Escola Tècnica Superior d'Enginyeria

Universitat Rovira i Virgili

[DΣIM]

LECTURE 4: Reactive and Hybrid Agents

Introduction to Multi-Agent Systems (MESIIA, MIA)

URV

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

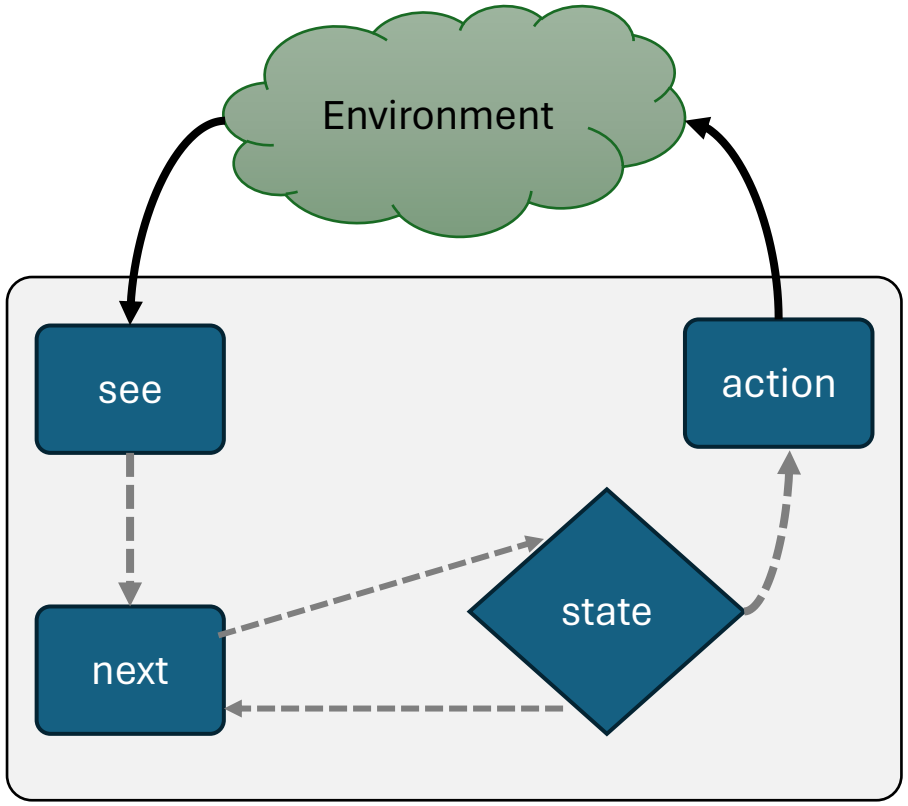
- 1990-present: Hybrid Agents
  - Hybrid architectures attempt to combine the best of reasoning and reactive architectures

Reactive Architectures

- There are many unsolved (some would say insoluble) problems associated with symbolic AI.
  - These problems have led some researchers to question the viability of the whole paradigm, and to the development of reactive architectures.
  - Although united by a belief that the assumptions underpinning mainstream AI are in some sense wrong, *reactive* agent researchers use many different techniques.
- In this lecture, we look at alternative architectures that better support some classes of agents and robots.
  - At the end, we then examine how *hybrid architectures* exploits the best aspects of deliberative and reactive ones.

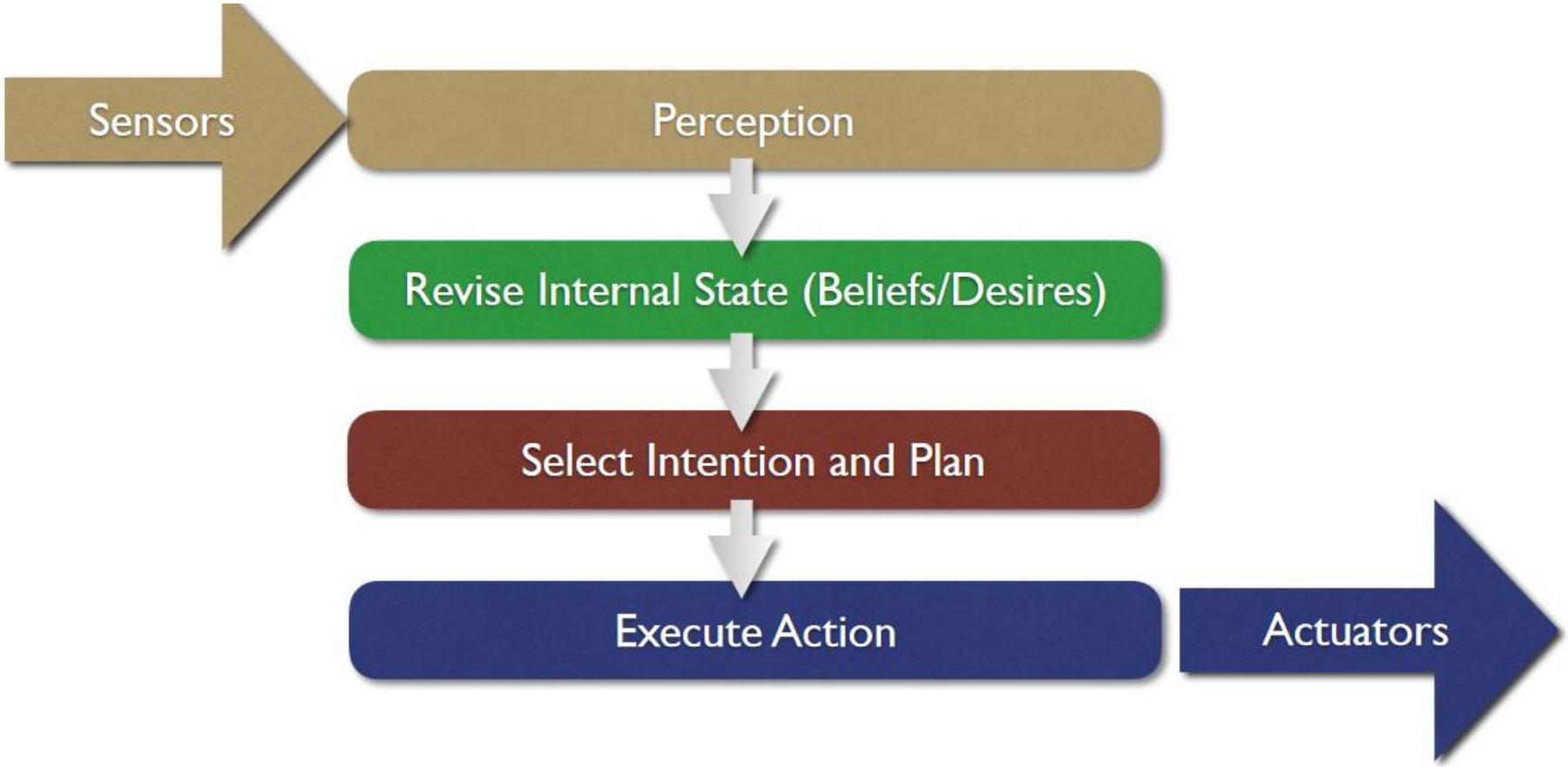
General Control Architecture

- So far, we have viewed the control architecture of an agent as one that:
  - *Perceives* the environment
  - *Revises* its internal state, identifying beliefs and desires
  - *Selects* actions from its intention and plan
  - *Acts*, possibly changing the environment
- Intention Reconsideration is important in highly dynamic environments

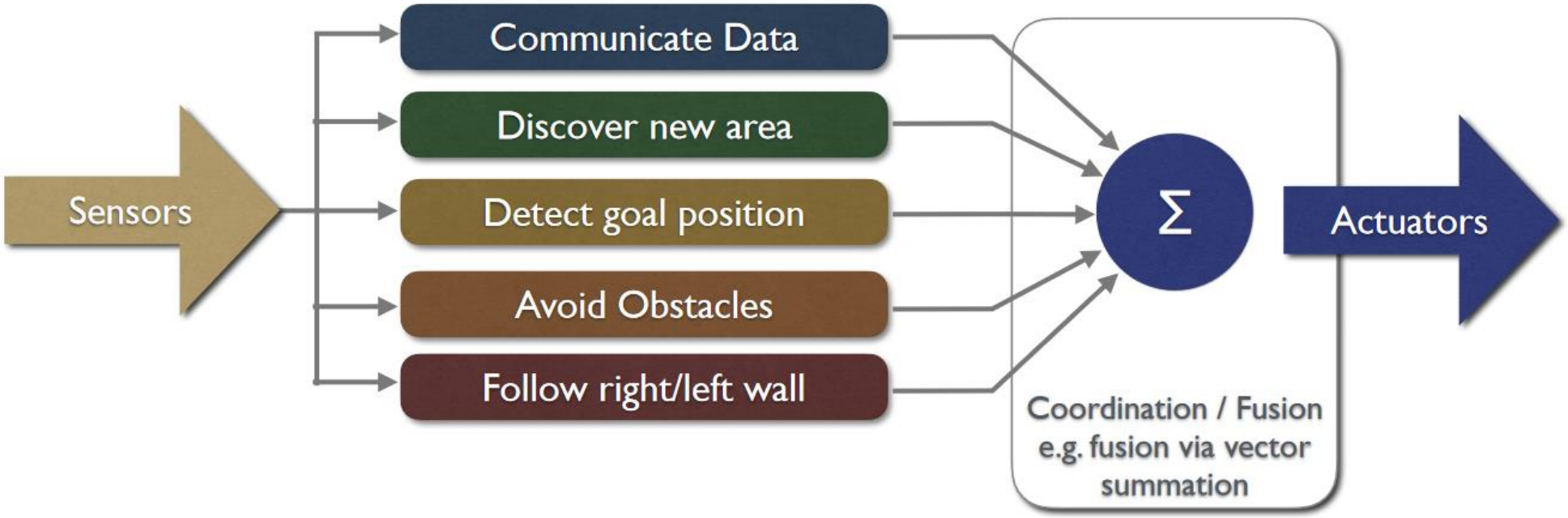


Agent Control Loop as Layers

The classic “Sense/Plan/Act” approach breaks it down serially like this



Behaviours



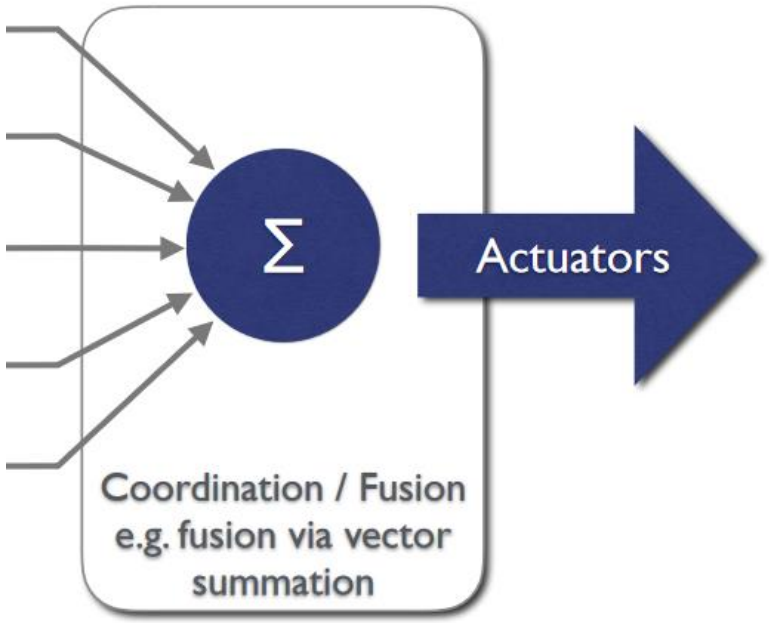
- Behaviour based control sees things differently
  - Behavioural chunks of control each connecting sensors to actuators
  - Implicitly parallel
  - Particularly well suited to Autonomous Robots

Behaviours

- Range of ways of combining behaviours.

- Some examples:
  - Pick the “best”
  - Sum the outputs
  - Use a weighted sum

- Flakey redux used a fuzzy combination which produced a nice integration of outputs.



Subsumption Architecture

- A subsumption architecture is a hierarchy of task-accomplishing behaviours.
  - Each behaviour is a rather simple rule-like structure.
  - Each behaviour ‘competes’ with others to exercise control over the agent.
  - Lower layers represent more primitive kinds of behaviour, (such as avoiding obstacles), and have precedence over layers further up the hierarchy.
- The resulting systems are, in terms of the amount of computation they do, extremely simple.
  - Some of the robots do tasks that would be impressive if they were accomplished by symbolic AI systems.

Rodney Brooks “subsumption architecture” was originally developed open Genghis

Brooks Behavioural Languages

- Brooks proposed the following three theses:

1. Intelligent behaviour can be generated *without* explicit *representations* of the kind that symbolic AI proposes.
2. Intelligent behaviour can be generated *without* explicit *abstract reasoning* of the kind that symbolic AI proposes.
3. Intelligence is an *emergent* property of certain complex systems.

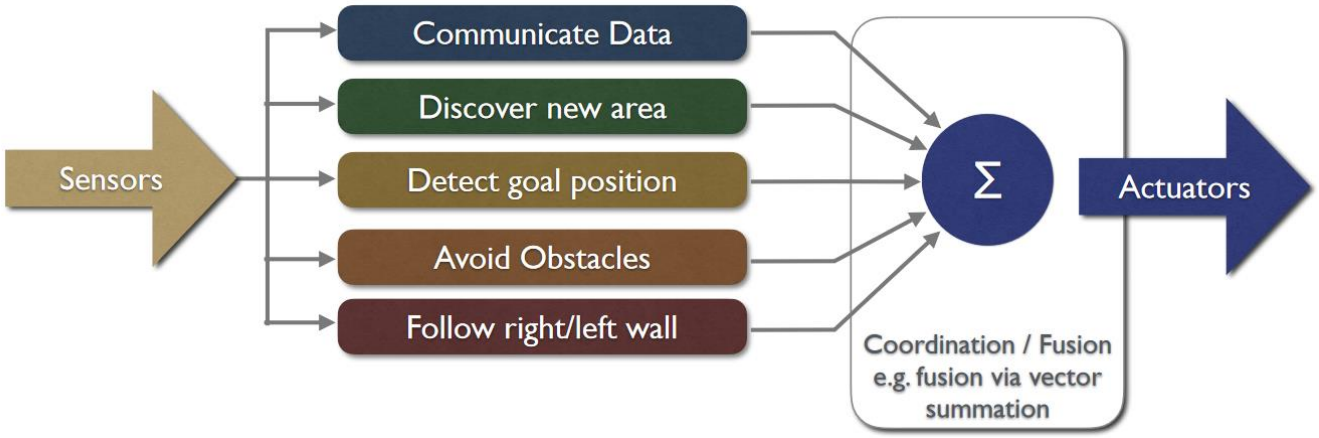
Brooks Behavioural Languages

- He identified two key ideas that have informed his research:
    1. ***Situatedness and embodiment:*** 'Real' intelligence is situated in the world, not in disembodied systems such as theorem provers or expert systems.
    2. ***Intelligence and emergence:*** 'Intelligent' behaviour arises as a result of an agent's interaction with its environment. Also, intelligence is 'in the eye of the beholder'; it is not an innate, isolated property.

- Brooks built several agents (such as Genghis https://en.wikipedia.org/wiki/Genghis_(robot)) based on his subsumption architecture to illustrate his ideas

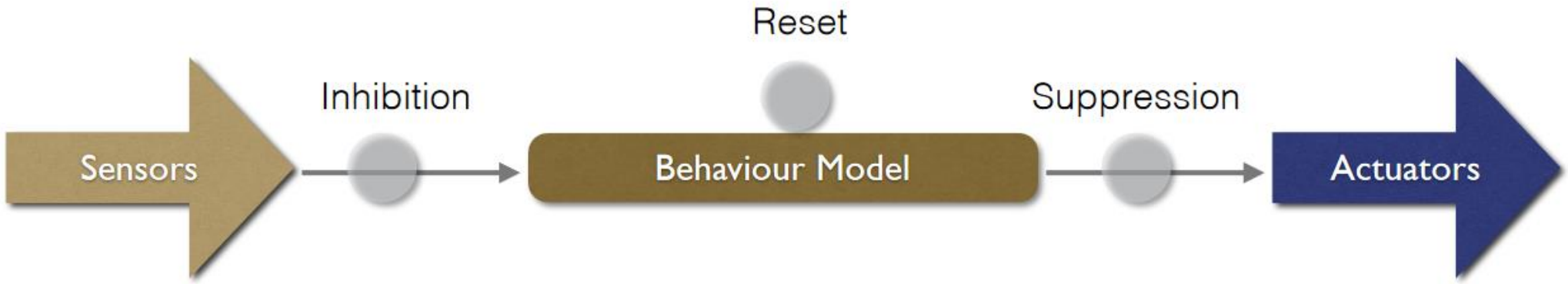
Subsumption Architecture

- It is the piling up of layers that gives the approach of its power.
  - Complex behaviour emerges from simple components
  - Since each layer is independent, each can independently be:
    - Coded / Tested / Debugged
  - Can then assemble them into a complete system



Abstract view of a Subsumption Machine

- Layered approach based on levels of competence
  - Higher level behaviours *inhibit* lower levels
- Augmented finite state machine:



Emergent behaviour

- Important but not well-understood phenomenon
  - Often found in behaviour-based/reactive systems
- Agent behaviours “emerge” from interactions of rules with environment.
  - Sum is greater than the parts.
  - The interaction links rules in ways that weren’t anticipated.
- Coded behaviour: In the programming scheme
- Observed behaviour: In the eyes of the observer
  - There is no one-to-one mapping between the two!
- When observed behaviour “exceeds” programmed behaviour, then we have emergence.

Emergent Flocking

Flocking is a classic example of emergence, e.g. Reynolds “Boids”, or Mataric’’s “nerd herd”.

Each agent uses the following three rules:

1. *Don’t run into any other robot*
2. *Don’t get too far from other robots*
3. *Keep moving if you can*

When run in parallel on many agents, the result is flocking

ToTo

- Maja Mataric ’s Toto is based on the subsumption architecture
  - Can map *spaces* and *execute plans* without the need for a *symbolic* representation.
  - Inspired by “…the ability of insects such as bees to identify shortcuts between feeding sites…”
- Each feature/landmark is a set of sensor readings
  - Signature
- Recorded in a behaviour as a triple:
  - Landmark type
  - Compass heading
  - Approximate length/size
- Distributed topological map

ToTo

- Whenever Toto visited a particular landmark, its associated map behaviour would become activated
  - If no behaviour was activated, then the landmark was new, so a new behaviour was created
  - If an existing behaviour was activated, it inhibited all other behaviours

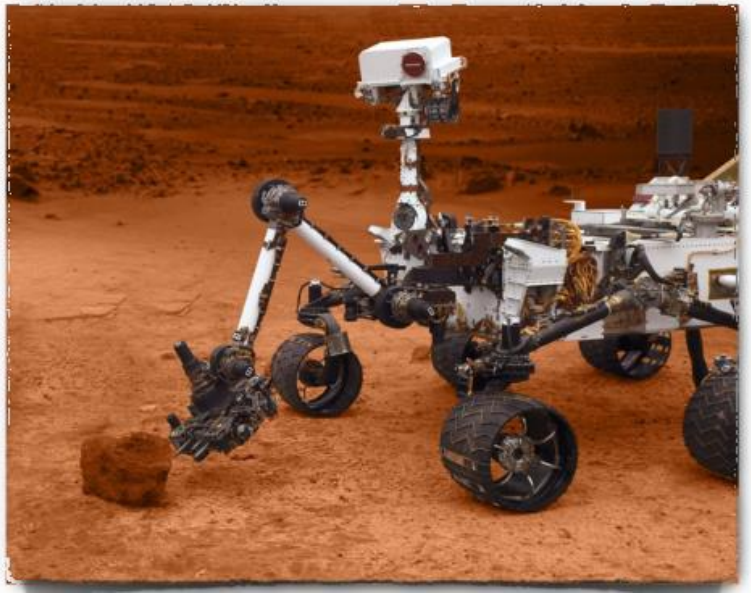
- Localization was based on which behaviour was active.
  - No map object, but the set of behaviours clearly included map functionality.

Steel's Mars Explorer System

- Steels' Mars explorer system
  - Uses the *subsumption* architecture to achieve near-optimal cooperative performance in simulated 'rock gathering on Mars' domain
  - *Individual behaviour* is governed by a set of simple rules.
  - *Coordination between agents* can also be achieved by leaving "markers" in the environment.

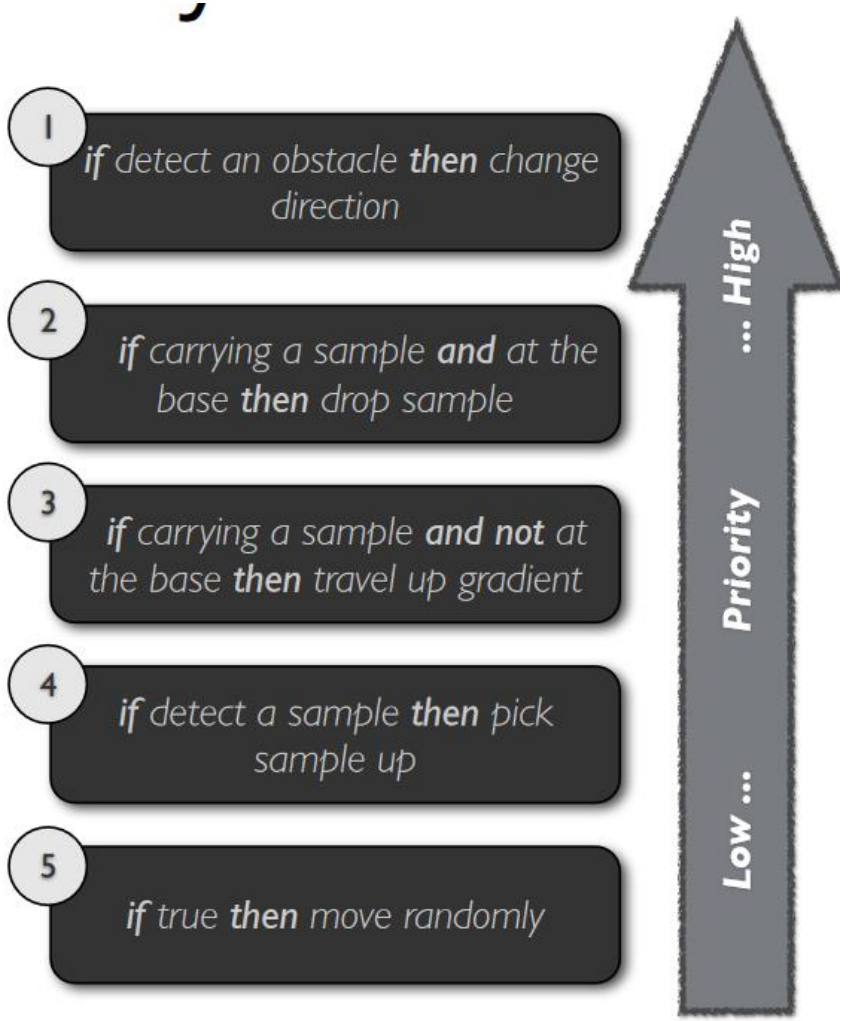
Objective

To explore a distant planet, and in particular, to collect sample of a precious rock. The location of the samples is not known in advance, but it is known that they tend to be clustered.



Steel's Mars Explorer System

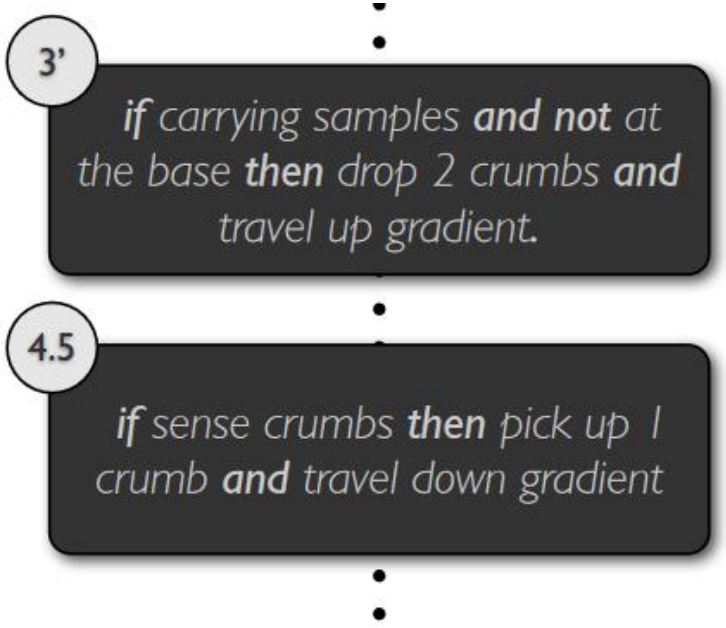
1. For individual (non-cooperative) agents, the lowest-level behaviour, (and hence the behaviour with the highest "priority") is obstacle avoidance.
2. Any samples carried by agents are dropped back at the mother-ship.
3. If not at the mother-ship, then navigate back there.
   - The "gradient" in this case refers to a virtual "hill" radio signal that slopes up to the mother ship/base.
4. Agents will collect samples they find.
5. An agent with "nothing better to do" will explore randomly. This is the highest-level behaviour (and hence lowest level "priority").



Steel's Mars Explorer System

- Existing strategy works well when samples are distributed randomly across the terrain.
  - However, samples are located in clusters
  - Agents should cooperate with each other to locate clusters

- Solution to this is based on foraging ants.
  - Agents leave a "radioactive" trail of crumbs when returning to the mother ship with samples.
    - If another agent senses this trail, it follows the trail back to the source of the samples
    - It also picks up some of the crumbs, making the trail fainter.
    - If there are still samples, the trail is reinforced by the agent returning to the mother ship (leaving more crumbs)
    - If no samples remain, the trail will soon be erased.



Situated Automata

- Approach proposed by Rosenschein and Kaelbling.
  - An agent is specified in a rule-like (declarative) language.
    - Then compiled down to a digital machine, which satisfies the declarative specification
  - This digital machine can operate in a provable time bound.
  - Reasoning is done off line, at compile time, rather than online at run time.

- The theoretical limitations of the approach are not well understood.
  - Compilation (with propositional specifications) is equivalent to an NP-complete problem.
  - The more expressive the agent specification language, the harder it is to compile it.

Situated Automata

- An agent is specified by perception and action
  - Two programs are used to synthesise agents:
    1. RULER specifies the perception component
       - (see opposite)
    2. GAPPS specifies the action component
       - Takes a set of goal reduction rules and a top-level goal
       - (symbolically specified) and generates a non-symbolic program

RULER takes as its input three components…

"…[A] specification of the semantics of the [agent's] inputs ("whenever bit 1 is on, it is raining"); a set of static facts ("whenever it is raining, the ground is wet"); and a specification of the state transitions of the world ("if the ground is wet, it stays wet until the sun comes out"). The programmer then specifies the desired semantics for the output ("if this bit is on, the ground is wet"), and the compiler … [synthesises] a circuit whose output will have the correct semantics… All that declarative "knowledge" has been reduced to a very simple circuit…"

Kaelbling, L.P. (1991) A Situated Automata Approach to the Design of Embedded Agents. SIGART Bulletin, 2(4): 85-88

Limitations of Reactive Systems

- Although there are clear advantages of Reactive Systems, there are also limitations!
  - If a model of the environment isn't used, then sufficient information of the local environment is needed for determining actions
  - As actions are based on local information, such agents inherently take a "short-term" view
  - *Emergent behaviour is very hard to engineer* or validate; typically a trial and error approach is ultimately adopted
  - Whilst agents with few layers are straightforward to build, *models using many layers are inherently complex* and difficult to understand.

Hybrid Architectures

- Many researchers have argued that neither a completely deliberative nor completely reactive approach is suitable for building agents.
- They have suggested using *hybrid* systems, which attempt to marry classical and alternative approaches.
- An obvious approach is to build an agent out of two (or more) subsystems:
  - a *deliberative* one, containing a symbolic world model, which develops plans and makes decisions in the way proposed by symbolic AI; and
  - a *reactive* one, which is capable of reacting to events without complex reasoning.

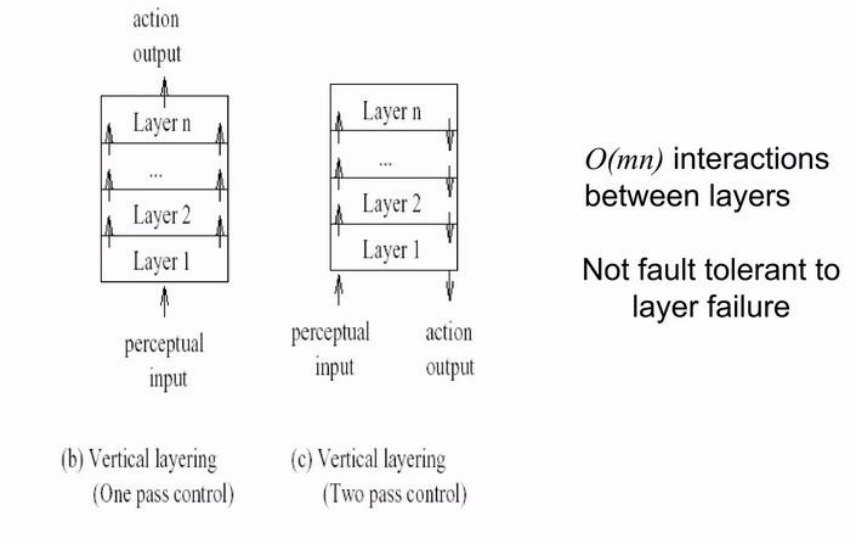
Hybrid Architectures

- Often, the reactive component is given some kind of precedence over the deliberative one.
- This kind of structuring leads naturally to the idea of a layered architecture, of which *InterRap* and *TouringMachines* are examples.
  - In such an architecture, an agent's control subsystems are arranged into a *hierarchy*…
  - …with higher layers dealing with information at *increasing levels of abstraction*.
- A key problem in such architectures is what kind control framework to embed the agent's subsystems in, to manage the interactions between the various layers.

Hybrid Architectures

Horizontal layering

- Layers are each directly connected to the sensory input and action output.
- In effect, each layer itself acts like an agent, producing suggestions as to what action to perform.

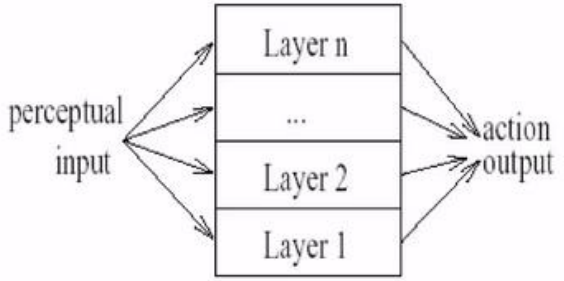


$O(mn)$ interactions between layers

Not fault tolerant to layer failure

Vertical layering

- Sensory input and action output are each dealt with by at most one layer each.

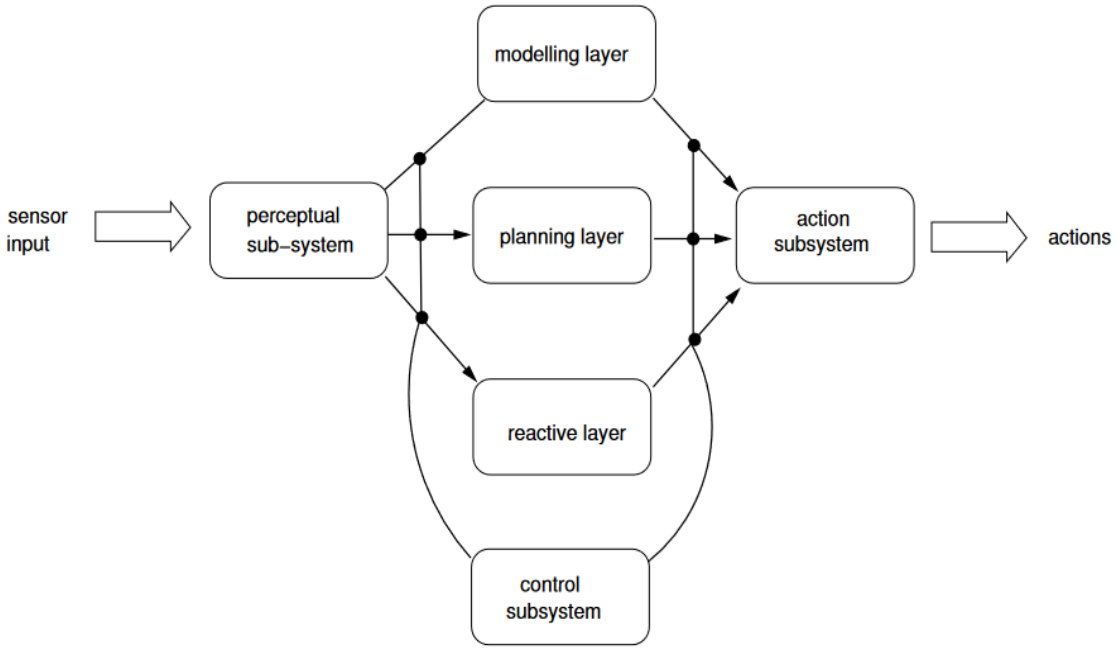


$O(m^n)$ possible options to be considered

Introduces bottleneck in central control system

Ferguson - TouringMachines

- The TouringMachines architecture consists of perception and action subsystems
  - These interface directly with the agent's environment, and three control layers, embedded in a control framework, which mediates between the layers.



Ferguson - TouringMachines

- The reactive layer is implemented as a set of situation-action rules, a` la subsumption architecture.
  - The planning layer constructs plans and selects actions to execute in order to achieve the agent's goals.

```
rule-1: kerb-avoidance
    if
        is-in-front(Kerb, Observer) and
        speed(Observer) > 0 and
        separation(Kerb, Observer) < KerbThreshHold
    then
        change-orientation(KerbAvoidanceAngle)
```

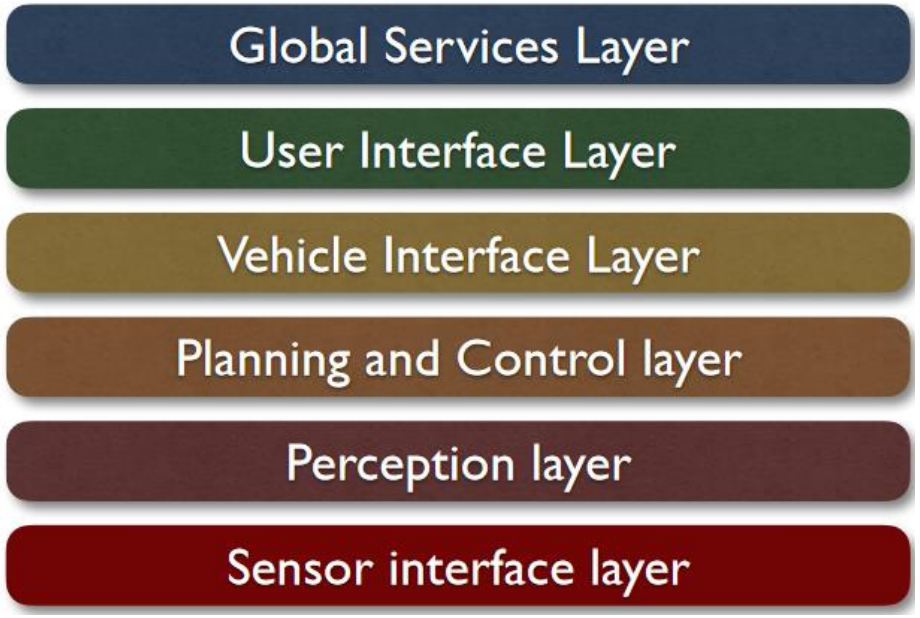
Ferguson - TouringMachines

- The modelling layer contains symbolic representations of the ‘cognitive state’ of other entities in the agent’s environment.
  - The three layers communicate with each other and are embedded in a control framework, which use control rules.
  - Such control structures have become common in robotics.

```
censor-rule-1:
    if
        entity(obstacle-6) in perception-buffer
    then
        remove-sensory-record(layer-R, entity(obstacle-6))
```

Real World Example: Stanley

- Won the 2005 DARPA Grand Challenge
  - Used a combination of the subsumption architecture with deliberative planning
  - Consists of 30 different independently operating modules across 6 layers



Summary

- This lecture has looked at two further kinds of agent:
  - Reactive agents; and
  - Hybrid agents.
- *Reactive agents* build complex behaviour from simple omponents.
- *Hybrid agents* try to combine the speed of reactive agents with the power of deliberative agents.

- Reading for this week:
  - M.Wooldridge: An introduction to MultiAgent Systems – Ch. 5 Reactive and Hybrid Architectures
  - A Robust Layered Control System for a Mobile Robot,", Rodney A. Brooks. IEEE Journal of Robotics and Automation, 2(1), March 1986, pp. 14–23. (also MIT AI Memo 864, September 1985)