Emergent functionality in robotic agents through on-line evolution.

Luc Steels
Artificial Intelligence Laboratory
Vrije Universiteit Brussel
Pleinlaan 2, B-1050 Brussels, Belgium
E-mail: steels@arti.vub.ac.be

Abstract
The paper proposes an architecture for the on-line evolution of new behavioral competences on a robotic agent. Some experimental results for evolving a set of primitive behaviors are presented.

Introduction
A central question in ALife research is how new complexity and new functionality may emerge [Stee1994]. Selectionism and self-organisation have so far been put forward as the key explanatory principles [Langto1989]. These principles have been applied at many level of biological systems, from the chemical reactions that explain the origin of life [Kaufma1993] to the interaction between individuals in societies [Denenbou1993]. This paper explores in how far selectionism and self-organisation may lead to the build up of behavioral complexity in animals. In the tradition of Alife research, this exploration takes place by building artificial systems, i.e. robotic agents.

There has already been a large amount of work attempting to use selectionist techniques for evolving behavioral competences. Holland for example has developed classifier systems and used genetic algorithms to evolve them [Hollan1975]. Koza has shown how the reactive finite state machines proposed by Brooks [Brooks1991] can be evolved using genetic programming techniques [Koza1991]. The Sussex group [Cliffs.et.al.1993] has proposed an experimental environment for using genetic techniques based on real sensory data. Selectionist mechanisms have also been proposed by neurobiologists, notably Edelman, as an alternative explanation to the inductive or associative mechanisms dominating the literature on neural networks [Edelman1987].

All this work is extremely valuable and has provided inspiration and techniques for the work reported here. Our own approach differs however in the following respects:

- **Subsymbolic vs symbolic.** The classifier systems of Holland are in the tradition of symbolic AI. They assume that the world needs to be categorised in terms of predicates like 'object in left center field of vision' and that behavior must be decomposed into actions like 'cause eyes to look left'. We want to stay at the sub- or presymbolic level in which the dynamics of the world is directly coupled to an internal dynamics without prior segmentation or categorisation. This implies that we must evolve dynamical systems instead of symbolic computation rules.

- **Cooperation vs Subsumption.** Many architectures for robotic agents, such as the subsumption architecture [Brooks1991], allow different behavioral modules to inhibit one another. We want instead to create a 'level playing field' in which one behavioral module cannot inhibit another one. Different modules must cooperate or compete with each other in order to achieve a coherent behavior. Genuine conflicts are handled by motivational systems, as studied extensively in ethology [McFarlan1992]. This 'level playing field' appears crucial to apply selectionism properly because otherwise some behavior systems get an unfair advantage.

- **On-line vs offline.** Most of the genetic experiments so far have been performed in a simulation setting. Instead, we want to work on real robots in view of the large discrepancies between simulations and artificial systems and the complexity of building realistic simulators [Brooks1991]. Moreover we want the agent to remain viable as a task-achieving agent instead of running the genetic algorithm off-line and then transplanting a solution for testing on the robot, as in [Cliffs.et.al.1993]. Running genetic mechanisms online puts heavy constraints on the selectionist mechanisms that can be used but it brings the experimental conditions closer to real autonomous robotic agents.

- **Open vs. closed functionality.** Most genetic experiments so far assume a fixed desired functionality so that the fitness function can be provided as input by the designer. We want instead an evolving open functionality with no *a priori* fitness function. To achieve this, we have created an experimental setup in which there are a large number of real world constraints acting on the agents [Stee1994b]. The constraints range from internal constraints (limited decision time, limited memory) to external constraints (pressure to obtain enough energy in time, pressure to make other robots survive, pressure to avoid bodily damage). This
way we obtain a situation in which there is natural selection as opposed to artificial selection, much in the same way the computational constraints in the Tierra system [Ray1992] constitute a natural selection environment for evolving copying programs.

This paper reports on progress towards the ambitious objectives stated above. Our explorations are far from finished but we have achieved the following results: (1) a selectionist architecture has been designed, (2) it has been shown experimentally that a primary repertoire of behavioral competences can be evolved, (3) an implementation has been constructed on real robots and experimental results with this implementation have been obtained. Much further work needs to be done, particularly towards the evolution of secondary repertoires and towards the mathematical investigation of the genetic mechanisms that have been proven to be successful so far in experiments. Moreover many variants of the architecture have not been explored yet.

The rest of the paper is in four parts. First we discuss the target of evolution, which are dynamical systems running on a cooperative dynamics architecture. Then the proposed selectionist mechanism, known as the selector, is described. Third some experimental results are reported. The paper concludes with some indications how we are tackling the evolution of a secondary repertoire.

The PDL Robot Architecture

Before we concentrate our efforts on evolving behavioral competences we must decide in which form this competence will be implemented on the robot. At the same time, the choice of robot architecture is influenced by whether it supports selectionist mechanisms. We have adopted a cooperative dynamics architecture which is implemented with an associated language called PDL [Steels1993]. This architecture assumes that the sensors deliver a continuous (discretised) stream of data and that there is a continuous (discretised) stream of parameters flowing to the effectors (e.g. the speed or acceleration of the motors). The time-varying data are made available as the values of quantities (sensory quantities and action parameters) which are stored in an array. There is also a set of internal quantities which are used to keep track of motivational states and world states, and to support internal bookkeeping. The architecture also features processes. A process establishes a dynamical relation between a set of quantities but is not in full control of the exact value. For example, there is no assignment. If a process wants a particular quantity to have a certain value, then it must drive the dynamics such that this quantity progressively reaches that value. A process is mathematically defined as a differential equation. It is discretised to a difference equation and then implemented by simulated parallelism similar to the way cellular automata approximate continuous dynamical systems.

The third basic unit in the architecture is called a behavior system. A behavior is a regularity in the interaction dynamics between an agent and the environment (for example a certain distance is maintained from the wall) [Smithers1992]. A behavior system is the set of internal processes that are active when the regularity is observed. Each behavior system can be said to establish a particular condition. Examples of conditions are: approaching zero translation speed (halting behavior), approaching minimum left photo and right photo sensing (orientating towards a light source), approaching minimum infrared reflection (turning away from an obstacle), etc. Often the robot can monitor the satisfaction of a condition if it can sense the behavioral regularity.

The PDL architecture is guided by two principles which are both adapted to create a 'level playing field':

- All behavior systems (or more precisely all the processes of all the behavior systems) are active at the same time, i.e. there are no subsumption relations between behavior systems. This means concretely that, for example, the forward movement behavior system and the backward movement behavior system are both operational. The ultimate direction taken depends on which behavior system influences the overall behavior in the strongest way.

- The influences of the different behavior systems are summed. Although a process can (and usually does) internally perform a non-linear mapping, the additive combination guarantees that no process is viewed as more important than another one.

Given a set of behavior systems, each consisting of a set of processes, then the overall execution algorithm is defined by the following procedure:

1. All quantities are frozen
2. All processes are executed and their influences combined.
3. All quantities are changed based on the influences.
4. The action parameter quantities are sent to the effectors.
5. The latest sensory quantities are read in.

Then the procedure starts again from 1. This cycle takes place on our current robots with PC-level processors at a speed of at least 40 cycles per second, ensuring a very reactive behavior (Figure 1.).

To regulate the interaction between different behavior systems motivations and behavioral tendencies have been introduced. Both terms are used in their ethological sense [McFarland1992], although we introduce them to constrain the internal architecture of the robot not to explain empirically observed behavior as is usually done in ethology. A motivation is a quantity which reflects the cost of reaching a particular state. A typical example is a quantity EnergyNeed which is inversely proportional to the level of the battery (Figure 2.): the less energy there is, the higher the quantity EnergyNeed. Certain behavior systems are influenced by motivational quantities. For example, because running out of energy is fatal to the robot, the behavior of moving towards the charging station should be stronger as energy becomes
Figure 1: Typical example of robotic agents we use in our experiments. The body has been built with LegoTechnics™. The main processor is a pocket PC computer inserted in the robot body. We use a custom-made sensory-motor board to buffer and pre-process the sensors and regulate the flow of action parameters to the actuators.

EnergyNeed

Battery level

Figure 2: The graph plots a typical dependency of a motivational quantity on an external quantity, in this case EnergyNeed (y-axis) versus battery level (x-axis). As a lethal region is approached, EnergyNeed goes up exponentially.

Motivations and behavioral tendencies are well established notions in ethology [McFarland1992] and have been suggested as applicable to robotics [McFarlandBoesser1994]. We incorporate them here as explicit internal quantities. There are processes which determine the level of motivations as well as the levels of behavioral tendencies. The influence imposed by a behavior system is always a (multiplicative) function of the tendency associated with the behavior.

The behavior of the robot is constrained by a set of built-in motivations and processes computed based on internal and external sensing. These built-in motivations set the physical boundaries of the robot (e.g., avoid overcharging the batteries, avoid running out of energy) and are also a way to (indirectly) specify a mission to the robot. For example, if the robot must perform measurements, then its motivational strength to collect more measurements depends on the amount of measurements it has been able to gather.

The selectionist mechanism

The task of the selectionist mechanism that we seek can now be defined more clearly: While remaining viable in the environment, the robot must evolve an appropriate set of behavior systems which will ultimately guarantee its survival. This means in the first place maintaining adequate energy levels (assuming recharging opportunities). We make a distinction between the acquisition of a primary repertoire of basic functionalities such as forward movement, turning away from obstacles, orientating towards a light source, etc. and a secondary repertoire which combines primary (or secondary) behaviors by manipulating their behavioral tendencies. So far we have only reached solid results for deriving the primary repertoire, which is however often viewed as the most difficult part.

The basic idea of our approach centers around conditions. As stated earlier, conditions are dynamical properties of the environment or the internal state of the agent that a particular behavior system attempts to satisfy. Conditions are related to the notion of goals in classical control theory, but not to the notion of goal in symbolic AI, where goals are predicates on symbolically described world states.

A typical example of a condition is: WheelCounter should approach zero. This is satisfied in the case of a halting behavior. To develop the primary repertoire, the robot generates conditions by combining randomly operators, constants, and quantities from a limited repertoire. For example, the condition "WheelCounter should approach the maximum value (which is 255)" could be generated. For each condition, a quantity corresponding to a behavioral-tendency is introduced. The higher this quantity, the faster we expect the condition to be satisfied. At the moment, the robot then concentrates on evolving a behavior system for the first condition generated, before moving on to the next condition. Evolving a behavior system means to derive a set of processes that are capable together to cause the condition to be satisfied.
There are many variants of selectionist mechanisms, but they typically involve the following steps [Koza1991]:

1. An initial population is generated.
2. The elements of the population are assigned a fitness value.
3. The constellation of the population is changed using copy and mutate operations whose probability of occurrence is a function of the fitness of the elements engaged in the operations.

We explore a particular variant of this general class of mechanisms which is adapted to the task at hand. We call this variant the selectionist.

The initial population

In the present case, the elements of a population are processes, such as 'increase the translation', or 'decrease the rotation'. The impact of these processes is always a (multiplicative) function of the behavioral tendency associated with the behavior system. The initial population may contain multiple copies of a process (in which case the effect is enforced) or completely opposing processes (for example some pushing the speed of the translation motors up and some others pushing the speed down).

Assigning a fitness value

To determine the fitness, all processes in the population are allowed to run for a particular length of time which is called the selection window \(W\). Typical lengths in our experiments are at least 10 cycles, i.e. 0.5 to 1 seconds. This is based on the experience that the quantities fluctuate too much for a step by step evaluation due to the tight interaction with the environment. After the test period the fitness is determined by testing how far the conditions are satisfied in relation to the level of their corresponding behavioral tendency. The test yields a vector indicating in how far the actual value deviates from the desired value (positive or negative). The test itself also takes place in a PDL process and is recorded in a quantity called satisfaction which gets higher when the current state is further away from being satisfied. The average satisfaction during the time window \(W\) is equal to \(S(T + W) = \sum_{j=1, W} s(t + j)/W\).

As processes are run, the system keeps track of the role of each process for the overall behavior. A process \(p_i\) has an impact \(v_i\) on a quantity \(q\) at time \(t\). For all processes, \(\Delta^+ = \sum v_i\) if \(v_i > 0\), and \(\Delta^- = \sum v_i\) if \(v_i < 0\). The fraction \(d_i\) of a process \(p_i\) at time \(t\) is equal to \(d_i = v_i/\Delta^+\) if \(v_i > 0\) or \(d_i = v_i/\Delta^-\) if \(v_i < 0\). The majority direction at time \(t\) is \(M = +1\) if \(\Delta^+ > \Delta^-\), \(M = -1\) otherwise. The role of a process \(p_i\) at time \(t\) is then defined as \(r_i = Md_i\). The average role of a process \(p_i\) in the time period \([t, t + W]\) is equal to \(a_i = \sum r_i/W\).

Changing the population

There are two ways in which the population changes: by copying which ensures that a process stays in the population or by mutation/creation which ensures that new processes enter. The latter is particularly needed if a needed candidate died off prematurely.

A process has a certain probability of being copied and/or surviving into the next generation (at time \(t + W\)). This probability is proportional to the change in average satisfaction between \(t\) and \(t + W\):

\[
\epsilon(t + W) = \begin{cases} 
1 & \text{if } S(t) < S(t + W) \\
-1 & \text{if } S(t) > S(t + W) \\
\text{random}, -1 & \text{if } S(t) = S(t + W)
\end{cases}
\]

The random choice between -1 and 1 helps to bring the system out of local minima. The probability of copying or deleting is such that the following constraints are satisfied:

\[
\gamma P_m(t + W) = \gamma a_i(t + W) \epsilon(t + W) + \gamma P_m(t) \quad (2)
\]

\[
\gamma P_m(t + W) > 0 \quad (3)
\]

where \(P_m\), \(\gamma\) is a constant which is in our current experiments set equal to 10.

The mutation/creation rate is influenced by the amount of processes in the population. This amount is kept constant. In our current experiments mutation is not yet related to fitness although we plan to do so in the future.

All the necessary data to execute the selectionist procedures are collected at every time step and in a parallel fashion (e.g. every process maintains itself what its role was). Moreover the overhead in temporal and spatial complexity for performing selection is small enough to incorporate selectionism as part of the robot's normal operation in the environment.

Experiments

The following examples illustrate the behavior of the selectionist and provide at the same time experimental evidence for the viability of the approach. We concentrate first on evolving behavior systems for moving forward and backward and for halting using only one sensory quantity (WheelCounter) and one action parameter (Translation speed of the motors). We first look at forward movement with tendency Tendency1. The quantities involved are WheelCounter, translation speed of the left and right motors, Tendency1, and satisfaction1. The latter quantities correspond respectively to the behavioral tendency for engaging in forward behavior and the satisfaction with which this behavior is actually observed.

The initial population is as follows. The number of copies of a process in the population is indicated in parentheses behind the name.

**process-0 (16):**
Translation <= Tendency1 *
approach(-255, WheelCounter)/1000

**process-1 (14):**
Translation <= Tendency1 *
approach(0, translation)/1000
Here are some steps in the execution of the algorithm. For each step the average satisfaction at time $t$ and $t+W$ are displayed as well as the direction $e$. For each process, the sum of the influences $r_i$ during the time period $[t, t + W]$ and the number of processes in the population after copying and elimination are indicated.

**Step 1. Satisf**: 25.800 $\Rightarrow$ 26.000; $e$ = -1

**process-0**: 1.94: 14
**process-1**: -0.03: 14
**process-2**: -1.79: 15
**process-3**: 2.06: 14
**process-5**: -2.16: 19
**process-7**: -0.02: 15

**Step 2. Satisf**: 26.000 $\Rightarrow$ 23.800; $e$ = 1

**process-0**: -0.06: 13
**process-1**: 0.07: 14
**process-2**: -0.03: 14
**process-3**: -0.10: 13
**process-5**: 0.01: 19
**process-7**: 0.12: 15

**Step 3. Satisf**: 23.800 $\Rightarrow$ 23.600; $e$ = 1

**process-0**: -0.01: 12
**process-1**: 0.01: 14
**process-2**: -0.00: 13
**process-3**: -0.02: 12
**process-5**: 0.00: 19
**process-7**: 0.02: 15

**Step 4. Satisf**: 23.600 $\Rightarrow$ 23.000; $e$ = 1

**process-0**: 1.79: 13
**process-1**: 0.20: 14
**process-2**: -1.63: 11
**process-3**: 1.78: 13
**process-5**: -2.37: 16
**process-7**: 0.22: 15

We observe during this short sequence a gradual improvement in the fitness of the population (because satisfaction evolves to zero). Usually there is a period of heavy competition until the winning processes finally dominate. This happens quite rapidly once these processes gain a slight upperhand, similar to phase transitions. The same data as above is displayed graphically but for a complete evolutionary sequence in Figure 4. We see that the satisfaction (directly related to fitness) evolves towards zero and that two processes dominate. These processes cause the wheel counter to approach the maximum value (255). The final population has the following constellation:

```
process-2 (15):
Translation <= Tendency1 * 
approach(255,WheelCounter)/1000
```

```
process-3 (15):
Translation <= Tendency1 * 
approach(-255,translation)/1000
```

```
process-5 (15):
Translation <= Tendency1 *
approach(255,WheelCounter)/1000
```

```
process-7 (15):
Translation <= Tendency1 *
approach(0,WheelCounter)/1000
```

**Figure 4**: Evolution of the population and of the fitness function. After a competition, the ‘winning’ processes (process 2 and 5) take over quickly as in a phase transition.

**Figure 5**: Evolution of the WheelCounter and translation quantity during the evolution of a forward movement behavior system.

```
process-0: 0
process-1: 0
process-2: 14
process-3: 0
process-5: 28
process-7: 0
```

The evolution of the WheelCounter (which is a sensory quantity) and the translational quantity (which is an action parameter) during the same time period are given in Figure 5. They indeed both reach their maximum values at the end of the evolutionary sequence.

Another typical run, now for acquiring the halting behavior, is shown in Figure 6. The evolution of WheelCounter and Translation speed as given in Figure 7, reflects the oscillation before the robot comes to a stand still. The processes at the end are:

```
process-0 (0):
translation <= tendency-2 *
approach (255, WheelCounter)/1000
```

```
process-1 (42):
translation <= tendency-2 *
approach (0, WheelCounter)/1000
```

```
process-4 (0):
translation <= tendency-2 *
approach (255, translation)/1000
```
Figure 6: Competition between different processes attempting to establish a halting behavior.

Figure 7: Evolution of the WheelCounter and translation quantities as the robot is evolving a halting behavior.

Figure 8: Evolution of the WheelCounter and translation quantities as the robot is evolving a halting behavior. We observe typically oscillations before the system settles in a solution.

process-5 (0):
  translation <= tendency-2 *
  approach (-255, translation)/1000
process-6 (6):
  translation <= tendency-2 *
  approach (0, translation)/1000

Process-1, which moves Translation towards a state where the WheelCounter is zero dominates, and process-6, which indirectly contributes, both contain a non-empty population. Often we note a quasi-periodic oscillation before the dynamics settles on the winning processes. This is illustrated in the results of another experiment displayed in Figure 8.

In yet another evolutionary run the following solution was obtained for the halting problem:

process-1 (11):
  translation <= tendency-1 *
  approach (-255, translation)/1000
process-2 (11):
  translation <= tendency-1 *
  approach (255, WheelCounter)/1000
process-0 (10):
  translation <= tendency-1 *
  approach (0, translation)/1000
process-12 (7):
  translation <= tendency-1 *

approach(0, WheelCounter)/1000
process-5 (6):
  translation <= tendency-1 *
  approach(-255, WheelCounter)/1000
process-6 (6):
  translation <= tendency-1 *
  approach(255, translation)/1000

Here we see a dynamic equilibrium between processes pushing towards the minimum (process-1 and process-5) and those pushing towards the maximum (process-2 and process-6) so that the net effect is to cause halting behavior.

We have consistently observed rapid success in generating such primitive behavioral competences where only few quantities are involved. In about 20 percent of the cases, ‘good’ processes are eliminated before a solution is found and then the system must rely on mutations to regenerate lost processes. But also in those cases adequate solutions are found.

Conclusion

The paper has discussed a selectionist mechanism, called the selectron, which has been designed to evolve new behavioral competences for a robotic agent. The mechanism satisfies the following constraints: (1) it can be used on-line, (2) it can be used on a real robot as it is operating in a dynamically changing real world environment, (3) there is no fixed fitness function imposed from the outside by the designer.

The selectron is based on a cooperative dynamics architecture which assumes that behavioral competence is decomposed into behavior systems. Each behavior system consists of a set of processes that establish a dynamical relationship between a set of sensory and action parameter quantities and a set of internal quantities. A behavior system, as it operates, satisfies a particular dynamic condition, for example approaching a zero wheel counter (halting behavior). To evolve a new behavior system, an initial population of processes is generated at random. These processes operate in parallel and determine the behavior of the robot in interaction with the
environment. At regular time intervals, the evolution of
the satisfaction of the condition is calculated and this
is used to vary the population using fitness proportion-
ate copy and elimination operations on processes. Initial
experimental results demonstrate the viability of the
approach.

There are many variations of the selectron algorithm
which we have not investigated yet and experiments are
in progress for deriving a secondary repertoire. Also a
mathematical investigation remains future work. Much
further work needs to be done to understand how the
power of evolution can be unleashed on the development
of behavior. But at the same time first results obtained
so far are highly encouraging.

Acknowledgements
A lot of the work discussed in this paper is based
on other work of members of the autonomous agents
group. In particular, contributions by Filip Vertommen
on the PDL architecture implementation on different
robot platforms, Danny Vereerbrughgen for construc-
tion of the hardware, and Peter Stuer for implementa-
tion of the overall self-sufficiency experiments has proven
highly valuable. Christophe Wauters has performed the
experimentation on physical robots. Many discussions
with David McFarland have helped to shape my ideas
on useful mechanisms in ethology. Also thanks to Wal-
ter Van de Velde for discussions and comments, and for
his help in preparing the final copy of this paper. The
writing of this paper was partly sponsored by the IUAP
Action of the Belgian Science Ministry. I am indebted
to Rodney Brooks for inviting me to the conference and
thus stimulating the writing of this paper.

References

plete Creature Architectures. In: Meyer, J-A.,
and S.W. Wilson (1991) From Animals to Ani-
mats. Proceedings of the First International
Conference on Simulation of Adaptive Behavior.
434-443.

[Cliff.et.al.1993] Cliff, D., P. Husbands, and I. Har-
vey (1993) Evolving Visually Guided Robots. In:
Meyer, J-A., H.L. Roitblatt, and S.W. Wil-
sion (1993) From Animals to Animats2. Pro-
cedings of the Second International Confer-
ence on Simulation of Adaptive Behavior. MIT
Press/Bradford Books, Cambridge Ma. p. 374-
383.

[Deneubourg1993] Deneubourg, J-L, et.al. (1993) Self-
organisation and life: from simple rules to global
complexity. Proceedings of the Second European
Conference on Artificial Life. ULB, Brussels.

The Theory of Neuronal Group Selection. Basic

ural and Artificial Systems. The University

order: self organization and selection in evolu-

Following Robotic Behavior sing the Genetic
Programming Paradigm. In: Varela, F.J. and
P. Bourgine (eds.) (1992) Toward a Practice of
Autonomous Systems. Proceedings of the First
European Conference on Artificial Life. MIT
Press/Bradford Books, Cambridge Ma. p. 110-
119.

Santa Fe Institute Studies in the Sciences of
Complexity. Proc. Vol VI. Addison-Wesley,
Reading Ma.

based Robots. International Studies in the Phi-

(1994) Intelligent Behavior in Animals and
Ma.

sis of Life. In: Langton, C.G., C. Taylor, J.D.
Farmer, and S. Rasmussen (1992) Artificial Life
II. Proceedings of the Workshop on Artificial
Life Held February, 1990 in Santa Fe, New Mex-
ico, p. 325-371.

Materialism Seriously: A Methodology for Au-
tonomous Systems Research. In Varela, F.J. and
P. Bourgine (eds.) (1992) Toward a Practice of
Autonomous Systems. Proceedings of the First
European Conference on Artificial Life. MIT

[Steels1993] Steels, L. (1993) Building Agents with Au-
tonomous Behavior Systems. In: Steels, L. and
to ‘artificial intelligence’. Building situated em-
body agents. Lawrence Erlbaum Associates,
New Haven.

Artificial Intelligence. Journal of Artificial Life.
MIT Press. 1,1/2, p. 89-125.

[Steels1994b] Steels, L. (1994b) A case study in the
behavior-oriented design of autonomous agents.
Proceedings of the Simulation of Adaptive Be-
havior Conference. Brighton. Cambridge: MIT
Press.