SHIFT OPERATORS AND COMPLEX SYSTEMS

C. Bertelle([1]), G.H.E. Duchamp([2])

([1]) LIH - University of Le Havre,
([2]) LIPN - University of Paris XIII,
France

LuaïJaff

LIPN - University of Paris XIII,
France

ABSTRACT

In a previous paper [13], some of the authors dealt with the use of automata with (various type of) multiplicities for modelizing agents with rational behaviour and how one could perform genetic operations on them. In this paper, we emphasize the role played by shift operators to identify automata with similar behaviour.

1. Introduction: `TODO`

2. Shift operators

One of the most notaural operations on functions is the shifting of the argument. Using the denotation of `http://mathworld.wolfram.com/ShiftOperator.html`, one has

$$E^a(f)[x] := f(x+a) \tag{1}$$

What is little less known (but not less useful) is the use of sifting operations on the exponents of a taylor series (which amounts to the same realm, considering a series as a function of the monomials).
To each series,

$$S = \sum_{n\geq 0} \alpha_n z^n \tag{2}$$

one make correspond the series

$$\gamma_z^{\dagger}(S) = \sum_{n\geq 0} \alpha_{n+1} z^n \tag{3}$$

In this manner one can deal not only with polynomials but with rational functions. This setting has a natural extension to the multivariate case allowing to dela with rational behaviours. Rational behaviours are exactly the behaviours of automata with multiplicities and these automta are the type which is well fitted for the use of genetic algorithms over populations of agents with rational behaviours [13].

3. Kinds of Automata

Automata theory is at the edge of many areas of human skill (mathematics, physics, engeneering, ...) and roughly speaking, could be defined as *all models dealing with transition rules*.
Several domains have been developped within automata theory including: *cellular automata, boolean automata (or finites-state automata) and multiplicity automata*.
The first kind deals with with *local trasition rules* whereas the two last are defined by *global rules*. As a matter of fact, the last one (*multiplicity automata theory [**?**, 2]*) includes the second by specilizing the scalars to the Boolean semiring. Indeed varying the scalars, one can reach, with the same theory, models as diverse as *deterministic finite automata*, *finite Markov chains* and *transducers* (kinds of automta where transitions produce words and languages).
This gives the general framework for *multiplicity automata* or *automata with multiplicities*. In this model, the output data of the automata with output belong to a specific algebraic structure, a semiring [10, 14, 24]. In that way, we will be able to build effective operations on such automata, using the power of the algebraic structures of the output data and we are also able to describe this automaton by means of a matrix representation with all the power of the new linear algebra (i.e. with semirings).
Moreover, the automaton can be observed by means of the function it generates. This function will be called the *behaviour* of the automaton.

Definition 1 (Automaton with multiplicities)
An automaton with multiplicities over an alphabet A and a semiring K is the 5-uple (A, Q, I, T, F) where

- $Q = \{S_1, S_2 \cdots S_n\}$ *is the (finite) set of states;*
- $I : Q \mapsto K$ *is a function over the set of states, which associates to each initial state a value in K (for instance probability, word or boolean), called entry cost, and to non-initial state a zero value ;*
- $F : Q \mapsto K$ *is a function over the set states, which associates to each final state a value in K, called final cost, and to non-final state a zero value;*
- T *is the transition function, that is* $T : Q \times A \times Q \mapsto K$ *which to a state S_i, a letter a*

and a state S_j associates a value z in K (the cost of the transition) if it exist a transition labelled with a from the state S_i to the state S_j and and zero otherwise.

Remark 1 `to redo` *i) As told above, automata with multiplicities are a generalisation of finite automata. In fact, finite automata can be considered as automata with multiplicities in the semiring K, the boolan set $B = \{0, 1\}$ (endowed with the logical "or/and"). To each transition, we affect 1 if it exists and 0 if not.*
ii) We have not yet, on purpose, defined what a semiring is. Roughly it is the least structure which allows the matrix "calculus" with unit (one can think of a ring without the "minus" operation).

With such a setting, the automaton can be seen as a machine which accepts a word and provides a scalar (for instance a probability, a word or a boolean). Let us describe now how the output is computed.

We associate to the support of T the weighted graph Γ_T with edges $edges(\Gamma_T) =$

$$\{(q, a, \alpha, r) \in Q \times A \times k \times Q\}_{T(q,a,r)\neq 0} \tag{4}$$

This means that every edge $(q \to r)$ is superscripted by the pair $(a|\alpha)$ such that $T(q, a, r) = \alpha \neq 0$. Classically call the **path**, a sequence of edges $(p_j, a_j, \alpha_j, q_j)_{1\leq j\leq n}$ such that, for all $j < n$, one has $q_j = p_{j+1}$.
Now, the cost of the path is $\prod_{1\leq j\leq n} \alpha_i$ (product in K) and its label is $a_1 a_2 \ldots a_n$ (concatenation)

$$A(w) = \sum_{q,r\in Q} I(q)\Big(\sum_{\substack{c\ path\ q\to r\\ label(c)=w}} \mathrm{cost}(c)\Big)F(r) \tag{5}$$

Remark 2 *The previous automata with multiplicities can be, equivalently, expressed by its matrix representation which is a triplet*

- $\lambda \in K^{1\times Q}$ *which is a row-vector which coefficients are $\lambda_i = I(S_i)$,*
- $\gamma \in K^{Q\times 1}$ *is a column-vector which coefficients are $\gamma_i = F(S_i)$,*
- $\mu : A \mapsto K^{Q\times Q}$ *is the mapping such that the coefficient on the q_ith row and q_jth column of $\mu(a)$ is $T(q_i, a, q_j)$*

Due to the effect of matrix computation we have

Theorem 1 *[2, 10] Still denoting μ the extension of μ to words, one has*

$$A(w) = I\mu(w)T$$

As the behaviour is a function on the words, we have will have advantage to consider operators on such functions.
In this paper we will concentrate on shifts which are generalizations the univariate case considered above (3).

The behaviour of an automaton $\mathcal{A}$ is, by definition,

$$Behaviour(\mathcal{A}) = \sum_{w\in A^*} \mathcal{A}(w)w \tag{6}$$

where $w \to \mathcal{A}(w)$, is the output function
`la définir en termes de chemins, puis de matrices et montrer que c'est la même chose`

4. Populations of automata

For complexity theory, it is important to deal with *populations of automata with multiplicities*. The first step, before considering neibourhood of *close behaviours* is

1. to be able to test whether two automata have the same behaviour
2. to be able to reduce the number of states of automata (the behaviour being preserved)

In fact, given an automaton $\mathcal{A}$ and if the scalars are taken in a field (also certain rings can do, see [**?**]) there is a minimal automaton having the same behaviour as $\mathcal{A}$. This automaton is the automaton of the *shifts of* $Behaviour(\mathcal{A})$. For a general function $f : A^* \mapsto k$,denoted as a sum $f = \sum_{w\in A^*} f(w)w$ one considers the *positive shifts* $(\gamma_u)_{u\in A^*}$

$$\gamma_u(f) = \sum w \in A^* f(w)uw \tag{7}$$

and the *negative shifts*

$$\gamma_u^\dagger(f) = \sum_{w\in A^*} f(uw)w \tag{8}$$

5. Topological considerations

If K is a field, one sees that the space $\mathcal{A}_{(n)}$ of automata of dimension n (with multiplicities in K) is a K-vector space of dimension $k.n^2 + 2n$ (k is here the number of letters). So, in case the ground field is the field of real or complex numbers [3], one can take any vector norm (usually one takes one of the Hölder norms $||(x_i)_{i\in I}||_\alpha := \left(\sum_{i\in I} |x_i|^\alpha\right)^{\frac{1}{\alpha}}$ for $\alpha \geq 1$, but any norm will do) and the distance is derived, in the classical way, by

$$d(\mathcal{A}_1, \mathcal{A}_2) = norm(V(\mathcal{A}_1) - V(\mathcal{A}_2)) \tag{9}$$

where $V(\mathcal{A})$ stands for the vector of all coefficients of $\mathcal{A} = (\lambda, \mu, \gamma)$ arranged in some order. One has then the result of Theorem 2. Assuming that K is the field of real or complex numbers, we endow the space of series/behaviours with the topology of pointwise convergence (Topology of F. Treves [25]).

Theorem 2 *Let $(\mathcal{A}_n)$ be a sequence of automata with limit $\mathcal{L}$ ($\mathcal{L}$ is an automaton), then one has*

$$Behaviour(\mathcal{L}) = \lim_{n\to\infty} Behaviour(\mathcal{A}_n) \tag{10}$$

where the limit is computed in the topology of Treves.

6. Genetic automata as efficient operators

We define the chromosome for each automata with multiplicities as the sequence of all the matrices associated to each letter from the (linearly ordered) alphabet. The chromosomes are composed with alleles which are here the lines of the matrix [6].

In the following, genetic algorithms are going to generate new automata containing possibly new transitions from the ones included in the initial automata.

The genetic algorithm over the population of automata with multiplicities follows a reproduction iteration broken up in three steps [15, 20, 19]:

- *Duplication*: where each automaton generates a clone of itself;
- *Crossing-over*: concerns a couple of automata. Over this couple, we consider a sequence of lines of each matrix for all. For each of these matrices, a permutation on the lines of the chosen sequence is made between the analogue matrices of this couple of automata;
- *Mutation*: where a line of each matrix is randomly chosen and a sequence of new values is given for this line.

Finally the whole genetic algorithm scheduling for a full process of reproduction over all the population of automata is the evolutionary algorithm:

1. For all couple of automata, two children are created by duplication, crossover and mutation mechanisms;
2. The fitness for each automaton is computed;
3. For all 4-uple composed of parents and children, the performless automata, in term of fitness computed in previous step, are suppressed. The two automata, still living, are the result of the evolution of the two initial parents.

Remark 3 *The fitness is not defined at this level of abstract formulation, but it is defined corresponding to the context for which the automaton is a model, as we will do in the next section.*

7. Applications to competition-cooperation modeling using prisoner dilemma

We develop in this section how we can modelize competition-cooperation processes in a same automata-based representation. The genetic computation allows to make automatic transitions from competition to cooperation or from cooperation to competition. The basic problem used for this purpose is the well-known prisoner dilemma [1].

7.1. From adaptive strategies to probabilistic automata

The prisoner dilemma is a two-players game where each player has two possible actions: cooperate (C) with its adversary or betray him ($\overline{C}$). So, four outputs are possible for the global actions of the two players. A payoff is defined relatively to these possible outputs, as described in the following table where the rows correspond to one player behaviour and the columns to the other player one.

	C	$\overline{C}$
C	(3,3)	(0,5)
$\overline{C}$	(5,0)	(1,1)

Table 1. Prisoner dilemma payoff

In the iterative version of the prisoner's dilemma, successive steps can be defined. Each player do not know the action of its adversary during the current step but he knows it for the preceding step. So, different strategies can be defined for a player behaviour, the goal of each one is to obtain a maximal payoff for himself.

In Figures **??** and **??**, we describe two strategies with transducers. Each transition is labeled by the input corresponding to the player perception which is the precedent adversary action and the output corresponding to the present player action. The only inital state is the state 1, recognizable by the incoming arrow labeled only by the output. The final states are the states 1 and 2, recognizable with the double circles.

In the strategy of Figure **??**, the player has systematically the same behaviour as its adversary at the previous step. In the strategy of Figure **??**, the player chooses definitively to betray as soon as his adversary does it. The previous automaton represents static strategies and so they are not well adapted for the modelization of evolutive strategies. For this purpose, we propose a model based on a probabilistic automaton described by Figure **??** [5].

This automaton represents all the two-states strategies for cooperation and competitive behaviour of one agent against another in prisoner's dilemma.

The transitions are labeled in output by the probabilities p_i of their realization. The first state is the state reached after cooperation action and the second state is reached after betrayal.

For this automaton, the associated matrix representation, as described previously, is:

$$I = \begin{pmatrix} p_1 & 1-p_1 \end{pmatrix}; \quad (11)$$

$$F = \begin{pmatrix} p_6 \\ 1-p_6 \end{pmatrix}; \quad (12)$$

$$T(C) = \begin{pmatrix} p_2 & 1-p_2 \\ p_3 & 1-p_3 \end{pmatrix}; \quad (13)$$

$$T(\overline{C}) = \begin{pmatrix} p_4 & 1-p_4 \\ p_5 & 1-p_5 \end{pmatrix} \quad (14)$$

7.2. From probabilistic automata to genetic automata

From the matrix representations of the automata, we can compute genetic automata as described in previous sections. Here the chromosomes are the sequences of all the matrices associated to each letter. We have to define the fitness in the context of the use of these automata. The fitness here is the value of the payoff.

7.3. General Genetic Algorithm Process for Genetic Automata

A population of automata is initially generated. These automata are playing against a predefined strategy, named S_0.

Each automaton makes a set of plays. At each play, we run the probabilistic automaton which gives one of the two outputs: (C) or $(\overline{C})$. With this output and the S_0's output, we compute the payoff of the automaton, according with the payoff table.

At the end of the set of plays, the automaton payoff is the sum of all the payoffs of each play. This sum is the fitness of the automaton. At the end of this set of plays, each automaton has its own fitness and so the selection process can select the best automata. At the end of these selection process, we obtain a new generation of automata.

This new generation of automata is the basis of a new computation of the 3 genetic operators.

This processus allows to make evolve the player's behavior which is modelized by the probabilistic multi-strategies two-states automaton from cooperation to competition or from competition to cooperation. The evolution of the strategy is the expression of an adaptive computation. This leads us to use this formalism to implement some self-organisation processes which occurs in complex systems.

8. Extension to Emergent Systems Modeling

In this section, we study how evolutive automata-based modeling can be used to compute automatic emergent systems. The emergent systems have to be understood in the meaning of complex system paradigm that we recall in the next section. We have previously defined some way to compute the distance between automata and we use these principles to define distance between agents behaviours that are modeled with automata. Finally, we defined a specific fitness that allows to use genetic algorithms as a kind of reinforcement method which leads to emergent system computation [16].

8.1. Complex System Description Using Automata-Based Agent Model

According to General System Theory [4, 21], a complex system is composed of entities in mutual interaction and interacting with the outside environment. A system has some characteristic properties which confer its structural aspects, as schematically described in part (a) of Figure **??**:

- The set elements or entities are in interactive dependance. The alteration of only one entity or one interaction reverberates on the whole system.

- A global organization emerges from interacting constitutive elements. This organization can be identified and carries its own autonomous behavior while it is in relation and dependance with its environment. The emergent organization possesses new properties that its own constitutive entities do not have. "The whole is more than the sum of its parts".

- The global organization retro-acts over its constitutive components. "The whole is less than the sum of its parts" after E. Morin.

The interacting entities network as described in part (b) of Figure **??** leads each entity to perceive informations or actions from other entities or from the whole system and to act itself.

A well-adapted modeling consists of using an agent-based representation which is composed of the entity called agent as an entity which perceives and acts on an environment, using an autonomous behaviour as described in part (c) of Figure **??**.

To compute a simulation composed of such entities, we need to describe the behaviour of each agent. This one can be schematically described using internal states and transition processes between these states, as described in part (d) of Figure **??**.

There are several definitions of "agents" or "intelligent agents" according to their behaviour specificities [11, 26]. Their autonomy means that the agents try to satisfy a goal and execute actions, optimizing a satisfaction function to reach it.

For agents with high level of autonomy, specific actions are realized even when no perception are detected from the environment. To represent the process of this deliberation, different formalisms can be used and a behaviour decomposed in internal states is an effective approach. Finally, when many agents operate, the social aspects must also be taken into account. These aspects are expressed as communications through agent organisation with message passing processes. Sending a message is an agent action and receiving a message is an agent perception. The previous description based on the couple: perception and action, is well adapted to this.

8.2. Agent Behavior Semi-Distance

We describe in this section the bases of the genetic algorithm used on the probabilistic automata allowing to manage emergent self-organizations in the multi-agent simulation.

For each agent, we define e an evaluation function of its own behaviour returning the matrix M of values such that $M_{i,j}$ is the output series from all possible successive perceptions when starting from the initial state i and ending at the final state j, without cycle. It will clearly be 0 if either i is not an initial state or j is not a final one and the matrix $M_{i,j}$ is indeed a matrix of evaluations [2] of subseries of

$$M^* := (\sum_{a \in A} \mu(a)a)^* \tag{15}$$

Notice that the coefficients of this matrix, as defined, are computed whatever the value of the perception in the alphabet A on each transition on the successful path[1]. This means that the contribution of the agent behaviour for collective organization formation is only based, here, on probabilities to reach a final state from an initial one. This allows to preserve individual characteristics in each agent behaviour even if the agent belongs to an organization.

Let x and y be two agents and $e(x)$ and $e(y)$ their respective evaluations as described above. We define $d(x, y)$ a semi-distance (or pseudometrics, see [3] ch IX) between the two agents x and y as $||e(x) - e(y)||$, a matrix norm of the difference of their evaluations. Let $\mathcal{V}_x$ a neighbourhood of the agent x, relatively to a specific criterium, for example a spatial distance or linkage network. We define $f(x)$ the agent fitness of the agent x as :

$$f(x) = \begin{cases} \dfrac{card(\mathcal{V}_x)}{\displaystyle\sum_{y_i \in \mathcal{V}_x} d(x, y_i)^2} & \text{if } \displaystyle\sum_{y_i \in \mathcal{V}_x} d(x, y_i)^2 \neq 0 \\ \infty & \text{otherwise} \end{cases}$$

8.3. Evolutive Automata for Automatic Emergence of Self-Organized Agent- Based Systems

In the previous computation, we defined a semi-distance between two agents. This semi-distance is computed using the matrix representation of the automaton with multiplicities associated to the agent behaviour. This semi-distance is based on the computation of successful paths which needs to define initial and final states on the behaviour automata. For specific purposes, we can choose to define in some specific way, the initial and final states. This means that we try to compute some specific action sequences which are chararacterized by the way of going from some specific states (defined here as initial ones) to some specific states (defined here as final ones).

Based on this specific purpose which leads to define some initial and final states, we compute a behaviour semi-distance and then the fitness function defined previously. This fitness function is an indicator which returns high value when the evaluated agent is near, in the sense of the behaviour semi-distance defined previously, to all the other agents belonging to a predefined neighbouring.

Genetic algorithms will compute in such a way to make evolve an agent population in a selective process. So during the computation, the genetic algorithm will make evolve the population towards a newer one with agents more and more adapted to the fitness. The new population will contain agents with better fitness, so the agents of a population will become nearer to each others in order to improve their fitness. In that way, the genetic algorithm reinforces the creation of a system which aggregates agents with similar behaviors, in the specific way of the definition of initial and final states defined on the automata.

The genetic algorithm proposed here can be considered as a modelization of the feed-back of emergent systems which leads to gather agents of similar behaviour, but these formations are dynamical and we cannot predict what will be the set of these aggregations which depends of the reaction of agents during the simulation. Moreover the genetic process has the effect of generating a feedback of the emergent systems on their own contitutive elements in the way that the fitness improvement leads to bring closer the agents which are picked up inside the emergent aggregations.

For specific problem solving, we can consider that the previous fitness function can be composed with another specific one which is able to measure the capability of the agent to solve one problem. This composition of fitness functions leads to create emergent systems only for the ones of interest, that is, these systems are able to be developed only if the aggregated agents are able to satisfy some problem solving evaluation.

9. Conclusion

The aim of this study is to develop a powerful algebraic structure to represent behaviors concerning cooperation-competition processes and on which we can add genetic operators. We have explained how we can use these structures for modeling adaptive behaviors needed in game theory. More than for this application, we have described how we can use such adaptive computations to automatically detect emergent systems inside interacting networks of entities represented by agents in a simulation.

[1]A *succesful path* is a path from an initial state to a final state

10. References

[1] R. Axelrod, *The complexity of cooperation*, Princeton University Press, 1997.

[2] J. Berstel and G. Reutenauer, *Rational series and their languages*, EATCS, 1988.

[3] Bourbaki N., *Elements of Mathematics: General Topology*, Chapters 5-10, Springer-Verlag Telos (dec. 1998).

[4] L. von Bertalanffy, *General System Theory*, Georges Braziller Ed., 1968.

[5] C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty "Adaptive behaviour for prisoner dilemma strategies based on automata with multiplicities." In *ESS 2002 Conf., Dresden* (Germany), October 2002.

[6] C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty "Genetic algorithms on automata with multiplicities for adaptive agent behaviour in emergent organizations" In *SCI'2001*, Orlando, Florida, USA, 22-25th July 2001.

[7] G. Duchamp, Hatem Hadj-Kacem and Eric Laugerotte, "Algebraic elimination of ϵ-transitions", *DMTCS, Volume 7* n. 1 (2005), pp. 51-70.

[8] G. Duchamp, J-M Champarnaud, *Derivatives of rational expressions and related theorems*, Theoretical Computer Science **313** (2004).

[9] N. Eber, *Théorie des jeux*, Dunod, 2004.

[10] S. Eilenberg, *Automata, languages and machines*, Vol. A and B, Academic press, 1976.

[11] J. Ferber, *Multi-agent system*, Addison-Wesley, 1999.

[12] L.J. Fogel, A.J. Owens, M.J. Welsh, *Artificial intelligence through simulated evolution*, John Wiley, 1966.

[13] R. Ghnemat, K. Khatatneh, S. Oqeili, C. Bertelle, G.H.E. Duchamp *Automata-based adaptive behavior for economic modeling using game theory*, Proc. of EPNADS05 (Emergent Properties In Natural and Artificial Dynamical Systems), Paris, November 17th, 2005, Springer (In press)

[14] J.S. Golan, *Power algebras over semirings*, Kluwer Academic Publishers, 1999.

[15] D.E. Goldberg, *Genetic Algorithms*, Addison-Wesley, 1989.

[16] J. H. Holland, *Hidden Order - How adaptation builds complexity*, Persus books ed., 1995.

[17] J.E. Hopcroft, R. Motwani, J.D. Ullman, *Introduction to automata theory, Languages and Computation*, Addison-Wesley, 2001.

[18] V. A. Kostyantyn, *A New Representation of Formal Power Series*, FPSAC'04 (Vancouver 2004).

[19] J. Koza, *Genetic programming*, Encyclopedia of Computer Sciences and Technology, 1997.

[20] M. Mitchell, *An introduction to Genetic Algorithms*, The MIT Press, 1996.

[21] J.-L. Le Moigne, *La modélisation des systèmes complexes*, Dunod, 1999.

[22] I. Rechenberg, *Evolution strategies*, Fromman-Holzboog, 1973.

[23] M.P. Schutzenberger, "On the definition of a family of automata", Information and Control **4**, 245-270 (1961).

[24] R.P. Stanley, *Enumerative combinatorics*, Cambridge University Press, 1999.

[25] F. Treves, *Topological Vector Spaces, Distributions and Kernels*, Acad. Press 1967.

[26] G. Weiss, ed., *Multiagent Systems*, MIT Press, 1999.