Automata-based adaptive behavior for economic modeling using game theory

Rawan Ghnemat, Khalaf Khatatneh, Saleh Oqeili, Cyrille Bertelle, Gérard Henry Edmond Duchamp

AUTOMATA-BASED ADAPTIVE BEHAVIOR FOR ECONOMIC MODELING USING GAME THEORY

R. Ghnemat, K. Khatatneh, S. Oqeili

Al-Balqa' Applied University,
Al-Salt, 19117
Jordan

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

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

ABSTRACT

In this paper, we deal with some specific domains of applications to game theory. This is one of the major class of models in the new approaches of modelling in the economic domain. For that, we use genetic automata which allow to buid adaptive strategies for the players. We explain how the automata-based formalism proposed - matrix representation of automata with multiplicities - allows to define a semi-distance between the strategy behaviors. With that tools, we are able to generate an automatic processus to compute emergent systems of entities whose behaviors are represented by these genetic automata.

1. Introduction: Adaptive Behaviour Modeling for Game Theory

Since the five last decades, game theory has become a major aspect in economic sciences modelling and in a great number of domains where strategical aspects has to be involved. Game theory is usually defined as a mathematical tool allowing to analyse strategical interactions between individuals.

Initially funded by mathematical researchers, J. von Neumann, E. Borel or E. Zermelo in 1920s, game theory increased in importance in the 1940s with a major work by J. von Neumann and O. Morgenstern and then with the works of John Nash in the 1950s [9]. John Nash has proposed an original equilibrium ruled by an adaptive criterium. In game theory, the Nash equilibrium is a kind of optimal strategy for games involving two or more players, whereby the players reach an outcome to mutual advantage. If there is a set of strategies for a game with the property that no player can benefit by changing his strategy while the other players keep their strategies unchanged, then this set of strategies and the corresponding payoffs constitute a Nash equilibrium.

We can understand easily that the modelization of a player behavior needs some adaptive properties. The computable model corresponding to genetic automata are in this way a good tool to modelize such adaptive strategy.

The plan of this paper is the following. In the next section, we present some efficient algebraic structures, the automata with multiplicities, which allow to implement powerful operators. We present in section 3, some topological considerations about the definition of distances between automata which induces a theorem of convergence on the automata behaviors. Genetic operators are proposed for these automata in section 4. For that purpose, we show that the relevant "calculus" is done by matrix represention unravelling then the powerful capabilities of such algebraic structures. In section 5, we focus our attention on the "iterated prisonner dilemma" and we buid an original evolutive probabilistic automaton for strategy modeling, showing that genetic automata are well-adapted to model adaptive strategies. Section 6 shows how we can use the genetic automata developed previously to represent agent evolving in complex systems description. An agent behavior semi-distance is then defined and allows to propose an automatic computation of emergent systems as a kind of self-organization detection.

2. Automata from boolean to multiplicies theory (Automata with scalars)

Automata are initially considered as theoretical tools. They are created in the 1950's following the works of A. Turing who previously deals with the definition of an abstract "machine". The aim of the Turing machines is to define the boundaries for what a computing machine could do and what it could not do.

The first class of automata, called finite state automata corresponds to simple kinds of machines [21]. They are studied by a great number of researchers as abstract concepts for computable building. In this aspect, we can recall the works of some linguist researchers, for example N. Chomsky who defined the study of formal grammars.

In many works, finite automata are associated to a recognizing operator which allows to describe a language [2, 10]. In such works, the condition of a transition is simply a symbol taken from an alphabet. From a specific state S,

the reading of a symbol a allows to make the transitions which are labeled by a and $come\ fromS$ (in case of a deterministic automaton - a DFA - there is only one transition - see below). A whole automaton is, in this way, associated to a language, the recognized language, which is a set of words. These recognized words are composed of the sequences of letters of the alphabet which allows to go from a specific state called initial state, to another specific state, called final state.

A first classification is based on the geometric aspect : DFA (Deterministic Finite Automata) and NFA (Nondeterministic Finite Automata).

- In Deterministic Finite Automata, for each state there is at most one transition for each possible input and only one initial state.
- In Nondeterministic Finite Automata, there can be none or more than one transition from a given state for a given possible input.

Besides the classical aspect of automata as machines allowing to recognize languages, another approach consists in associating to the automata a functional goal. In addition of accepted letter from an alphabet as the condition of a transition, we add for each transition an information which can be considered as an output data of the transition, the read letter is now called input data. We define in such a way an *automaton with outputs* or *weighted automaton*.

Such automata with outputs give a new classification of machines. *Transducers* are such a kind of machines, they generate outputs based on a given input and/or a state using actions. They are currently used for control applications. *Moore machines* are also such machines where output depends only on a state, i.e. the automaton uses only entry actions. The advantage of the Moore model is a simplification of the behaviour.

Finally, we focus our attention on a special kind of automata with outputs which are efficient in an operational way. This automata with output are called *automata with multiplicities*. An automaton with multiplicities is based on the fact that the output data of the automata with output belong to a specific algebraic structure, a semiring [13, 22]. 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 (i.e. with semirings) linear algebra.

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 state;*
- *$I : Q \mapsto K$ is a function over the set of states, which associates to each initial state a value of K, 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 of 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 of 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 *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.*

Remark 2 *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). The previous automata with multiplicities can be, equivalently, expressed by a 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 a morphism of monoids (indeed $K^{Q\times Q}$ is endowed with the product of matrices) such that the coefficient on the q_ith row and q_jth column of $\mu(a)$ is $T(q_i, a, q_j)$*

3. 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

$$d(\mathcal{A}_1, \mathcal{A}_2) = norm(V(\mathcal{A}_1) - V(\mathcal{A}_2)) \qquad (1)$$

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 1. 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 [23]).

Theorem 1 *Let $(\mathcal{A}_n)$ be a sequence of automata with limit $\mathcal{L}$ ($\mathcal{L}$ is an automaton), then one has*

$$Behaviour(\mathcal{L}) = \lim_{n\to\infty} Behaviour(\mathcal{A}_n) \qquad (2)$$

where the limit is computed in the topology of Treves.

4. 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 [14, 18, 17]:

- *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, result from 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.*

5. 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 transition from competition to cooperation or from coopeartion to competition. The basic problem used for this purpose is the well-known prisoner dilemma [1].

5.1. 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 relative 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 maximal payoff for himself.

In Figures 1 and 2, 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 1, the player has systematically the same behaviour as its adversary at the previous step. In the strategy of Figure 2, 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 3 [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

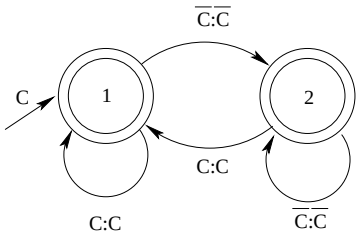


Figure 1. Tit-for-tat strategy automaton

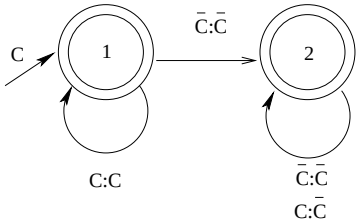


Figure 2. Vindictive strategy automaton

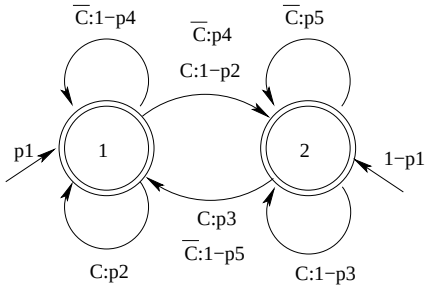


Figure 3. Probabilistic multi-strategies two-states automaton

after cooperation action and the second state is reached after betrayal.

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

$$I = (\, p_1 \quad 1 - p_1 \,); \tag{3}$$

$$F = \begin{pmatrix} p_6 \\ 1 - p_6 \end{pmatrix}; \tag{4}$$

$$T(C) = \begin{pmatrix} p_2 & 1 - p_2 \\ p_3 & 1 - p_3 \end{pmatrix}; \tag{5}$$

$$T(\overline{C}) = \begin{pmatrix} p_4 & 1 - p_4 \\ p_5 & 1 - p_5 \end{pmatrix} \tag{6}$$

5.2. From probabilistic automata to genetic automata

With the matrix representation 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.

5.3. 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 genetics 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.

6. 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 [15].

6.1. Complex System Description Using Automata-Based Agent Model

According to General System Theory [4, 19], 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 4:

- 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 don't 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 4 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 4.

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 4.

There are several definitions of "agents" or "intelligent agents" according to their behaviour specificities [11, 24]. 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 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.

6.2. 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

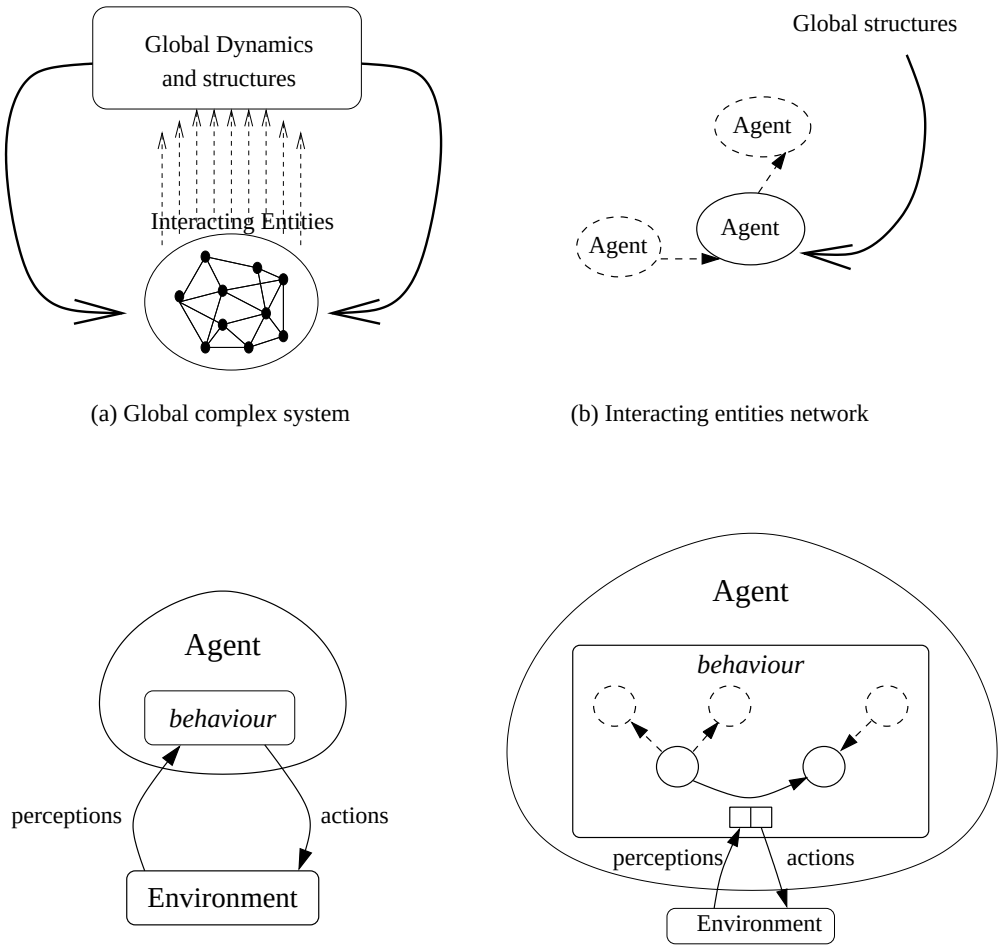
$$M^* := (\sum_{a \in A} \mu(a)a)^* \tag{7}$$

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]. That 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 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 :

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

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



(a) Global complex system

(b) Interacting entities network

(c) Agent−based model for entity

(d) automata−based model for agent behaviou

Figure 4. Multi-scale complex system description: from global to individual models

6.3. 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 successful paths computation 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 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 feed-back of the emergent systems on their own contitutive elements in the way that the fitness improvement lead 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.

7. 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.

8. References

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

[2] J. Berstel and G. Reutenauer, *Rational series and their language*, 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] J.S. Golan, *Power algebras over semirings*, Kluwer Academic Publishers, 1999.

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

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

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

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

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

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

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

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

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

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

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