Escola Tècnica Superior d'Enginyeria
Universitat Rovira i Virgili

# LECTURE 3: Deductive Reasoning Agents

Introduction to Multi-Agent Systems (MESIIA, MIA)

URV

# Agent Architectures

**Pattie Maes (1991)**

"… [A] particular methodology for building [agents]. It specifies how . . . the agent can be decomposed into the construction of a set of component modules and how these modules should be made to interact. The total set of modules and their interactions has to provide an answer to the question of how the sensor data and the current internal state of the agent determine the actions . . . and future internal state of the agent. An architecture encompasses **techniques** and **algorithms** that support this methodology …"

**Leslie Kaebling (1991)**

"… [A] specific collection of software (or hardware) modules, typically designated by boxes with arrows indicating the **data and control flow** among the modules. A more abstract view of an architecture is as a general methodology for designing particular modular decompositions for particular tasks …"

# Classes of Architecture

- 1956–present: Symbolic Reasoning Agents
  - Agents make decisions about what to do via symbol manipulation.
  - Its purest expression, proposes that agents use explicit logical reasoning in order to decide what to do.

- 1985–present: Reactive Agents
  - Problems with symbolic reasoning led to a reaction against this
  - led to the reactive agents movement

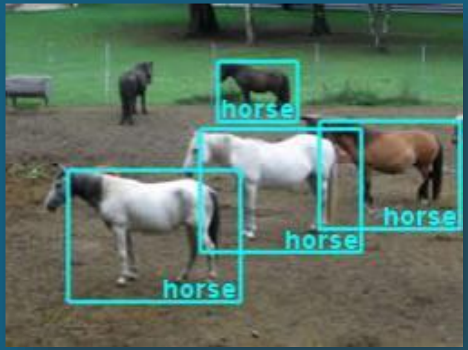
- 1990-present: Hybrid Agents
  - Hybrid architectures attempt to combine the best of reasoning and reactive architectures

# Symbolic Reasoning Agents

- The classical approach to building agents is to view them as a particular type of knowledge-based system and bring all the associated methodologies of such systems to bear.
  - This paradigm is known as symbolic AI.
- We define a deliberative agent or agent architecture to be one that:
  - contains an explicitly represented, symbolic model of the world;
  - makes decisions (for example about what actions to perform) via symbolic reasoning.

# Two issues

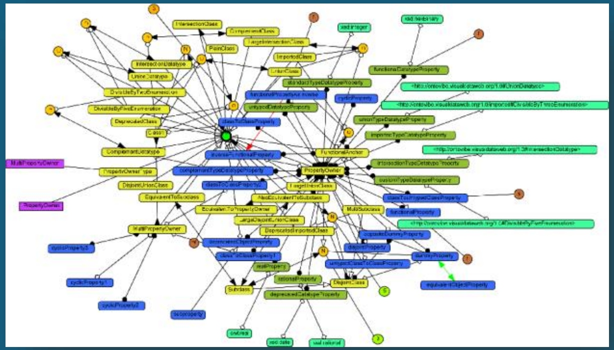
## The Transduction Problem
## Identifying objects is hard!!!




The transduction problem is that of translating the real world into an accurate, adequate symbolic description, in time that description to be useful.

This has led onto research into vision, speech, language understanding, etc.

## The Representation/Reasoning Problem
## Representing objects is harder!!!



How to symbolically represent information about complex real-world entities and processes, and how to get agents to reason with this information in time for the results to be useful.

This has led onto research into knowledge representation, automated reasoning, planning, etc.

Most researchers accept that neither problem is anywhere near solved.

# The representation / reasoning problem

- The underlying problem with knowledge representation/ reasoning lies with the complexity of symbol manipulation algorithms.
  - In general, many (most) search-based symbol manipulation algorithms of interest are highly intractable.
  - Hard to find compact representations.

- Because of these problems, some researchers have looked to alternative techniques for building agents; we look at these later.

# Deductive Reasoning Agents

- How can an agent decide what to do using theorem proving?
  - Basic idea is to use logic to encode a theory stating the best action to perform in any given situation.

- Let:
  - $\rho$ be the theory (typically a set of rules);
  - $\Delta$ be a logical database that describes the current state of the world;
  - $Ac$ be the set of actions the agent can perform;
  - $\Delta \vdash_{\rho} \varphi$ means that $\varphi$ can be proved from $\Delta$ using $\rho$.

# Deductive Reasoning Agents

- How does this fit into the abstract description we talked about last time?
  - The perception function is as before:

$$see: E \rightarrow Per$$

  - of course, this is (much) easier said than done.
  - The next state function revises the database Δ:

$$next: \Delta \times Per \rightarrow \Delta$$

  - And the action function?

# Action Function

for each $\alpha \in Ac$ do ; *try to find an action explicitly prescribed*
    if $\Delta \vdash_\rho Do(\alpha)$ then
        return $\alpha$
    end if
end for

for each $\alpha \in Ac$ do ; *try to find an action not excluded*
    if $\neg\Delta \vdash_\rho Do(\alpha)$ then
        return $\alpha$
    end if
end for

return *null* ; *no action found*

# An example: The Vacuum World

### *The Vacuum World*

The goal is for the robot to clear up all the dirt.

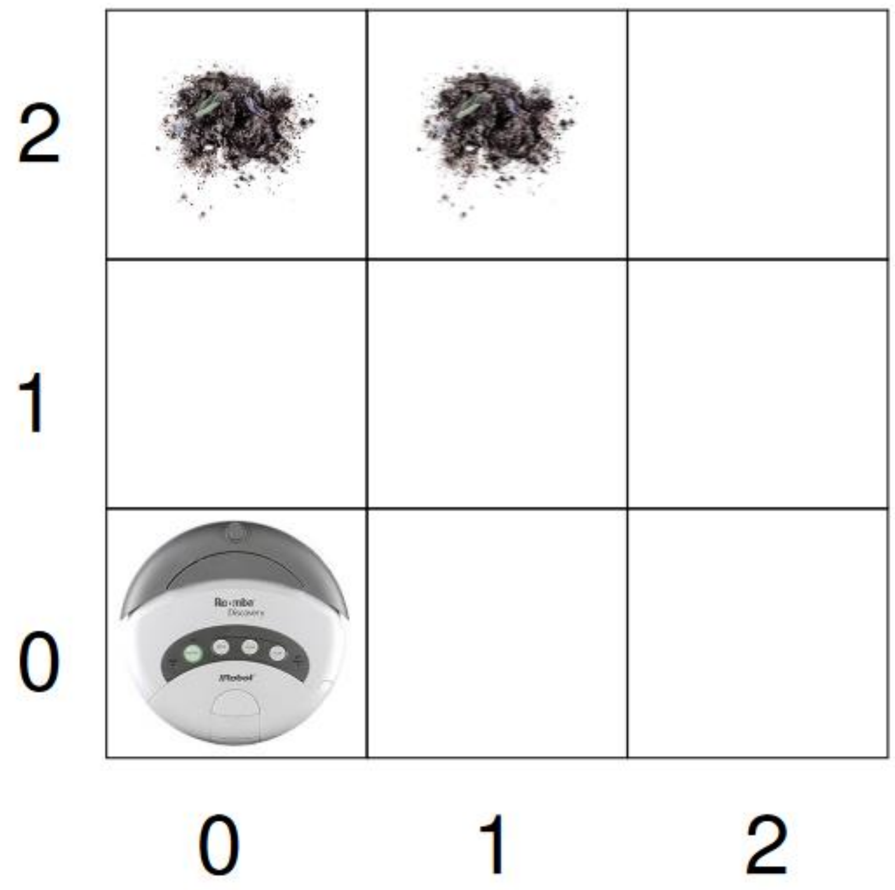
**Uses 3 domain predicates in this exercise:**

- $In(x, y)$ Agent is at $(x, y)$ cell
- $Dirt(x, y)$ There is a dirt at $(x, y)$
- $Facing(d)$ The agent is facing direction $d$

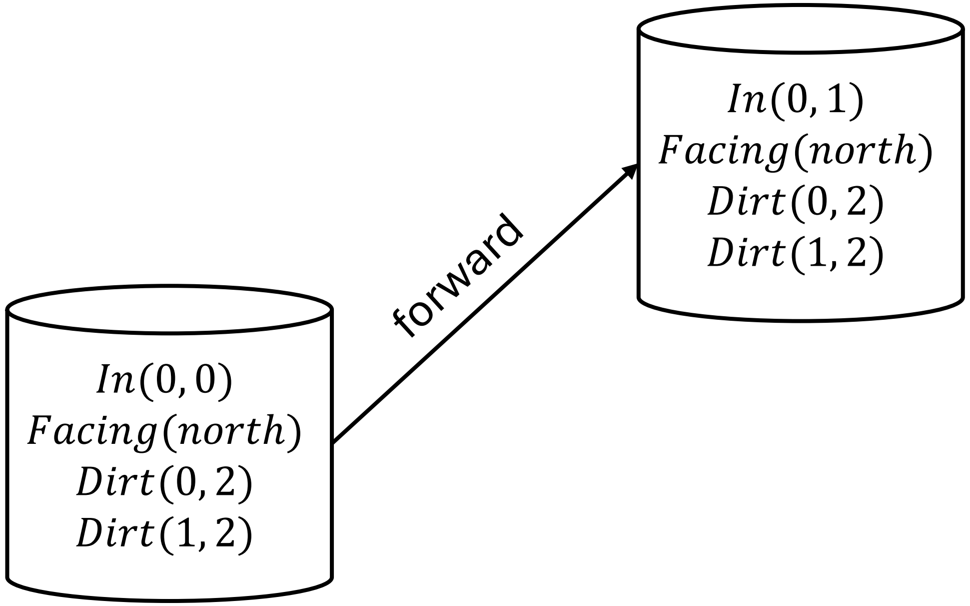
**Possible Actions:**

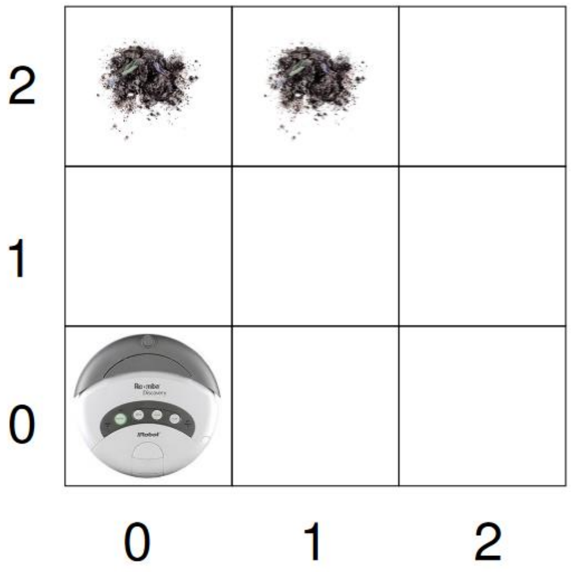
$Ac = \{turn, forward, suck\}$

Note: $turn$ means "$turn\ right$"

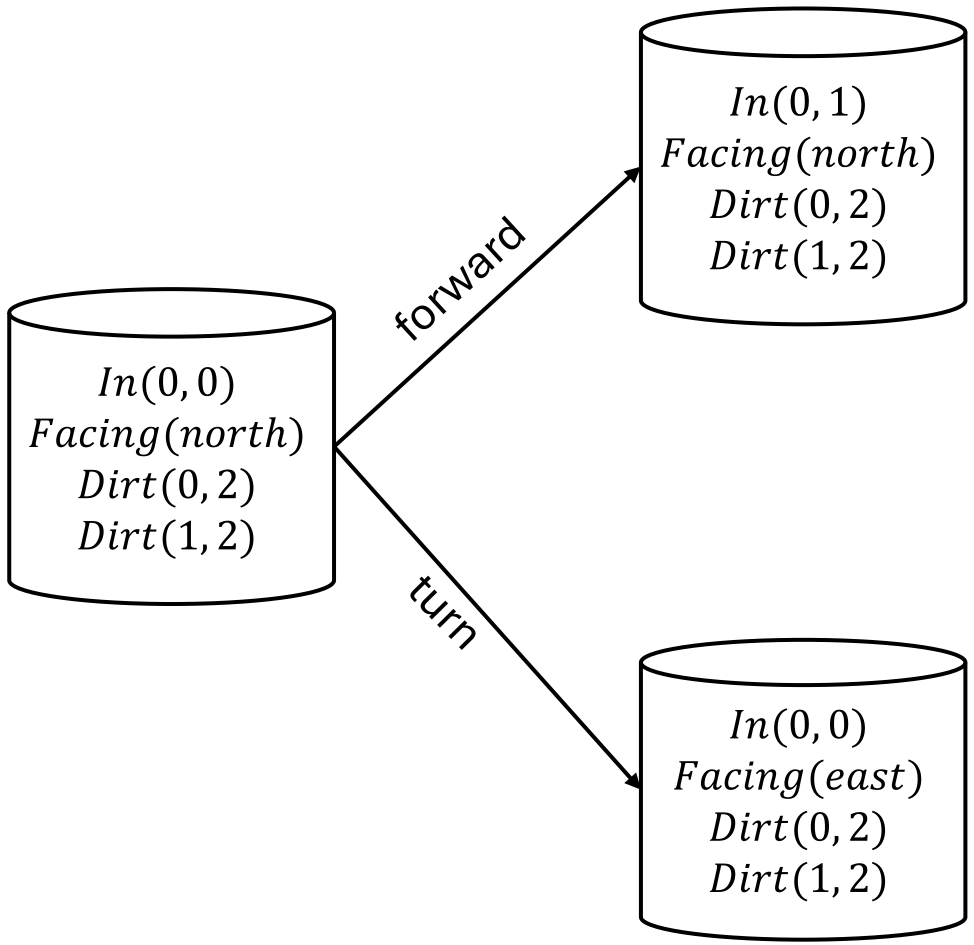


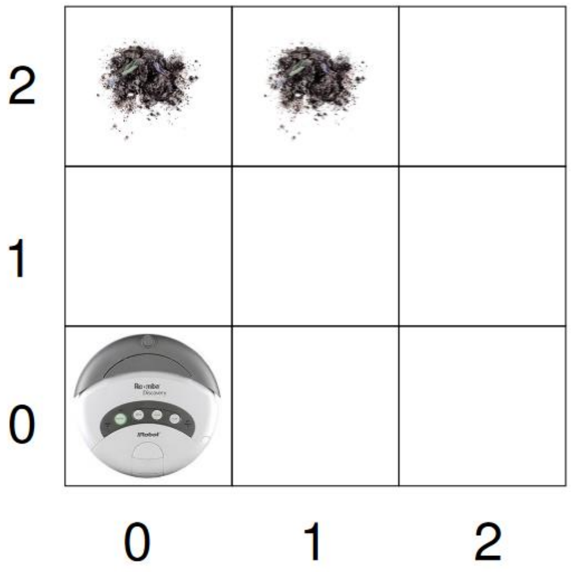
# The Vacuum World



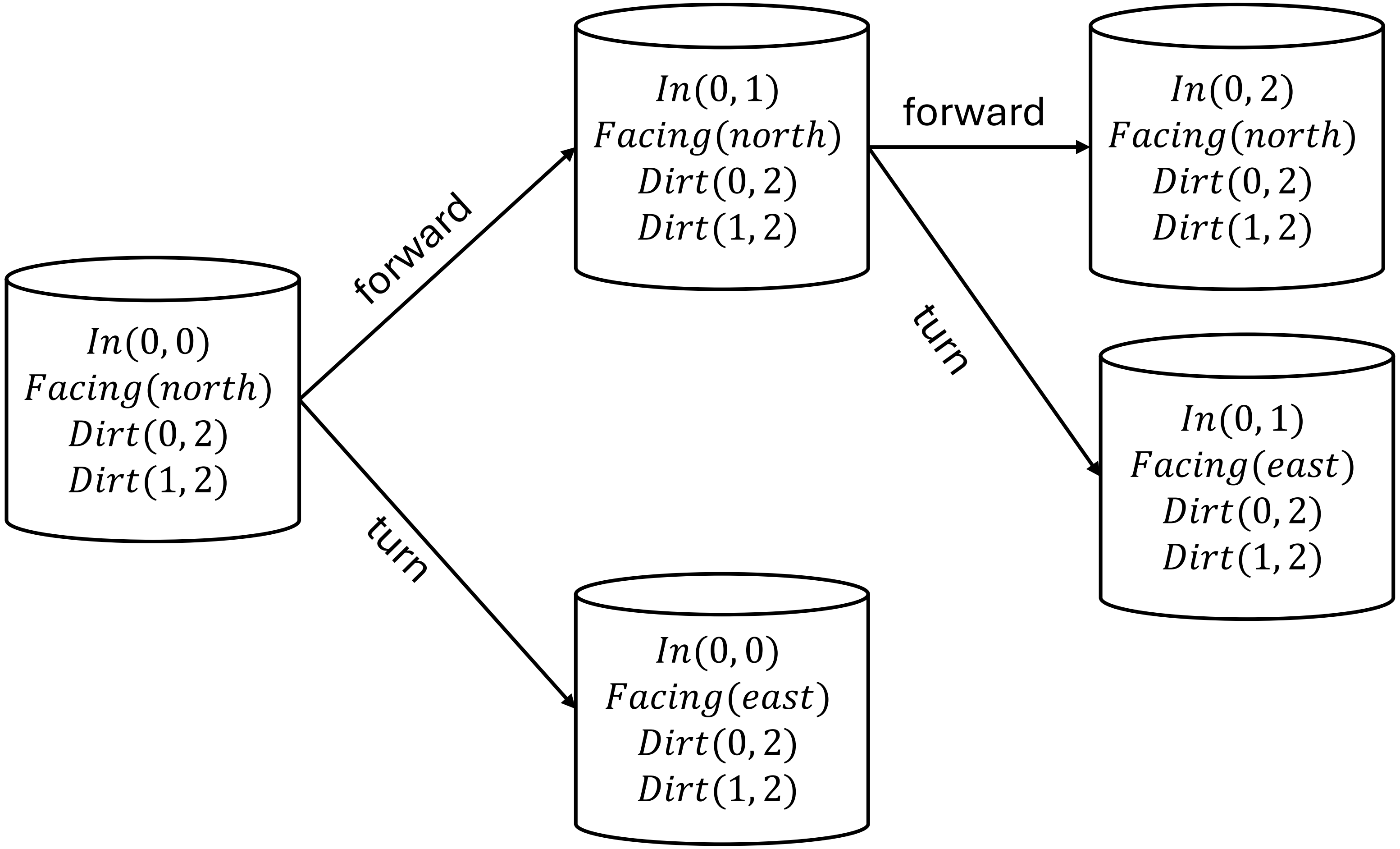



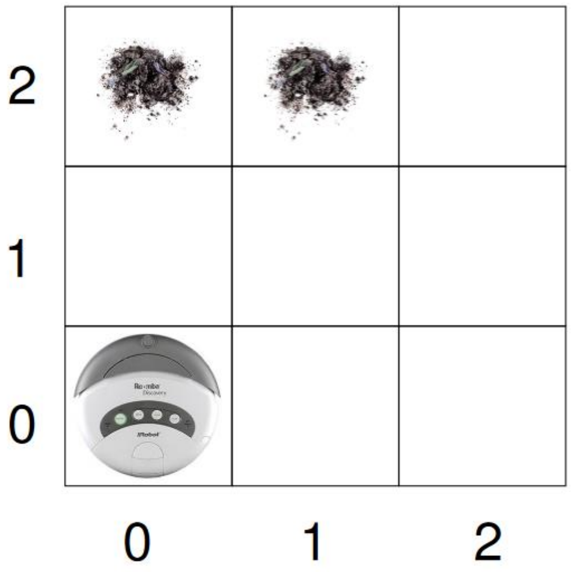
# The Vacuum World



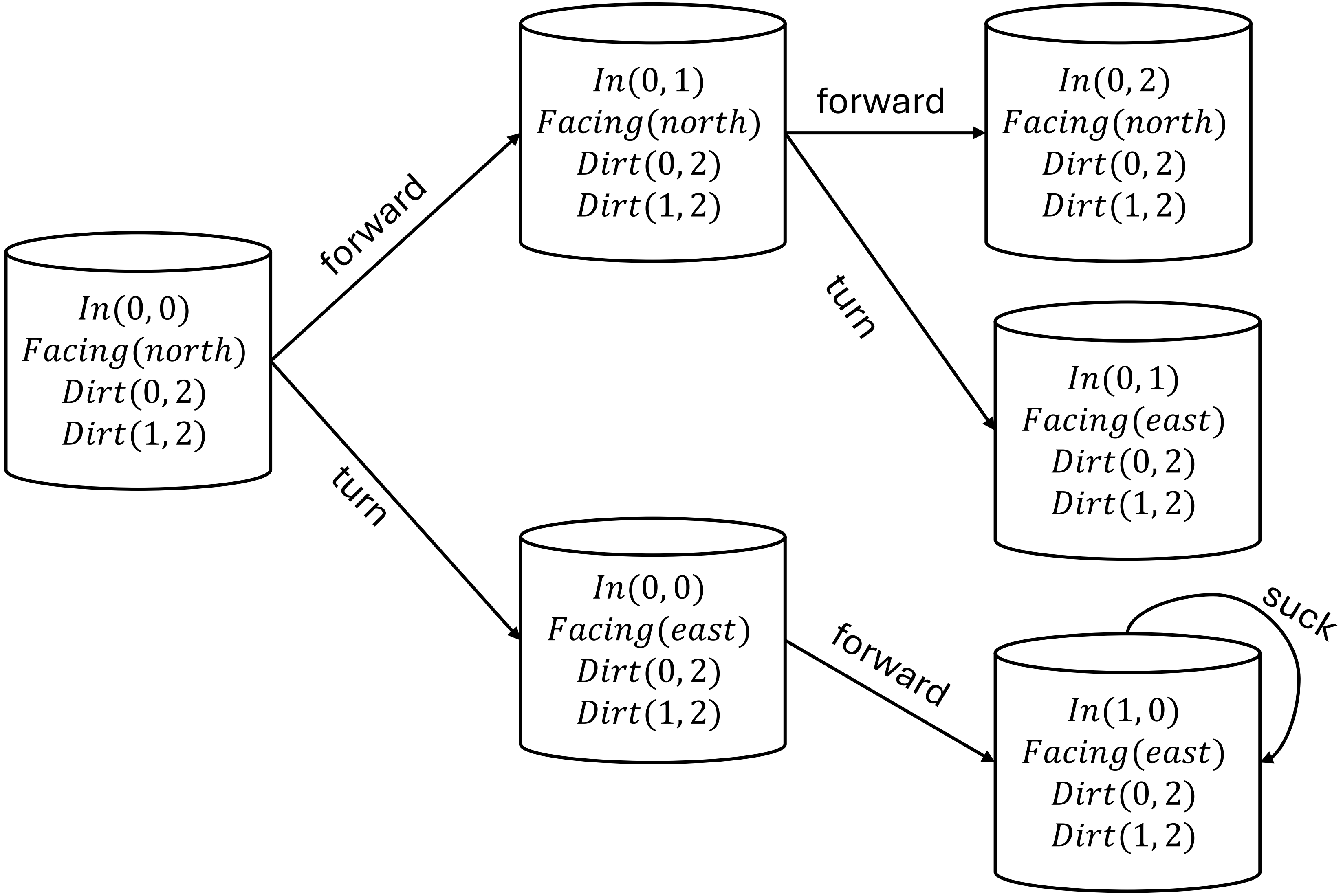



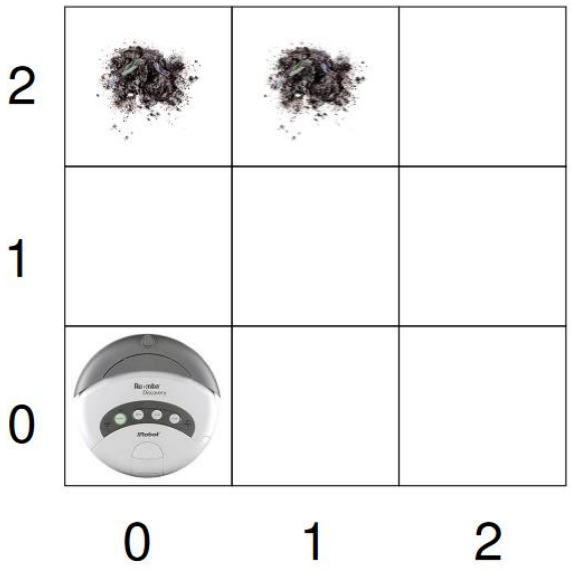
# The Vacuum World



# The Vacuum World

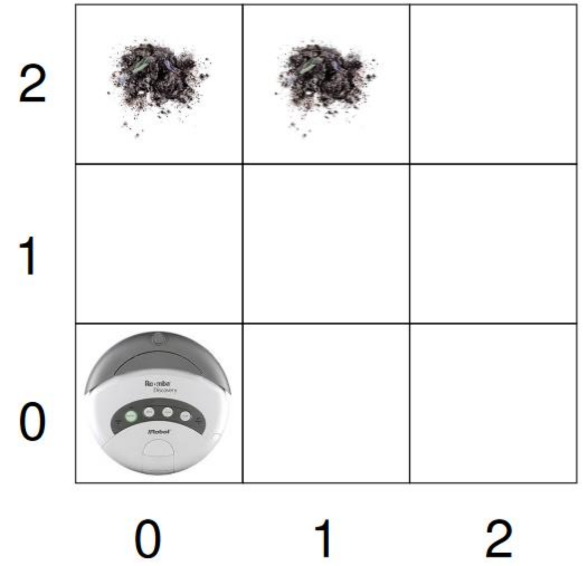


# The Vacuum World

- Rules ρ for determining what to do:

$$In(0,0) \land Facing(north) \land \neg Dirt(0,0) \rightarrow Do(forward)$$
$$In(0,1) \land Facing(north) \land \neg Dirt(0,1) \rightarrow Do(forward)$$
$$In(0,2) \land Facing(north) \land \neg Dirt(0,2) \rightarrow Do(forward)$$
$$In(0,2) \land Facing(east) \rightarrow Do(forward)$$

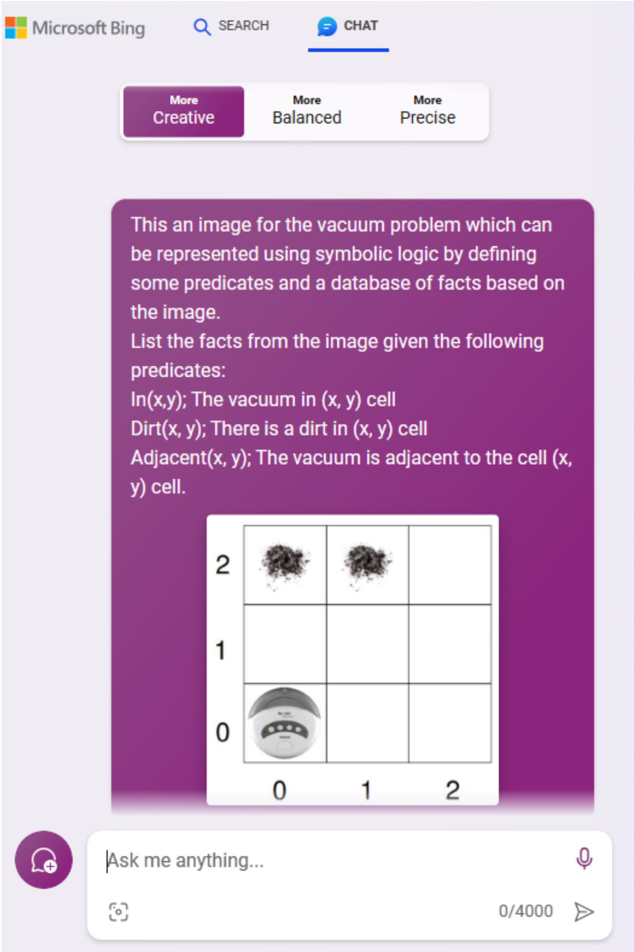


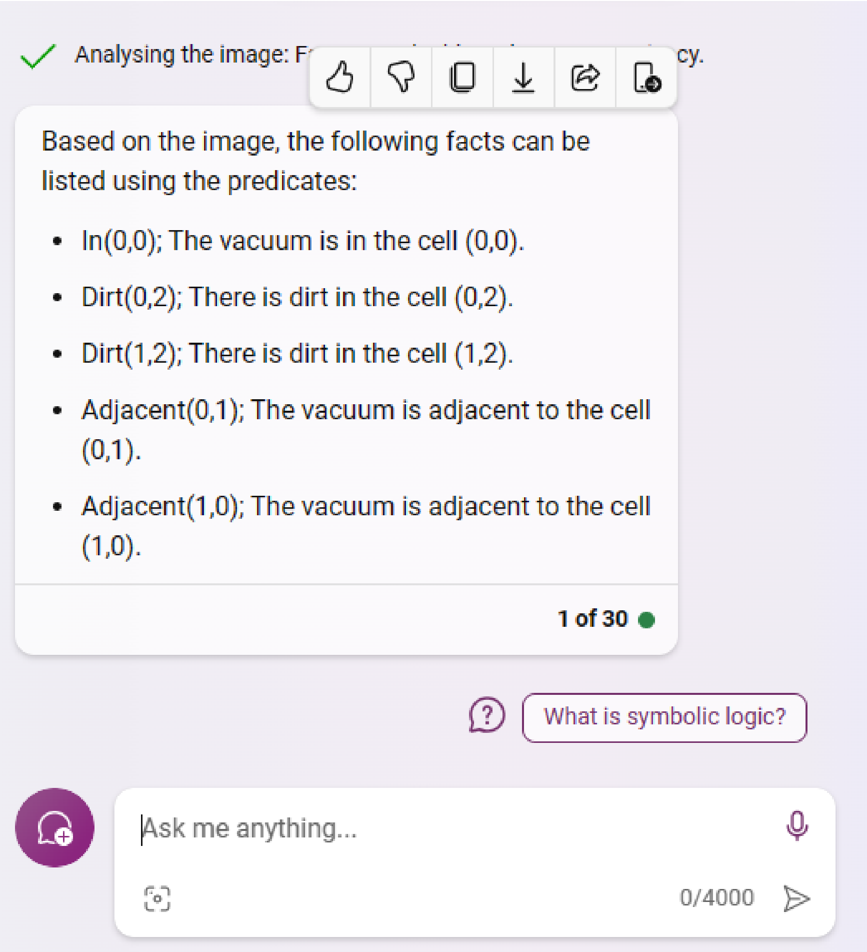

- … and so on!
- Using these rules (+ other obvious ones), starting at (0, 0) the robot will clear up dirt.

# The Vacuum World

- Problems:
  - how to convert video camera input to Dirt(0, 1)?
  - decision making assumes a static environment (calculative rationality)

# The Vacuum World



# Agent-oriented programming

- Yoav Shoham introduced “agent-oriented programming” (AOP) in 1990

“… new programming paradigm, based on a societal view of computation …”

- The key idea:
  - directly programming agents in terms of intentional notions
    - like belief, desire, and intention
    - Adopts the same abstraction as humans
  - Resulted in the Agent0 programming language

# Agent0

- AGENT0 is implemented as an extension to LISP.
- Each agent in AGENT0 has 4 components:
  - a set of *capabilities* (things the agent can do);
  - a set of *initial beliefs*;
  - a set of *initial commitments* (things the agent will do); and
  - a set of *commitment rules*.

- The key component, which determines how the agent acts, is the commitment rule set.
  - Each commitment rule contains:
    - a message condition;
    - a mental condition; and
    - an action.

# Agent0 Decision Cycle

## On each decision cycle . . .

- The message condition is matched against the messages the agent has received;
  - The mental condition is matched against the beliefs of the agent.
  - If the rule fires, then the agent becomes committed to the action (the action gets added to the agents commitment set).

## Actions may be . . .

- Private
  - An externally executed computation
- Communicative
  - Sending messages

## Messages are constrained to be one of three types . . .

- requests
  - To commit to action
- unrequests
  - To refrain from action
- Informs
  - Which pass on information

# Commitment Rules

- This rule may be paraphrased as follows:
  - if I receive a message from *agent* which requests me to do *action* at *time*, and I believe that:
    - *agent* is currently a friend;
    - I can do the *action*;
    - at *time*, I am not committed to doing any other *action*,
  - then commit to doing *action* at *time*.

```
COMMIT(
    (agent, REQUEST, DO(time, action)
    ), ;;; msg condition
    (B,
        [now, Friend agent] AND
        CAN(self, action) AND
        NOT [time, CMT(self, anyaction)]
    ), ;;; mental condition
    self,
    DO(time, action)
)
```

# Agent0

- Agent0 language can model *complex* agent behaviours.

- However, it is essentially a *prototype*, not intended for building large scale production systems.

# Concurrent MetateM

- Concurrent METATEM is a multi-agent language, developed by Michael Fisher
  - Each agent is programmed by giving it a temporal logic specification of the behaviour it should exhibit.
  - These specifications are executed directly in order to generate the behaviour of the agent.
- Temporal logic is classical logic augmented by modal operators for describing how the truth of propositions changes over time.
  - Think of the world as being a number of discrete states.
  - There is a single past history, but a number of possible futures

# MetateM Agents

- A Concurrent MetateM system contains a number of agents (objects)
  - Each object has 3 attributes:
    - A name
    - An interface
    - A MetateM program
  - An agent's interface contains two sets:
    - messages the agent will accept;
    - messages the agent may send.

# MetateM

- The root of the MetateM concept is Gabbay's separation theorem:
  - Any arbitrary temporal logic formula can be rewritten in a logically equivalent past $\Rightarrow$ future form

- Execution proceeds by a process of continually matching rules against a "history", and firing those rules whose antecedents are satisfied.
  - The instantiated future-time consequents become commitments which must subsequently be satisfied.

# Examples

| | |
|---|---|
| $\Box \mathrm{important(agents)}$ | *means "it is now, and will always be true that agents are important"* |
| $\Diamond \mathrm{important(ConcurrentMetateM)}$ | *means "sometime in the future, ConcurrentMetateM will be important"* |
| $\blacklozenge \mathrm{important(Prolog)}$ | *means "sometime in the past it was true that Prolog was important"* |
| $(\neg \mathrm{friends(us)})\, \mathcal{U}\, \mathrm{apologise(you)}$ | *means "we are not friends until you apologise"* |
| $\bigcirc \mathrm{apologise(you)}$ | *means "tomorrow (in the next state), you apologise"* |
| $\bullet \mathrm{apologise(you)} \Rightarrow \bigcirc \mathrm{friends(us)}$ | *means "if you apologised yesterday, then tomorrow we will be friends"* |
| $\mathrm{friends(us)}\, \mathcal{S}\, \mathrm{apologise(you)}$ | *means "we have been friends since you apologised"* |

# Readings for this week

- M.Wooldridge: An introduction to MultiAgent Systems – Ch. 3 Deductive Reasoning Agents
- "Agent Oriented Programming", Yoav Shoham