Behaviour-based control in ARGoS: The Subsumption Architecture

Observations

– Intelligent Robotic Systems –

Andrea Roli
andrea.roli@unibo.it
Dept. of Computer Science and Engineering (DISI)
Alma Mater Studiorum Università di Bologna

Most common solution proposed by the class: Competences implemented as functions that return either velocities or a "subsumption message" (e.g. inhibition flag for a level) or both.

- It is important that the functions set to `true` the "subsumption message" on the basis of the readings from the sensors, so that each function has its own mechanism for deciding whether to take control. This is in general the most appropriate implementation for the subsumption architecture.
- These functions can be either nested or simply called sequentially with final selection of the signals to be sent to the wheels.
- The order in which levels are called depends on che choice of the control scheme.
- A good practice consists in separating conditions from actual activation code.
- In some solutions levels send outputs to the level above them. This is mainly to achieve a kind of fusion of behaviors (as indeed we see in the paper by Brooks).

Levels

In all the solutions proposed, `halt` is the uppermost level. Moreover, usually obstacle avoidance has a higher priority than phototaxis (i.e. it is more important to avoid collisions, so this level takes control in these cases). This is probably the easiest way for implementing the subsumption architecture in a control software, because it preserves independence and responsibility of individual behaviours. However, in the classical hardware implementation of a controller (as from the example provided by Brooks in his foundational paper), the lowermost levels correspond to a kind of reflex behaviours. Accordingly, in our task obstacle avoidance should be at the bottom and phototaxis the uppermost. In this case, the decision that the phototaxis level takes has to be based also with respect to to proximity sensor values; in other terms, the level decides whether it is safe to act, otherwise it "knows" that a lower level will take control. This latter option is viable and even unavoidable if the robot is already equipped with obstacle avoidance and a further level with a heading direction has to be added. Nevertheless, it requires some degree of redundancy or reuse of software that composes other levels.

Observations

- The implementation on the robot model in ARGoS requires some adjustments, as the parallelism of the architecture can not be achieved. However, we can encapsulate all the computation related to a competence in one function, and set a priority among these levels; then we can either call them either from the highest priority function (and we use locks or similar mechanisms to suppress outputs of lower levels) or from the lowest one (implementing an overriding of the variables used to control the actuators).
- The temptation of programming the overall behaviour simply by means of nested IF-THEN-ELSE structures is natural, but this makes the code less extendible (not in the spirit of the subsumption architecture) and error prone.
- It is not trivial to decide the hierarchy of the levels and often this depends upon the personal viewpoint of the task (remember that the segmentation of behaviour depends on the observer).
- The issue of setting thresholds has to be faced.

Take-home messages

- The architecture forces designers to neatly structure their code.
- The simpler the implementation of competences the easier to combine them. It is preferable to have simpler competences, than just a few but rather complicated.
- If the architecture is clean then it is easy to add new competences.