Emerging Behaviors in Fuzzy Evolutionary Agents

Antonio Di Nola, Antonio Gisolfi, Vincenzo Loia and Salvatore Sessa*
Dipartimento di Matematica e Informatica
Università di Salerno
84081 Baronissi (SA) Italy
email:{dinola, gisolfi, loia}@unisa.it
*Dipartimento di Costruzioni e Modelli Matematici in Architettura
Università di Napoli
email: sessa@unina.it

ABSTRACT: We extend the actor-based model of concurrent and distributed programming toward a framework in which the agents can learn emerging behaviours through a fuzzy evolutionary structure. The framework is based on the notion of *FuzzyEvoAgent*, i.e. an entity that exploiting the basic issues of actors (asynchronous message passing, concurrent computation) is submitted to evolutionary laws that reinforce the most suitable behaviours respecting the environment. We propose a formal definition as well as an implementation model of FuzzyEvoAgents that has been verified via simple simulation of Artificial Life.

KEYWORDS: Fuzzy Evolutionary Models, Agent-Based Systems, Actor Paradigm.

1. MOTIVATION

Agent technology has recently become one of the most important growing areas in information technology. Within this attractive domain, the efforts accomplished by the research and industrial community in the development of intelligent information agents have reported a remarkable progress in practical applications as well as in theoretical results [2].
The starting point of this work is the improvement of a design paradigm useful to describe medium-grained distributed and parallel system, the actor model [1]. This model is based on a essential entity, named actor, a computation object whose states are shown in Figure 1.

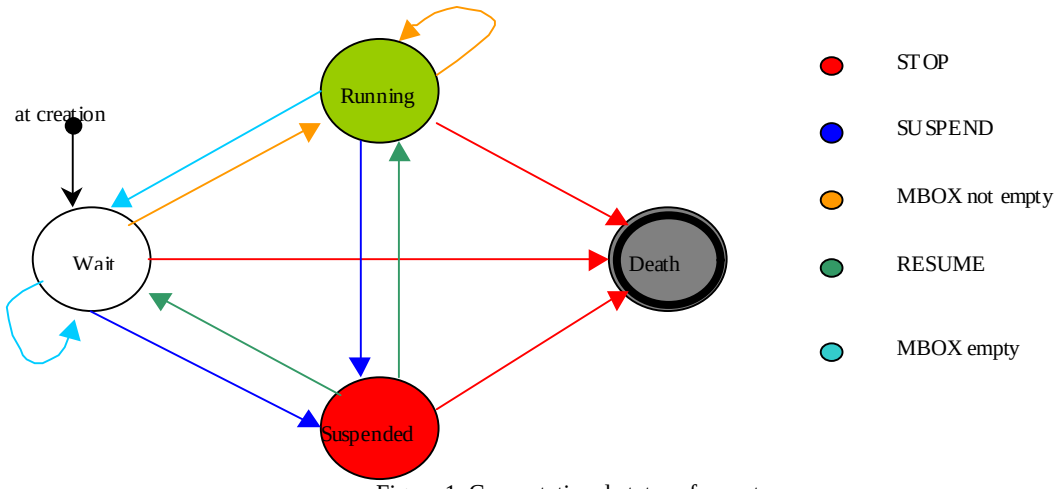


Figure 1. Computational states of an actor.

Modelling software as collections of distributed, cooperative actors is a natural evolution of object-level languages. Object-oriented programming is based on a philosophy of software development through progressive refinements, strengthening the abstraction level as a result of the use of abstract data types and information hiding. The Actors combine object-oriented and functional programming in order to make the management of concurrency easier for the user. Briefly, the actor model can be presented as:

- the universe contains computational agents, called *actors*;
- actors perform computation through asynchronous, point-to-point message passing;
- each actor is defined by its state, a mail queue to store external messages and internal behaviour;
- an actor's state is defined by its internal data, not sharable by other actors. These local variables are named *acquaintances*;
- an actor reacts to the external environment by executing its procedural skills, called *scripts*.

The actor model is at the basis of the so-called object-oriented concurrent programming which constitute one of the most important paradigms employed to realise DAI (Distributed Artificial Intelligence) level architectures.

Despite the considerable effort spent on the agent theory, a consistent definition has not been fully acknowledged. In general, agents own local duties which increase their ability in reasoning, in refusing orders, in negotiating commitments. To improve the intelligence of the behaviour, the agents must be equipped of autonomous duties that allows them to generate emerging behaviours that strengthen the performance of the agent without being forced to foresee all the (hardwired) possible behaviours.

For this reason, our main goal was to extend the notion of actor towards that of agent, i.e. to inject in the actor the competence to adapt its behaviour according to the fulfilment of its local (sub)goals. Our result was the definition and implementation of the *FuzzyEvoAgent*.

We point out the basic issues of our model.

- The model is based on the notion of “FuzzyEvoAgents”.
- FuzzyEvoAgents perform their computation in an asynchronous and independent way, exploiting concurrency and point-to-point and multicasting message passing.
- FuzzyEvoAgents are defined by their internal states, by a list of messages to process, and by a behaviour.
- FuzzyEvoAgents reacts to external stimulus, coming from other EvoAgents or from the environment.
- The scripts of an FuzzyEvoAgent are executed through an internal fuzzy control sub-system that takes into account a knowledge bases handled by a deduction engine.
- The rules represent the genotype of each FuzzyEvoAgent.
- FuzzyEvoAgent are adaptive entities thanks to an evolutionary law to which they are submitted.

2. FORMAL DEFINITIONS

As previosly stated, FuzzyEvoAgents are an extesion of the basic notion of actor. Differently from the actors, the FuzzyEvoAgents are capable to ameliorate their performances during the life of the system.
Figure 2 shows the basic components of a FuzzyEvoAgents:

1) A data section, containing the acquaintances.
2) A script section, containing the tasks that the agent may perform.
3) A Rule Base, formed by control fuzzy rules.
4) A Fuzzy Logic Control (FLC) that enables the execution of a task according to its knowledge base.
5) A genetic engine (MEET) that allows the creation of new FuzzyEvoAgents by processing the genetic information contained in the knowledge-base of the agent.

Following [7] we use (1) to define our rule-base as a set R :

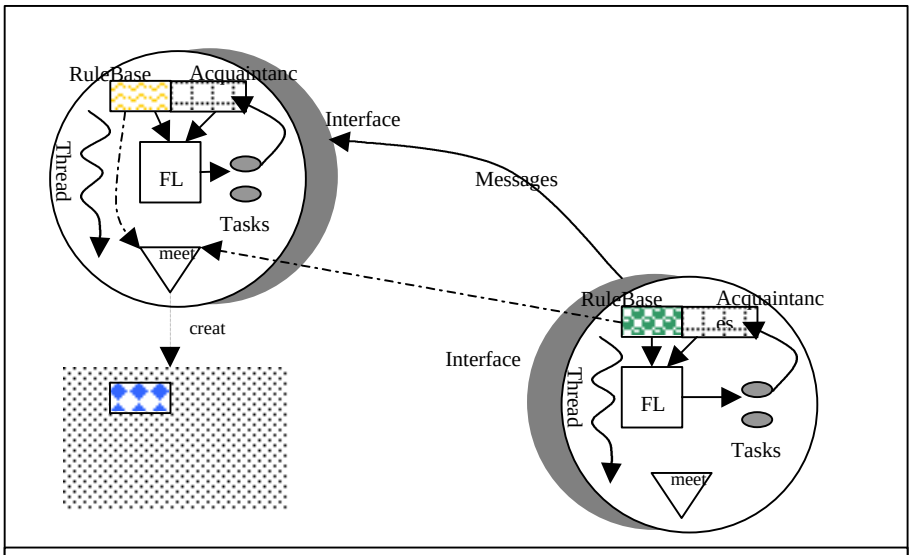


Figure 1: Fuzzy EvoAgent: they own a thread of control, an internal state, a rule base and a fuzzy control system.

$$(1)\quad R=\bigcup_{i=1}^{m} r_i=\bigcup_{i=1}^{m}(X_i \rightarrow Y_i)$$

In our model, a *fuzzy control rules* takes the form (2):

(2) ***if*** acq_1 *is* a_1, acq_2 *is* a_2, ..., acq_n *is* a_n
then $task_1$ *is* b_1, $task_2$ *is* b_2, ... $task_m$ *is* b_n (*)

where α_1, α_2, ..., α_n, are fuzzy number representing the evaluation of the data variables acq_i ; and β_1, β_2, ..., β_m, are fuzzy numbers representing the causality values of tasks with respect to the given set of data, having represented:

X_i=IF acq_1 is α_1 and…and acq_n is α_n , Y_i=$task_k$ is β_k and … and $task_t$ is β_t .

In the definition of a FuzzyEvoAgents, many fuzzy control rules may appear. The task activated is determined by a fuzzy control system [8]:

(3) $\mu_{Y(y)}=\cup_{i=1..m}$ (min[λ_i, $\mu_{Yi}(y)$])

with λ_i equal to the degree of applicability of rule r_i

(4) $\lambda_i=\cap_{j=1..n}\prod(X_{i,j} \mid I_j)$

with $\prod(X_{i,j} \mid I_j)$ representing the measure of the "matching" between the vector of variables X_i and the input vector I_i.and $\prod(X_{i,j} \mid I_j)=\cup_{xj}$ (min[$\mu_{Xi,j}(x_j)$, $\mu_I(x_j)$]).

Let *t* be the number of fuzzy rules; the i-th fuzzy rule will be referred as:

if acq_1 is α_{i1}, acq_2 is α_{i2}, . . . , acq_n is α_{in} then $task_1$ is β_{i1}, $task_2$ is β_{i2}, . . ., $task_m$ is $\beta_{im,}$ $1 \le i \le t$

hence, given the status {acq_1, acq_2, ..., acq_n}, the task to be activated will be $task_k$ if and only if:
From a logical point of view, if {acq_1, acq_2, ..., acq_n} is the status, the ith rule says that the jth task can be activated with possibility equal to $g_i = a_{i1}(acq_1) \cap a_{i2}(acq_2) \cap \ldots a_{in}(acq_n) \cap b_{ij}$. By taking the logical maximum

$p_j = \bigcup_{i=1}^{t} g_i$ we have the overall possibility value for the jth task to be executed, with respect to the all set of fuzzy rules. The task actually activated is the one for which the value π is maximum among the m possible tasks.
Fuzzy rule (2) can be encoded as a string like the following:

(2') $r_i = c_s^{\gamma_s} \ldots\ldots c_1^{\gamma_1}$

where $\{\gamma_1 \ldots \gamma_k\} = \{\alpha_1 \ldots \alpha_n\} \cup \{\beta_1 \ldots \beta_m\}$and any c_i is the subset of $\{acq_1, ..., acq_n\} \cup \{task_1,..., task_1\}$, whose elements all have the evaluation γ_i
We remark that the string (2') is a compact, and value ordered version of the rule (2), where each evaluation appears exactly once.
In this new form two fuzzy rules can be aggregated by using the following procedure [3]:
Given

$A = a_n \alpha n\ a_{n-1} \alpha_{n-1} \ldots a_1 \alpha_1$

$B = b_m \beta_m b_{m-1} \beta_{m-1} \ldots b_1 \beta_1$

then:

$A \Delta B = (a_n \alpha_n a_{n-1} \alpha_{n-1} \ldots a_1 \alpha_1) \Delta (b_m \beta_m\ b_{m-1} \beta_{m-1} \ldots b_1 \beta_1) =$

$c_{m+n-1} \gamma_{m+n-1} \ldots\ldots\ldots c_1 \gamma 1$

here, for n > m

$$c_i = \begin{cases} \bigcup_{j=1}^{i} a_{i-j+1} \cap b_j & 1 \le i < m \\ \bigcup_{j=1}^{m} a_{i-j+1} \cap b_j & m \le i < n \\ \bigcup_{j=i-n+1}^{i} a_{i-j+1} \cap b_j & n \le i < m+m \end{cases} \qquad g_i = \begin{cases} \frac{1}{i} \sum_{j=1}^{i} \frac{1}{2}(a_{+j-1} + b_j) & 1 \le \le -1 \\ \frac{1}{} \sum_{j=1}^{m} \frac{1}{2}(a_{i+j-1} + b_j) & \le \le -1 \\ \frac{1}{+-} \sum_{j=i-n+1}^{m} \frac{1}{2}(a_{i+j-1} + b_j) & \le \le +-1 \end{cases}$$

where the $\cap$ and $\cup$ are the well known operations of set-join and set-meet;
From rules

if acq_1 is a, acq_2 is b then $task_1$ is g, $task_2$ is a
if acq_1 is b, acq_2 is a then $task_1$ is b, $task_2$ is g

we get the representation (supposing $\gamma < \beta < \alpha$):

$R_1 = [acq_1, task_2]^{\alpha}[acq_2]^{\beta}[task_1]^{\gamma}$
$R_2 = [acq_2]^{\alpha}[acq_1, task_1]^{\beta}[task_2]^{\gamma}$

and aggregating R_1 and R_2 we obtain a string like:

$R_3 = [acq_2, acq_1]^{\delta 1}[task_2]^{\delta 2}[task_1]^{\delta 3}$

which represents the rule:

if acq_1 is d_1, acq_2 is d_1 then $task_1$ is d_3, $task_2$ is d_2

where δ_1, δ_2 and δ_3, are fuzzy numbers yielded by the fuzzy aggregation operators.
This aggregation is the basis for the evolution of the actors' knowledge base. We refer to R_3 as the *offspring* of the *mating* rules R_1 and R_2. Given a population of FuzzyEvoAgents Agt_1, Agt_2, ..., Agt_p, each agent keeps on with its fuzzy rules for a certain amount of time. During this period it would gain credit as the value of the function it has to maximize (agent's target or fitting function). Agent with large credit can continue their life. Agent with small credit finish to die and be replaced by new agents, whose fuzzy rules are obtained by mating the knowledge base of the well performing agents. Moreover the resulting new population of agents undergoes a mutation operator. Mutation means that agents fuzzy rules can be modified (with small probability) by changing the fuzzy numbers attached to the tasks and acquaintances.

3. TESTBED

The term "artificial life" represents a complex discipline that covers different areas, such as biology, chemical economics, philosophy; its main goal the study and the design of computational framework able to make sufficiently

abstract the principles of biological phenomena, managing them so to reproduce the dynamic behavior of evolutionary creatures, reducing the cost and complexity of alternative (biological) approaches [6].
In order to test the functionality of our model and its ability in representing ecomodels, i.e. models where the evolution changes the environment and vice versa, we sketched, with a rough approximation of the biological model, a naive representation of a simple natural scenario constituted by flies and frogs, that play the role of pray and predator.
We focus our attention on the "action selection", that is the election of the more appropriate behaviour in the range of the possible actions, taking into account the cognitive and perceptive information of the agents. The agents can accomplish actions that are "biologically" wrong (such as do not being able to eat for a long time and so to die). The applicability of our testbed is measured in terms of simulating the evolution of the system, in particular the ability to propagate to the future generations of unpredictable emerging behaviours that lead to the survival of the species [4,5].
The framework is essentially composed of two basic modules, as shown in Figure 3:

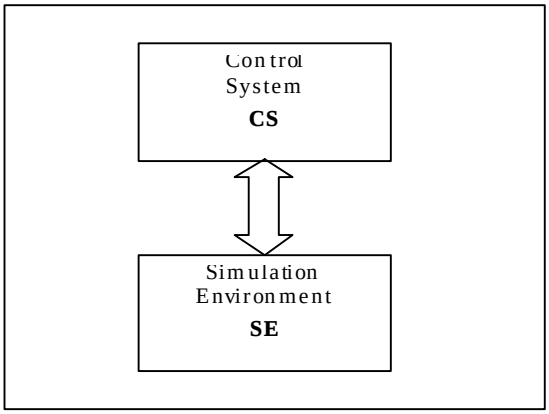
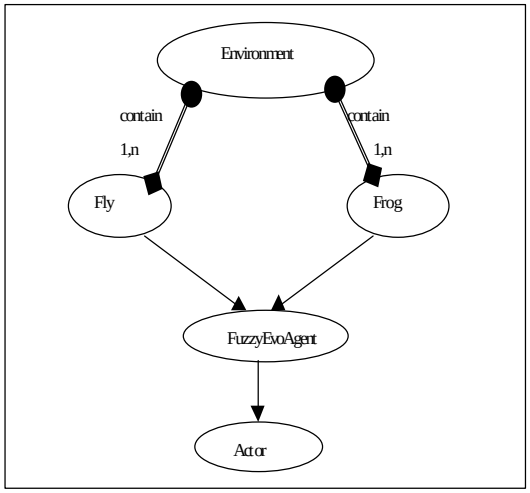


Figure 3.



- Figure 4: The Simulation Environment module in an ER representation

- *SE (Simulated Environment)* that provides the simulation environment;
- *CS (Control System)* that provides the primitives of the control flow management.
-

The simulation environment is shown in Figure 4 according with an object-oriented notation.
The entities *FuzzyEvoAgent* and *Actor* constitute the abstract agents, that is that are not strictly dependent from the specific application but they realise the basic primitives (concurrent and distributed evofuzzy reasoning) of the model.
The Control System module, given in Figure 5, is composed of two layers, the *GEC (Generational Evolutionary Controller)*, and the *PEC* (*Population Evolutionary Controller)* ,i.e. the synchronizer of the overall simulation that manages the evolutionary phase of a certain population (in our case, fly and frog).

The GEC communicates the corresponding PEC to trigger the simulation for a given duration time T.

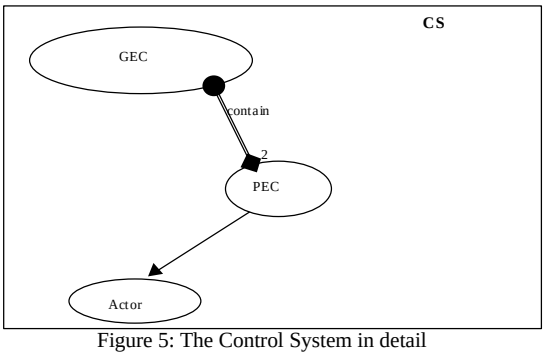


Figure 5: The Control System in detail

In order to prevent dangerous side-effects, the GEC, before triggering the evolutionary phase, must await that all the FuzzyEvoAgents have suspended their actions: this constraint is due to the necessity to assure the accomplishment of "atomic" actions

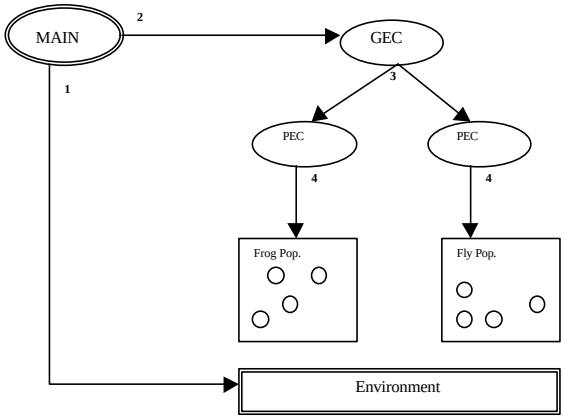


Figure 6: Representation of Step 1

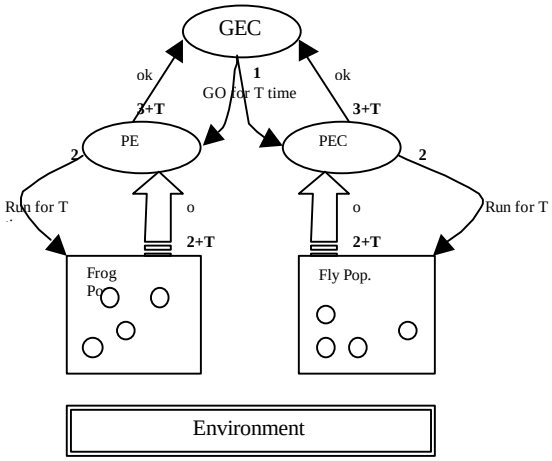


Figure 7: Representation of Step 2

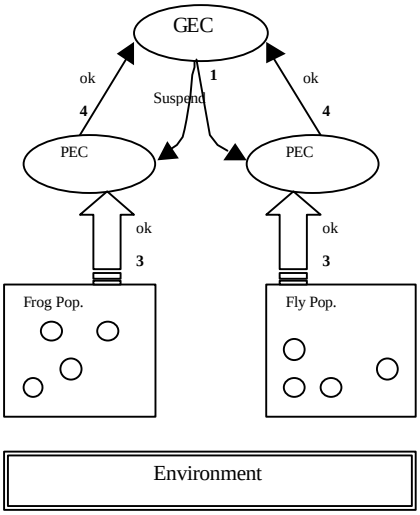


Figure 8: Representation of Step 3

Essentially, the basic steps of the simulation control consists of four steps:

- *init of CS and SE.*
- *Trigger and simulate SE for a time T*
- *Suspend the Se activity*
- *Apply evolution*
-

We describe these phases visually using circle to represent FuzzyEvoAgents, rectangles with single line to represent a set of FuzzyEvoAgents, whereas a double line means statical objects. Pointed arrows show the creation of the objects, continuos arrows stands for messages. We use labels to indicate the kind of message and to show temporal order

In the second step the Control System delivers the FuzzyEvoagents handled by the PEC that waits for the interaction between the agents and the environment for a time T (see Figure 7). The third step is characterised by the suspension done by the PEC of the FuzzyEvoagents (see Figure 8). In the last step, the PEC, stimulated from the GEC, performs concurrently the evaluation of the FuzzyEvoAgents, choosing those to reproduct according to the law of legge "survival of the fittest" (see Figure 9).

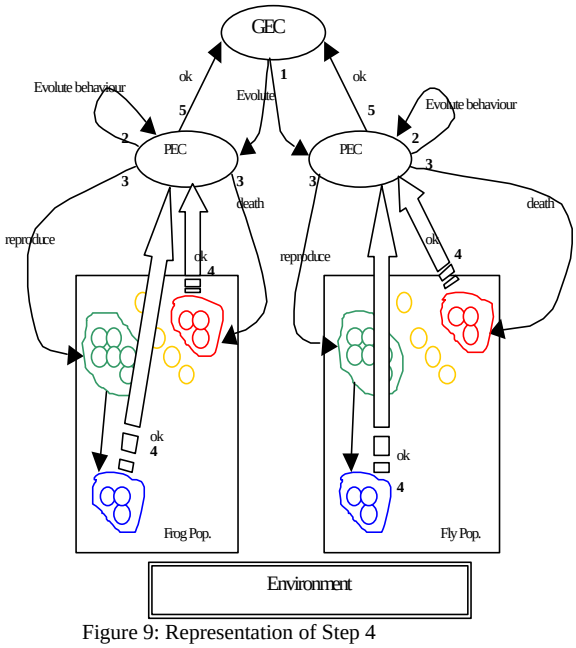


Figure 9: Representation of Step 4

To better understand the evolution phase, we must deepen the behavior of the PEC after the reception of the message "evolute behaviour" and before the sending of the "reproduce" and "death".
At the reception of the message "Evolute behaviour" the PEC executes an internal script that individuates three classes, composed respectively from the best FuzzyEvoAgents, the worst ones and the intermediate ones (that will be maintained for another generation).
We note that these classes are determined exploiting the concurrency of the platform. More in detail, considering a set $G=\{(f_1,m_1),..,(f_{n/4}, m_{n/4})\}$ with f_i and m_i respectively the father and mother selected from the PEC over the n/2 "best" agents, then the PEC sends in parallel n/4 messagges *"riproduce with m_i"*, to each f_i in G. This leads to the generation of n/4 new agents, (the blue group) that substitute the dead agents, characterized by the red group. This mechanism maintain constant the numbers of FuzzyEvoAgents, as depicted in Figure 10..

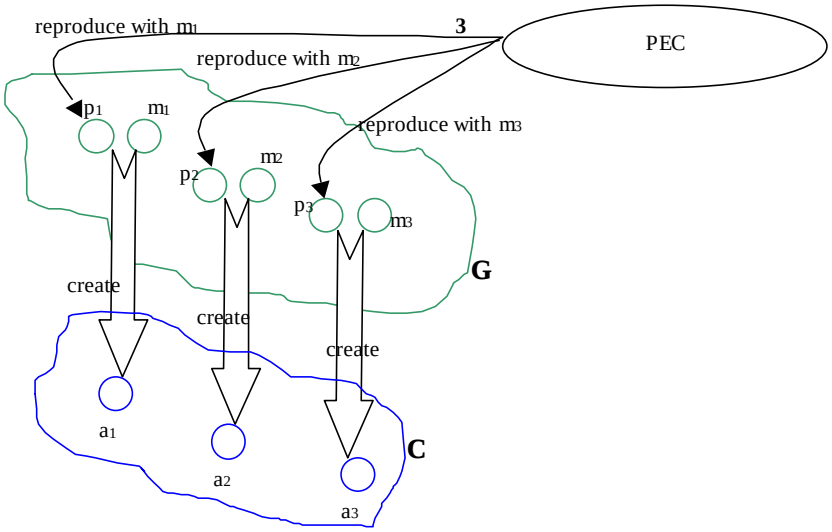


Figure 10: Parallel reproduction

4. CONCLUSIONS

We have described a general framework of distributed, concurrent computation based on the notion of FuzzyEvoagents. They are active objects capable to modify their inner capabilities according to a scheme of evolutionary learning. The platform has been implemented in Java by exploiting multi-threads-level tools.

REFERENCES

1. Agha G. Actors. *A Model of Concurrent Computation in Distributed Systems*, MIT Press, MA, 1986.
2. Genesereth M.R. and Ketchpel S.P. *Software agents: Communications of the ACM*, 37(7):48-53, 1994.
3. Gisolfi A. and Loia V. *A Complete Flexible Fuzzy-Based Approach to the Classification Problem*, Int. Journal of Approximate Reasoning, Vol. 13, 15-183, 1995.
4. Goldberg D. *Genetic Algorithms in Search, Optimization and Machine Learning.* Reading, MA:Addison-Wesley, 1989.
5. Holland J.H. *Adaptation in Natural and Artificial Systems.* Ann Arbor, Michigan: The University of Michigan Press. 1975
6. Langton C.G. Preface in C.G. Langton, C. Taylor, J.D. Farmer and S. Rasmussen, editors, *Artificial Life II*, vol. X of *SFI Studies in Sciences of Complexity*, pg. xiii-xviii, Redwood City, CA, Addison-Wesley, 1992.
7. Mandami E.H. and Assilian S. *An experiment in linguistic synthesis with a fuzzy logic controller.* Int. J. Man Machine Studies, 7(1):1-13, 1975.
8. Zadeh L.A. *A theory of approximate reasoning.* In P. Hayes, D. Nuchie, and L.I. Mikulich editors, Machine Intelligence, pg. 149-194. Halstead Press, New York, 1979.