Multi-Agent Systems

Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Felix Lindner, and Thorsten Engesser

Winter Term 2018/19

Definition: Agent Architecture

An agent architecture proposes a particular methodology for building an autonomous agent: Set of component modules and interaction of these modules determines how perception and current state of the agent determine its next action and next internal state.

Agents: Standard View

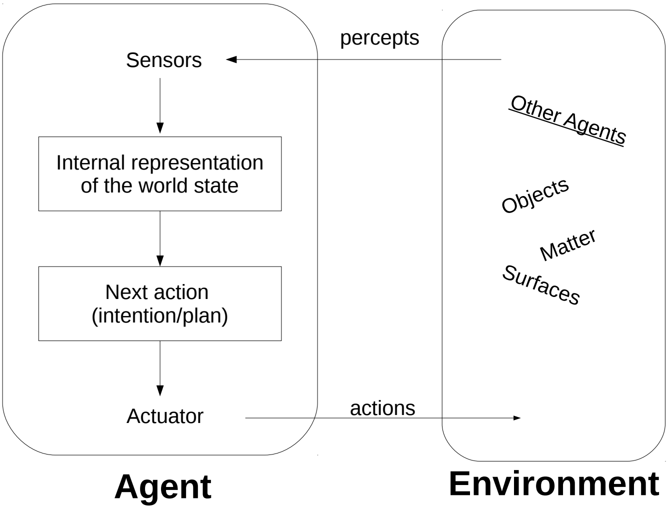


Table-Driven Agent

function TABLE-DRIVEN-AGENT(*percept*)
  global *table*, *percepts*
  percepts ← APPEND(*percepts*, *percept*)
  action ← LOOKUP(*percepts*, *table*)
  return *action*
end function

- Epistemic state is the list of percepts so far perceived.
- Practical reasoning based on look-up table.
- How large will the look-up table grow?

Simple Reflex Agent

function SIMPLE-REFLEX-AGENT(*percept*)
  global *rules*
  state ← INTERPRET-INPUT(*percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← RULE-ACTION(*rule*)
  return *action*
end function

- Epistemic state is just the current percept.
- Practical reasoning based on condition-action rules.

Swarms of Simple Reflex Agents




- Swarm formation control: How to design programs that result into a particular swarm formation when executed on each simple reflex agent. Video: EPFL Formation

Formation Control: General Setting

- Problem
  - Form an approximation of a simple geometric object (shape)
  - Problem not yet solved in general!
  - Algorithms exists that make simplifying assumptions about the agents' capabilities and the shape.
- Assumptions shared by the algorithms proposed by Sugihara & Suzuki (1996)
  - Each robot can see all the other robots
  - Shapes are connected
  - But ...
  - Total number of robots unknown
  - No common frame of reference (i.e., one cannot program the robots "to meet at point (X, Y)" or "to move north")
  - robots cannot communicate with each other
  - Local decision making

Formation Control: CIRCLE

- Problem: Move a group of robots such that they will eventually approximate a circle of a given diameter D.

- Algorithm [Sugihara & Suzuki, 1996]: The robot R continuously monitors the position of a farthest robot R_{far} and a nearest robot R_{near}, and the distance d between R (itself) and R_{far}.
  1. If $d > D$, then R moves towards R_{far}
  2. If $d < D - \delta$, then R moves away from R_{far}
  3. If $D - \delta \leq d \leq D$, then R moves away from R_{near}

Formation Control: POLYGON

- Problem: Move a group of N robots such that they will eventually approximate an $n \ll N$-sided polygon.
- Algorithm [Sugihara & Suzuki, 1996]:
  1. Run the CIRCLE algorithm until each robot R can recognize its immediate left neighbor $l(R)$ and right neighbor $r(R)$.
  2. Selection of n robots to be the vertices of the n-sided polygon.
  3. All robots R execute the CONTRACTION algorithm
     1. Continuously monitor the position of $l(R)$ and $r(R)$
     2. Move toward the midpoint of the segment $\overline{l(R)r(R)}$

Formation Control: FILLCIRCLE

- Problem: Move a group of robots such that they will eventually distribute nearly uniformly within a circle of diameter D.

- Algorithm [Sugihara & Suzuki, 1996]: The robot R continously monitors the position of a farthest robot R_{far} and a nearest robot R_{near}, and the distance d between R (itself) and R_{far}.
  1. If $d > D$, then R moves toward R_{far}.
  2. If $d \leq D$, then R moves away from R_{near}.

Formation Control: FILLPOLYGON

- Problem: Move a group of N robots such that they will eventually distribute nearly uniformly within an $n \ll N$-sided convex polygon.
- Algorithm [Sugihara & Suzuki, 1996]: First n robots are picked as vertices of the polygon and moved to the desired position. All other robots R execute FILLPOLYGON:
  1. If, as seen from R, all other robots lie in a wedge whose apex angle is less than π, then R moves into the wedge along the bisector of the apex.
  2. Otherwise, R moves away from the nearest robot.

Formation Control: LINE

- Problem: Move a group of robots such that they will eventually connect to points. (In fact, just a special case of FILLPOLYGON.)
- Algorithm [Sugihara & Suzuki, 1996]: First, two robots are picked as vertices of the line and moved to the desired position. All other robots *R* execure FILLPOLYGON.

When Memory Helps

- Simple reflex agent's do not make use of memory. This can be a severe limitation:
  - Imagine you are at a crossing and you have to decide to either go left or right. You go left and find out it's a dead end. You return to the crossing. Again, you have the choice between going left and going right ...
  - Possible solutions:
    - Change the environment (pheromones, bread crumbs)
    - Put your previous actions and experiences into your memory

Reflex Agent With State

function REFLEX-AGENT-WITH-STATE(*percept*)
  global *rules*, *state*
  state ← UPDATE-STATE(*state*, *percept*)
  rule ← RULE-MATCH(*state*, *rules*)
  action ← RULE-ACTION(*rule*)
  state ← UPDATE-STATE(*state*, *action*)
  return *action*
end function

- Epistemic state is updated over time (takes both state and percept into account and thus can also update currently unobserved aspects).
- Practical reasoning is based on rules applied in this state and leads to another state update.

Agent-Based Modeling

Definition (Wilensky & Rand, 2015)

Agent-based modeling is a form of computational modeling whereby a phenomenon is modeled in terms of agents and their interactions.

- Agents are entities that have state variables and values (e.g., position, velocity, age, wealth)
  - Gas molecule agent: mass, speed, heading
  - Sheep agent: speed, weight, fleece
- Agents also have rules of behavior
  - Gas molecule: Rule to collide with another molecule
  - Sheep: Rule to eat grass
- Universal clock: At each tick, all agents invoke their rules.

Wolves and Moose

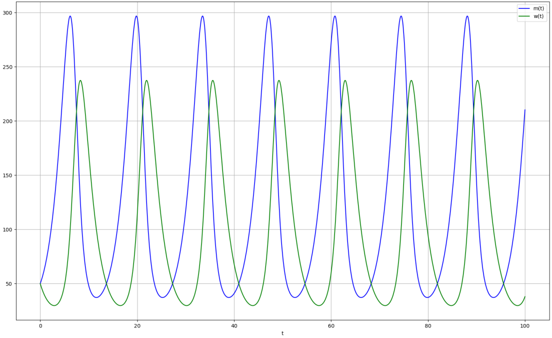
The populations of wolves and moose of Isle Royale have been observed for more than 50 years. Result: Dynamic variation rather than ‘balance of nature’.

- More wolves
- ... leads to less moose
- ... leads to less wolves
- ... leads to more moose.

Wolves and Moose: Classical Model

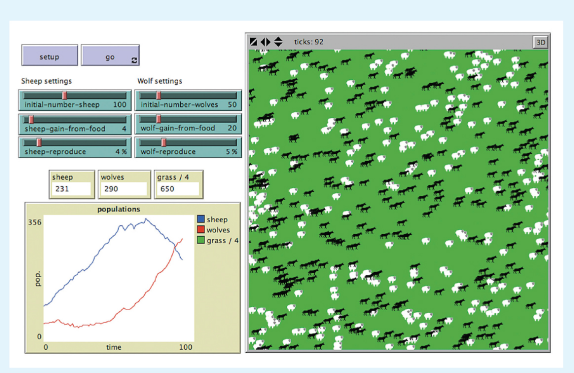
Lotka-Volterra model for wolf (w) and moose (m) populations:

$$\frac{\delta m}{\delta t} = k_1 m - k_2 wm, \frac{\delta w}{\delta t} = -k_3 w + k_4 k_2 wm$$



Wolves and Moose: Agent-Based Model

- Spawn *m* moose and *w* wolves and invoke each agent's behavior in each loop:
  - ask moose [move death reproduce-sheep]
  - ask wolves [move set energy energy - 1 catch-sheep death reproduce-wolves]



Discussion: Pros and Cons

Differential Equations

- Pro: Mathematically well understood, analytical inference by using calculus, many tools available (e.g., Matlab)
- Con: Hard to explain, models phenomenon rather than behavior, harder to extend

Agent-Based Model

- Pro: Easy to understand and to explain to stakeholders, models individual beahvior and observes emergent phenomenon, easy to extend
- Con: Tool support improves slowly, no analytical tools comparable to calculus

Modeling Traffic

- Observation: Traffic on the motorway produces certain patterns.
- Question: Can similar patterns be algorithmically reproduced?
- Agent-Based Simulation approach:
  - Modeling traffic on the motorway as a multi-agent system
  - Cars (drivers) as agents
    - Percepts: Distance to next car in front
    - Internal State: Current Speed
    - Actions: Speeding, braking

Nagel-Schreckenberg Model: Motivation

- Research Question: How do traffic jams emerge?
- Research Hypothesis: Might be due to the local behaviour of individual agents.
- Approach: Model traffic as a MAS and study the resulting system's behavior. If the systems' behavior matches empirical phenomenon, then the model might be an acceptable explanation.

Cellular Automaton

- A cellular automaton is a quad-tuple $A = < R, Q, N, \delta >$
- A cell space R
- A set Q of states each cell can be in
- A neighborhood $N : R \rightarrow 2^R$
- A transition function $\delta : Q^{|N|} \rightarrow Q$
  - For a probabilistic cellular automaton, δ is a probability distribution $P(r = q | N(r))$
- The configuration of A can be written as $x_1 x_2 \ldots x_n$ with x_i being the state of the cell r_i.

Nagel-Schreckenberg Model: Representation

- Traffic is modeled as $A = < R, Q, N, \delta >$
- Entities of $R = \{c_1, c_2, \ldots\}$ stand for parts of the lane
  - Each cell corresponds to a discrete part of the lane (roughly the space needed by a car)
- $Q = \{0, \ldots, v_{max}, free\}$: Each cell is either occupied by one car with velocity $v \leq v_{max}$, or it is empty.
- $N(c_i) = \{c_{i-v_{max}}, \ldots, c_{i+1}\}$
- δ is realized by a set of four rules executed by each driver

Nagel-Schreckenberg Model: Rules

- Each car at cell c_i with velocity v performs four consecutive steps:
  - Acceleration: If $v < v_{max}$ and gap to next car is larger than $v+1$, then increment speed by 1.
  - Slowing down: If the next car is at cell $i+j$ with $j \leq v$, then reduce speed to $j-1$.
  - Randomization: If $v > 0$, then decrement v by 1 with probability p.
    - Car does not accelerate although it could (takes back Acceleration)
    - Car reached maximal velocity but slows down again
    - Overreaction when braking
  - Car motion: Move forward v cells.

Nagel-Schreckenberg: Example

2 _ _ 2 _ _ 2 _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

_ _ _ _ _ _ _ _ _

Nagel-Schreckenberg: Example

2 _ _ 2 _ _ 2 _ _

_ _ 2 _ _ 2 _ _ 2

_ 2 _ _ 2 _ 1 _ _

Nagel-Schreckenberg: Example

2	_	_	2	_	_	2	_	_
_	_	2	_	_	2	_	_	2
_	2	_	_	2	_	1	_	_
_	_	_	2	0	_	_	_	2

Nagel-Schreckenberg: Example

```
2 _ _ 2 _ _ 2 _ _
_ _ 2 _ _ 2 _ _ 2
_ 2 _ _ 2 _ 1 _ _
_ _ _ 2 0 _ _ _ 2
_ 2 _ 0 _ 1 _ _ _
```

Nagel-Schreckenberg: Density and Flow

- Assume constant system density: $\rho = \frac{|Ag|}{|R|}$
- For a fixed cell c_i, time-averaged density over time interval T:

$$\bar{\rho}^T = \frac{1}{T} \sum_{t=t_0+1}^{t_0+T} n_i(t)$$

- ... with $n_i(t) = 1$ if i is occupied, else $n_i(t) = 0$
- Time-averaged flow $\bar{q}$ between i and $i+1$:

$$\bar{q}^T = \frac{1}{T} \sum_{t=t_0+1}^{t_0+T} n_{i,i+1}(t)$$

- ... with $n_{i,i+1}(t) = 1$ if some car moved between i and $i+1$ at t, else $n_{i,i+1}(t) = 0$

Nagel-Schreckenberg: Fundamental Diagram

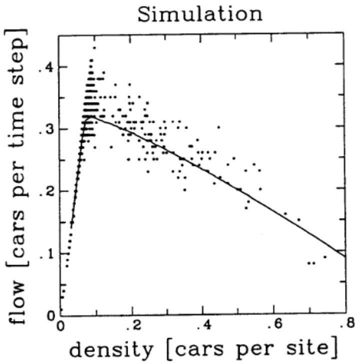


Fig.: Source: [5]

Goal-Based Agent

function GOAL-BASED AGENT(*percept*)
    global *state*, *actions*, *goals*
    state ← UPDATE-STATE(*state*, *percept*)
    predictions ← PREDICT(*state*, *actions*)
    action ← BEST-ACTION(*predictions*, *goals*)
    state ← UPDATE-STATE(*state*, *action*)
    return *action*
end function

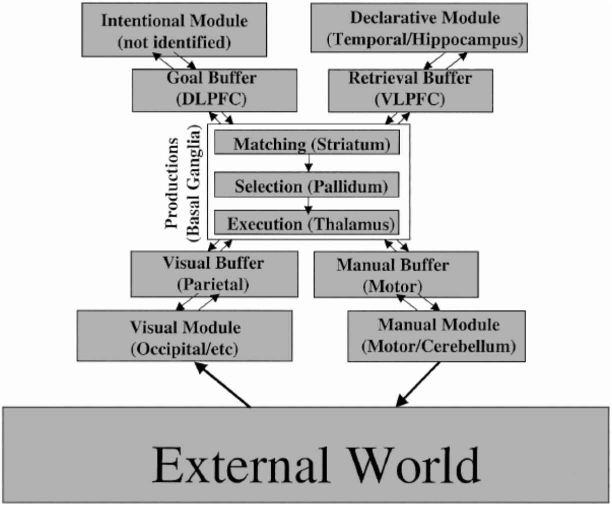
- Practical reasoning more flexible due to explicitly representing actions and goals instead of rules, i.e., “Will the world state be consistent with my goals if I execute action A?”

Utility-Based Agent

function UTILITY-BASED-AGENT(*percept*)
  global *state*, *actions*, *utilities*
  state ← UPDATE-STATE(*state*, *percept*)
  predictions ← PREDICT(*state*, *actions*)
  action ← BEST-ACTION(*predictions*, *utilities*)
  state ← UPDATE-STATE(*state*, *action*)
  return *action*
end function

- Practical reasoning more decisive due to the ability to take utilities into account, i.e., “Is action A the best action among the available actions?”

Cognitive Agent: ACT-R



ACT-R: Activation and Learning

- Activation
  - Entries in the declarative memory are called chunks
  - Chunks have a degree of activation
  - Activation of chunks activates associated chunks
  - Chunks' activation descreases over time and fall below the retrieval threshold (forgetting)
- Utility Learning
  - The rules of an ACT-R agent are called productions
  - Production have utility: $U_i = P_i G - C_i$
  - Probability of success: $P = success / (success + failures)$
  - Cost equation: $C = \sum_j effort_j / (successes + failures)$
  - G: Some fixed importance of the current goal
  - Production choice: $Prob_i = e^{U_i / noise} / (\sum_j^n e^{U_j / noise})$

BDI Agent

function BDI-AGENT(*percept*)
  global *beliefs*, *desires*, *intentions*
  beliefs ← UPDATE-BELIEF(*beliefs*, *percept*)
  desires ← OPTIONS(*beliefs*, *intentions*)
  intentions ← FILTER(*beliefs*, *intentions*, *desires*)
  action ← MEANS-END-REASONING(*intentions*)
  beliefs ← UPDATE-BELIEF(*action*)
  return *action*
end function

- BDI agents start out with some beliefs and intentions.
- Intentions are goals the agent has actually chosen to bring about (can be adopted and dropped).
- Beliefs and intentions constrain what the agent desires.
- Together, B, D, and I determine the agent's future intentions.

BDI Frameworks

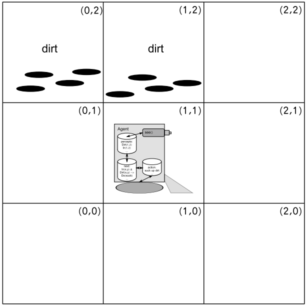
- Just to name a few
  - Jason: `http://jason.sourceforge.net/`
  - 3APL: `https://en.wikipedia.org/wiki/3APL`
  - 2APL: `http://apapl.sourceforge.net/`
  - JADEX: `http://vsis-www.informatik.uni-hamburg.de/projects/jadex/`
  - GOAL: `https://goalapl.atlassian.net/wiki`
- Different technologies, e.g., Prolog-style knowledge bases vs. XML files vs. Java Objects
- Different formalizations of BDI, e.g., AgentSpeak, GOAL

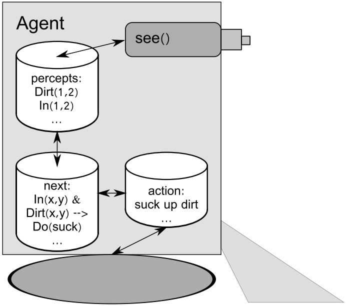
Cognitive Agents in GOAL

- GOAL emphasizes programming cognitive agents.
- Cognitive agents maintain a cognitive state that consists of knowledge and goals.
  - Knowledge: Facts the agent believes are true.
  - Goals: Facts the agent wants to be true.
- Cognitive state is represented in some knowledge representation (KR) language.
- Cognitive agents derive their choice of action from their knowledge and goals.

Example: The Vacuum World

- Percepts: dirt, orientation (N, S, E, W)
- Knowledge: In/2, dirt/0, clean/0. initial KB: In(0, 0), $\neg$*clean*
- Goal: clean [*Note: clean cannot be perceived but must be inferred!*]
- Actions: suck, step forward, turn right (90°)





Programming language GOAL

- Mind-body metaphor:
  - Agents (mind) are connected to controllable entities (body) living in some environment.
  - Agents receive percepts from the environment through their controlled entities.
  - Agents decide what the controlled entities will do.

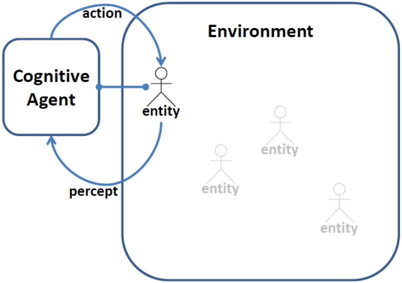
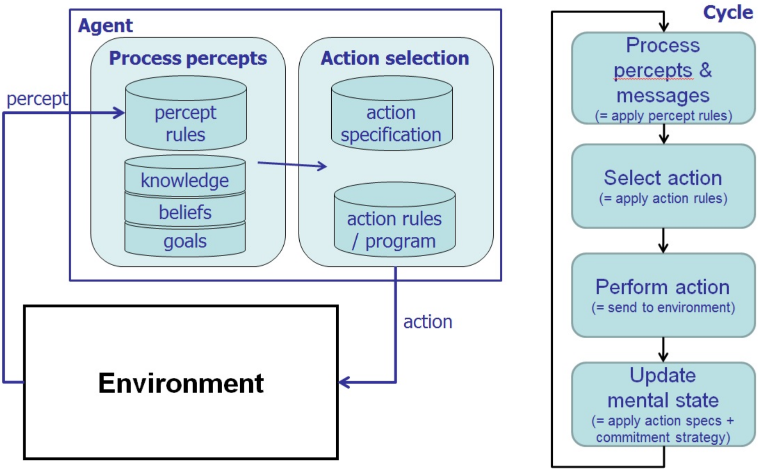


Fig.: Source [1]

- Controlled entities: a bot in Unreal Tournament, a robot, …

GOAL Execution Cycle



Literature

- Hindriks, K. V., Programming Cognitive Agents in GOAL, Technical Manual, 2017, https://goalapl.atlassian.net/wiki/.
- Brachmann, R. J. & Levesque, H. J., Knowledge Representation and Reasoning, 2004, Morgan Kaufmann Publishers.
- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, second edition, Prentice Hall, 2003.
- U. Wilensky, W. Rand, An Introduction to Agent-Based Modeling, MIT Press, ISBN: 9780262731898, 2015.
- K. Nagel, M. Schreckenberg (1992), A cellular automaton model for freeway traffic, J. Phys. I France 2, pp. 2221–2229.