Behaviour-based control in ARGoS: The Subsumption Architecture

– Intelligent Robotic Systems –

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

The subsumption architecture

The *subsumption architecture* is one the most used models for implementing behaviour-based control. In this lab activity you are asked to program a robot for a given task according to this architecture. Therefore, a major requirement of this activity is to implement the behaviour trying to keep the main features of the subsumption architecture, with the obvious limitation that here it is not possible to implement actual parallel processes.

Task

The robot is expected to be able to find a light source and go towards it, while avoiding collisions with other objects, such as walls, boxes and other robots. The robot should reach the target as fast as possible and, once reached it, it should halt. A black spot underneath the light bulb identifies the halt region. For your convenience, an `argos` file defining the arena is provided. This defines the distribution of typical arenas for which you have to devise your controller. For physical constraints the wheel velocity cannot exceed the value 15 (i.e., 15^{-2} m/s). The robot (a *footbot*) is equipped with light sensors, proximity sensors, and ground sensors. In addition, the robot is also equipped with the positioning sensor, which however can only be used for testing and evaluation purposes.

Exercise

Implement the robot control program in ARGoS on the basis of the subsumption architecture. Before starting to code, define the basic task-achieving behaviours (i.e. *competences* according to Brook's terminology) and detail their relations, mainly their position in the stack of competences. Design your code such that a further level can be added with a rather limited amount of changes in the previous code. Suggestion: implement each level by means of a function that reads from the sensors all the data needed for its competence and implement `step()` by properly combining the functions without a central control (i.e. avoid using a centralized dispatcher that checks the conditions and chooses the function to be run); instead, functions should communicate and coordinate e.g. via arguments (input and output ones), such that the addition of an extra layer would consists in just adding a function.

Compare the behaviour of the robot in a condition without noise and another with noise (set noise level in both actuator and sensors; you may want to set the noise level at about 0.01).

Some suggestions:

- Consider the fact that you have an abstract machine in which only sequential processes can be run; therefore, some variants with respect to the subsumption architecture are in fact unavoidable.
- Try to think of the possible ways to implement a competence and do not stop at the first idea that comes to your mind. You will be surprised to discover that there are several possibilities, even for this simple task.
- Try the controller also with more than one robot in the arena (just place n robots randomly by using `distribute`). This will be useful to test the collision avoidance module in presence of moving objects.
- As the position of the light bulb is fixed and known in advance, you can compute the distance between the robot and the target by means of the positioning sensor. Therefore, you can evaluate the performance of the robot by estimating the time needed to reach the light, e.g. in terms of number of steps. Alternatively, you can either count how many times the robot can reach the light in a given time (just set the simulation duration in the ARGoS configuration file) or take the distance between the robot and the light at the end of the simulation. Question: what is a sound way for statistically estimating the performance of the controller?