Some problems of Programming in Robotics

Eleonora Bilotta
Pietro Pantano

Centro Interdipartimentale della Comunicazione
Università della Calabria
eleb@abramo.it
piepa@abramo.it

Keywords: POP-I.C. Behaviour-Based Robotics, Educational Technology.

Abstract

Although there is a vast bibliography on robot control, only in recent year a lot of attention is placed on how to teach control to children and the consequent problems related to this programming activity. In fact, many are the design problems in relation with a good programming activity in robotics.

In this paper, we analyse some of these problems. In particular:

- programming the robot control;
- the organisation of the program in relation to hardware, software, behaviours and performance design in robotics;
- some educational remarks.

Introduction

Artificial intelligent behaviour has been thought as input, control and output as separate functions. This idea, penetrated in Psychology from Physiology, has influenced some research approaches in Psychology, in Artificial Intelligence in Robotics. Alternative approach came from Zoology, in which it has been recognised that, very simple animals, without a central nervous system, have complex behaviours and are well integrated with their environment. It has been argued that robots, which are still at a very early stage of evolution, could be designed better following the evolving taxonomic scale of animal species, instead of modelling their behaviours based on a central activity, in analogy with the human higher cognitive activities. This kind of *bottom-up robotics* or *behaviour-based robotics* is the to day standard approach for designing robots.

Braitenberg (1984), with his *synthetic experiments* put in evidence that simple machine, equipped with sensors and motors, can exhibit very complex behaviours. Brooks (1986) argued that intelligent behaviours could emerge even from very simple stimulus-response activity.

In this approach, many researchers distinguish between traditional decomposition of intelligent behaviours by functions and the alternative decomposition by activity. The former involves a serial input-output architecture; the latter involves a number of task-accomplishment procedures, acting in parallel. In defining intelligent behaviours what matters is the behavioural outcome, not the nature of the mechanism by which the outcome is achieved.

Some key-points of this approach are:

- intelligent behaviour needs a body: it is not possible to behave without a body. So, robots have to be real;

- animal and robots affect the environment in which they live and they are affected by it. So intelligent behaviour has to be grounded in the environment;
- it is also necessary to establish a scale of assessment to evaluate intelligence in robots.

The main features of behaviour-based systems are functionality, behaviour, mechanism and components.

Functionality is something robot has to reach or to acquire (for examples, locomotion, recharge, obstacle avoidance, search for the recharge station, measurement, communication with other robots). Other words to identify functionality are task, target and competence. Functionality is related with the *observer* or *designer.*

Behaviour is described as regularity in the dynamic of interaction between robot and its environment (for example, to have the same distance from a wall, to change continually its own position in a specified direction). One or more behaviours give birth to functionality. Also, behaviours are related with the *observer* vocabulary.

Mechanism it is a principle or a technique to establish a particular behaviour (for example, a specified pair between sensing and acting).

Component is the hardware side of the robot, a physical piece or a process that is used to implement the mechanism. Components are sensors, body pieces, actuators, data structures, programs, hardware and software of communication.

How it is possible to design and build up intelligent behaviour-based robots?

There are three design approaches (Arkin, 1998):

The first, *ethologically, guided/constrained* design is based on the following step by step methodology:

- a phenomenon to be reproduced in a real robot (for example, avoiding obstacle behaviour) it is identified;
- an artificial system, which has this ability, it is built up;
- the system is included in the environment to work;
- all resulting phenomena are recorded;
- results are compared with original phenomenon in other animals;
- bad functionality is improved.

The results of these experiments have two customers, roboticists who can use these insights to produce machines that are more intelligent and experimental biologists, which can develop and test their theories of animal behaviours.

In the *situated activity-based design*, the robot's actions are strictly integrated in the situations in which it finds itself. Hence, the problem is to make the robot to perceive the situations it is in and then choosing one action to undertake, and so on: when the robot is in another situation, it selects a new and more appropriate action. Designing a robot based on this methodology means designer has to understand the relationship between the robotic agent and its environment and to describe and specify by means of micro-behaviours, all possible actions the robot can undertake.

Usually the methodology is the following:

- assess robot-environment dynamics;
- partition into situations;
- create situational responses;
- import behaviours to robot;

- run robotic experiments;
- evaluate results;
- enhance, expand, correct behavioural responses.

The third approaches, *experimentally driven design*, is based on the assumption to endow a robot with a limited set of capabilities, run the experiment in real world, see what works and what doesn't, debug imperfect behaviours and then add new behaviours, until the overall system exhibits a good performance. The methodology is the following:

- build minimal system;
- exercise robot;
- evaluate results;
- add new behavioural competence.

Starting from these complex set of concepts, and related designing and implementation techniques, how to teach children robotics? New tools and new teaching methodology are necessary to transfer robotics background to children.

Many constructivist environments have been created to teach children to programme in robotics. Resnick and his group (1989; 1994), for example, experimented how children build up "artificial organisms". In the first phase of the experiment, children build the *body* of the robot (the hardware). In the second, they programme the *mind* (the program), which controls the behaviour of the robots. The principal achievement of this experimentation is that children realise that, as with living organisms, there are different solutions in programming robot's behaviour: these solutions, produce change in the *mind* and in the body of the *robot*. Martin (1994), who worked with undergraduate and graduate students, pointed out some important problems relevant to robotics design. He proposed a "living laboratory" to explore design in robotics. He said that students make mistakes, due to the unrealistic view of their robot's capabilities: students fail to view the robot from its own point of view, relying on the sort of unrealistic abstractions they have been trained to use for most of their education. Trough experimentation with sensors, motors and control, students modify their view of interaction between the robot and real world, continuously modifying their design process, until it becomes a realistic one and according with their needs.

Some other researchers (Lund et al., 1998) explored the concept of *development* (of robot behaviour) *without programming*, creating an experimental situation in which children, without any programming knowledge, developed robot control systems.

In this paper, we want to make evident that there are many problems and many approaches to teach control to young people, related to different styles of programming and designing the robot behaviours.

Programming in Robotics

Programming in robotics (as in other context) is a complex activity because user has to accomplish different steps. If we could utilise a graphical representation of these steps, we should use a diagram organised as a graph, in which every node has a function and in which the user can make specific activities, linked to the same function (see Figure 1). So the user has different view of the programming activity and can utilise multiple view of the process, to organise better the robot's behaviour.

In the first step user has to define some functionality the robot could do. In the second step, user defines the HW components his/her robot has to have. It is possible to use some pre-arranged platforms, making the user to choose among some HW, (like in Khepera - *http://www.k-team.com/* or Aibo- *http://www.world.sony.com/aibo/index.html*), or to realise it physically. This is possible with some educational Lego kits, actually available. This step is very important from the education point

of view. In the third step, user has to set sensors and motors in order to let the robot to get information from the environment and to modify it.

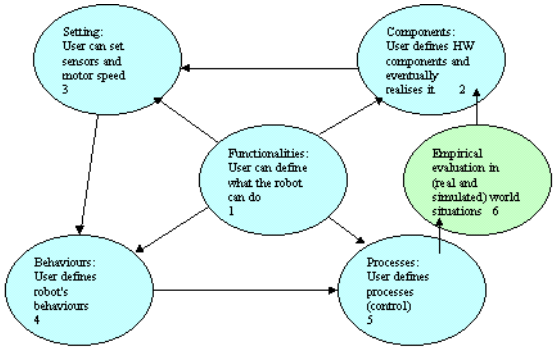

Figure 1: A graphical representation of programming in robotics

In the fourth step, user has to define the robot behaviour. This stage is closed linked with the following, in which user organises the program in order to let the robot to work in a real environment situation. This step is really like an open-ended problem, since user has to plan all the possibility the robot can find in the environment. The fifth step is about programming activity. This means user realises the robot behaviour by a program. The following step regards to run the robot in the environment and to evaluate its performance. In the following, we'll examine deeper these tasks and the related problems.

Planning functionality

To comprehend how behaviour should be programmed in robots, we need to identify some basic functionality the robot has to exhibit. In designing the system functionality, it is necessary to consider the environmental space in which the system will work, the behaviour space and the task space. The *environmental space* defines the constraints, the topology and the laws of movement within which the robot can move. The *behaviour space* includes the variables to which the robot responds in its behaviour and the variables that could be encountered in a dynamic environment. The robot needs to respond only to those variables that are relevant to its behaviour. The robot requires only a simplified view of the environment, indicating which behaviours and movements are possible.

The *task space* includes all possible activities it executes in pursuing the goals robot has.

For example, a robot that has to solve a maze lives in a static environment with walls, streets and has a starting point (usually called *home*) and a location to which it has to arrive to solve the problem (the winner location). The behaviour space of this robot is the set of movements which lets the robot to go forward and backward, to turn left and right. The task space is to solve the maze and to arrive to the winner position.

From the programming point of view, children could use two different strategies:

- begin from simple program which identifies a very simple behaviour (for example, move) and then go on (adding other behaviours, turn right, turn left), until the principal task (solve the maze) is achieved;
- decompose the principal task in a deductive manner, from general (solve the maze) to particular (move).

This strategy implies children know exactly all the behaviours necessary to arrive to the winner position and know how to decompose the principal task in many short pieces. Experiments to ascertain mental models children use to programme functionality should be done.

Defining HW equipment

There is a close correspondence between functionality and hardware equipment. Sometimes, the hardware design that students realise is not well fitted with the environment and most simulated experiments don't evidence these problems. Martin said many students were frustrated when they run their robots in the real situation: they didn't work. Sometime, to make them to work correctly, it is necessary to change the hardware.

It is important, at this stage, to write a very little control program and to run the robot to test if the hardware design is well fitted with the environment. This activity could be a sort of *virtual/real debug* of the program and of the robot. It is also possible to use the above-mentioned experimentally driven methodology.

Some actually available kits to build robots, like Mindstorms are shape oriented, in the sense that it is very easy for children to create or re-arrange the robot's shape. In other pre-organised systems, like Kephera and Koala, this is not so obvious.

Evolutionary (or evolutive) robotics is the sector that deals with changing shape and emergent behaviours in robots (Husbands and Meyer, 1998).

In fact, the principles that apply to the design of robots are similar to those that apply to the design of living organism by natural selection. It has been argued that animals and plants are completely well adapted to the environment they live in. Darwin pointed out that this adaptation could be accounted for in terms of natural selection, which in the evolution has adopted good behaviours and rejected behaviours not very important for survival. Also in robotics, natural selection can be thought as an agent who models robotic behaviours. Therefore, principles that are applied in robotic design are like to those involved in biological processes: the robot should be designed to behave in such a manner that the greatest benefit is attained. In other words, the form of the behaviour is the outcome of a design project that aims at the optimal compromise between characteristic of the environment and its program: form and function of behaviour are related to the environment and to the program the robot has.

Setting sensors and motors

Robotic sensors are related with information that comes from outsider of the robot. The dominant view in recent years has been to create robots essentially reactive, although it is possible to postulate considerable intrinsic constraints on responsiveness. Environmental conditions are held to change the robot behaviour, which in turn changes the robot's relation to its environment. The new stimuli from the environment produce fresh changes in behaviour and so on in chain-like fashion.

Every robot has sensor-actuator sets and a correct setting let the robot to interact in an efficient way with the environment.

In Lego robots, for example, it is possible to connect sensors to the robot and it is possible to make the robot to react to the information sensors carry out from the environment. There are touch, light and rotation sensors.

In a NQC very simple program in which the robot drives forwards until it hits something, a line of the program tells the robot what kind of sensor we have activated. SENSOR_1 is the number of the input to which the sensor is connected. The other two sensor inputs are called SENSOR_2, SENSOR_3. SENSOR_TOUCH indicates it is a touch sensor activated. For the light sensors, we would use SENSOR_LIGHT.

Khepera has 6 infrared sensors and two motors, to which it is possible to add a camera and a gripper.

The emphasis is placed on the interactions taking place between the robot and its environment. Robot's behaviour is the product of continuous interaction between sensors and its environment.

Programming Behaviour

In robotics domain, behaviour means a co-ordinate activity of the robot to an environmental stimulus and a reflex activity is the simplest form of this activity. Stimulus works as a trigger and there isn't a fixed relation between the stimulus and the robot's activity. Repenning (1993) has utilised this model in Agentsheets and the agent's behaviour is programmed by a combined relation between a set of conditions (the agent can sense) and a set of actions (the agent can reply to), triggered by some special variables.

The programme is a set of combined stimulus response behaviours, organised in a chain-like fashion. The programme is sequential; each behaviour runs after another behaviour and so on.

Another approach for designing robots is related to Ethology. Some of these ideas are:

- hierarchical organisation of behaviour;
- action selection mechanisms related to the specific survival value of the chosen behaviour;
- the role of learning in adaptiveness;

This approach stresses the importance of classifying the single behaviour according to spatial and temporal parameters that define the robot's characteristics as embedded in an environment. Two ways are utilised. The first belongs to the robot (the development of its program). The second is external to the system (it is conditioned by the environmental niche the robot lives in and by the sets of sign-stimuli the robot is planned to react to), to whom the robot tends to adapt itself. So we have to design the sets of sign-stimuli our solving a maze robot has to respond to. To build up behaviour based robot it is necessary to consider not only the characteristics of behaviour patterns subject to modification, but also the order in which they are performed. Moreover, the order in which behaviour patterns occur is essential for the study of the mechanism responsible for the temporal/parallel organisation of behaviour.

From the programming point of view, the behaviour space of the robot is defined by the locations the robot can reach (or by the set of actions it has to exhibit in the physical space) and by the transition between those locations. Even if the robot can attain a nearly infinite number of states, it is better to design a useful behaviour space in which the programmer limits him/herself to a small number of states. This limited view of the situation is sufficient so long as the robot remains within the designated environmental and tasks spaces. The behaviour space may be represented as a graph, a state transition network, and a set of propositions. The fact that behaviour patterns tend to occur clustered in time, the clusters constituting functionality related groups. Behaviour patterns doubtless do form clusters (as evidence in itself for underlying hierarchical organisation) in that each act is likely to be followed by another member of the same cluster.

A task is solved by a sequence of movements in the behaviour space of the robot. The behaviour space implies that a constrained number of behaviour sequences can lead from the initial state to the end state. A single sequence of operations, or state transitions, in the behaviour space is called procedure. A programme is an ordered sequence of operations. A procedure is constrained by the dependencies between successive operations. It is not possible, for example, to turn to left or right if the robot doesn't arrive to the corner (in the functionality we have choose as example: solving a maze robot). In addition to a description of the states, and in addition to the operations and state variables modified during the performance of a procedure, the procedure may also be characterised by the costs occurring during the execution of the same procedure.

Programming the robot's control

There are two approaches in programming behaviour and control in robotics. One possibility is to pre-programme the robots with procedures for all tasks the designer and the functionality foresees. The second possibility is to provide the robot with the initial task and the task the robot has to solve and let the behaviour autonomously arises by evolution and by learning processes. This second approach is called evolutive robotics. To pre-programme the robot's behaviour means that all

potential behaviours (or all conceivable procedures) should be well known. Pre-programmed behaviour has some difficulties. All tasks must be well known ad anticipated by the designer, who must explicitly design the procedures for all the tasks that the robot will be required to solve. The programme should be prohibitively large. When the robot encounters unexpected states, it will behave inappropriately or stop because it doesn't know how to go on.

In the evolutive robotics approach, not all the procedures have to be explicitly programmed. The robot has an improved ability to deal with unexpected or unknown states. The behaviour repertoire of the robot (or the programme) is smaller and it is possible to modify it faster, without changing the structure of the programme.

As we have already said, Khepera, a simple locomotion robot, is a modular platform, equipped with six sensors and two motors. It could be furnished with a gripper, a camera and some communication tools. MathLab, Labview or C can program control in Khepera. But these languages are too difficult for young people. Furthermore, in Khepera it is also possible to use genetic programming algorithms and neural nets.

LegoMindstorms uses a more simplified kit of programming activity but is rather limited in functionality. Hence, it can only be used for programming only very simple task in robots. To unleash the power of robot behaviours, Dave Baum built up NQC, especially designed for LEGO robots. NQC, which stands for Not Quite C is a very simple programming language and is well fitted for non-programming users. RCX is an application that lets to control Lego robot directly and has some specific tools which help user in programming with NQC. These applications also (used in university course in robotics) are very difficult for children.

In NQC it is possible to have multiple tasks. It is also possible to put piece of code in so called subroutines that we can use at different places in the program. In NQC, each task has a name. One task must have the name *main* and this task will be executed. The other tasks will be executed when a running task tells them to be executed using a start command. From this moment on, both tasks are running simultaneously (while the first task continues to run). A running task can also stop another running task by using the stop command. Later this task can be restarted again, but it will start from the beginning, not from the place where it was stopped.

If we use a touch sensor in a robot, we want to make a program in which the robot drives around in squares. But when it hits an obstacle, it should to react to it. It is difficult to do this in one task, because the robot must carry out two different behaviours at the same moment: drive around (that is switching on and off motors at the right moments) and watch for sensors. So it is better to have two separate tasks: one task which drives the square and the other which reacts to sensors. In the *main task*, it is necessary to set sensors and to recall the task move-square that moves the robot forever in squares. Task check-sensor checks whether the touch sensor is pushed. If so, it takes the following action. First, it stops move-square. Check-sensor takes control over the motion of the robot. Next it moves the robot back a bit and makes it turns. Then move-square can start again. It is very important to remember that tasks that it is possible to start are running at the same moment. This can lead to unexpected results.

Running the robot and performance evaluation

To evaluate robot's behaviour it is necessary to establish which behaviour is best fitted in that particular environment, to solve the task the robot has to carry on.

The first method could be to compare the behaviour of other robots with the same function, selecting the best behaviour as the result of selective pressure (since it solves some implementation problems and clarify the programming process).

The second method involves experimental demonstration that a particular feature of the robot's behaviour is likely to have consequences that affect the *survival* of the robot (since it makes the robot carries on the task it has been designed for). The third method involves direct comparison of

percentage of success in accomplishing the task, for robots differing in some behavioural characteristic. It is possible to evaluate the robot's performance in a simulated world or in a physical world.

Some educational remarks

Robotics is spreading out. It's possible to argue that in few years many people will become literate in this topic. This raising of interest invests first school people. Thanks to the pioneer works of Papert first, Resnick and Martin after, a great family of educational kits, devoted to robotics have been developed.

LEGOMINDSTORMS, KHEPERA, AIBO and others (now available at low cost) give children the possibility to play with mechanical robots, to programme them and to make them to behave in a real world situation. But, while the hardware components is very easy to use and to compose (even if it is hard to make it to work), to design complex behaviour the robot has to exhibit in the world it is difficult. As we have seen, there are many concepts to achieve and a lot of designing and programming activity to carry out.

What would be interesting is to teach children how to programme control in robotics.

Learning to programme robot's control means children acquire some specific competence. This could be very important in the educational domain for the following reasons. First, children learn to think to a complex problem, to solve it, it is necessary to decompose it in small pieces. Every piece has to be processed separately. So, they have to plan functionality, thinking about hardware design, sensors and motors activity, behaviour, and how the robot's performance could be in a real situation. Second, they learn to co-ordinate all these separate pieces by programming activity, setting sensors and putting in relation the motor activity, writing control procedures, evaluating the robot's performance in the world. Third, the learning process can be seen as an open ended problem: the student achieve a real feel for the discrepancies between the results predicted at the design stage and those actually produced by their robots, learning to reduce this discrepancy during design, programming and construction. In this way, a circular relationship between theory and practice is acquired.

Many experiments have to be done in educational robotics to ascertain what is the better methodology to let children, but also undergraduate and graduate students, to acquire this complex body of knowledge.

References

Arkin R.C. (1998), Behavior-Based Robotics, *MIT Press*, Cambridge, MA

Braitenberg,V. (1984) Vehicles – Experiments in Synthetic Psycology. *MIT Press*, Cambridge, MA.

Brooks, R.A. (1986), *A robust layered control system for a mobile robot*, IEEE Journal of Robotics and Automation, RA-2, April, 14-23

Husbands, P., Meyer, J (1998), *Evolutionary Robotics*, Springer-Verlag, Berlin

Lund, H.H., Miglino, O., Pagliarini, L., Billard, A., Ijspeert, A. (1998), *Evolutionary Robotics – A Children's Game*, Proc. of IEEE 5th International Conference on Evolutionary Computation, IEEE Press, NJ.

Martin, F. (1996), *Ideal and Real Systems: A study of notions*. In Kafai Y. and Resnick M. (eds.) Undergraduates Who Design Robots. In Constructionism in Practice: Designing, Thinking and Learning in a Digital World. Mahwah, NJ: Lawrence Erlbaum

Repenning, A. (1993) Agentsheets: A Tool for Building Domain-Oriented Dynamics, Visual Environments. University of Colorado at Boulder. Ph.d Dissertation, Department of Computer Science.

Resnick, M (1989), *Lego, Logo, and Life*. In Langton C. (eds.) Artificial Life, Addison-Wesley, Boston, MA

Resnick, M. (1994), *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*, MIT Press, Cambridge, MA