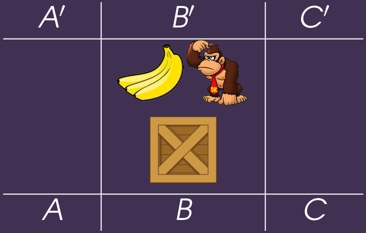
STRIPS example — Solution



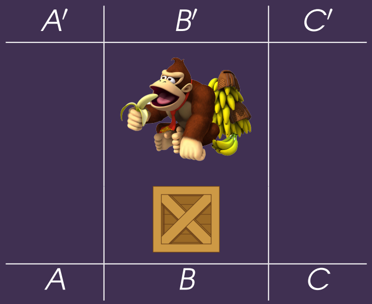
STRIPS example — Solution



STRIPS example — Solution



STRIPS example — Solution



Finding the solution

Finding the solution

- For a given state, we can construct a list of all **valid actions** based on their **preconditions**

Finding the solution

- For a given state, we can construct a list of all **valid actions** based on their **preconditions**
- We can also find the **next state** resulting from each action based on their **postconditions**

Finding the solution

- For a given state, we can construct a list of all **valid actions** based on their **preconditions**
- We can also find the **next state** resulting from each action based on their **postconditions**
- We can construct a **tree** of states and actions

Finding the solution

- For a given state, we can construct a list of all **valid actions** based on their **preconditions**
- We can also find the **next state** resulting from each action based on their **postconditions**
- We can construct a **tree** of states and actions
- We can then **search** this tree to find a goal state

Tree traversal

Tree traversal

procedure DEPTHFIRSTSEARCH

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack
    push root node onto S

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack
    push root node onto S
    while S is not empty **do**

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S

Tree traversal

```
procedure DepthFirstSearch
    let S be a stack
    push root node onto S
    while S is not empty do
        pop n from S
        push children of n onto S
```

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S
        push children of n onto S
    end while
end procedure

procedure BREADTHFIRSTSEARCH

Tree traversal

procedure DepthFirstSearch
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S
        push children of n onto S
    end while
end procedure

procedure BreadthFirstSearch
    let Q be a queue

Tree traversal

procedure DepthFirstSearch
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S
        push children of n onto S
    end while
end procedure

procedure BreadthFirstSearch
    let Q be a queue
    enqueue root node into Q

Tree traversal

procedure DEPTHFIRSTSEARCH
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S
        push children of n onto S
    end while
end procedure

procedure BREADTHFIRSTSEARCH
    let Q be a queue
    enqueue root node into Q
    while Q is not empty **do**

Tree traversal

procedure DepthFirstSearch
    let *S* be a stack
    push root node onto *S*
    while *S* is not empty **do**
        pop *n* from *S*
        push children of *n* onto *S*
    end while
end procedure

procedure BreadthFirstSearch
    let *Q* be a queue
    enqueue root node into *Q*
    while *Q* is not empty **do**
        dequeue *n* from *Q*

Tree traversal

procedure DepthFirstSearch
    let S be a stack
    push root node onto S
    while S is not empty **do**
        pop n from S
        push children of n onto S
    end while
end procedure

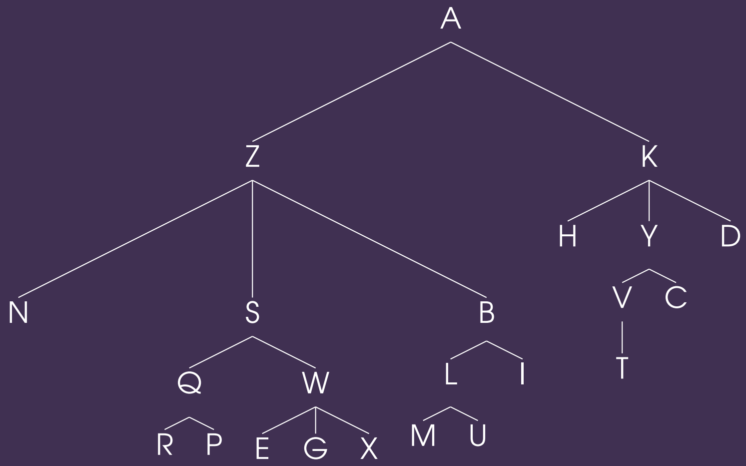
procedure BreadthFirstSearch
    let Q be a queue
    enqueue root node into Q
    while Q is not empty **do**
        dequeue n from Q
        enqueue children of n into Q

Tree traversal

procedure DepthFirstSearch
  let *S* be a stack
  push root node onto *S*
  while *S* is not empty **do**
    pop *n* from *S*
    push children of *n* onto *S*
  end while
end procedure

procedure BreadthFirstSearch
  let *Q* be a queue
  enqueue root node into *Q*
  while *Q* is not empty **do**
    dequeue *n* from *Q*
    enqueue children of *n* into *Q*
  end while
end procedure

Tree traversal example



FALMOUTH
UNIVERSITY

Assignment check-in

AI component

- Assignment brief on LearningSpace
- For **next week**: prepare your **proposal**

Research journal

Final check-in