Reactive and Hybrid Agents
Based on “An Introduction to MultiAgent Systems” and slides
by Michael Wooldridge
Reactive Architectures

- Problems with symbolic/logical approaches (transduction, computational complexity) resulted in the reactive paradigm

- Common theses occurring in the reactive approaches
  - the rejection of symbolic representations and decision making based on such representations (e.g., logical reasoning)
  - the idea that intelligent, rational behavior is linked to the environment an agent functions in (intelligent behavior is a product of interaction the agent maintains with its environment)
  - the idea that intelligent behavior emerges from the interaction of various simpler behaviors
The best-known reactive agent architecture based on the following theses:

- Intelligent behavior can be generated without explicit representations of the kind that symbolic AI proposes.
- Intelligent behavior can be generated without explicit abstract reasoning of the kind that symbolic AI proposes.
- Intelligence is an emergent property of certain complex systems.
Characteristics of the Subsumption Architecture

1. An agent’s decision making is realized through a set of *task-accomplishing* behaviors

   1. Each behavior can be seen as an individual selection function, which continually maps perceptual input to an action to perform
   2. Behaviors (behavior modules) are implemented as finite-state machines and include no complex symbolic representation (situation $\rightarrow$ action rules)

2. Many behaviors can fire simultaneously

   1. Modules are arranged into a *subsumption hierarchy*, with the behaviors arranged into *layers*
   2. Lower layers in the hierarchy can *inhibit* higher layers (the lower a layer is, the higher is its priority)
Action Selection in Layered Architectures

- Raw sensor input is not processed or transformed much.
- Action selection is realized through a set of behaviors together with an inhibition relation.
- We write $b_1 \prec b_2$, and read this as ‘$b_1$ inhibits $b_2$’ – $b_1$ is lower in the hierarchy than $b_2$, and will hence get priority over $b_2$. 

![Diagram of layered architecture with levels 0 to 3, showing sensor input leading to action output.](image-url)
The objective is 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. A number of autonomous vehicles are available that can drive around the planet collecting samples and later re-enter a mother ship spacecraft to go back to Earth. There is no detailed map of the planet, though it is known that the terrain is full of obstacles which prevent vehicles from exchanging any communication.
Mechanisms used in the Explorer

1. A gradient field
   1. The mother ship generates a radio signal so that agents can know in which direction the mother ship lies
   2. An agent needs to travel ‘up the gradient’ of signal strength
   3. The signal need not carry any information

2. Indirect communication
   Agents will carry ‘radioactive crumbs’, which can be dropped, picked up and detected by passing robots
Individual Behaviors

- $b_0$: if detect an obstacle then change direction
- $b_1$: if carrying a sample and at the base then drop sample
- $b_2$: if carrying a sample and not at the base then travel up gradient
- $b_3$: if detect a sample and not at the base then pick up sample
- $b_4$: if true then move randomly

The above behaviors are arranged into the hierarchy: $b_0 \prec b_1 \prec b_2 \prec b_3 \prec b_4$
Cooperative Behaviors

- $b_5$: if carrying a sample and not at the base then drop 2 crumbs and travel up gradient
- $b_6$: if sense crumbs then pick 1 crumb and travel down gradient

The modified subsumption hierarchy is the following:
$b_0 \prec b_1 \prec b_5 \prec b_3 \prec b_6 \prec b_4$
Limitations of Reactive Agents

1. If agents do not employ models of their environment, then they must have sufficient information available in their local environment to determine an acceptable action.

2. It is difficult to see how decision-making could take into account non-local information (a ‘shot-term’ view).

3. The relationship between individual behaviors, environment and overall behavior is not understandable. It is difficult to engineer agents for specific tasks and there is no methodology for building such agents.

4. It is difficult to build agents that contain many layers (>10) due to dynamics and complexity of interactions between the different behaviors.
Hybrid Agents

- It is claimed that neither a completely deliberative (pro-active) nor completely reactive approach is suitable for building agents.
- 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.
  - A **reactive** one, which is capable of reacting to events without complex reasoning.
- Subsystems are arranged into hierarchy of interacting *layers* (→ layered architectures).
Types of Layered 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.

- **Vertical layering**
  Sensory input and action output are each dealt with by at most one layer each.
Types of Layered Architectures
TouringMachines consists of three horizontal activity-producing layers.

Each layer continually produces suggestions for what actions the agent should perform.

Demonstration scenario – autonomous vehicles driving between locations through streets populated by other similar agents.
Architecture of Touring Machines

Sensor input

Perceptual subsystem

Modeling layer

Planning layer

Reactive layer

Action subsystem

Action output

Control subsystem
Layers in Touring Machines

- Reactive layer
  - provides immediate response to changes that occur in the environment
  - implemented as a set of situation-action rules, similarly to subsumption architecture

\textbf{rule -1: kerb-avoidance}

\begin{verbatim}
if
  is-in-front(Kerb, Observer) and
  speed(Observer) > 0 and
  separation(Kerb, Observer) < KerbThreshold
then
  change-orientation(KerbAvoidanceAngle)
\end{verbatim}
Layers in TouringMachines

- **Planning layer**
  - achieves the agent’s proactive behavior (‘day-to-day’ running of the agent)
  - constructs plans employing a library of ‘skeletons’ called *schemas* (hierarchically structured plans)
  - elaborates at run-time in order to decide which plan to follow

- **Modeling layer**
  - represents various entities in the world (including the agent itself)
  - predicts conflicts between agents and generates new goals to resolve these conflicts
  - these goals are posted down to the planning layer
Control Subsystem in Touring Machines

- It is effectively responsible for deciding which of the layers should take control over the agent.
- It is implemented as a set of control rules.
- Control rules can either suppress sensor information, or censor action outputs from the control layers.

**censor-rule-1:**

```
if entity(obstacle-6) in perception-buffer
then remove-sensory-record(layer-R, entity(obstacle-6))
```
InterRRaP uses a vertically layered two-pass architecture

It contains three layers

- *behavior-based* layer deals with reactive behavior
- *local planning* layer deals with everyday planning to achieve the agent’s goals
- *cooperative planning* layer deals with social interactions

Each layer has an associated *knowledge base* (i.e., representation of the world appropriate for this layer) – from ’raw’ information to complex models
Layer Interactions in InterRRaP

- **Bottom-up activation**
  a lower layer passes control to a higher layer because it is not competent to deal with the current situation

- **Top-down execution**
  a higher layer makes use of the facilities provided by a lower layer to achieve its goals

- The basic flow of control begins when perceptual input arrives at the lowest layer, and control may flow to higher layers, and then back again
DARPA Challenge and Stanley

- “Robot” (?) that won the DARPA Grand Challenge in 2005
- Developed at Stanford by the team of Sebastian Thrun (→ Google autonomous car)
- Complex software system (+ sensors) fitted in a standard VW Touareg R5 expanded with drive-by-wire capabilities
Computer Hardware

- 6 servers (Pentium M) running Linux
- 3 for running the software, 1 for logging, 2 idle
  - 1 server responsible for video processing
  - 2 servers responsible for remaining activities (e.g., planning)
No need for long-term planning
  - Organizers supplied a file with the route
  - The file included waypoints, corridor widths, speed limits

Short-term planning (search with cost functions)
  - driving close to the center of the road (corridor)
  - avoiding obstacles
  - maximizing speed (within limits) and minimizing driving time
Planning
Autonomous Driving

Google’s modified Toyota Prius uses an array of sensors to navigate public roads without a human driver. Other components, not shown, include a GPS receiver and an inertial motion sensor.

**LIDAR**
A rotating sensor on the roof scans more than 200 feet in all directions to generate a precise three-dimensional map of the car’s surroundings.

**POSITION ESTIMATOR**
A sensor mounted on the left rear wheel measures small movements made by the car and helps to accurately locate its position on the map.

**VIDEO CAMERA**
A camera mounted near the rear-view mirror detects traffic lights and helps the car’s onboard computers recognize moving obstacles like pedestrians and bicyclists.

**RADAR**
Four standard automotive radar sensors, three in front and one in the rear, help determine the positions of distant objects.

Source: Google

THE NEW YORK TIMES; PHOTOGRAPHS BY RAMI NABHAN FOR THE NEW YORK TIMES