Symbolic architectures

1. They use an explicit and symbolic representation of the environment.
2. They take actions by using symbolic reasoning and pattern matching.

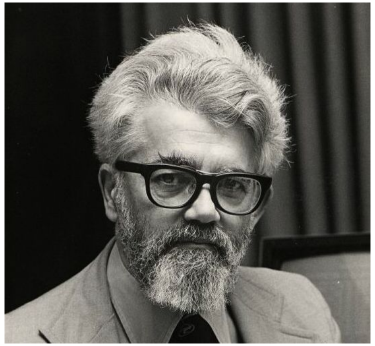
However, classical mathematical logic reveals limits for programming flexible behaviors.

Modal logics are proposed (they are classic logics augmented with new operators) to deal with flexible behaviors.

Intentional systems

Philosopher Daniel Dennett (https://ase.tufts.edu/cogstud/dennett/) coined the term intentional system to describe entities "whose behavior can be predicted through the attribution of beliefs, desires, and rational perspective."

John McCarthy (http://jmc.stanford.edu/) expresses that, although mental activities could be only used to human beings, occasionally it can also be applied when referring to things (Ascribing Mental Qualities to Machines: Formalizing Common Sense: 1990):



“Ascribing beliefs, free desires, intentions, consciousness, abilities or needs to a machine is legitimate when such ascription expresses the same information about the machine as it does about a person. It is useful when the ascription helps us understand the structure of the machine, its past or future behavior, or how to repair or improve it. It is perhaps not necessarily necessary, even among or by people, but it expresses in a reasonably brief way what is known about the state of a machine in a particular situation. Theories of beliefs, knowledge, and need can be constructed more easily for machines than for people, and then later applied to people. The ascription of mental qualities is easier for machines of known structure, such as thermostats or OS, but it is more useful when applied to entities whose structure is quite unknown.

Practical reasoning

It is directed to run actions. It involves the search of what actions to run, how, when and in which order. Theoretical reasoning is directed to the believes, not to the actions to hold a goal.

As stated by M. Bratman (https://philosophy.stanford.edu/people/michael-e-bratman): "Practical reasoning is a way to deal with competitive options, where relevant options are given by what the agent desires, values or wants and what the agent believes."

How practical reasoning can be used to program intelligent agents?

Practical reasoning

Two activities:

1. Deliberation: which states can be achieved.
2. Means/ends reasoning: how to achieve these states

The result of the deliberation process are the **intentions**

Intentions is a concept with many deep lateral consequences:

Intentions in practical reasoning

Intentions are problems for agents to solve.

If I got intention i, you will hope that I will put resources to decide how to get i

Intentions are a “filter” for adopting other intentions

If I got intention i, you will not hope that I will have another active intention j such that i and j will be mutually exclusive

Agents try to reach intentions, and if one try fails, agents figure out other ways to get that intentions

If my first try to get i fails, and the environment did not change significantly , I will try to search an alternative plan to get i

Intentions in practical reasoning

Agents believe that intentions are possible.

It must be believed that there are some way to fulfill these intentions.

Agents believe that intentions can be reached by their means.

It is not rational to adopt an intention i if I think i is not possible.

Agents can fail under certain circumstances.

It is not rational for an agent to believe that intentions will be arranged without doubt.

Agents do not need to calculate lateral consequences that can be obtained when an intention is reached.

If I think that i => j and I've got intention i, does not imply that I have also intention j. (intentions are not closed under implication)

Intentions are not desires

“My desire to play football this afternoon is nothing more than a potential influence on my behavior in the morning. It must compete with other relevant desires [. . .] before being fulfilled. On the other hand, if I intend to play football this afternoon, I will no longer continue weighing the pros and cons with other intentions. M. Bratman

Intentions are stronger that desires (Desires still continues to be an important concept)

Practical reasoning agents implementation.

Control Loop version 1

```
while true {
    Sense the environment.
    Update the internal model of the environment.
    Deliver about what to be the next intention.
    Use the medium/ends reasoning to achieve the intention chosen.
    Execute the plan.
}
```

Practical reasoning agents implementation.

- Medium-ends reasoning and deliberation processes are not instantaneous. They've got a **temporal cost**.

- Let the begining of deliberation process be in t_0, in t_1 starts the process of medium-ends reasoning and the obtained plan starts to run on t_2. Then, deliberation time is:

$$t_{deliberation} = t_1 - t_0$$

- And medium-ends reasoning time is:

$$t_{me} = t_2 - t_1$$

Implementing practical reasoning agents.

- Let's suppose that deliberation is optimum, in the sense that the intention chosen is the best for the agent.
- Then, this optimum intention, is still optimum in t_1? If $t_{deliberation}$ is not very small, maybe this intention is no longer the best.
- So, its is important to do *reasoning calculability* (trying to get a very good intention in a short $t_{deliberation}$ time).
- Deliberation is just the half of the problem: agent already has to figure out how to reach that intention (planification).

Implementing practical reasoning agents.

Therefore, agent behaviour is optimum if the next conditions hold:

- Deliberation of medium-ends reasoning are very fast; or
- Environment remains static while an agent is deliberating or doing the medium-ends reasoning processes; or
- When an intention that is optimal in t_0 still is optimum until t_2

Practical reasoning agents implementation.

```
Control Loop version 2
Set<Belief> beliefs = initialBeliefs();
while(true) {
     Percept percept = getNextPercept();
     beliefs = brf(beliefs, percept);
     Set<Intention> intentions = deliberate(beliefs);
     Plan currentPlan = plan(beliefs, intentions);
     execute(currentPlan);
}
```

How to deliberate?
It starts trying to calculate which are the available *options*.
Choose among them and it compromised with one
Intentions are the chosen options

Practical reasoning agents implementation.

```
Control Loop version 3
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initialIntentions();
while(true) {
    Percept percept = getNextPercept();
    beliefs = brf(beliefs, percept);
     Set<Desire> desires = options(beliefs, intentions);
     intentions = filter(beliefs, desires, intentions);
    Plan currentPlan = plan(beliefs, intentions);
    execute(currentPlan);
}
```

Deliberation function is composed of other two functions::

- *Option generator: it generates a whole set of alternatives taking as reference its current beliefs and intentions*
- *Filter: agent chooses between options and it compromises to reach one of them*.

Practical reasoning. Commitment strategies.

There are three possible *commitment strategies*:

- *Blind commitment*

  An agent with a blind commitment holds the intention until is fulfilled (It is also called as fanatic strategy).

- *Unique commitment*

  An agent with unique commitment still holds with an intention until achieved or believes that is not possible to achieved.

- *Open commitment*

  An agent with an open commitment still holds the intention whilst be possible, but it can change to another more interesting intention.

```
Control Loop version 4
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initialIntentions();
while(true) {
    Percept percept = getNextPercept();
    beliefs = brf(beliefs, percept);
    Set<Desire> desires = options(beliefs, intentions);
    intentions = filter(beliefs, desires, intentions);
    Plan currentPlan = plan(beliefs, intentions);
     while (!currentPlan.isEmpty()) {
          Action head = currentPlan.removeFirst();
          execute(head);
          percept = getNextPercept();
          beliefs = brf(beliefs, percept);
          if (!sound(currentPlan)) {
               currentPlan = plan(beliefs, intentions);
          }
     }
}
```

Blind commitment: An agent has commitment on medium-ends.
Here, the control loop of our agent is over-commitment in medium-ends
Modification: *replan if current plan goes wrong*

Control Loop version 5

```
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initialIntentions();
while(true) {
    Percept percept = getNextPercept();
    beliefs = brf(beliefs, percept);
    Set<Desire> desires = options(beliefs, intentions);
    intentions = filter(beliefs, desires, intentions);
    Plan currentPlan = plan(beliefs, intentions);
    while (!currentPlan.isEmpty() ||
           !succeeded(intentions, beliefs) ||
           !impossible(intentions, beliefs)) {
      Action head = currentPlan.removeFirst();
      execute(head);
      percept = getNextPercept();
      beliefs = brf(beliefs, percept);
      if (!sound(currentPlan)) {
           currentPlan = plan(beliefs, intentions);
      }
    }
}
```

Unique commitment: Still is over-commitment with intentions. It never stops if they are still valid ones. Modification: stop to calculate if they have been yet achieved or if they are impossible to achieve.

Practical reasoning: reconsidering intentions.

Current loop reconsidered the intention once per loop iteration:

- It has executed totally the plan; or
- It thinks that intention is yet achieved; or
- It thinks current intention is no longer possible.

Another approach: Reconsider intentions after execution of each action in the current plan.

Control Loop version 6

```
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initialIntentions();
while(true) {
    Percept percept = getNextPercept();
    beliefs = brf(beliefs, percept);
    Set<Desire> desires = options(beliefs, intentions);
    intentions = filter(beliefs, desires, intentions);
    Plan currentPlan = plan(beliefs, intentions);
    while (!currentPlan.isEmpty() ||
           !succeeded(intentions, beliefs) ||
           !impossible(intentions, beliefs)) {
     Action head = currentPlan.removeFirst();
     execute(head);
     percept = getNextPercept();
     beliefs = brf(beliefs, percept);
     desires = options(beliefs, intentions);
     intentions = filter(beliefs, desires, intentions);
     if (!sound(currentPlan)) {
           currentPlan = plan(beliefs, intentions);
     }
    }
}
```

Open commitment

But this process has a *¡high cost!*

But:

- If reconsideration is not executed the agent will be wasting its time executing a plan that it will at the end be successful.
- An agent with *continuous reconsideration* spends too much time to his task and this time will be required to executes the plan.

Solution: introducing a *meta-level control* which decides when to reconsidering and whose cost will be a lot of less that the one of deliberation(how?)

An heuristic function may help.

How it is now the control loop of the practical reasoning agent?

Control Loop version 7

```
Set<Belief> beliefs = initBeliefBase();
Set<Intention> intentions = initialIntentions();
while(true) {
    Percept percept = getNextPercept();
    beliefs = brf(beliefs, percept);
    Set<Desire> desires = options(beliefs, intentions);
    intentions = filter(beliefs, desires, intentions);
    Plan currentPlan = plan(beliefs, intentions);
    while (!currentPlan.isEmpty() ||
            succeeded(intentions, beliefs) ||
            impossible(intentions, beliefs)) {
      Action head = currentPlan.removeFirst();
      execute(head);
      percept = getNextPercept();
      beliefs = brf(beliefs, percept);
      if (reconsider(intentions, beliefs)) {
            desires = options(beliefs, intentions);
            intentions = filter(beliefs, desires, intentions);
      }
      if (!sound(currentPlan)) {
            currentPlan = plan(beliefs, intentions);
      }
    }
}
```

Practical reasoning: BDI logic.

With the purpose to give semantic to BDI architectures, Rao & Georgeff develop *BDI logic*: *not classical logic* with modal operators toi represent beliefs, desires and intentions.

‘*BDI logic*’ of Rao & Georgeff is an extended and quantified version of the branched temporal logic CTL*

The deep semantic structure is a net of labelled temporal branches

From classical logic: $\wedge$ (and), $\vee$ (or), $\neg$ (not), …

Path quantifiers CTL*:

$\forall\phi$ (for all paths ϕ)

$\exists\phi$ (in some path ϕ)

BDI operators:

(Bel *i* ϕ) *i* believes ϕ

(Des *i* ϕ) *i* desires ϕ

(Int *i* ϕ) *i* intention ϕ

Hybrid architectures.

Neither totally deliberative nor reactive are satisfactory for some authors

It is suggested to use hybrid systems that try to combine both approaches

An obvious approach is to build an agent from two (or more) subsystems:

- **Deliberative**, containing the symbolic model of the world, developing plans and making decisions in the way proposed by symbolic AI
- **Reactive**, capable of reacting to events without complex reasoning

A key problem in such architectures is what kind of control framework to include in the agent subsystems to manage the interactions between the different levels.

- **Horizontal Leveling**
  - The levels are all directly connected to the sensors and to the actuators.
  - Each level is like an agent, issuing suggestions on what action to take.
- **Vertical Leveling**
  - Sensors and actuators are connected to a single level at most

Reactive architectures

Reactive agent researchers employ a variety of techniques.

In this presentation, we will start by reviewing the work of one of the most critical of symbolic AI: Rodney Brooks (http://people.csail.mit.edu/brooks/)

Brooks proposes three theses:

1. Intelligent behavior can be generated without the explicit representations proposed by symbolic AI.
2. Intelligent behavior can be generated without the explicit reasoning mechanisms proposed by symbolic AI.
3. Intelligence is an emergent property of certain complex systems.

Reactive architectures

Two key ideas that have motivated his research are identified:

1. Situation and Non-Isolation: 'Real' intelligence is embedded in the world, not isolated in systems such as theorem provers or expert systems.
2. Intelligence and Emergency: 'Intelligent' behavior appears as a result of the agent's interaction with its environment. Furthermore, intelligence is in the eye of the beholder, it is not an isolated or innate property.

To illustrate these ideas, Brooks proposed the architecture of subsumption

- A subsumption architecture is a hierarchy of task-accomplishment behaviors.
- Each behavior has a simple rule structure.
- Each behavior 'competes' with the others to exercise control over the agent.
- The lower layers represent the most primitive behaviors (such as “obstacle avoidance” for a robot's movement) and take precedence over those higher in the hierarchy.
- The systems obtained are, in terms of the amount of computation performed, extremely simple.
- Some systems perform tasks based on these ideas that would be very difficult to achieve with symbolic AI systems.

Reactive architectures: Emergence.

“But even as the influence of decentralized ideas grows, there is a deep-seated resistance to such ideas. At some deep level, people seem to have strong attachments to centralized ways of thinking. When people see patterns in the world (like a flock of birds), they often assume that there is some type of centralized control (a leader of the flock). According to this way of thinking, a pattern can exist only if someone (or something) creates and orchestrates the pattern. Everything must have a single cause, and ultimate controlling factor. The continuing resistance to evolutionary theories is an example: many people still insisted that someone or something must have explicitly designed the complex, orderly structures that we call Life”

Mitchel Resnick (1956-)

Reactive architectures: Emergence.

Luc Steels' (http://arti.vub.ac.be/~steels/) Mars rover, based on the subsumption architecture, achieves near-optimal cooperative performance in the simulated domain of "collecting samples of minerals on Mars" :

The objective is:

"Explore a distant planet and collect samples of specific minerals. The location of the samples is not known but it is known that they tend to form groups. There is no direct communication between agents (due to the irregularities of the terrain), but they can all pick up a signal coming from the mothership"

Reactive architectures: Emergence.

For individual (non-cooperative) agents, the lowest-level (and therefore highest-priority) behavior is obstacle avoidance:

if I detect an obstacle then it changes direction (1)

The samples collected by the agents are deposited in the mother ship:

if I carry samples and I am at the base then deposit samples (2)

Agents carrying samples must return to the mothership:

if I carry samples and I am not in the base then it travels following the signal (3)

Agents collect samples they find:

if i detect a sample then i collect it (4)

An agent with "nothing better to do" randomly explores the world:

if true then move randomly (5)

How can it be improved? **Ant colony**

Reactive architectures: Advantages and disadvantages.

Simplicity

Economy

Computational tractability

failure robustness

Elegance

Agents without environment models must have sufficient local environment information available

If decisions are based on local environments, how do they take non-local information into account? (i.e. they have a short-term vision)

Difficulty making reactive agents that learn

As the behavior emerges from the interaction of the components and the environment, it is difficult to see how to design specific agents (there is no methodology based on principles)

It is difficult to design agents with a large number of behaviors (the dynamics of the interactions are too complex to understand a priori)

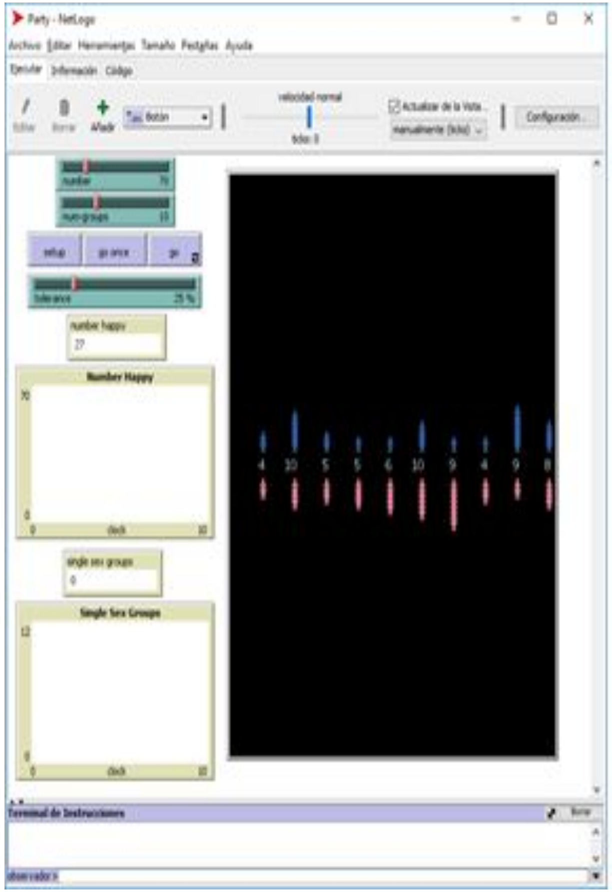
Reactive architectures: Netlogo

The design principles of reactive architectures have given rise to the development of agent-based models that are not only concerned with designing intelligent systems but also with experimenting with the result of the interaction between a large number of simple agents. This model of experimentation is frequently used in video games, ethology, sociology, molecular biology, network analysis, and many much more.

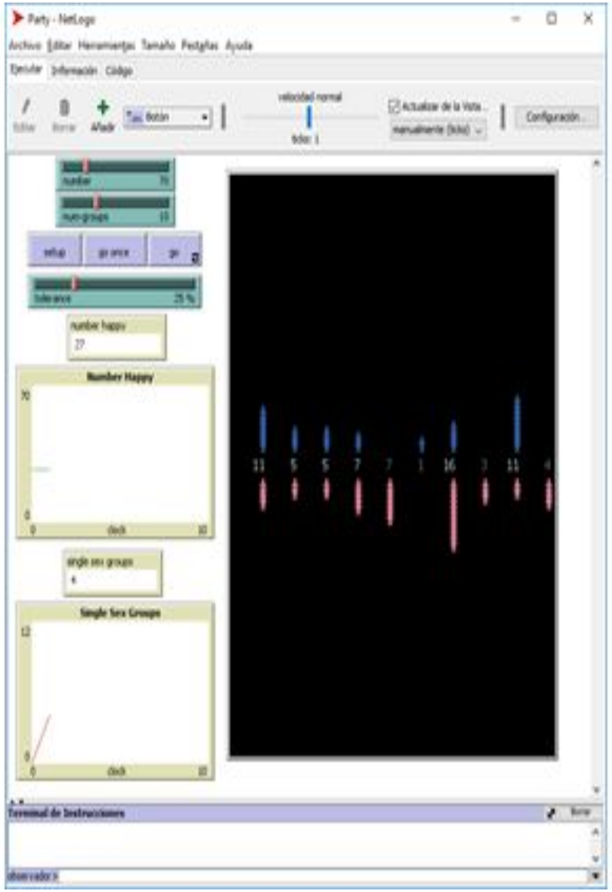
Netlogo is a programmable modeling environment for the simulation of social and natural phenomena.

Netlogo contains agents: ”observer” (only one), “turtle” (multiple and mobile), “patches” (multiple and static) and “links” (multiple, each connecting 2 turtles).

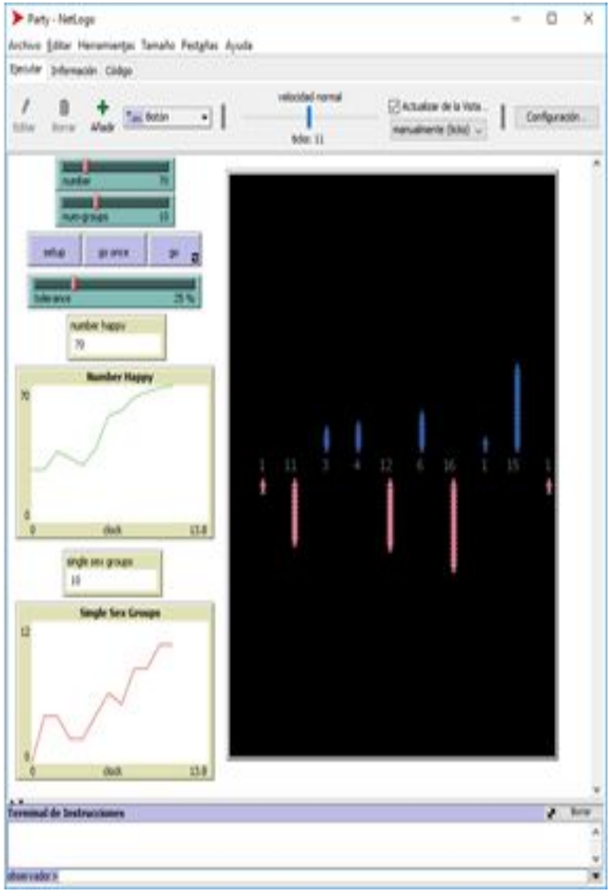
Each type of agent can be further specialized. These subspecies are generically called "breeds". There are global variables to all agents, variables just for one type or subtype of agent, and local variables just for use in the code block in which they are defined.



ticks: 0
number 70
num-groups 10
tolerance 25 %
number happy 27
single sex groups 0



ticks: 1
number 70
num-groups 10
tolerance 25 %
number happy 27
single sex groups 4



ticks: 11
number 70
num-groups 10
tolerance 25 %
number happy 70
single sex groups 10