Design of Physically Grounded Communication System
実世界コミュニケーション特論

Michita Imai
今井 倫太

Intelligent System in Real World

- The system must deal with
  - Dynamic changes in environments
  - Unpredictable events
  - Vast number of entities
    - Difficulty to make a model
- Vertical System and Horizontal System

Typical design of a system

- Horizontal decomposition with vertical slices
  - A traditional decomposition of a system is based on functional modules
    - Functional modules: Sensing, Mapping sensor data into a world representation, Planning, Task execution, Motor control
    - Modeling and planning has an expression corresponding to the virtual expressions of SHRDLU.

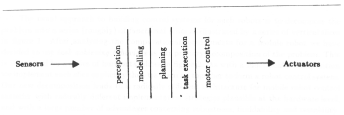
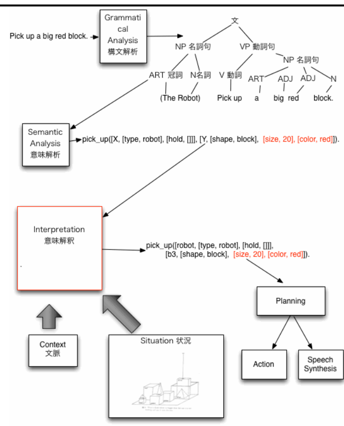


Figure 1. A traditional decomposition of a mobile robot control system into functional modules.



Typical design of a system

- A designer must consider the overall performance of a robot.
- Difficulty of modification

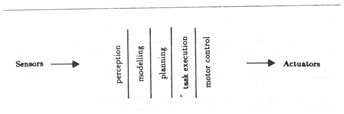


Figure 1. A traditional decomposition of a mobile robot control system into functional modules.

Typical design of a system

- Disadvantage of the traditional decomposition is that the system cannot react the changes in an environment immediately.

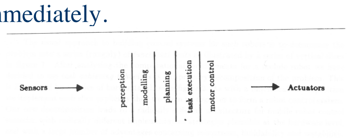


Figure 1. A traditional decomposition of a mobile robot control system into functional modules.

Horizontal System

- A decomposition of a control system based on achieving behaviors
- Subsumption Architecture (SSA)

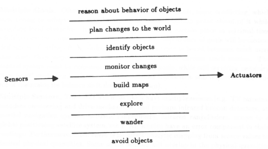


Figure 2. A decomposition of a mobile robot control system based on task achieving behaviors.

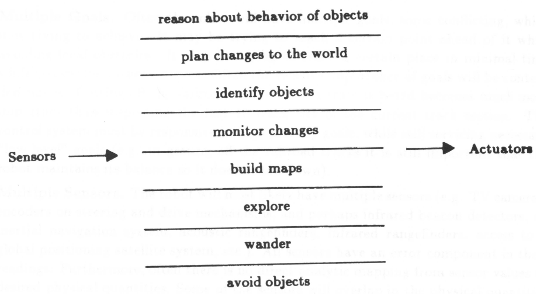
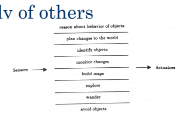


Figure 2. A decomposition of a mobile robot control system based on task achieving behaviors.

- https://www.youtube.com/watch?v=K2xUHYFcYKI

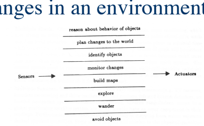
Horizontal System

- How to design
  - Decomposing the problem vertically
  - Rather than the slice based on internal workings of the solution, a designer should slice the problem on the basis of desired external manifestations of the system.
  - The designer is able to design each module independentlv of others



Horizontal System

- Complex behaviors are the reflection of a complex environment.
  - The complex behaviors of traditional systems come from the internal computations.
- Easy to react the changes in an environment immediately



Requirements of a robot in Real World

- Getting sensor input and generating behaviors
- Multiple Goals
- Multiple Sensors
- Robustness
- Additivity

Multiple Goals

- There are several goals simultaneously
  - EX.
    - A robot tries to reach a certain place while avoiding local obstacles.
    - It tries to reach a certain place in minimal time while conserving power reserves.
  - The relative importance of goals will be context dependent.
  - Behaviors of the horizontal system are selected in response to the structure of environments.
- It need not have a model of the environments.

Multiple Sensors

- Many sensors
  - CCD cameras, encoders on steering and drive mechanism, infrared beacon detectors, an internal navigation system, laser rangefinders, sonic sensors, GPS, and etc.
- The robot must make a decision under the conditions.
  - No direct analytic mapping from sensor values to desired physical quantities
  - They overlap in the physical quantities they measure.
  - Inconsistent readings
    - Normal errors
    - Errors based on use outside its domain of applicability
    - No analytic characterization of the domain of applicability

Robustness

- A robot must be robust to deal with
  - Sensor fails: when some sensors fail, the robot should be able to adapt and cope by relying on those still functional.
  - Dynamic environment: when the environment changes drastically, it should be able to still achieve some sensible behaviors.
  - Hardware fails: when there are faults in parts of its processors...

Additivity

- Hardware requirement
  - When more sensors and capabilities are added, it needs more processing power.

Robot

- Mobile robot
- Sensors
  - a ring of twelve sonar
  - pan tilt camera

Levels of competence

- *Level 0:* Avoid contact with objects
- *Level 1:* Wander aimlessly around without hitting things
- *Level 2:* Explore the world by seeing places in the distance which look reachable and heading for them.
- *Level 3:* Build a map of the environment and plan routes from one place to another.
- *Level 4:* Notice changes in the static environment.
- *Level 5:* Reason about the world in terms of identifiable objects and perform tasks related to certain objects.
- *Level 6:* Formulate and execute plans which involve changing the state of the world in some desirable way.
- *Level 7:* Reason about the behavior of objects in the world and modify plans accordingly.

Level of competence

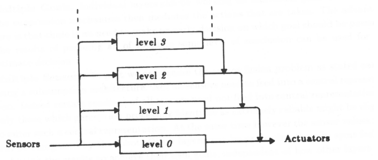


Figure 3. Control is layered with higher level layers subsuming the roles of lower level layers when they wish to take control. The system can be partioned at any level, and the layers below form a complete operational control system.

Design process of Layers of control

- Building layers of a control system corresponding to each level of competence
- Adding the developed competence on the lower layer.
  - Easy to add a new layer to an existing set to achieve a new behavior
- Starting the development from level0
- Add a higher layer on the lower layer.

Design process of Layers of control Development of *level 0*

- Starting by building a complete robot system which achieving *level 0* competence.
  - Never alter *level 0* after it is debugged thoroughly

Design process of Layers of control Development of *level 1*

- Next building *level 1*
  - Examining data from *level 0*
  - Permitted to inject data into the internal interface of *level 0* suppressing the normal data flow in *level 0*.
- The achievement of *level 1* competence with the aid of *level 0*
- *Level 0* continues to run unaware of the layers above level 0 which sometimes interfere with its data paths.

Design process of Layers of control Development of higher levels

- The work you must do in designing the rest of the layers is to repeating the same process for the higher levels.
  - Developing a competence and adding it on the lower layer.
- The feature of the incremental development
  - In the scheme of the development, the robot is able to run as soon as you have built the first layer.
  - Additional layers can be added later
  - The developed lower layers need never be changed.

Robot control system instance

- *level 0*: prevent a robot from contacting with an object.
- *level 1*: wander avoiding obstacles.
- *level 2*: generate a path to reach a certain place.

Robot control system instance
Instance of *level 0*

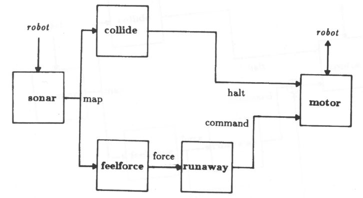


Figure 5. The level *0* control system.

Robot control system instance
Instance of *level 1*

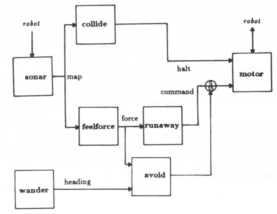


Figure 6. The level *0* control system augmented with the level *1* system.

Instance of *level 2*

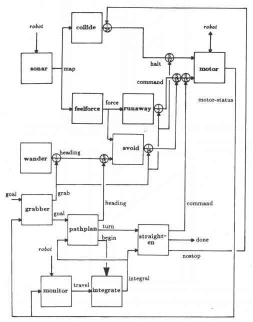


Figure 7. The level *0* and *1* control systems augmented with the level *2* system.

Components for SSA

- Internal structure of modules
- Communication structure between modules

Components for SSA
Internal Structure of Modules

- Each module is a finite state machine.
  - Lisp data structure
  - Four state types
    - Output: An output message is sent to line.
    - Side effect: One of the module's variables is set to a new value.
    - Conditional dispatch: A predicate on the module's variables and input buffers is computed. Depending on the outcome, one of two subsequent states is entered. Branch.
    - Event dispatch: A pair of conditions and states to branch to are monitored until one of the event is true. The events are in combinations of arrivals of messages on input lines and expiration of time delays.

Components for SSA
Example of avoid module

- *(defmodule avoid*
  :inputs (force heading) ← *force = distance*^{5}
  :outputs (command)
  :instance-vars (resultforce)
  :states
  ((nil (event-dispatch (and force heading) plan))
  (plan (setf resultforce (select-direction force heading)) go)
  (go (conditional-dispatch (significant-force-p resultforce 1.0)
  start
  nil))
  (start (output command (follow-force resultforce))
  nil)))

Components for SSA
Communication between modules

- Output may be inhibited.
- Input may be suppressed.

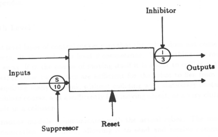


Figure 4. A module has input and output lines. Input signals can be suppressed and replaced with the suppressing signal. Output signals can be inhibited. A module can also be reset to state NIL.

- *(defwire (feelforce force) (runaway force) (avoid force))*
- *(defwire (avoid command) ((suppress (motor command) 1.5)))*

Simulation of SSA

- Sonar data

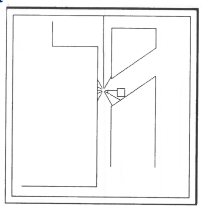


Figure 8. The simulated robot receives 12 sonar readings. Some sonar beams glance off walls and do not return within a certain time.

Simulation of SSA
Simulation of level 1

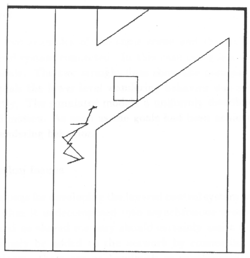


Figure 9. Under levels *0* and *1* control the robot wanders around aimlessly. It does not hit obstacles.

Simulation of SSA
Simulation of level 2

- After tracing the given path, level 1 controls the robot.

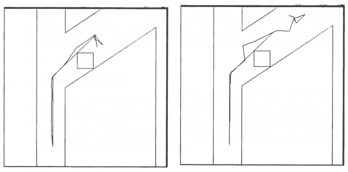


Figure 10. With level *2* control the robot tries to achieve commanded goals. The nominal goals are the two straight lines. After reaching the second goal, since there are no new goals forthcoming, the robot reverts to aimless level *1* behavior.

Problems tackled by SSA

- Multiple Goals
  - Individual layers works on individual goals concurrently.
  - The suppression mechanism between layers mediates the action selection.
  - The advantage is that there is no need to make an early decision which goal should be pursued.
- Multiple Sensors
  - SSA can ignore the sensor fusion problem (No general expression)
  - Not all sensors need to feed into a central representation.
    - Only those which perception processing identifies as extremely reliable might be eligible to enter it.
  - At the same time, other layers may be processing them to achieve their own goals independent of how other layers use them.

Problems tackled by SSA

- Robustness
  - The use of multiple sensors
  - Lower levels have been well debugged and continue to run independent of higher layers.
    - The lower level continues to produce outputs if the higher layers cannot produce results in a timely fashion.
- Additivity
  - Easy to spread the layers over many loosely coupled processors.
  - Each layer can be implemented on its own processor.
  - Communications between them require fairly low bandwidth.

Advantage of Layers design

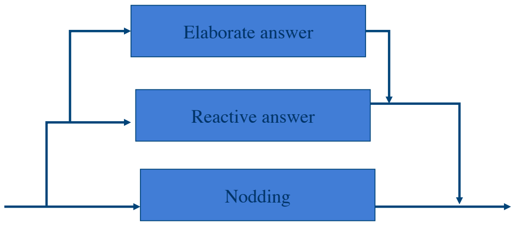
- Behavior selection reflects the structure of environments themselves.
  - No need to knowledge to select them.

Advantage of Layers design

- The designers are free to use different decomposition for different sensor-set task-set pairs.
- The processors send messages over connecting wires.
  - There is no handshaking or acknowledgement of messages.
  - There is no other communication path. Ex. Shared memory.
  - The processors run completely asynchronously monitoring input wire and sending messages on output wires.
- There is no central control.
  - All processors are created equal.
- Higher level layers subsume the role of lower levels.
  - Inputs to modules can be suppressed and outputs can be inhibited by wires terminating form other modules.

Example of implemented systems
Robot with SSA

Example of implemented systems
SSA for Communication



Example of implemented systems
SSA for Communication

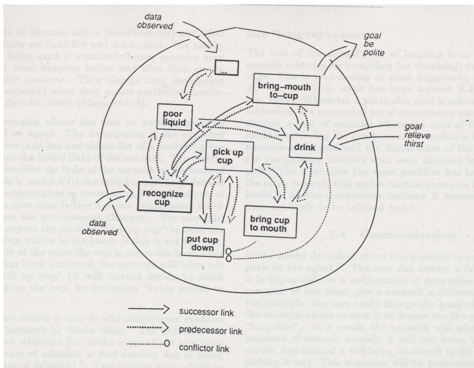
- Stammering Computer (Incremental utterance generation)
  - 口ごもるコンピュータ
- Computer keeping its ears cocked
  - 聞き耳をたてるコンピュータ

Disadvantage of SSA

- Consistency between goal-oriented behavior selection and environment-oriented behavior selection
- Difficulty of developing actual implementation
- Self-recognition

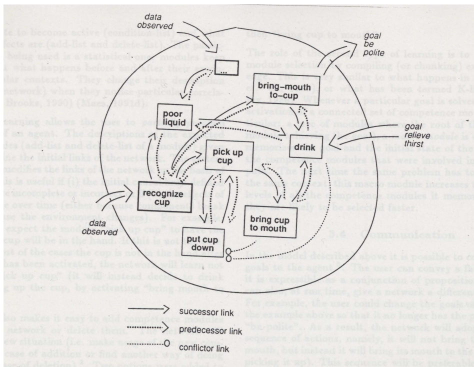
3.5 ANA (Agent Network Architecture)

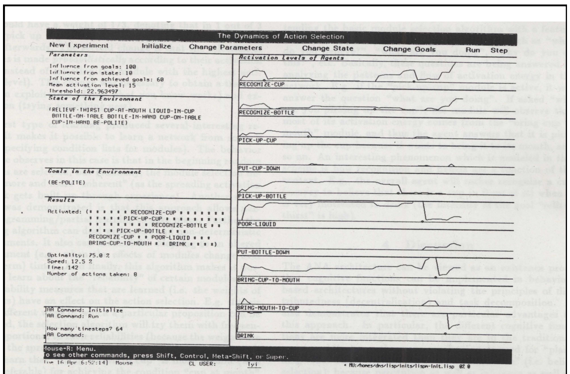
- Behavior generation satisfies a goal-oriented method and an environment oriented method.
- ANA has both features of planner and SSA.
- The features vary depending on parameters.



3.5.1 ANA

- Calculation process
  1. Energy from environmental information
  2. Energy from goals
  3. Energy from other agents
  4. Agents which has highest energy and executable conditions and whose energy is beyond a threshold θ are executed.
  5. Update world model (add-list and delet-list)
  6. Try 1-5 until given goals are satisfied.





ure 2: An example run of the module selection algorithm. The panes on the right side of the interface show the activa
ls of the modules evolving over time. The small circles on the time axis denote when a module has been activated.
es on the left show (from top to bottom): the global parameter values, the current state of the environment, the cu
s of the agent, the list of modules that have been activated (chronologically, where "*" stands for a timestep in whic

3.5.4 The effect of parameter setting

- The effect of parameter values

$$\theta \propto \frac{1}{speed} \qquad \theta \propto optimality$$

$$\beta \propto goal-oriented$$

Env-orienteted ⟵⟶ *Goal-oriented*

$0 < \beta < 1.0$

3.5.5 Effect of environments and goals

- The effect of environments and goals

$$\#\,agents-realizing-goals \propto time-necessary-for-goal$$

$$\#\,goals \propto \frac{1}{goal-orientedness}$$

$$\#\,propositions-in-state \propto \frac{1}{data-orientedness}$$

Applications of ANA

- Applications for dialogue system
  - ANA as dialogue planner
  - Easy to adapt changes in subjects

Disadvantages of ANA

- Disadvantage
  - The problem of planner
    - Symbol grounding problem

Focus of Attention

- They are talking about what she is reading.

Top-down manner
Focus of Attention

Intentionally select information relevant to conversation

Focus of Attention

Bottom-up manner
Focus of Attention

Direct Attention in response to an event

Real World

- Situation is important.
  - If you want a book,
  - If you are searching for your key,
  - If you are searching for someone,

Focus of Attention

- They are talking about what she is reading.

Top-down manner
Focus of Attention

Intentionally select information relevant to conversation