# Reactive agents & Simulation

S Luz
luzs@cs.tcd.ie

October 7, 2014

### Concrete vs. abstract architectures

- Different ways of specifying the *action*
  - Logic-based: decision function implemented as a theorem prover (plus control layer)
  - *Reactive: (hierarchical) condition → action rules*
  - BDI: manipulation of data structures representing *Beliefs*, *Desires* and *Intentions*
  - Layered: combination of logic-based (or BDI) and reactive decision strategies

### Stimuli and responses

- Behaviour: $\boxed{\text{the product of an agent's interaction with its environment}}$
- Intelligence: $\boxed{\text{patterns that } \textit{emerge} \text{ from the interactions triggered by different behaviours}}$
- Emergence: $\boxed{\text{The transition from } \textit{local feedback} \text{ (human designed) and } \textit{global feedback} \text{ (product of agent autonomy).}}$

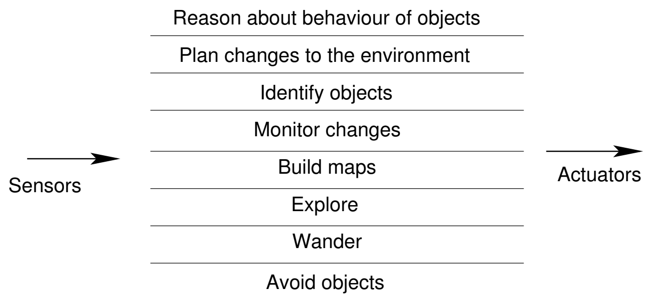
### A typical scenario

- Multiple goals
  - sometimes conflicting or inconsistent
- Multiple sensors
  - dealing with varied, sometimes inconsistent readings
- Robustness and fault tolerance
  - w.r.t. loss of agents
- Additivity
  - the more sensors and capabilities, the more processing power the agent needs

**Comparing action models**

- Cognitive Agent action model


- Reactive Agent action model

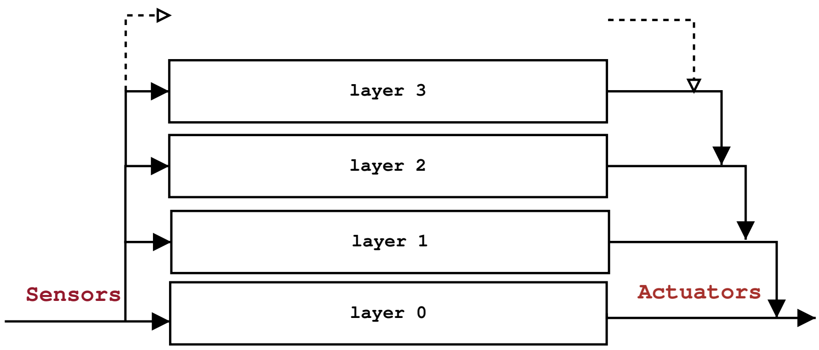



**Some reactive architectures**

- Situated rules:
  - PENGI (Chapman and Agre, 1987)
- *Subsumption architecture*
  - (Brooks, 1986)
- Competing tasks (Maes, 1989)
- Eco Agents (Drogoul and Ferber, 1992)
- Neural nets??
- ...

**A simple reactive architecture**

- The subsumption diagram



## A Formalisation

- Situated Rules represented as pairs $< c, a >$ (*behaviours*)

  - The set of all possible behaviours is then:
    $Beh = \{< c, a > | c \in P \wedge a \in A\}$

- The subsumption hierarchy will be represented by a total ordering, $\prec$, on the *behaviour relation*, $R \subseteq Beh$: $\prec \subseteq R \times R$

  - We say that "$b$ inhibits $b'$" if $b \prec b'$

## A reactive decision function

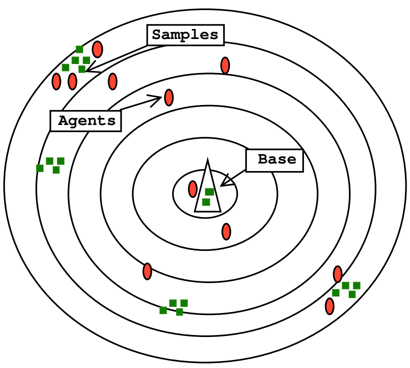
```
function action(p : P) : A
var fired : ℘(R)
begin
        fired := {⟨c, a⟩|⟨c, a⟩ ∈ R ∧ p ∈ c}
        for each ⟨c, a⟩ ∈ fired do
                if ¬(∃⟨c′, a′⟩ ∈ fired ∧
                      ⟨c′, a′⟩ ≺ ⟨c, a⟩)
                then return a
                end if
        end for
        return noop
end function action
```

## Time complexity

- For the "naive" algorithm above...
  - $action() = O(n^2)$, where $n = max(|R|, |P|)$
- N.B.: Complexity for *each* agent
- (In practice, one can often do better than $O(n^2)$, low time complexity being one of the main selling points of reactive architectures.)

## Example: collective problem solving

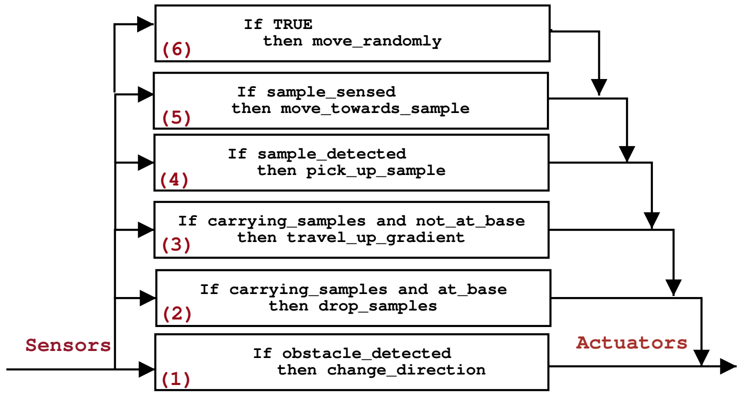
- Case study: Foraging Robots (Steels, 1990; Drogoul and Ferber, 1992):




- Constraints:
  - No message exchange
  - No agent maps
  - obstacles
  - gradient field
  - clustering of samples

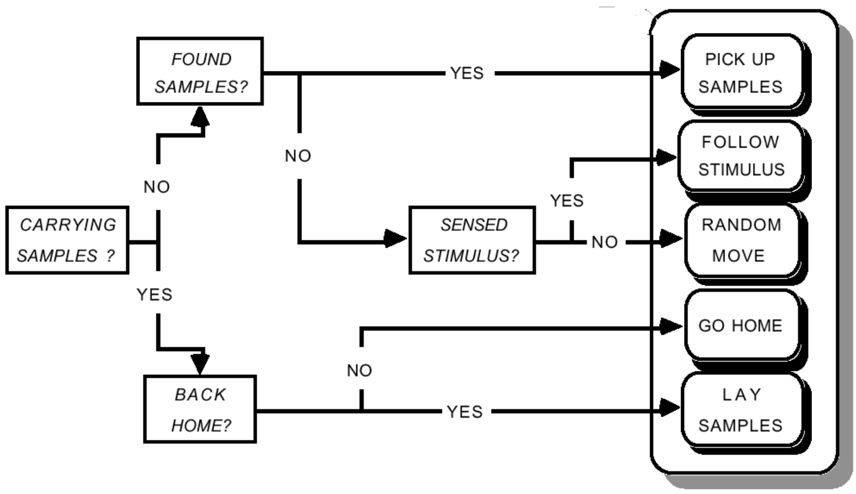
### Simple collecting behaviour

- Subsumption ordering: (1) $\prec$ (2) $\prec$ (3) $\prec$ (4) $\prec$ (5) $\prec$ (6)




### Behaviour diagram

- Same rules (roughly), as described in (Drogoul and Ferber, 1992)




### Can we improve on this design?

- What is the problem with the proposed system?
  - Too many random trips when samples in clusters.
- Try Indirect communication: "bread crumbs", ant pheromones, etc
- Replace rules (3) and (5) by:

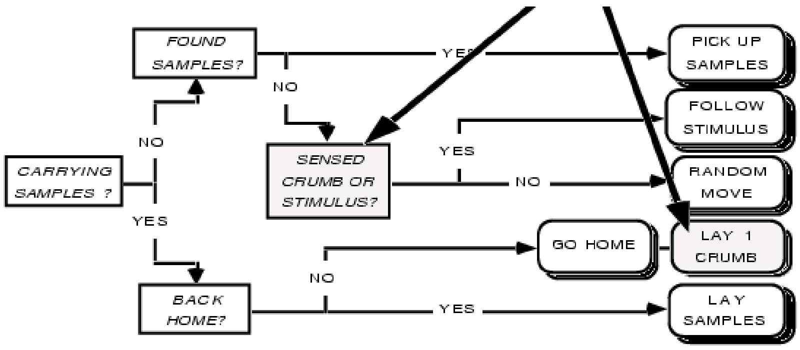
```
(3') If carrying samples
        and not_at_base
```

```
      then drop_crumb
        and travel_up_gradient
```

```
(5') If sample_sensed
       or crumb_sensed
     then
       move_towards_sample_or_crumb
```

**Improved architecture I**

- After replacing (3) and (5):




**Further improvement?**

- What is the long-term effect of laying bread crumbs? Is there room for improvement?
- Change (3') into:

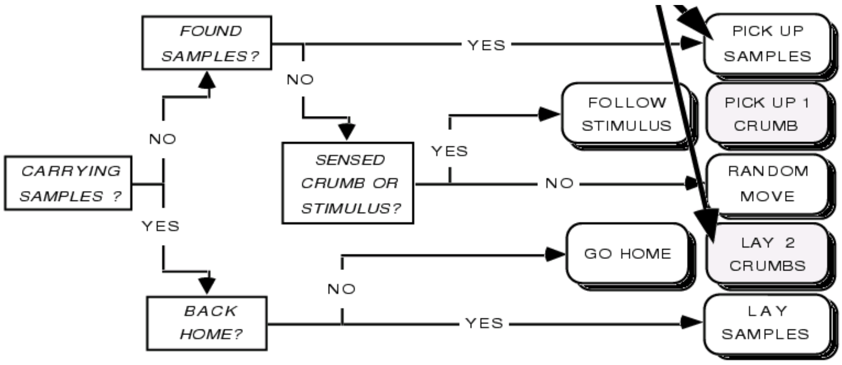
```
(3") If carrying_samples
          and not_at_base
        then drop_2_crumbs and
             travel_up_gradient
```

- and add the following:

```
(7) If crumb_detected
      then pick_up_1_crumb
```

**Improved architecture II**

- The subsumption ordering becomes
  - $(1) \prec (2) \prec (3'') \prec (4) \prec (7) \prec (5') \prec (6)$




**Advantages of this approach**

- Low time (and space) complexity
- Robustness
- Better performance
  - (near optimal in some cases)
  - problems when the agent population is large

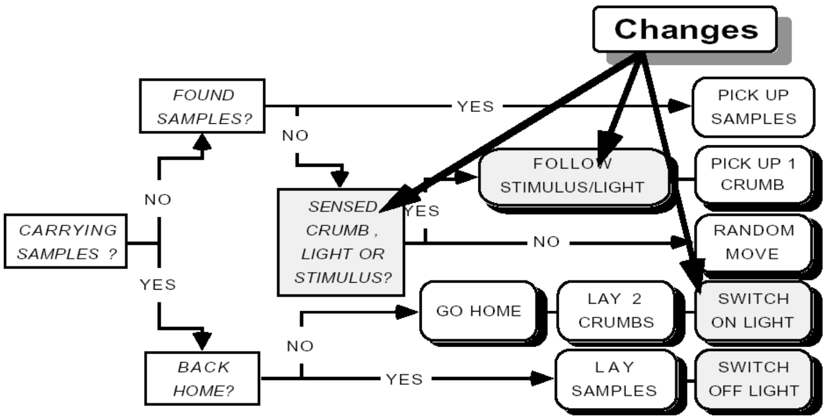
**Agent chains**

- New rules ("docker robots") (Drogoul and Ferber, 1992)

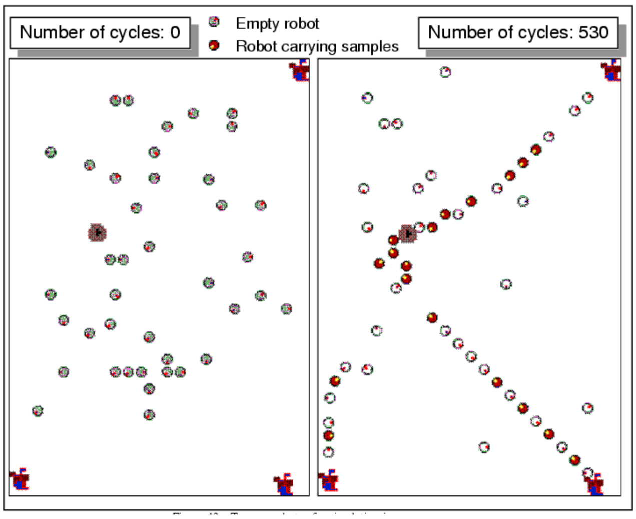
```
(a) If not_carrying_sample and
       travelling_down_gradient and
       detected_sample_carrying_agent
    then pick_up_other_agents_sample and
         travel_up_gradient

(b) If carrying_sample and
       travelling_up_gradient and
       detected_empty_agent
    then deliver_sample_to_other_agent and
         travel_down_gradient
```

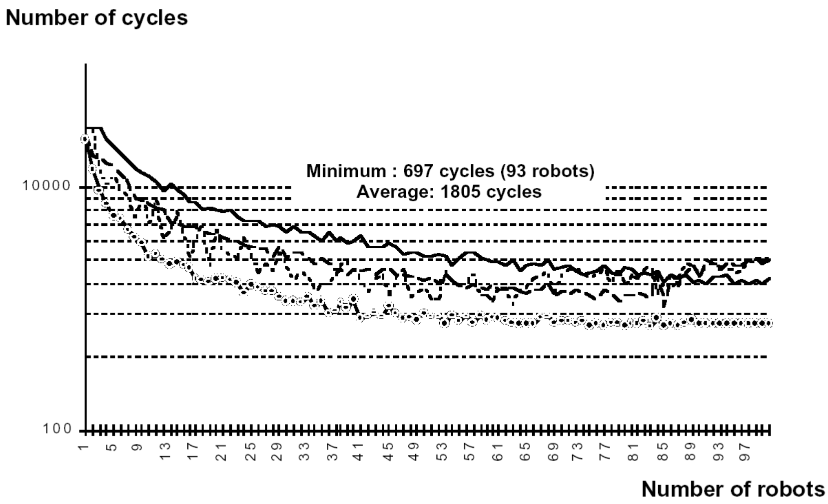
**Behaviour diagram**



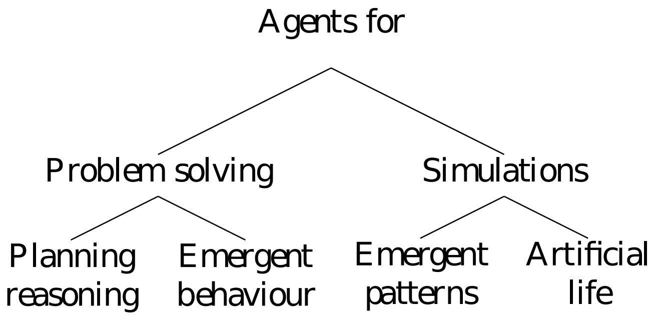

### A simulation




### How do the different architectures perform?




**Simulation Agents and Reactive Architectures**




**Why use agents in simulations**

- challenges for "traditional" modelling methods:
  - how to extrapolate a model to situations where the assumptions on which it is based (described through ODEs or PDEs, for instance) no longer hold;
  - how to *explain* the macroscopic properties of the systems modelled;
  - how handle heterogenity;
  - how to handle discontinuity...

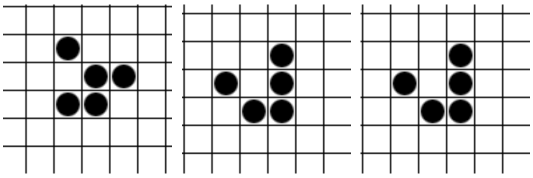
**Background**

- A workshop (Langton, 1989)
- Nonlinear models and complex systems:
  - A few phenomena which resist linearisation: plant growth, weather, traffic flow, stock market crashes, intelligence, ...
- "Understanding by building":

– Individual-based Modelling in Biology (population biology, ecology, ...)
– principles of intelligent behaviour
– and practical applications

### Applications to games

- Artificial Life: cellular automata and the "Game of Life"



- Tamagotchi, The Sims$^{tm}$, etc
- Distributed search and problem solving (e.g. for path-finding)

### Examples: Conway's Life

- A "zero-player" game invented by John H Conway in the 70s.
- Rules:
  1. Any live cell with fewer than two live neighbours dies, as if caused by under-population.
  2. Any live cell with two or three live neighbours lives on to the next generation.
  3. Any live cell with more than three live neighbours dies, as if by overcrowding.
  4. Any dead cell with exactly three live neighbours becomes a live cell, as if by reproduction.
- (Demo in emacs...)

### Entertaining examples: flocking behaviour

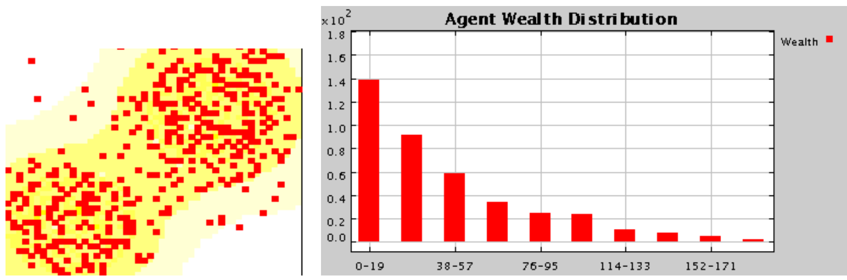
- Craig Reynolds page at Sony Entertainment: http://www.red3d.com/cwr/boids/
- (demo Video and applets)
- Steering behaviours:
  – *Separation*: steer to avoid crowding local flockmates
  – *Alignment*: steer towards the average heading of local flockmates
  – *Cohesion*: steer to move toward the average position of local flockmates

**Emergence in agent-based simulations**

- Emergence in complex systems:

> *Stable macroscopic patterns arising from the local interaction of agents.*

- Example: Skewed "wealth" distribution in (Epstein and Axtell, 1996, ch 2)




**Advantages of agent-based modelling**

- Simplicity: shift from mathematical descriptions of entire systems to rule based specifications of agent behaviour.

- Implementation of complex boundary conditions: in agent-based simulations, environments with irregular shape are not more complex to model than regular ones.

- Inherent parallelism: no code changes when porting to parallel architectures

- Adequacy to modelling of small populations

- Realism (??)

NB: Complex boundary conditions are handled in traditional (e.g. PDE) modelling by methods such as multigrid, at the price of complicated complications in ensuring consistency of the conditions in the embedded problems.

**Disdvantages of agent-based modelling**

- Memory and processing speed might constrain the size of the agent population in the model

- Difficulties in exploring the parameter space, if the simulation comprises a large number of rules

- Understanding how simple local behaviour gives rise to complex global behaviour is not always an easy task; if a model captures too much of the complexity of the world, it may become just as difficult to understand as the world itself.

- "Noise" introduced by the model or its implementation might give rise to phenomena not present in the real system

### Agent modelling toolkits

- Swarm, RePast, StarLogo, Ascape, ...
- What they provide
  - mechanisms for managing resource allocation
  - a schedule
  - basic environment topography
  - graphics, (media handling etc)
  - a scientific computing library
  - basic statistics
  - Usually no built-in agent semantics
- Agent development library: `mason`: "a fast, discrete-event multiagent simulation library core in Java" (Luke et al., 2005).

### Applications of agent-based modelling

- Sociological models (e.g. (Epstein and Axtell, 1996))
- Biological simulations
  - Insect societies
  - bacterial growth
  - forest dynamics
- Molecule interaction in artificial chemistry
- Traffic simulations
- Computer networks (see `http://www.crd.ge.com/~bushsf/ImperishNets.html`, for instance)

### RePast: A (Pure) Java Simulator

- Repast is an acronym for *REcursive Porous Agent Simulation Toolkit.*

  > *"Our goal with Repast is to move beyond the representation of agents as discrete, self-contained entities in favor of a view of social actors as permeable, interleaved and mutually defining, with cascading and recombinant motives."*
  >
  > *From the Repast web site*

- **??????**

**Two simulations: 1 - Mouse Traps**

- A demonstration of "nuclear fission":
  - Lay a bunch of mousetraps on the floor in a regular grid, and load each mousetrap with two ping pong balls.
  - Drop one ping pong ball in the middle...
- A discrete-event simulation that demonstrates the dynamic scheduling capabilities of Repast
- The agent programmer defines:
  - an agent (`MouseTrap`)
  - a model (`MouseTrapModel`)

**Two simulations: 2 - Heat Bugs**

- "(...) an example of how simple agents acting only on local information can produce complex global behaviour".
- Agents: `HeatBug`s which absorb and expel heat
- Model: `HeatBugsModel` has a spatial property, heat, which diffuses and evaporates over time. (green dots represent `HeatBug`s, brighter red represents warmer spots of the world.)
- A `HeatBug` has an ideal temperature and will move about the space attempting to achieve this ideal temperature.

# References


Brooks, R. A. (1986). A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, RA-2(1):14–23.

Chapman, D. and Agre, P. E. (1987). Pengi: An implementation of a theory of activity. Technical report, Massachusetts Institute of Technology, A.I. Lab., Cambridge, Massachusetts. Subm. to the Cognitive Modeling/Science Section of AAAI-87.

Drogoul, A. and Ferber, J. (1992). From tom thumb to the dockers: Some experiments with foraging robots. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 451–459, Honolulu, Hawaii.

Epstein, J. and Axtell, R. (1996). *Growing Artificial Societies: Social Science From The Bottom Up.* MIT Press.

Langton, C. G. (1989). *Artificial Life: Proceedings of the First Int. Workshop on the Synthesis and Simulation of Living Systems.* Addison-Wesley.

Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., and Balan, G. (2005). Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527.

Steels, L. (1990). Cooperation between distributed agents through self organization. In Demazeau, Y. and Müller, J. P., editors, *Decentralised AI, Proceedings of MAAMAW '89*, pages 175–196, Amsterdam. Elsevier.