Reasoning

- Deductive Reasoning
  - Deliberative Agent

    Traditionally, AI systems were built using symbolic AI, which is the manipulation of symbols and logical formulae. This approach can be used to make agents. They will need a symbolic representation of the environment and of their desired behaviour and will be using syntactic manipulation (like theorem proving and logical deduction).

    A deliberative agent is an agent that contains an explicitly represented symbolic model of the world and makes decision via symbolic reasoning. This will lead to some problems

    - Transduction problem

      Identifying objects is a hard job for agents. This problem consists on how do we translate real world entities into an accurate and adequate symbolic description.

      There is a lot of research in this area, like vision, speech understanding, learning…

    - Representation/reasoning problem

      Representing objects is a difficult problem. This problem encompasses how do we symbolically represent information about complex real-world entities, like the relationships between entities.

      It also encompasses how to get agents to manipulate and reason with this representation.

      This has led to research in knowledge representation, automated reasoning and planning.

    Neither of these problems are considered nearly solved, and most algorithms of interest are highly intractable, meaning they require an amount of resources way too high, being NP-HARD, for example.

  - Deductive-reasoning Agent

    We have a set of rules Rho

$$\mathrm{R} = \{\rho_1, \rho_2, ...\}$$

We have a database Delta that holds all the perceived information about the world (similar to an internal state)

$$\Delta = \{\delta_1, \delta_2, ...\}$$

We have a set of Actions which we know about

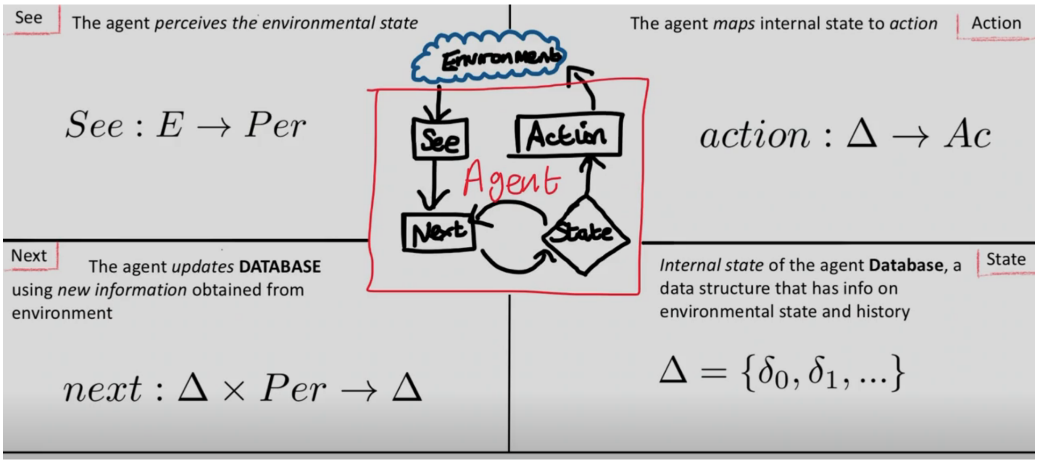
$$Ac = \{\alpha_1, \alpha_2, ...\}$$

The information in the database can (or not) allow an action phi to happen.

▼ Agent Control Loop

We need to alter the Agent Control Loop to allow for Deductive-reasoning Agents.

It will now look like the following



The state changes to be a database.

The next function has to change to both take in and update the database.

The Action function becomes the following

Algorithm 1 Action Funcion

for $\alpha \in Ac$ **do**
    if $\Delta \vdash_\rho Do(\alpha)$ **then** return α
for $\alpha \in Ac$ **do**
    if $\Delta \nvdash_\rho \neg Do(\alpha)$ **then** return α
return null

The first for loop looks for an action prescribed by the database and the second for loop looks for an action that is not excluded by the database.

▼ Example

Imagine we have a vacuum world. Our agent is a hoover that traverses and cleans a room, represented by a 3×3 grid. We have the following domain predicates an action.

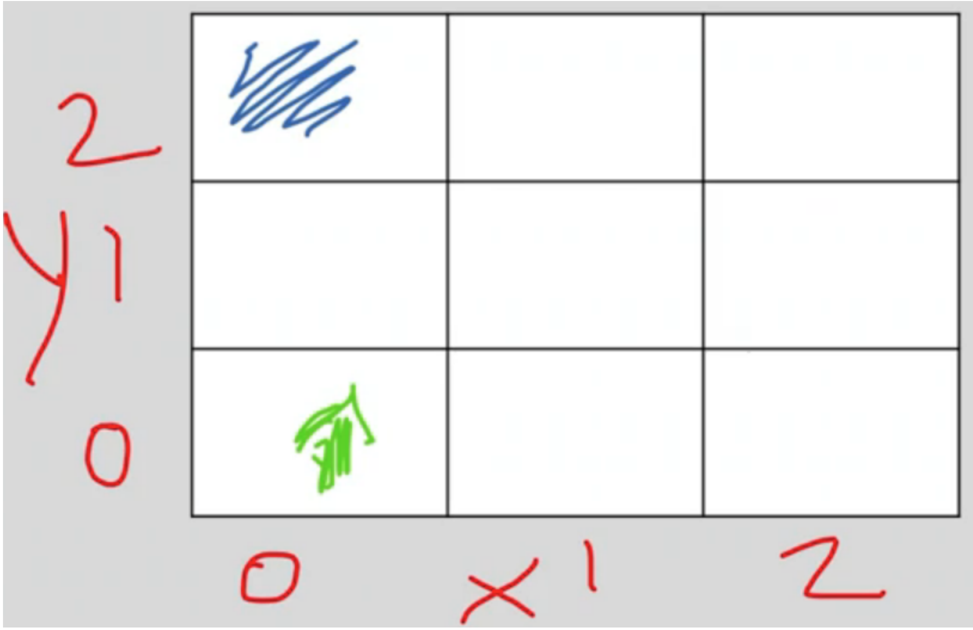
$In(x, y)$ – Agent is in tile (x, y)
$Dirt(x, y)$ – There is dirt in tile (x, y)
$Facing(d)$ – Agent is facing the direction $d \in \{N, S, E, W\}$
Domain Predicates
$Ac = \{turnRight, forward, clean\}$ — Actions

Remember a domain predicate is something that can be true or false.

We have the following set Rho of rules

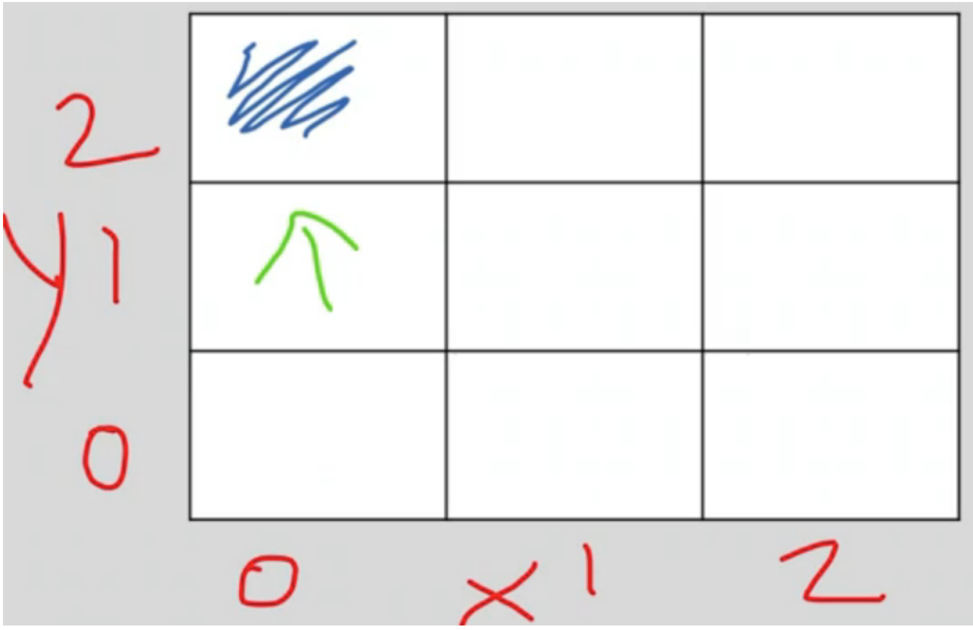
$$
\begin{aligned}
\mathrm{P} = \{&In(x,y) \wedge Dirt(x,y) \rightarrow Do(Clean),\\
&In(0,0) \wedge Facing(N) \wedge \neg Dirt(0,0) \rightarrow Do(Forward),\\
&In(0,1) \wedge Facing(N) \wedge \neg Dirt(0,1) \rightarrow Do(Forward),\\
&In(0,2) \wedge Facing(N) \wedge \neg Dirt(0,2) \rightarrow Do(turnRight),\\
&In(0,2) \wedge Facing(E) \wedge \neg Dirt(0,2) \rightarrow Do(turnRight),\\
&In(0,2) \wedge Facing(S) \wedge \neg Dirt(0,2) \rightarrow Do(Forward), ...\}
\end{aligned}
$$

The initial state of the environment is



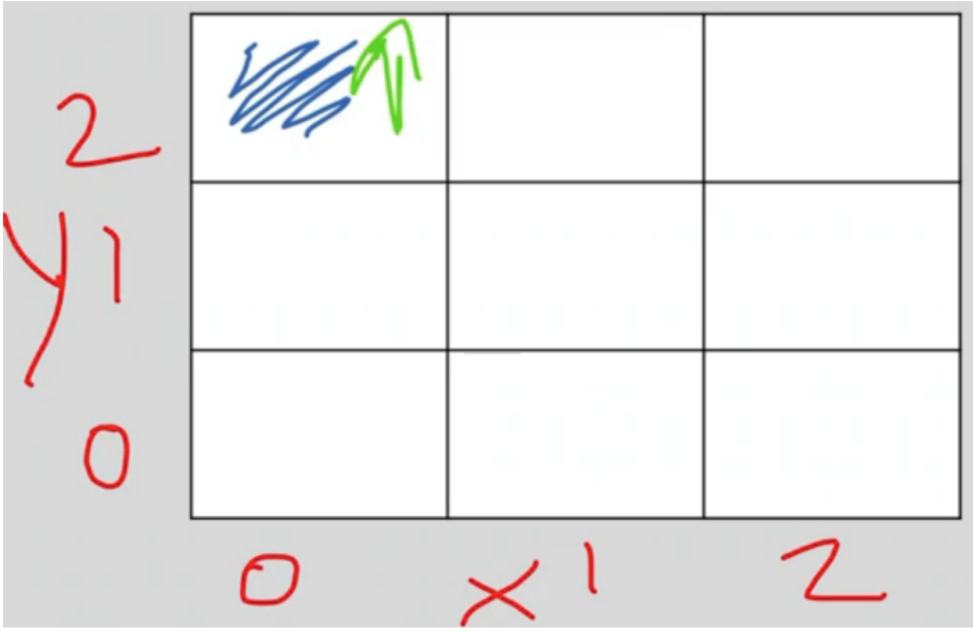
with the agent in (0,0) facing North and dirt in (0,2).

This matches the second rule, so we move forward. The new state of the environment is

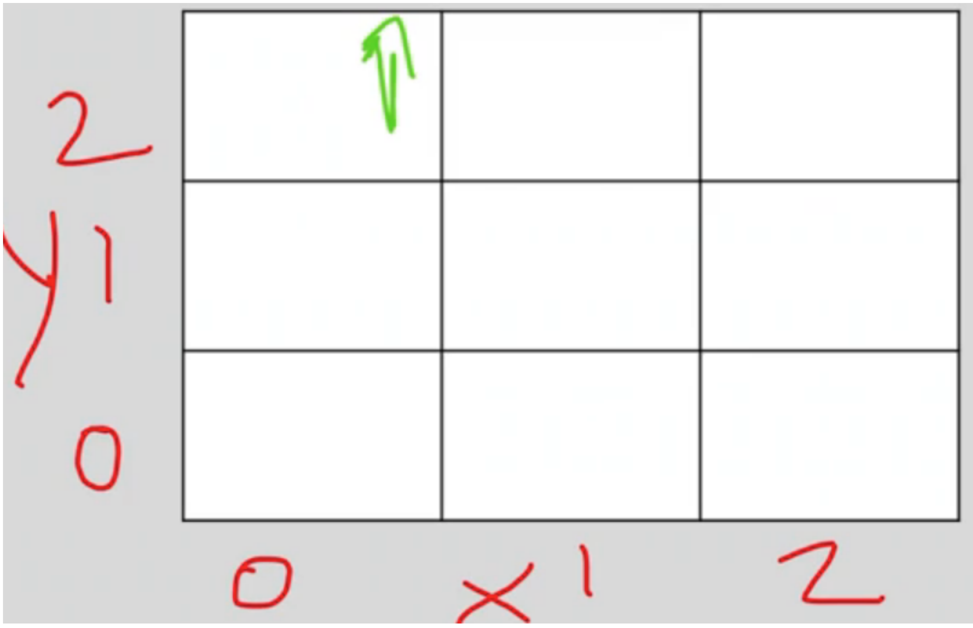


This also matches the second rule, so we move forward again.

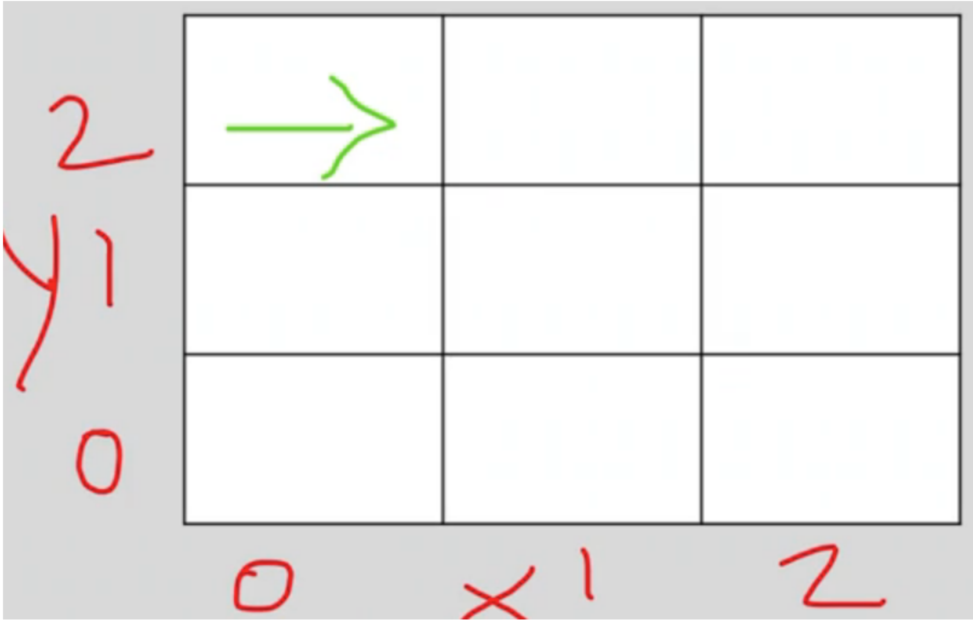
The new state is



The agent is in (0,2) and there is dirt in (0,2), which matches the first rule, therefore we do the "clean" action. The new state is



This matches the forth rule, so we turn right



This matches the fifth rule, so we turn right again.

And so on.

- Practical reasoning

  We are going to introduce formal notation for the following

  - Agent

    An agent has a set of Actions

    $$Ac = \{\alpha_1, \alpha_2, ...\}$$

    Each action has a structure

    - Preconditions: things that have to be true to carry out the action

      $$P_{\alpha_i}$$

    - Delete list: things that will no longer be true once the action is done

      $$D_{\alpha_i}$$

    - Add list: things that will become true once the action is carried out.

$$A_{\alpha_i}$$

We can represent an action as a triple

$$\alpha_i = \langle P_{alpha_i}, D_{alpha_i}, A_{alpha_i} \}$$

▼ Planning Problem

A planning problem is represented by a triple of

- A set of initial beliefs about the world

$$B_0$$

- A set of actions we can carry out

$$Ac$$

- An intention (goal state)

$$I$$

The formal representation of that is

$$\langle B_0, Ac, I \rangle$$

This outputs a plan, which is a sequence of actions. It's represented by pi

$$\pi = \{\alpha, \alpha', ...\}$$

Each of these actions have to be in the set Ac.

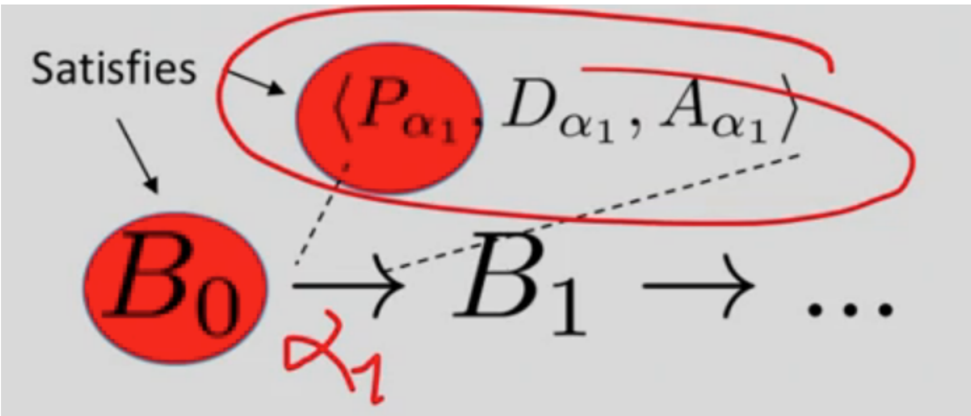
$$\alpha \in Ac$$

Every time we carry out an action it changes the world, so beliefs should also change as we carry out actions.

A plan pi is acceptable with regards to a particular planning problem if and only if:

- Our belief about the world satisfies the preconditions for an action

Imagine we want to carry out action alpha



The precondition list P_alpha_i has to be satisfied by the beliefs of the world B_0. This is expressed by

$$\forall 1 \leq j \leq n : B_{j-1} | = P_{\alpha_i}$$

For all j between 1 and n, the belief of the world before we carry out the action satisfies the preconditions of the action we wish to carry out.

A plan is acceptable if all the actions we want to carry out are possible.

- A plan is correct if it's acceptable (each action is possible at the point we wish to carry it out) and our final set of beliefs of the world (after carrying all the actions) satisfies our intentions, expressed as

$$B_n \models I$$