Multi-Agent Systems

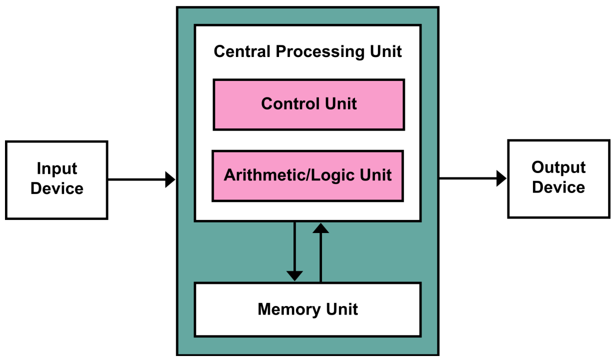
Albert-Ludwigs-Universität Freiburg

Bernhard Nebel, Felix Lindner, and Thorsten Engesser

Summer Term 2017

Course outline

Von-Neumann Architecture



Agent Architectures

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

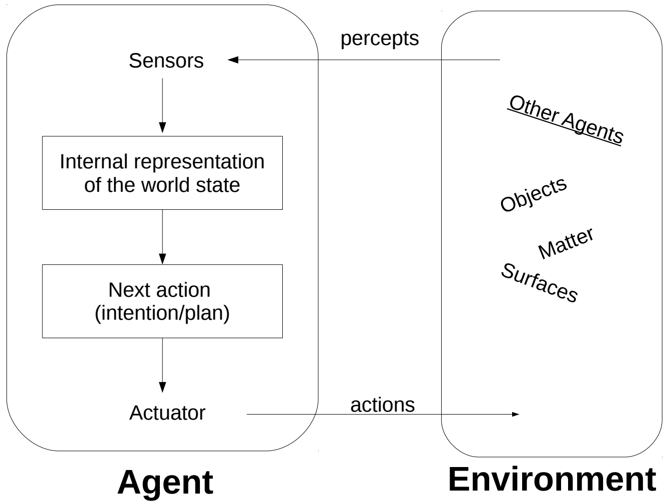


Table-Driven Agent

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

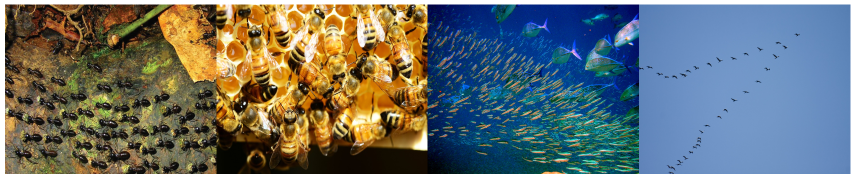
- Epistemic state is the list of percepts so far perceived.
- Practical reasoning based on look-up table.

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 is 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.

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

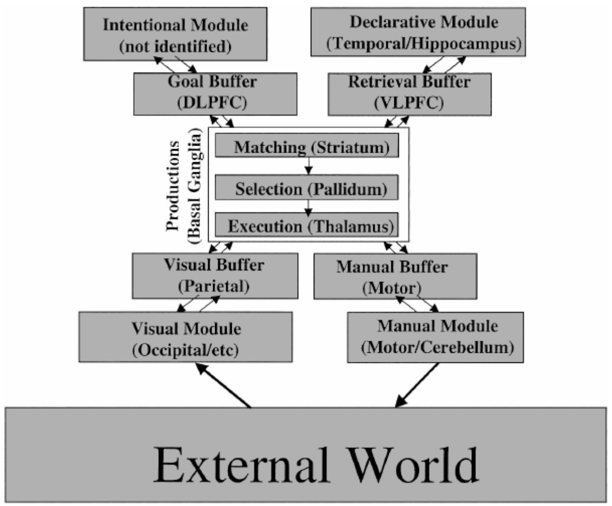
- Practical reasoning more flexible due to explicitly representing actions and goals instead of rules, i.e., "Will the world state be consistent with my goals if I execute action A?"

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.

Quiz

- Let us think back to the first session of this lecture and to the entities we identified as agents:
  - Humans
  - Animals
  - Plants
  - (Self-driving) cars
  - Light switches
- Which architecture would you pick to implement each of these agents?

Course outline

Literature I

Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, second edition, Prentice Hall, 2003.

K. Sugihara, I. Suzuki, Distributed Algorithms for Formation of Geometric Patterns with Many Mobile Robots, Journal of Robotic Systems, Vol. 13, No. 3, pp. 127–139, 1996.