COMP702: Classical Artificial Intelligence

8: Planning

FALMOUTH
UNIVERSITY

STRIPS

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:

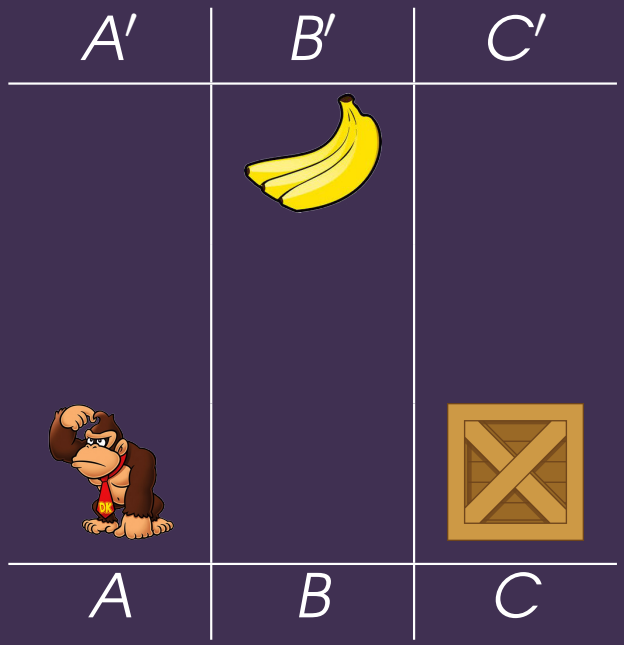
STRIPS planning

- **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)

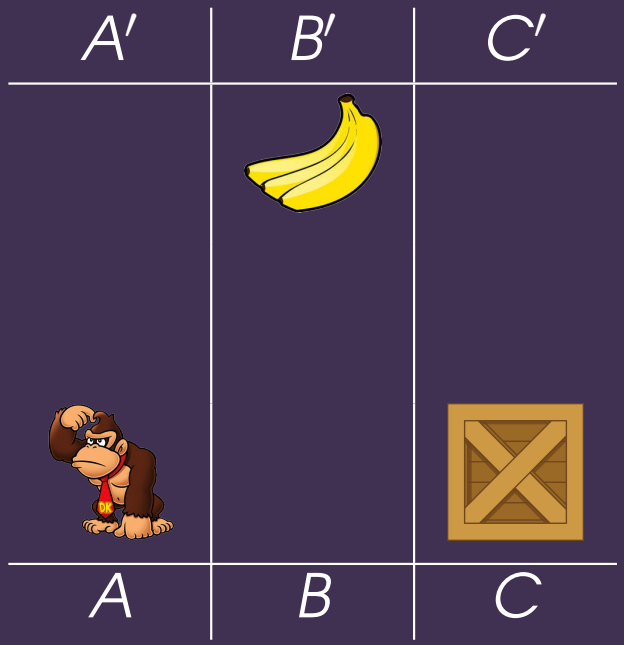
STRIPS planning

- **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



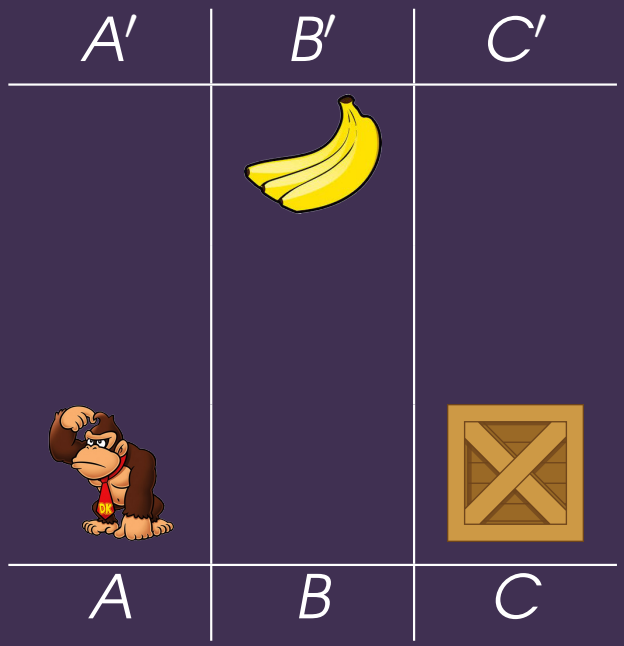
STRIPS example



Initial state:

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

STRIPS example



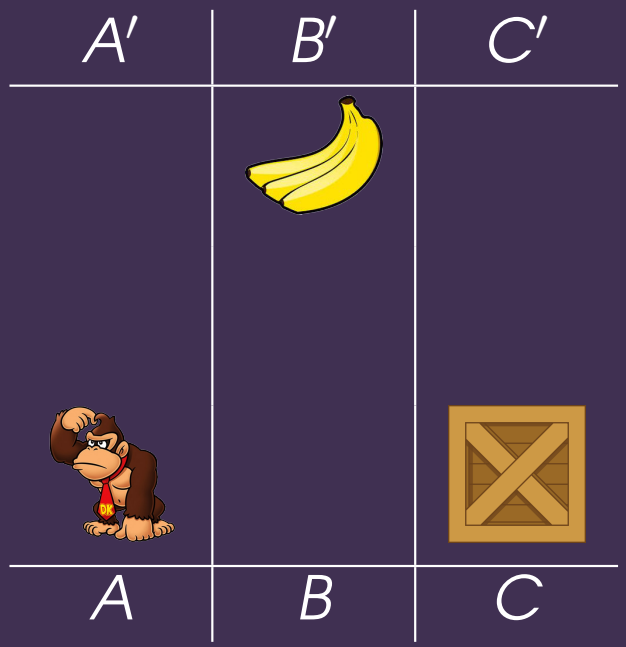
Initial state:

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

Goal:

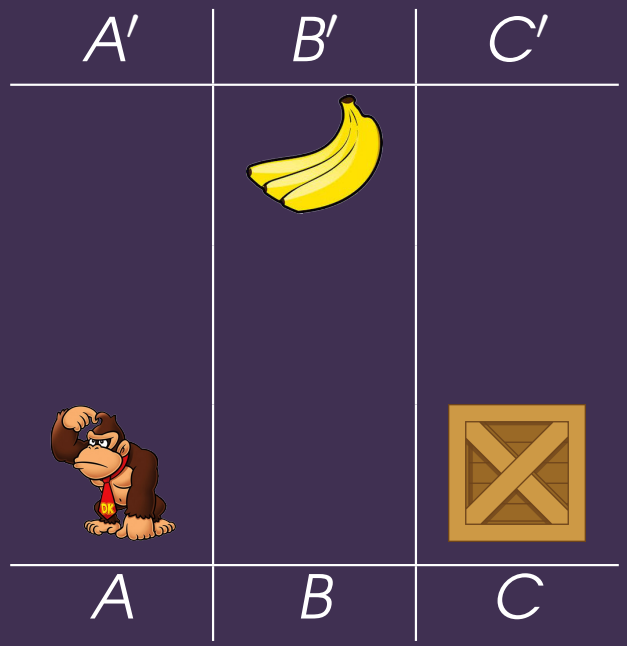
```
HasBananas
```

STRIPS example — Actions

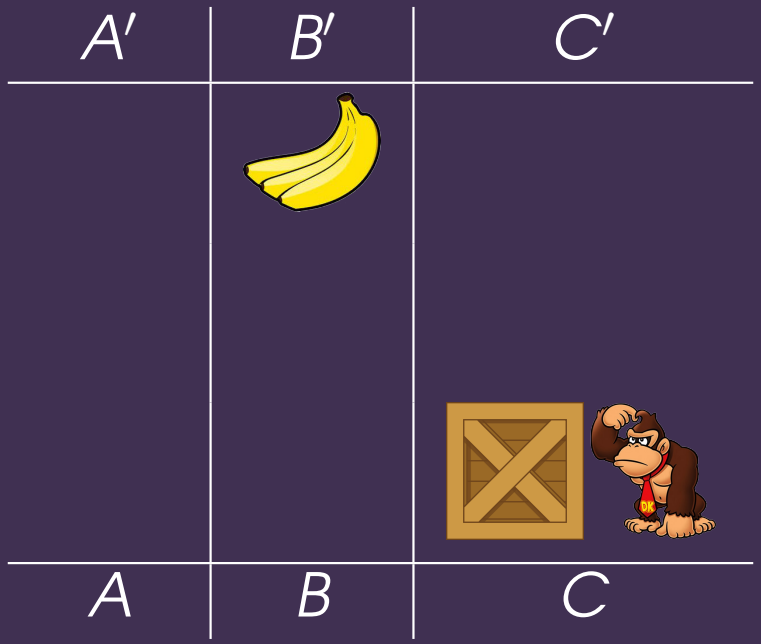



```
Move(x, y)
  Pre:  At(x)
  Post: !At(x), At(y)

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
```

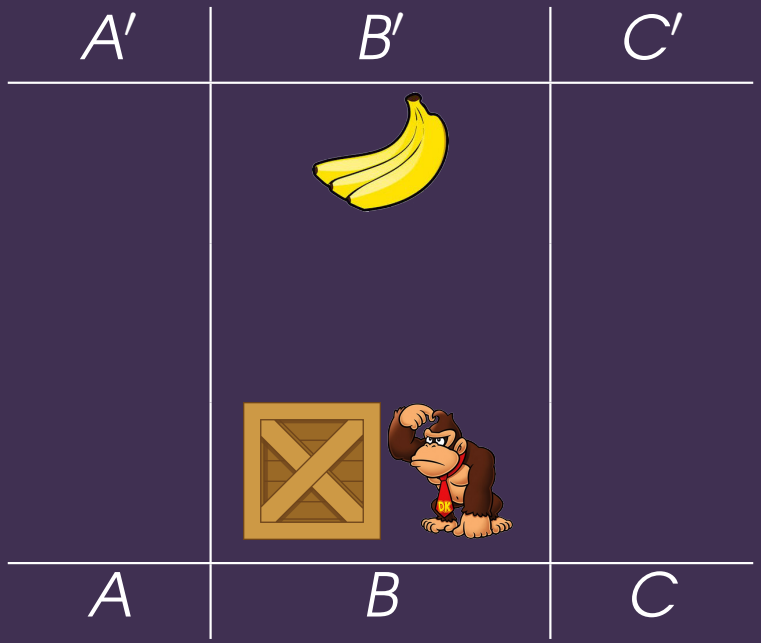
STRIPS example — Solution



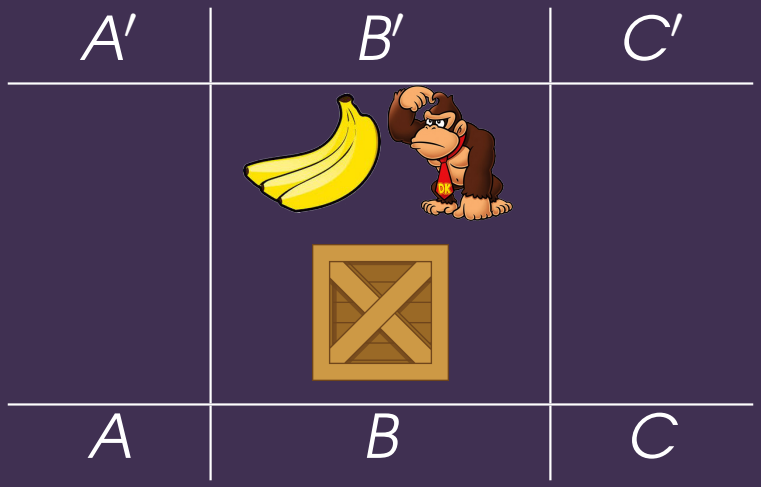
STRIPS example — Solution



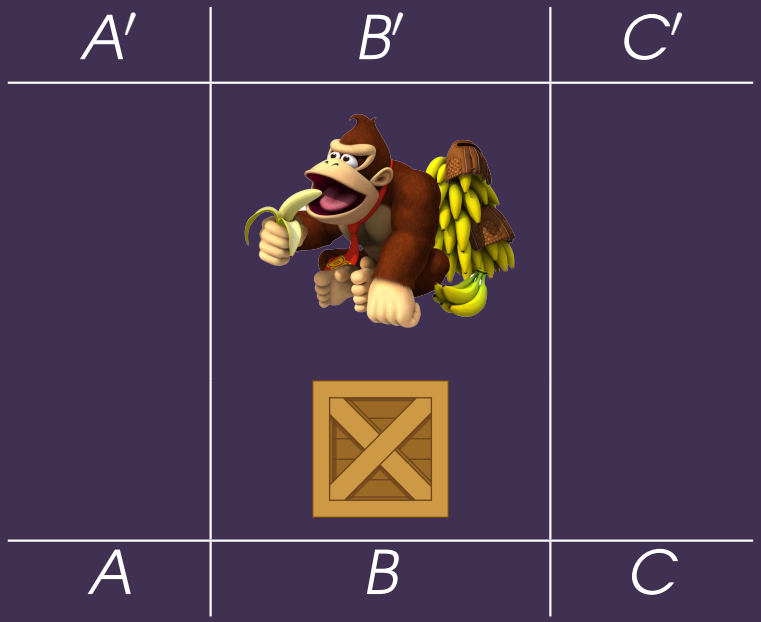
STRIPS example — Solution



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

FALMOUTH
UNIVERSITY

GOAP

GOAP

GOAP

- **G**oal **O**riented **A**ction **P**lanning

GOAP

- **G**oal **O**riented **A**ction **P**lanning
- Originally developed for F.E.A.R. (2005), since used in several games

GOAP

- **G**oal **O**riented **A**ction **P**lanning
- Originally developed for F.E.A.R. (2005), since used in several games
- A modified version of STRIPS specifically for real-time planning in video games

GOAP

GOAP

- Each agent has a **goal set**

GOAP

- Each agent has a **goal set**
  - Multiple goals with differing **priority**

GOAP

- Each agent has a **goal set**
  - Multiple goals with differing **priority**
  - Goals are like in STRIPS — sets of predicates that the agent wants to satisfy

GOAP

- Each agent has a **goal set**
  - Multiple goals with differing **priority**
  - Goals are like in STRIPS — sets of predicates that the agent wants to satisfy
- Each agent also has a set of **actions**

GOAP

- Each agent has a **goal set**
  - Multiple goals with differing **priority**
  - Goals are like in STRIPS — sets of predicates that the agent wants to satisfy
- Each agent also has a set of **actions**
  - Like in STRIPS — actions have preconditions and postconditions

GOAP

- Each agent has a **goal set**
  - Multiple goals with differing **priority**
  - Goals are like in STRIPS — sets of predicates that the agent wants to satisfy
- Each agent also has a set of **actions**
  - Like in STRIPS — actions have preconditions and postconditions
  - Unlike STRIPS, each action also has a **cost**

Action sets

Action sets

- Different types of agent could have the **same goals** but **different action sets**

Action sets

- Different types of agent could have the **same goals** but **different action sets**
- This will result in those agents achieving those goals in **different ways**

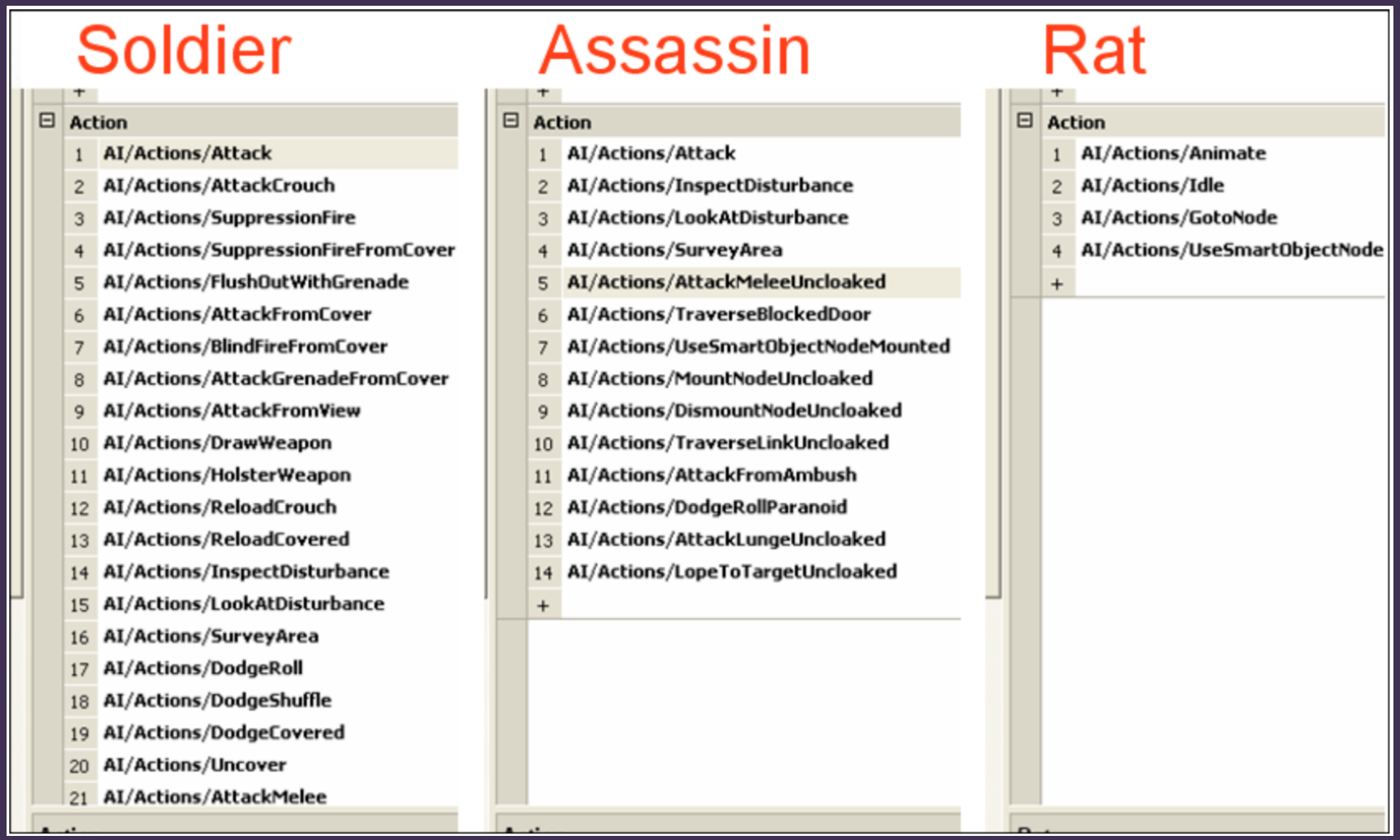
Action sets

- Different types of agent could have the **same goals** but **different action sets**
- This will result in those agents achieving those goals in **different ways**
- NB this doesn't have to be explicitly coded — it **emerges** from the GOAP system

Action sets

- Different types of agent could have the **same goals** but **different action sets**
- This will result in those agents achieving those goals in **different ways**
- NB this doesn't have to be explicitly coded — it **emerges** from the GOAP system
- E.g. this was used by the F.E.A.R. team to quickly add new enemy types

Action sets


Soldier
Action
1 AI/Actions/Attack
2 AI/Actions/AttackCrouch
3 AI/Actions/SuppressionFire
4 AI/Actions/SuppressionFireFromCover
5 AI/Actions/FlushOutWithGrenade
6 AI/Actions/AttackFromCover
7 AI/Actions/BlindFireFromCover
8 AI/Actions/AttackGrenadeFromCover
9 AI/Actions/AttackFromView
10 AI/Actions/DrawWeapon
11 AI/Actions/HolsterWeapon
12 AI/Actions/ReloadCrouch
13 AI/Actions/ReloadCovered
14 AI/Actions/InspectDisturbance
15 AI/Actions/LookAtDisturbance
16 AI/Actions/SurveyArea
17 AI/Actions/DodgeRoll
18 AI/Actions/DodgeShuffle
19 AI/Actions/DodgeCovered
20 AI/Actions/Uncover
21 AI/Actions/AttackMelee
Assassin
Action
1 AI/Actions/Attack
2 AI/Actions/InspectDisturbance
3 AI/Actions/LookAtDisturbance
4 AI/Actions/SurveyArea
5 AI/Actions/AttackMeleeUncloaked
6 AI/Actions/TraverseBlockedDoor
7 AI/Actions/UseSmartObjectNodeMounted
8 AI/Actions/MountNodeUncloaked
9 AI/Actions/DismountNodeUncloaked
10 AI/Actions/TraverseLinkUncloaked
11 AI/Actions/AttackFromAmbush
12 AI/Actions/DodgeRollParanoid
13 AI/Actions/AttackLungeUncloaked
14 AI/Actions/LopeToTargetUncloaked
Rat
Action
1 AI/Actions/Animate
2 AI/Actions/Idle
3 AI/Actions/GotoNode
4 AI/Actions/UseSmartObjectNode

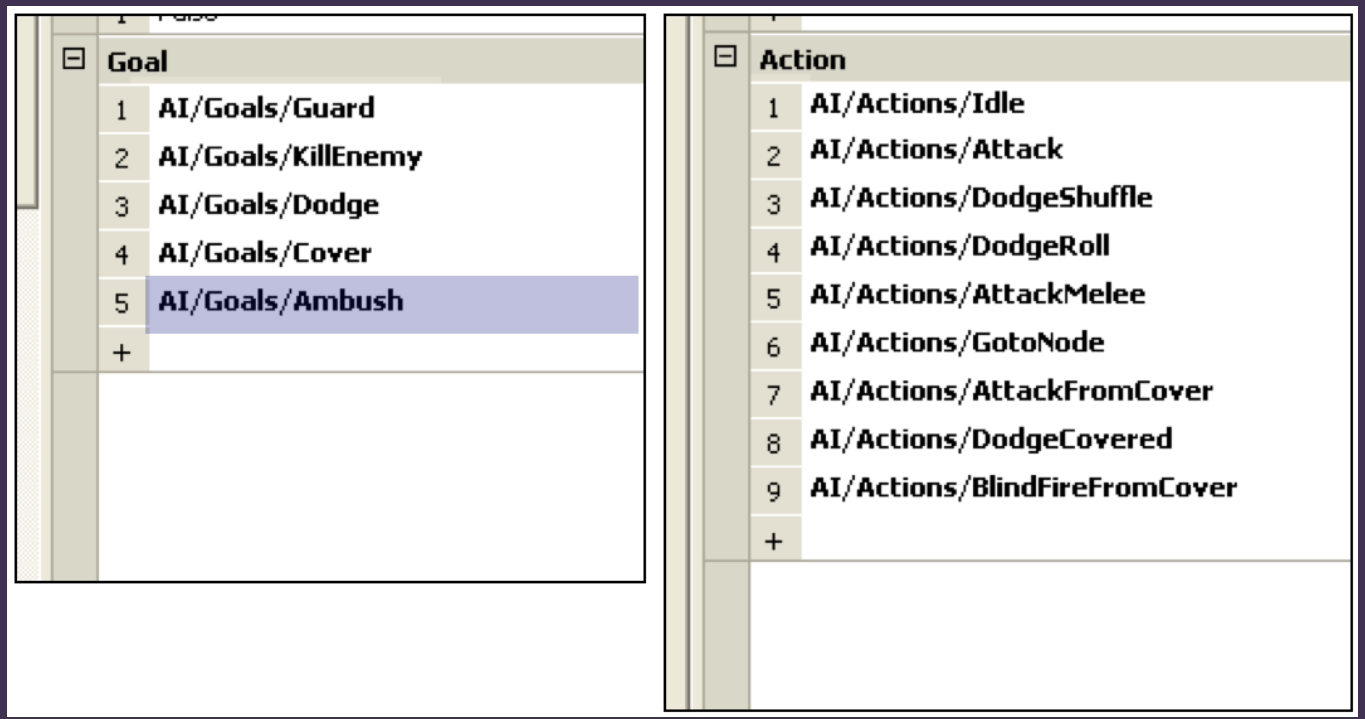
Layering

Layering

- Goal set allows different behaviours with different priorities to be **layered**

Layering

- Goal set allows different behaviours with different priorities to be **layered**
- E.g. enemy AI in F.E.A.R.:



Implementing GOAP

Implementing GOAP

- An **abstracted** view of the game world is used for planning

Implementing GOAP

- An **abstracted** view of the game world is used for planning
- Represented as a fixed-length array (or struct) of values

Implementing GOAP

- An **abstracted** view of the game world is used for planning
- Represented as a fixed-length array (or struct) of values
- Predicates (preconditions, postconditions, goals) represented in terms of this array representation

Implementing GOAP

- An **abstracted** view of the game world is used for planning
- Represented as a fixed-length array (or struct) of values
- Predicates (preconditions, postconditions, goals) represented in terms of this array representation
- Most implementations also allow for **programmatic** preconditions (e.g. calling the pathfinding system to check availability of a path)

Implementing GOAP

Implementing GOAP

- Not difficult to implement

Implementing GOAP

- Not difficult to implement
- Open-source implementations do exist

Implementing GOAP

- Not difficult to implement
- Open-source implementations do exist
- Not built into Unity or Unreal, but asset store packages are available

Finding the plan

Finding the plan

- As in STRIPS, we can build a **tree** whose nodes are world states and edges are available actions

Finding the plan

- As in STRIPS, we can build a **tree** whose nodes are world states and edges are available actions
- Since actions have costs, we can use A* to find the lowest cost path to the goal

Finding the plan

- As in STRIPS, we can build a **tree** whose nodes are world states and edges are available actions
- Since actions have costs, we can use A* to find the lowest cost path to the goal
- Plan is a **queue** of actions that the agent then executes

Finding the plan

- As in STRIPS, we can build a **tree** whose nodes are world states and edges are available actions
- Since actions have costs, we can use A* to find the lowest cost path to the goal
- Plan is a **queue** of actions that the agent then executes
- If the plan is interrupted or fails then the agent can **replan**

GOAP vs behaviour trees

GOAP vs behaviour trees

- BT: Designer specifies “how”

GOAP vs behaviour trees

- BT: Designer specifies “how”
- GOAP: Designer specifies “what” — “how” is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)

GOAP vs behaviour trees

- BT: Designer specifies “how”
- GOAP: Designer specifies “what” — “how” is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)
- Both: actions (tasks in BT) are modular and reusable between agents

GOAP vs behaviour trees

- BT: Designer specifies “how”
- GOAP: Designer specifies “what” — “how” is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)
- Both: actions (tasks in BT) are modular and reusable between agents
- GOAP: goals are also modular and reusable

GOAP vs behaviour trees

- BT: Designer specifies “how”
- GOAP: Designer specifies “what” — “how” is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)
- Both: actions (tasks in BT) are modular and reusable between agents
- GOAP: goals are also modular and reusable
- BT: goals are not represented explicitly

GOAP vs behaviour trees

- BT: Designer specifies "how"
- GOAP: Designer specifies "what" — "how" is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)
- Both: actions (tasks in BT) are modular and reusable between agents
- GOAP: goals are also modular and reusable
- BT: goals are not represented explicitly
- BT can be classified as **authored behaviour**

GOAP vs behaviour trees

- BT: Designer specifies "how"
- GOAP: Designer specifies "what" — "how" is in whatever system is used to implement actions (FSMs in F.E.A.R.; could use BTs or hand coding)
- Both: actions (tasks in BT) are modular and reusable between agents
- GOAP: goals are also modular and reusable
- BT: goals are not represented explicitly
- BT can be classified as **authored behaviour**
- GOAP can be classified as **computational intelligence**

FALMOUTH
UNIVERSITY

Workshop