Rouen University

PhD Thesis

Speciality: Computer Science

presented by

Khalaf KHATATNEH

Subject:

Operators for complex modeling

Defended on July 12th, 2005
For the obtention of
Doctorat de l'Université de ROUEN

Jury composition

Michel COTSAFTIS,	Prof. LTME/ECE, Paris	Referee
Mohammad S. OBAIDAT,	Prof. Monmouth University - USA	Referee
Saleh OQEILI,	Prof. Al-Balqa University - Jordan,	Referee
Joël COLLOC,	Prof. Le Havre University,	Examiner
Habib ABDULRAB,	Prof. Rouen University,	Guest
Éric LAUGEROTTE	A. Prof. Rouen University	Guest
Jean-Gabriel LUQUE,	A. Prof. Gaspar Monge Institute, UMLV	Guest
Gérard DUCHAMP	Prof. Galilée Institute, Paris XIII	Advisor
Cyrille BERTELLE	A. Prof. Le Havre University	Co-advisor

PRELIMINARY DOCUMENT
July 11, 2005

Contents

List of Figures

Acknowlegments

THANKS

A lot of people have supported this work in many different ways and I would like to thank all of them and with the risk of forgetting someone I would like to point out some people. I especially want to thank them for helping me during this work.

Without doubt the most important person has been my advisor Prof. Gérard Duchamp with his sincere ambition to let me develop and find myself very privileged and thankful for being allowed to pursue my work and guide me under his supervision. “ Merci beaucoup Cher Gérard ”.

Also this work would not have been possible without my fourth brother co-advisor Prof. Cyrille Bertelle. My deepest thanks go to my wonderful brother who put our relationship in this way by his support and really I loved our trips to Paris together that made me relax a lot.

Dear Gérard and Cyrille may be I can’t find the words to thank you as you deserves.

Many thanks to my family - my father, my mother, and all my brothers and sisters - for all of love and support which they have given to me, without forgetting Jehad who was very near of me during all the difficult times.

I am grateful to Prof. Mohammad Obaidat for giving opportunity to be one of referee members and for his great support of our workgroup.

Thanks also for all the referees Prof. Michel Cotsaftis and Prof. Saleh Oqeili who will be also one of my best partner in BAU soon.

Also special thanks to the examiners for reading the manuscript and to have spent their time to finish this work (Prof. Joel Colloc, Prof. Habib Abdulrab and Dr.

Jean-Gabriel Luque)

To all my friends in France I am proud to be friend with each of you as individuals (Hatem, Fisal, Jalol and Houda) thank you for your continued courage during this period.

I would like to acknowledge the structural laboratory of LIFAR for their help during my work, specialy the Director of LIFAR Prof. J.-F. Michon and all the staff of the laboratory. Special thank also for Eric Laugerotte for his help, courage and nice discusions with him, thanks also to all Engineering Department staff.

To all people who I have not named explicity or inadvertently left out, thank you. Finally, I would like to thank those who have been my best teachers, the ones who inspired me to learn and to continue learning: My parents.

Many thanks to all of you !

Chapter 1

Introduction

Computer Science finds its origins in computation and its development was highly improved by the need of applications and modelizations. During the half-past century, formal aspects were developed using the huge background of algebraic structures. The aim of this scientific activities consists in finding versatile structures. The genericity is the essential need in terms of programming efficience. In the same period, applications and modelizations grow fast in importance with the hardware improvement. After a long period while technology focused its attention to building mainframes and super-computers, more recently, distributed computing shows its efficience because of the wide spreading of many computer networks (see for example the review from M.S. Obaidat and G.I. Papadimitriou [131]) and because of the fault-tolerent aspects allowed by such distributed computing.

The work presented here take advantage of two domains in Computer Science. The first one can be considered as the natural evolution of research on computation and efficient algebraic data stuctures in modern computation. The second one can be considered as the novative development in modelization and found basis inside artificial intelligence and its capability in problem solving, using distributed approaches.

Complex systems are considered as a major concept for this new century. General system theory introduced by von Bertalanffy [157], are now largely spread in a very huge spectrum of applications (see for example, M. Cotsaftis [52]). The modelization of life cannot today propose some new models without taking into account such a paradigm. The modelization of life is presently largely used as a metaphor in numerous engineering applications. Evolutive systems, agent-based modelizations are some of the new illuminating methodologies that are used for complex modeling.

In this work, we first present in chapter 2, the general concepts of systemic as the background of complex systems paradigm. Emergence and self-organization are the basic phenomena that charaterize complex systems. How to compute in an efficient way these phenomena is the lighting question of this chapter but also of this whole work. We show some of these implementations which deal with automata based models for self-organization.

Chapter 3 presents the basis of efficient algebraic data structures. As heralded by Kleene-Schützenberger theorem in the [147] and confirmed by the recent development of fast and distributed computation, well mastered algebraic structures allow not only to gain in understanding but also to gain in efficiency. Automata in their more general formulation are so introduced and are recently strengthen by semirings support to allow operative aspects [66].

In chapter 4, we present how these operative aspects of automata can be used for agent-based modeling as the basis of complex system simulation. Classical algebraic operators are used for computation of aggregative aspects as an essential key for self-organized structures. Genetic operators are well suited to the management of the dynamic aspects which are indissociable of self-organized phenomena, enlighted by the dissipative structures theory developed by I. Prigogine [140].

Chapter 5 is devoted to applications about economical domains. We show how genetic automata are efficient structures in game theory and in the more versatile way for all the modelizations that deal with cooperative and competitive aspects. We end this application section with the use of evolutive computation through feed-back processes for decision support systems, using cognitive sciences paradigm.

The work presented here deals with a huge knowledge in both theoritical aspects and application modeling in Computer Science. Its power consists in showing the interest of mixing this two aspects. We focus our attention on the needed background both in term of conceptual representation (complex systems paradigm) and in computation theory. More than focus our attention on a specifical application, the aim of this work is to show a huge range of modelization developments which can be expanded in more specific ways in many future works.

Chapter 2

Complex systems and self-organization

Contents

In this first chapter, we introduce the basic concept of the modelling approach with which we deal in this work. Systemic is the first basis and we draw up the general context. We explain why and how we use it. Complex system theory allows to introduce some additional concepts which give some functional description of a large kind of models dealing with various applications. We end this chapter with the first ground of complex system building, as the interaction networks. As modelling activity finds its justification in the applications adressed, we draw a general context of applications, concerning social sciences and life cycle systems which means that evolution aspects is a major concept in the simulations addressed here.

2.1 General system theory and the complex system paradigm

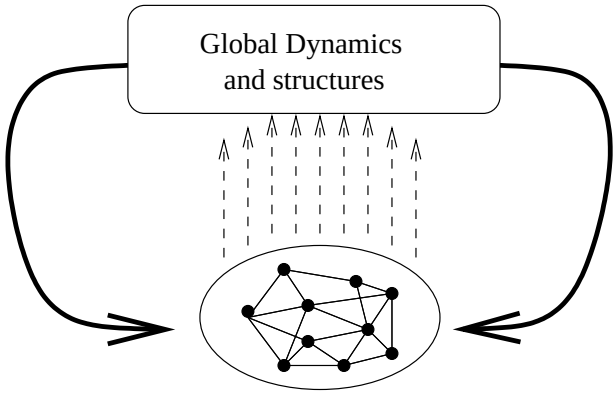


Figure 2.1: Basic system representation

According to systemic, a system is a set composed with entities in mutual interaction and interacting with outside environment. A system has characteristic properties which confer its structural aspects:

- 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 on its constitutive components. "The whole is less than the sum of its parts" after E. Morin.

2.1.1 Closed systems and open systems

A system can be in one of the two following states:

- **Open systems** interact with its environment by means of energy, potential information or matter flux transfer. These fluxes are the catalysts of organization formation which emerges and structures the system.

- **Closed systems** are cut from the outside environment. They are not able to generate dynamically emergent formations.

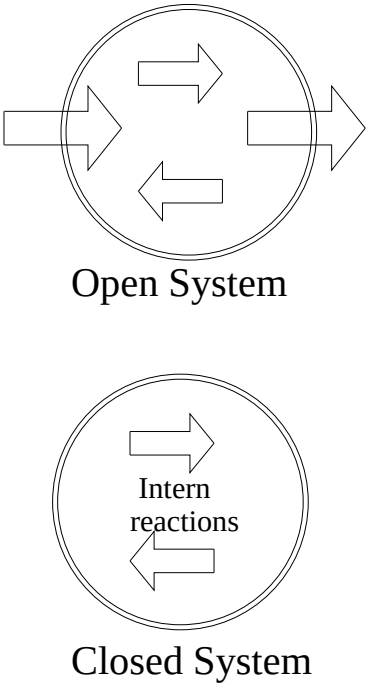


Figure 2.2: Open systems and closed systems after [57]

2.1.2 Complex systems

Complex qualification on a system means heterogeneous caracteristics of the system components. Naturals systems, biological ones for example, or artificial systems, economical ones for example.

J.-L. Le Moigne [112] dissociates "complicated systems" and "complex systems":

- **A complicated system** can be reduced to be better understood;
- **A complex system** cannot be reduced without losing its intelligibility. We need to take into account at once all its components.

A complex system is ruled by the major principle of the preservation of some kind of collective function, as survival or adaptation. For this reason, the system uses a *teleologic* behavior which means that the system is led by its goals. To be able to achieve this functionality, a system has to use some cybernetic rules. The major one is the "feed-back" process which is a kind of global control.

2.1.3 Hierarchical representation

A complex system is generally composed of a lot of sub-systems which aggregate each others in a hierarchical way. There is so a wide huge set of multi-scale organizations, as the result of these aggregations.

2.1.4 Development strategies

The system development can be represented as the result of the application of some of the two following strategies. These two strategies can be mixed or evolve from one to the other:

- **Adaptive strategy** is based on structural reorganizations against some fluctuations;
- **Paradoxical strategy** is based on the existence of antagonist and concurent entities.

2.1.5 System evolution

The complex systems evolve in dynamical way as the succession of two kinds of periods, self-organization ones and stress ones:

- **Self-organization** is a period without major fluctuation. The system seems to complexify itself in terms of the creation of many multi-scale emergent organizations. Such systems develop a great aptitude for adaptive functionality.

- **Stress** is a period with one or more significant fluctuations. The system is simplified and is being destructured partially or totally.

2.1.6 From natural to artificial complex systems

Natural complex systems are characterized by a set of natural entities of different kind, like physical ones. The interactions system leads to a collective behavior which influence the dynamic of the whole system. We give some samples of such systems in the following:

- In Biology: organism cell behavior will contribute to define its global metabolism which constraints each cell.

- In Ecology: the evolution of the major part of species depends on complex organization describing many interacting other species. From this whole organization, some global tendency can be defined. These tendencies are able to increase or not the development of the constitutives populations.

- In Economy: each consumer acts according to its social behavior on the global market which finally constraints each consumer.

- In Transport: the vehicles moves are the basis of traffic which can evolve in organization like traffic jams which constraints in turn the vehicle evolution.

Besides these "natural" systems that we can try to modelize and implement on computers, the present evolution of computation lead to buid complex systems. Decentralized approaches for computation leads to buid computable systems composed of formal communicating entities characterized with some kind of autonomous behavior. In huge networks managing such decentralized programmation, we are facing an artificial complex system. We show in the following (cf. interaction networks) how such systems can lead to some specific form of self-organization.

2.2 From interaction to organizations, emergence and self-organization

Self-organization is a phenomenon which from interactions between elements and other factors tends to create and improve order inside the whole complex system. Such phenomenon go against the increase of entropy and leads to energy dissipation. This dissipation has as effect to maintain the structure generated in that way.

So this phenomenon is a natural tendency of physical dissipative systems or social systems to generate organization from themselves [140].

2.2.1 Interaction networks

Complex systems consists of many very simple entities which interact. The amount of interaction among entities partially determines the overall behaviour of the whole system. On one extreme, systems with little interactions fall into static patterns, while on the other extreme, overactive systems reach the chaos. Between this two extrema exists an area where the phenomenon are particularly interesting. One main key is the notion of interaction and the relationships generate by this last one. So what are the interactions?

If it is difficult to answer to this question, one can be: influences between parts due to their interconnections. These interconnections can be of many forms (e.g. wiring, gravitational or electromagnetic fields, physical contact or logical information channels). We assume that the influence can act in such a way as to change the part state or to cause a signal to be propagated in some way to other parts. Thus the extent of the interactions determines the behavioural richness of the system.

This generate another question, how the interactions are organized? In a general way, the answer is in a network. Thus an interactions network is constituted by a set of interacting entities. Different kinds of interactions networks can be identified [129]:

- social networks ;
- information networks ;
- technology networks ;
- biological networks.

Each one shows an anatomy and characteristics.

Why is network anatomy so important to characterize? Because structure always affects function. For instance, the topology of social networks affects the spread of information and disease, and the topology of the power grid affects the robustness and stability of power transmission. From this perspective, the current interest in networks is part of a broader movement towards research on complex systems.

In the words of E. O. Wilson [162] "The greatest challenge today, not just in cell biology and ecology but in all of science, is the accurate and complete description of complex systems. Scientists have broken down many kinds of systems. They think they know most of the elements and forces. The next task is to reassemble them, at least in mathematical models that capture the key properties of the entire ensembles."

If the networks can be modeled by graphs they bare inherently difficult to understand.

1. Structural complexity;
2. Network evolution;
3. Connection diversity;
4. Dynamical complexity: the nodes could be nonlinear dynamical systems;
5. Node diversity;there could be many different kinds of nodes.
6. Meta-complication: the various complications can influence each other.

Traditionally the study of complex interactions networks has been the field of graph theory. Different structures has been observed:

- Random graph ;
- Small World network ;
- Scale free networks.

2.2.2 A classification for self-organization

We have four broad classes for self-organization each of which include both natural and artificial processes [26]:

- Emergence.
- Inherent computation.
- Non-linear and non-equilibrium processes.
- Evolution.

and these classes account for most of current research and progress on self-organization in complex systems.

The complex systems paradigm uses systemic inquiry to build fuzzy, multivalent, multi-level and multi-disciplinary representations of reality, and systems can be understood by looking for patterns with their complexity, patterns that describe potential evaluations of the systems.

2.2.3 Emergence

The system diverges from its initial state and after a transient period settles into some attractor states. These attractors may be a simple equilibrium or cycle, or may be a strange attractor if the process is chaotic, so settling into the basin of an attractor seems to be a general way for properties and patterns to emerge.

As concerns networks in social and life cycle systems we have to talk about the pattern,the idea of a pattern of organization of a configuration of relationships characteristic of a particular system become the explicit focus of systems thinking in cybernetics and has been a crucial concept ever since.The study of pattern was always present. It began with Pythagoras and Euclid in Greece and was continued by the alchemists, Newton and Galileo, the Romantic poets, arabic scientists, Al-Kuwarizmi, Abu Jafar and various other intellectual movements. However, for most of the time the study of pattern was eclipsed by the study of substance until it re-emerged forcefully in our century, when it was recognized by systems thinkers as essential to the understanding of life [34],[156]. The key to a comprehenive theory of living systems lies in the synthesis of those two very different approaches, the study of substance (or structure) and the study of form (or pattern). In the study of structure we measure and weight things. Patterns, however, cannot be measured or weighted; they must be mapped. To understand a pattern, we must map a configuration of relationships. In other words, structure involves quantities, while pattern involves qualities. The study of pattern is crucial to the understanding of living systems because systemic properties, as we have seen, arise from a configuration of ordered relationships. Systemic properties are properties of a pattern. The components are still there, but the configuration of relationships between them is destroyed, and thus the organism dies.

2.2.4 Inherent computation

We refer with this denomination to systems which evolve with fixed rules. Usually, this rules are computed using automata [98] or discrete events systems or interacting networks of automata or cellular automata [108, 54].

It provides a discrete basis for understanding condensed phase properties, this is especially the case for systems that are discrete in structure and iterative in behavior.

Some examples of effective computations are described in the following paragraph (2.3.1).

2.2.5 Non-linear and non-equilibrium processes

A description of self-organized systems consists in open systems far from equilibrum. They are crossed by energetic and matter flux which leads to such non-equilibrium processes. In this context, feed-back phenomena occurs and are expressed in a mathematical way with non-linear equations[139].

Based on these concepts, a major representation of self-organized open systems is the theory of dissipative structures. This theory has been elaborated by I. Prigogine [140]. Initially, he studies living organisms capabilities to maintain live process in non-equilibrium conditions. In 1960s, I. Prigogine points out the link between non-equilibrum and non linearity. Dissipative systems are able to decrease entropy and so to increase order. This order can be expressed by the fact that organizations emerge from such dissipative systems. These organizations can be the cause of the reinforcement of some irregularities that grow into large scale patterns.

2.2.6 Evolution

Evolution refers to the process of overall change in complex systems. Entities of the system are represented in natural way or artificial one with a genetic information. Advancing to ever smaller levels in their explorations of the phenomena of life, biologists found that the characteristic of the living organisms were encoded in their chromosomes in the same chemical substance, using the same code script.

Simultaneously, the problems which resisted the mechanistic approach of molecular biology became ever more apparent during the second half of the century.

The inspiration for how to conduct experiments of genetics and evolution theory comes from Artifical Intelligence , it has been generally accepted that natural selection is the mechanism by which species evolve (e.g genetic variation). The idea is based on the way in which chromosomes serve a dual purpose, this can represent what the organism will become, and also the actual material that could be transformed to yield new genetic material which we can use it for the new or next generation. Within the *neo-Darwinian* theory we can summarise it as :

- We can get more individuals born in a population that can stand the environment, so they must rival one another for water, food and all other resources in the land.
- The individuals vary in their 'fitness', that is their capability, efficiency and the ability to subsist and reproduce.
- As a result of competition, suitable individuals produce more offspring, so the frequency of their genes increases within a population.
- Natural selection removes incapable individuals from a population, with the time the population becomes better and adapted with environment.
- New species arise when a population becomes isolated for long enough that individuals are no longer capable of breeding with members of the parent species.

Individuals are not fully able to analyze the situation and calculate their optimal strategy.The results described earlier about criticality and connectivity can help to explain some aspects of species evolution. The mechanisms that have let biological evolution to be very well at adaptation have been employed in artifical intelligence technique called "genetic algorithm".

Genetic algorithms [82], [96] solve complex problems of search and optimisation by emulating. The "standard" genetic algorithm (GA) works as follows:

1. START Generate a random population of n chromosomes as suitable solutions for the problem.
2. FITNESS Evaluate the fitness $f(x)$ of each chromosome x in the popluation.
3. NEW POPULATION Create a new popluation by repeating these steps until we will have a complete population:

- $Selection$ select two parent chromosomes from a population according to their fitness .
- $Crossover$ With a crossover probability, cross over the parents to form new offsprings (if no crossover was performed, the offspring is the exact copy of its parents.
- $Mutation$ With a mutation probability, mutate new offsprings at each position in chromosome.
- $Accepting$ Place the new offsprings in the new population

4. REPLACE use new generated population for a further run of the algorithm.
5. TEST if the end condition is satisfied, stop, and return the best solution in current population.
6. LOOP start from step 2.

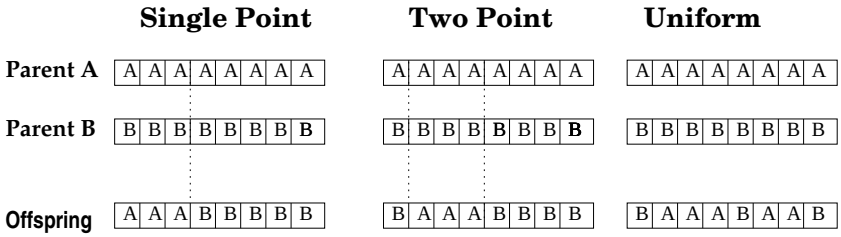


Figure 2.3: Crossover in genetic algorithms, showing three separate mechanisms that can be invoked. Each box denotes a separate 'gene', which may be (say) a parameter in a model. In "crossover" , offspring are formed by selecting genes from one or other of the parents.

As we have seen above a string of genes on a chromosome can undergo genetic transformations, such as mutation, then the initial population is constructed from the allowable set (perhaps by simply picking at random). In each generation, the effectiveness of each individual in the population is determined by running the individual in the current strategic environment.

- Parameters, assumptions and other mutable features of the problem are all expressed as "genes". The combination of genes for each model defines its "genotype". For discrete parameters, possible values are represented as

alleles. Numerical parameters are allowed to vary freely. however, unlike biological genes, alleles that are paired together by Genetic Algorithms (GA) usually represent different features entirely. So dominant and recessive properties do not exist in the standard algorithm.

- The algorithm operates not on a single model, but on an entire population of models, each represented by its set of genes.
- Each "generation" is an iteration of the algorithm. It consists of "reproduction", in which new models are generated from existing ones, and "selection", in which poor models are removed. The important procedures in the reproduction phase are point mutation, by which individual parameters are changed, and crossover, in which features of separate models are mixed in their "offsprings" see the figure 2.3.

Genetic Algorithms (GA) have several practical uses. For instance they deal with a whole population of examples, so they are ideal for exploring the variety in a given system. The result might well be a population that is substantially more successful in the given strategic environment than the orignal population. Finally the genetic algorithm is a highly effective method of searching effective strategies in a huge space of possibilities, the problem for evolution can be conceptualized as a search for relatively high points in a multidimensional field of gene combinations, where height corresponds to fitness.

2.3 From models to simulation, from calculability to operators in complex systems - Some examples

In the previous sections, we present the major concepts about complex systems and self-organization. This allows to describe the frame of the phenomena in which we focus our attention. A phenomenon is firstly observed and the complex system paradigm allows to draw some lighting to make pertinent observations. Once the phenomena are observed, we need to define operative process to buid a simulation. In this section, we focus our attention on some samples of operators build with respect to the complexity of the phenomena observed.

Some of these operators are deeply inscribed in a decentralized computation which is inspired from DAI (Distributed Artificial Intelligence) and MAS (Multi-Agent Systems). Others show how rules can give efficient processes or elsewhere

how some kind of automata can be powerful in term of operative aspect. Such rules or automata can be the basis of emergent formations.

2.3.1 L-system

The name L-system is short of Lindenmayer system. Formally a L-system is a set of syntactic rules and symbols (i.e. formal grammar) that represents discrete steps and units in growth process, and it contains four elements:

- *variables*: are symbols denoting elements that can be replaced.
- *constant*: are symbols denoting elements that remain fixed.(i.e. we can replace each variables by constants English words or phrases to produce sentences in English, such as 'That cat sat on the mat', constants and variables together constitute the alphabet of the system.
- *Start (axioms)* words are expression defining the system's initial state.
- *Rules (syntax)* define how the variables are to be replaced by constants or other variables.

$\langle$ sentence $\rangle \rightarrow \langle$ subject $\rangle$ $\langle$ verb $\rangle$ $\langle$ predicate $\rangle$
$\langle$ subject $\rangle \rightarrow$the cat.
$\langle$ verb $\rangle \rightarrow$sat.
$\langle$ predicate $\rangle \rightarrow$the mat.

The language $L(G)$ generated by a grammar G is the set of all words that can be obtained from the axioms by applying rules in a finite number of steps.

A word which contains no variables is termed a sentence, we can classify languages by properties of their syntax (following Chomsky classification).
- A syntax is said to be regular if every rule is of the form

$$A \rightarrow aB$$
$$A \rightarrow a$$

where A, B are variables and X, Y are any permissible expressions, the context-sensitive grammar include rules of the form

$$AX \rightarrow Y$$
$$XA \rightarrow Y$$

where X and Y are any permissible expressions. We can take an example to consider the simple grammar.

Example 1 *Fibonacci numbers.*

variables :*A,B*
constants :*none*
start :*A*
rules :$A \rightarrow B$
:$B \rightarrow AB$

This L-system produces the following sequence of strings

Stage 0 : A
Stage 1 : B
Stage 2 : AB
Stage 3 : BAB
Stage 4 : ABBAB
Stage 5 : BABABBAB
Stage 6 : ABBABBABABBAB
Counting the length of each string yields the famous Fibonacci sequence:

1 1 2 3 5 8 13 21 34

This simple sequence also demonstrates the way in which iterative properties of L-systems do operate, and if we denote stage n above as S_n , then we observe that for $n \leq 2$ we have $S_n = S_{n-2}S_{n-1}$

Theorem 1 *Let L(G) be a context-free language that is generated from the syntax G, if there exists an integer m, such that S_m expressed in terms of $S_m - i$ where $i < m$, then this same relationship holds for S_n, for all $n > m$.*

L-system employ two distinct kinds of semantics (i.e. the meaning of symbols). In one approach a model gives a complete description of the structure at any growth stage.

Theorem 2 *Any context-free language can be generated by a grammar in which all production rules are the form*

$$A \rightarrow BC$$
$$A \rightarrow a$$

where A,B,C are variables and a is a constant.

Some models introduce intermediate states simply to ensure that the timing of particular events is correct. The context-sensitive models are often more appropriate, and we can see this by the next example:

Example 2 *In L-system also we can model the animal behavior by representing the turtle graphics, this idea finds its simplest expression is so-called "turtle geometry" , introduced by Seymour Papert (1973), deals with patterns produced by the path of an imaginary turtle moving around on a plane. We can define by a grammar such as the following:*

Constant= {nF,nB,aR,aL,Stop}
Variables =$\{< Path >, < Design >, < Arm >$*,etc..}*
Start =$< Path >$

where

nF denotes "n steps Forward"
nB denotes "n steps Back"
aR denotes "n steps turns a degrees Right"
aL denotes "n steps turns a degrees Left"

and basic production rules are:

$$< Path > \rightarrow nF < Path >$$
$$< Path > \rightarrow aR < Path >$$
$$< Path > \rightarrow nB < Path >$$
$$< Path > \rightarrow aL < Path >$$
$$< Path > \rightarrow Stop.$$

In this grammar, the variables $< Path >$ denotes part of the turtle's trail. The transitions represent moves made by the turtle.

At any time, the completed portion of the turtle's path is specified by a sequence of individual moves, with the following rules we can use the variables $< Design >, < Arm >, etc...$ to describe the formation of a simple design (2.4).

$$< Path > \rightarrow < Design > Stop$$
$$< Design \twoheadrightarrow 4 < Arm >$$
$$< Arm > \rightarrow 4F3 < Corner > 1F$$
$$< Corner \twoheadrightarrow 2F3 < Turn >$$
$$< Turn > \rightarrow 90RF.$$

From turtle geometry it is only a small step to syntactic models that describe the organization of animal behavior, so we saw how any animal must interact with its environment.

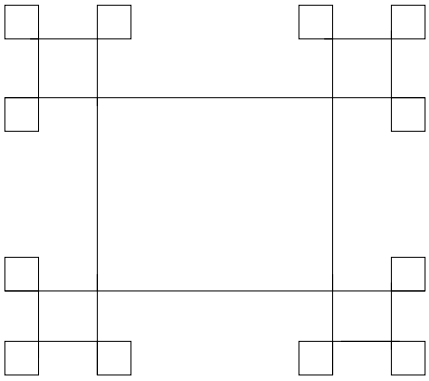


Figure 2.4: A simple design produced by trutle geometry

2.3.2 Adaptive behaviors for social robots from Samuel Landau

We present in this part, the works of Samuel Landau concerning his PhD. The goal is to generate with automatic process some adaptive behaviors for social robots. These works is part of a major scientific project lead in LIP6 by Alexis Drogoul and called "MICRobES". The corresponding basic thematics are

- autonomous robotic in a real environment,
- collective robotic,
- evolutionist robotic.

The evolutionist paradigm allows to go forward emergent and adaptive robots behavior. This emergent behavior is not previously plan and come from complex interactions between robotic society and its environment. We deal here with self-learning.

S. Picault and S. Landau define a new concept called *Ethogenetic* which deals with general principles for genetic models building applied to evolvables agent behavior. The two major aspects are:

- Continuity: this notion concerns progressive adaptation in terms of progressive adding of small improvments for individual, during few generations. This notion is implemented using indirect codage which leads to separate genetic substratum from evaluated behavior description.

- Expressivity: This notion consist in the fact that agent behaviors are automatically created with a minimum limit concerning research space. The generated structures are of arbitrary complexity but they are modular.

Stack use for the building of the semantic structure

The semantic structure is built using a stack. The principles are inspired from a natural process: the protein synthesis (see figure 2.5).

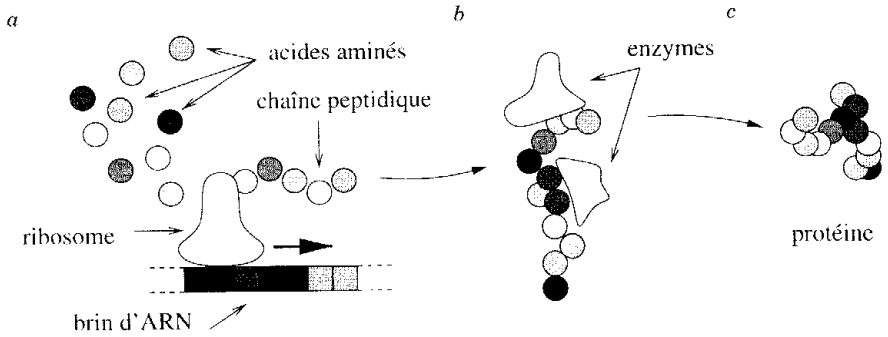


Figure 2.5: Protein synthesis from [106]

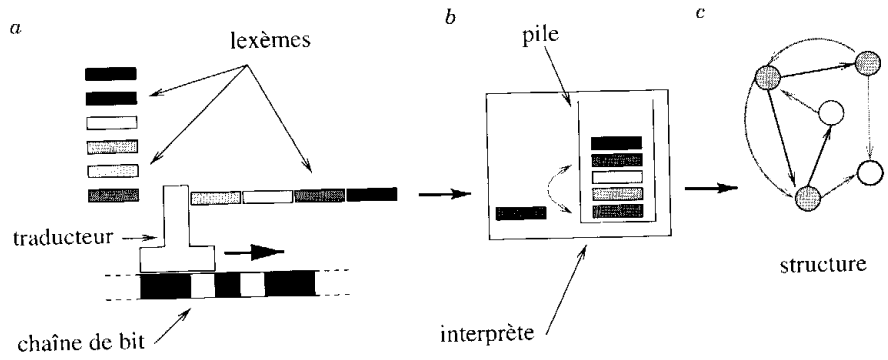


Figure 2.6: processus inspirated from protein synthesis from [106]

The computed process is bio-inspired by the protein synthes (see figure 2.6), using a framework composed of 2 elements:

- SFERES concerns artificial evolution for multiagent simulations;
- ATNoSFERES is based on SFERES and allows the automatic building of evolvable agents which are represented with an oriented and labelled graph, ATN.

SFERES

This package which is described in figure 2.7 manages the evolutionist part. In such a way, there is a separation between the genetic evolution and the simulation part (evaluation).

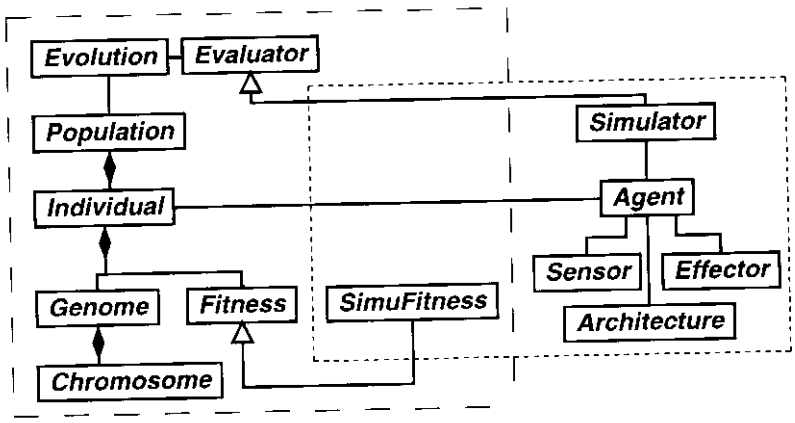


Figure 2.7: Global view of sphere project from [106]

ATN based modelling

The basic model used called ATN is an oriented and labelled graph as described in figure 2.8. This graph is evaluated like a automaton:

- We start in "Start".
- We manage the actions:
  - the reachable edges are selected (the conditions on the edge must be satisfied)
  - one acceptable edges is choosen and then it's crossed, realized the labelled actions.
- We stop when we are on "End".

From binary genetic code to ATN

The genetic code is translated in lexemes which belong to one of the 3 following kinds:

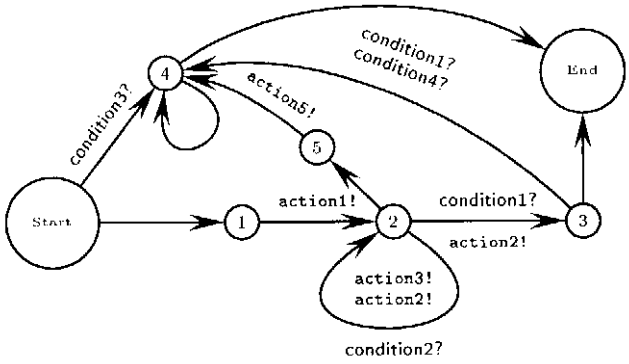


Figure 2.8: ATN representation from [106]

- Agent lexemes: conditions and actions;
- ATN lexemes: nodes and connexions creation;
- Stack lexemes: stack management.

The interpret reads lexeme flux, build and manage stack, buid ATN:

- Stack lexemes are executed;
- Agent lexemes are stacked;
- Node creation lexeme stacked the node;
- connexion lexeme connect two internal nodes and unstack

In the figure 2.9, we find the array describing the ATN manufacturing, using stack which are modified with new token.

token	(état initial de la pile)		(pile résultante)
dup	$(x\ y\ ...)$	$\longrightarrow$	$(x\ x\ y\ ...)$
del	$(x\ y\ ...)$	$\longrightarrow$	$(y\ ...)$
dupNode	$(x\ y\ N_i\ z\ ...)$	$\longrightarrow$	$(N_i\ x\ y\ N_i\ z\ ...)$
delNode	$(x\ N_i\ y\ N_i\ z\ ...)$	$\longrightarrow$	$(x\ y\ N_i\ z\ ...)$
popRoll	$(x\ y\ ...\ z)$	$\longrightarrow$	$(y\ ...\ z\ x)$
pushRoll	$(x\ ...\ y\ z)$	$\longrightarrow$	$(z\ x\ ...\ y)$
swap	$(x\ y\ ...)$	$\longrightarrow$	$(y\ x\ ...)$
node	$(x\ ...)$	$\longrightarrow$	$(N_i\ x\ ...)$ [a]
startConnect	(c1? c2? c1? x a2! a1!y N_i...)	$\longrightarrow$	$(x\ y\ N_i...)$ [b]
endConnect	(c1? c2? c1? x a2! a1!y N_i...)	$\longrightarrow$	$(x\ y\ N_i...)$ [c]
connect	(c1? c2? $x\ N_i\ y$ c1? z a2! a1! $t\ N_j\ u$ ...)	$\longrightarrow$	$(x\ N_i\ y\ z\ t\ N_j\ u\ ...)$ [d]
condition?	$(x\ ...)$	$\longrightarrow$	(condition? x ...)
`action!`	$(x\ ...)$	$\longrightarrow$	(`action!` x ...)

Figure 2.9: ATN manufacturing from [106]

Chapter 3

Semirings, Automata and applications

Contents

3.1 Functions on monoids

3.1.1 Introduction

Language Theory deals with sets of words (or subsets of some free monoid), System Theory deals with actions (which constitute monoids), transducers deal with correspondences (with or without weights). All these theories can be placed in a unified framework: the space of functions on a suitable monoid and taking their values in a suitable (commutative or not) semiring. In fact, we want to maintain the non-commutative possibility as in the next chapter *"Evolutive agent behaviour modelling based on genetic automata"* it will be made use of a *transducer* which, for our needs, will just be an automaton with multiplicities taken in a noncommutative semiring.

3.1.2 Semirings: the scalars of Computer Science

Etymologically, the word *scalar* has to do with scale and scaling. In physics, *scalar quantities* are distinguished from *vector quantities*. The second add the first add and multiply. In Computer Science, they are *costs, probabilities, truth values, words*. This time they have kept the historical and functionnal virtues of being able to add and multiply, but contrary to the philosophy of rings, aditive symmetry is not required. In fact we ask the minimal properties for which computations on matrices (with unit) can be performed.
For more details, and an impressive review covering not less than 400 references (with applications in: Fuzzy logic and sets, Probability measures, Command algebras, Information algebras, Automata theory, Operator theory, Schedule algebras, Signal processing, Economic agents, Bi- and Multivalued Logic, $\cdots$), the reader is referred to [80, 81].

In fact, the first semiring encountered by mankind through history (and children during their classes) is not a ring. This is $(\mathbb{N}, +, .)$ where

$$\mathbb{N} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \cdots, \cdots, 10000, \cdots, 10^{100}, \cdots\} \tag{3.1}$$

Let's define the structure of semirings more formally.
Let k be a set. A structure of *semiring* n k is the data of two internal compostion laws $(+, .)$ with the following properties:

- SR1) $(k, +)$ is a commutative monoid with neutral 0_k
- SR2) $(k, .)$ is a monoid with neutral 1_k
- SR3) the product is (left and right) distributive with respect to the addition

- SR4) 0_k is an annihilator ($0_k \times x = x \times 0_k = 0_k$)

Remark 1 *i) If the unity matrix is not required, one can withdraw the need of neutrals. See*
`http://mathworld.wolfram.com /Semi ring .html`
Such a semiring can be naturally embedded in a semiring in our sense.
ii) SR4 is unrelated to SR1..3 as shows the model $(\mathbb{N}, max, +)$.

The semirings form a category for which we give the morphims.

Definition 1 *Let* $(k_1, +_1, \times_1)$ *(resp.* $(k_2, +_2, \times_2)$*). We say that a mapping* $k_1 \overset{\phi}{\mapsto} k_2$ *is a morphism of semirings iff* ϕ *is a morphism of the additive and multiplicative monoid structure i.e.*

- $(\forall x, y \in k_1)(\phi(x +_1 y) = \phi(x) +_2 \phi(y));\ \phi(0_{k_1}) = 0_{k_2}$
- $(\forall x, y \in k_1)(\phi(x \times_1 y) = \phi(x) \times_2 \phi(y));\ \phi(1_{k_1}) = 1_{k_2}$

As usual, if $\phi : k_1 \mapsto k_2$ is an inclusion mapping, we say that k_1 is a subsemiring of k_2. If ϕ is onto, we say that k_2 is a quotient of k_1.

As for the case of rings and fields, the subsemiring of k generated by 1_k is of great importance and gives rise to the notion of characteristic which, in this case, is two fold.

Proposition 1 *Let* $(k, +, .)$ *then:*
1) i) there is an unique morphisms of monoids $\phi_k : \mathbb{N} \mapsto k$.
ii) one has, $\phi(n) = \underbrace{1_k + 1_k \cdots 1_k}_{n \text{ times}}$ *(denoted below* $n1_k$*) and then, the subset* $\phi(\mathbb{N})$ *is the additive submonoid generated by* 1_k *i.e.*

$$\phi_k(\mathbb{N}) = \{0_k, 1_k, 1_k + 1_k, \cdots \underbrace{1_k + 1_k \cdots 1_k}_{n \text{ times}} \cdots\} \tag{3.2}$$

iii) $\phi(\mathbb{N})$ *is a subsemiring of* k*, in fact the smallest of all the subsemirings of* k.
2) (Structure of $\phi(\mathbb{N})$*) [2] Either* $(\forall n > 0)(\phi(n) \neq 0_k)$ *and* $\phi(\mathbb{N}) \simeq \mathbb{N}$ *or it exists an unique couple* $(e, p) \in \mathbb{N} \times \mathbb{N}^+$ *such that*

1. $\{0, \cdots e + p - 1\}$ *is a section of* ϕ *(hence* $|\phi(\mathbb{N})| = e + p$*)*
2. $\phi(m) = m1_k$ *if* $m < e$ *and* $\phi(m) = \Big((m - e \mod p) + e\Big)1_k$ *if* $m \geq e$

Definition 2 *We say that the characteristic of a semiring k is 0 if $k_{(0)} \simeq \mathbb{N}$ and $(e,p) \in \mathbb{N} \times \mathbb{N}^+$ if (as above) $|\phi(\mathbb{N})| = e + p$ and $\phi(e) = \phi(e+p) = 0_k$ and one sets $ch(k) = (e,p)$.*

Remark 2 *i) The semiring $\phi(\mathbb{N})$ is called the* basic semiring *of k and denoted $k_{(0)}$ in [80]. In [80], the characteristic is also 0 if $k_{(0)} \simeq \mathbb{N}$. Otherwise it is $(e+p, e)$.*
ii) We will denote by $\mathbb{K}_{e,p}$ the model of the basic semirings (which are all isomorphic) of characteristic (e,p), realized as follows.
One can endow $\mathbb{K}_{e,p} = \{0, 1, \cdots e, \cdots e+p-1\}$ with a unique structure of semiring such that the mapping $r : \mathbb{N} \mapsto \mathbb{K}_{e,p}$ defined by

$$r(n) = \begin{cases} n \text{ si } 0 \leq n \leq e+p-1 \\ (n - e \; mod \; p) + e \text{ sinon} \end{cases} \tag{3.3}$$

be a morphism. iii) A semiring of characteristic (e,p) is a ring iff $e = 0$.
iv) A semiring is additively idempotent (i.e. $x + x = x$ identically, see below) iff it is of characteristic $(1,1)$ which is equivalent to the fact that $k_{(0)}$ be the boolean semiring.

Example 3 *We have the following first examples*

- *a1) The boolean semiring $(\mathbb{B}, +, \times)$.*
- *a2)$(\mathbb{N} \cup \{+\infty\}, min, +)$,*
- *a3) $(\mathbb{R} \cup \{-\infty\}, max, +)$ (known as the "Tropical semiring") [79].*
- *b1) $(\mathbb{Z}, +, \times)$*
- *b2) $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ with n composite (i.e. non-prime).*
- *c1) $(\mathbb{Q}, +, \times)$, $(\mathbb{R}, +, \times)$, $(\mathbb{C}, +, \times)$ and $(\mathbb{Z}/n\mathbb{Z}, +, \times)$ with n prime*
- *c2) $(\mathbb{H}, +, \times)$*

Comment. — *"ai"s are pure semirings (i.e. not rings), "bi"s are rings but not fields, "ci"s are fields and c2 (Cayley's quaternion numbers) is not commutative.*

A semiring k is said commutative if and only if the monoid (k,.) is commutative. If k is a semiring and Q is a finite set, the set $k^{Q \times Q}$ of the square matrices with coefficients in k is naturally endowed with a structure of a semiring by (the usual matrix operations).

Remark 3 *If $1 \neq 0$, $k^{2\times 2}$ is not a commutative as in fact*

$$\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

whereas

$$\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

3.1.3 Valued graphs for automata

Extensive use will be made of weighted graphs, such a graph is the data of:

- a set of arrows $\mathbf{E}$ and of vertices $\mathbf{V}$
- two functions **tail** $t : \mathbf{E} \mapsto \mathbf{V}$ and **head** $h : \mathbf{E} \mapsto \mathbf{V}$
- two functions $\mathbf{E} \xrightarrow{\lambda} \mathcal{A}$ and $\xrightarrow{\omega} k$.

Where $\mathcal{A}$ is an alphabet of commands and k is some set of coefficients where the computations have to be done (indeed a semiring).
Usually the symbol $e = p \xrightarrow{a|\alpha} q$ means

$$t(e) = p,\ h(e) = q,\ \lambda(e) = a,\ \omega(e) = \alpha \tag{3.4}$$

A path is a sequence $e_1 \cdots e_n = \pi$ of arrows such that $(\forall i < n)(h(e_i) = t(e_{i+1}))$. One extends at once the four functions t, h, λ, ω to the paths by

$$\begin{array}{rcl} t(\pi) & = & t(e_1) \\ h(\pi) & = & h(e_n) \\ \lambda(\pi) & = & \lambda(e_1)\cdots\lambda(e_n) \text{ (Concatenation)} \\ \omega(\pi) & = & \omega(e_1)\cdots\omega(e_n) \text{ (Product in the semiring))} \end{array} \tag{3.5}$$

From this definition we see that, given a family of paths.
PS) If they are in series $\pi_1 \cdots \pi_k$ (which means that $(\forall i < n)(h(\pi_j) = t(\pi_{j+1}))$ then

$$\lambda(\pi_1 \cdots \pi_k) = \lambda(\pi_1)\cdots\lambda(\pi_k) \text{ and } \omega(\pi_1 \cdots \pi_k) = \omega(\pi_1)\cdots\omega(\pi_k) \tag{3.6}$$

PP) If the paths are in parallel (same head, tail and label) then

$$\omega(\{\pi_1 \cdots \pi_k\}) = \sum_{i\in I} \omega(\pi_i) \tag{3.7}$$

the axioms of semiring appear them as completely determined by this structure.

Diagram	Identity
$p \overset{\overset{\alpha}{\to}}{\underset{\gamma}{\overset{\beta}{\to}}} q$	$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
$p \overset{\overset{\alpha}{\to}}{\underset{\beta}{\to}} q$	$\alpha + \beta = \beta + \alpha$
$p \xrightarrow{\alpha} q \xrightarrow{\beta} r \xrightarrow{c\|\gamma} s$	$\alpha(\beta\gamma) = (\alpha\beta)\gamma$
$p \overset{\overset{\alpha}{\to}}{\underset{\beta}{\to}} q \xrightarrow{\gamma} r$	$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$
$p \xrightarrow{\alpha} q \overset{\overset{\beta}{\to}}{\underset{\gamma}{\to}} r$	$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$

Remark 4 *These axioms ensure at once that that the matrix computations with coeficients in k are feasible.*

3.1.4 Modules

This section is rather algebraic in flavor and can be skipped with no major harm. However a deep understanding of the most efficient results would require it. For further studies, the reader is referred to [80] where modules are called *semimodules*.

Let $(M, +)$, be a commutative monoid. A left K-module structure on M is defined by the data of a mapping $\phi : K \mapsto End(M)$ such that :

$$\textbf{(MG)}\ \phi \text{ is morphism} \tag{3.8}$$

where, as this can be checked straightforwardly, $End(M)$ is considered as endowed with its natural structure of semiring $(End(M), +, \circ)$.

In general αx denotes the element $\phi(\alpha)(x)$ and this defines, in an equivalent manner, an external law $K \times M \mapsto M$.

The data **(MG)** is the equivalent to the following requirements

$$(\forall \alpha, \beta \in K)(\forall x, y \in M)$$

1. $\alpha(x + y) = \alpha x + \alpha y;\ \alpha 0_M = 0_M$
2. $(\alpha + \beta)x = \alpha x + \beta x;\ 0_K x = 0_M$
3. $\alpha(\beta x) = (\alpha\beta)x;\ 1_K x = x$

In a dual way, a right K-module structure on M is defined as soon as one has an anti-morphism[1] $\phi : K \mapsto End(M)$. In order to express these axioms as

[1] $(\forall \alpha, \beta \in K)(\phi(\alpha\beta) = \phi(\beta)\phi(\alpha))$

natural, one creates at once an external law $M \times K \mapsto M$ defined by $(x, \alpha) \mapsto \phi(\alpha, x)$. The resulting axioms can be written without difficulty.

Notions of morphisms of modules, of submodules and bimodules are defined as in standard linear algebra.

The bi-module of fonctions $X \mapsto K$

Let X be a set and K a non-trivial (i.e. $0_k \neq 1_K$) semiring. One calls K-subset of X[72] (or set with multiplicities in K), every mapping $X \stackrel{f}{\mapsto} K$. Following the general conventions [28], the set of Ksubsets of X is denoted K^X.
This set in naturally endowed with a structure of K-bimodule by (given $(\alpha, x) \in K \times X$)

$$\alpha f : x \mapsto \alpha f(x);\ f\alpha : x \mapsto f(x)\alpha \tag{3.9}$$

some submodules, in particular those *of finite type*, of K^M defined by automata, will be of utmost importance in the sequel.

Definition 3 *(Summability) A family $(f_i)_{i\in I}$ of K-subsets will be said summable [10] iff*

$$(\forall x \in X)((f_i(x))_{i\in I} \textit{ is finitely supported}) \tag{3.10}$$

where the support *of f is*

$$Supp(f) = \{x \in X | f(x) \neq 0\} \tag{3.11}$$

We are now in position of enriching our space with a natural pairing. Let $k^{(M)}$ denote the space (set) of finitely supported $f \in k^M$, that is

$$k^{(M)} = \{f \in k^M | Supp(f) \text{ is finite}\} \tag{3.12}$$

then, for $(f, g) \in k^{(M)} \times k^M$ or $(f, g) \in k^M \times k^{(M)}$, one defines the scalar product of f and g with the finite supported sum

$$\langle f|g\rangle = \sum_{x\in X} f(x)g(x) \tag{3.13}$$

3.2 The case when $X = M$ is a monoid: behaviour of automata

3.2.1 Series

In case $X = M$ is endowed with a structure of monoid, the K-subsets are usually called series over the monoid M. The reason of this is the multiple specialisations

of this construction [27] which served, with a considerable amount of studies, as spaces of series. This feature as even gained in importance with the advent of Computer Science(s) where this functions serve to measure the behaviour of automata. Let us summarize in a tabular form the different species of series.

Series	Monomials	Restrictions	Space
Univariate in z	$z^{\mathbb{N}} = \{z^n\}_{n\in\mathbb{N}}$	None	Noted $k[[z]]$
Polynomials in z	$z^{\mathbb{N}} = \{z^n\}_{n\in\mathbb{N}}$	Finite support	Noted $k[z]$
Several commutative variables (X)	$\mathbb{N}^{(X)}$, fonct. $X \mapsto \mathbb{N}$ with finite support	None	Noted $k[[X]]$
Polynomials with several commutative variables (X)	$\mathbb{N}^{(X)}$, fonct. $X \mapsto \mathbb{N}$ with finite support	Finite support	Noted $k[X]$
Several noncommutative variables (X)	X^*, free monoid over the alphabet X	None	Noted $k\langle\langle X\rangle\rangle$
Polynomial with several noncommutative variables	X^*, free monoid over the alphabet X	Finite support	Noted $k\langle X\rangle$
Laurent series	$z^{\mathbb{Z}} = \{z^n\}_{n\in\mathbb{Z}}$	$n \geq N;\ N \in \mathbb{Z}$	Noted $k((z))$
Laurent polynomials	$z^{\mathbb{Z}} = \{z^n\}_{n\in\mathbb{Z}}$	Finite support	Noted $k(z, z^{-1})$
Puiseux series	$z^{\mathbb{Q}_+} = \{z^\alpha\}_{\substack{\alpha\in\mathbb{Q}\\ \alpha>0}}$	None	
Malcev series	$(\Gamma, \prec)$ linearly ordered group	Well ordered support	Noted $k((\Gamma))$
Taylor series	$z^{\mathbb{N}} = \{z^n\}_{n\in\mathbb{N}}$	Convergence radius	Noted $k[[\{z\}]]$ $k = \mathbb{R}$ ou $\mathbb{C}$
Exponentials	$\{e^{nz}\}_{n\in\mathbb{N}}$		
Dirichlet	$\{n^{-z}\}_{n>0 \text{ entier}}$		
Bertrand	$e^{\alpha z} ln(z)^\beta n^\gamma$		

Thus, series must be understood as functions $M \mapsto k$ where M is the space of monomials (endowed with a structure of monoid) and k, some semiring. Let us examine the extra operations on the series induced by the product of M.

First, we remark that, given any function $f : M \mapsto k$, the family $(f(m)m)_{m\in M}$ is summable with sum f. This provides us another denotation for f (which denotes however the same object)

$$f = \sum_{m\in M} f(m)m \tag{3.14}$$

and we immediately, in case f, g be finitely supported, have the usual product within the algebra $k[M]$ of the monoid M.

3.2.2 Products, inversion and star

First extension: Cauchy product

When M is a monoid, one has an additional structure on the space $k^{(M)}$ which is called the *algebra* of M. Let us recall it's product called the convolution or

Cauchy product [10]. With $f, g \in k^{(M)}$ one has (all the sums can be checked to be finite supported)

$$\Big(\sum_{u\in M} f(u)u\Big)\Big(\sum_{v\in M} g(v)v\Big) = \sum_{u,v\in M} f(u)g(v)uv = \sum_{w\in M}\Big(\sum_{uv=w} f(u)g(v)\Big)w$$

so, we take it as definition of the Cauchy product which extends linearily the product within M. Let us state

$$f * g := \sum_{w\in M}\Big(\sum_{uv=w} f(u)g(v)\Big)w \tag{3.15}$$

Proposition 2 *The space $(k^{(M)}, +, *)$ is a semiring which contains M as a (multiplicative) submonoid.*

We are now looking to extend this product to series in k^M and we see that the sum (3.15) is well defined. This condition is the following [27]

$$\textbf{Condition (D)}\ (\forall w \in M)(\#(\{(u,v) \in M^2 | uv = w\}) < +\infty) \tag{3.16}$$

under this condition the space k^M can be extended with a structure of semiring extending that of $k^{(M)}$.

Proposition 3 *Provided that M fulfills condition* **D**, *the space $(k^M, +, *)$ is a semiring which contains $(k^{(M)}, +, *)$ as a subsemiring.*

Second extension: star

Star is unary operation which is of central importance in language theory [10, 104, 117, 144]. But, in the multiplicity real, when one has a ring of scalars and $M = A^*$ (a free monoid), a series S has an inverse iff $\langle S|1_{A^*}\rangle$ is invertible, in fact this is true in a wider class, the *locally finite* monoids [72].

Definition 4 *(Locally finite monoids) Let M be a monoid and set $M^+ = M - \{1_M\}$ then M is said locally finite iff*

$$\bigcap_{n\geq 1}(M^+)^n = \emptyset \tag{3.17}$$

in fact this is equivalent to the summability of the family $\Big((M^+)^n\Big)_{n\geq 0}$ (a subset being identified with its characterisc function in $\mathbb{N}^M$) and the sum

$$(M^+)^* = \sum_{n\geq 0}(M^+)^n = 1_M + (M^+) + (M^+)^2 + \cdots (M^+)^k \cdots \tag{3.18}$$

makes sense. The star operation, which will be developped below, is of great importance in Computer Science as it corresponds to the measures of iterations.

Definition 5 *Let k be a semiring, we will say that $y \in k$ is a right (resp. left) star of x iff $xy + 1 = y$ (resp. $yx + 1 = y$).*

The case of series without constant term is of special interest as thy alway admit a two-sided star.

Definition 6 *Let k be a semiring and M a locally finite monoid. We call* proper *a series $S \in k^M$ such that $\langle S|1_M\rangle = 0$.*

Proposition 4 *(Inversion and star) Let M be a locally finite monoid and $S \in k^M$, then*
1) If k is a ring, S has an inverse iff $\langle S|1_M\rangle$ does.
2) If S is proper, the family $(S^n)_{n\geq 0}$ is summable and

$$S^* := \sum_{n\geq 0} S^n \tag{3.19}$$

is the unique star of S. It is a two-sided star
3) In general S admits a star (left or right) iff $\langle S|1_M\rangle$ does.

3.3 Automata with Multiplicities

3.3.1 Generalites

An autmaton is a device with permits to assign to every word a coefficient in a semiring and this in an implementable form (mainly using matrix computations). Let k a semiring, then an automaton (or, more precisely, a k-automaton) is the data a five-uplet $(Q, A, \mu, \lambda, \gamma)$ with :

- Q: the (finite) set of states,
- A: a finite set (alphabet),
- $\mu : A \to k^{Q\times Q}$, the transition function.
- $\lambda \in k^{1\times Q}$: the set of initial states together with initial values,
- $\gamma \in k^{Q\times 1}$: the set of final states together with initial values.

Remark 5 *i) For graphical reasons, one often derives the following*

- $T = \left\{(q_1, a|\mu(a)_{q_1,q_2}, q_2)\right\}_{\substack{q_1,q_2\in Q \\ \mu(a)_{q_1,q_2}\neq 0}}$ *is the set of transitions of non zero weight.*

- $I = \{(q \in Q : \lambda(q) \neq 0)\}$ *the set of initial states,*
- $F = \{(q \in Q : \gamma(q) \neq 0)\}$ *is the set of final states*

the set T characterises (i.e. is equivalent to the data of) μ.
ii) When $k = \mathbb{B}$ (the boolean semiring), the $\mathbb{B}$-automata are exactly the classic automata (NFA) which are then often called boolean automata since to each transition we affect the coefficient 1 if it exists, 0 if not.

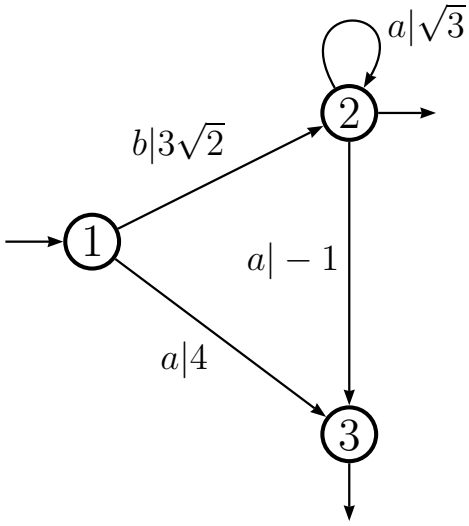


Figure 3.1: Example of automaton with multiplicities

Example 4 *Let* $A = (Q, A, \lambda, \gamma, \mu)$ *where*

- $Q = \{1, 2, 3\}$
- $A = \{a, b\}$
- $\delta = \{(1, b|3\sqrt{2}, 2), (1, a|4, 3), (2, a|-1, 3), (2, a|\sqrt{3}, 2)\}$

Let $f = (q_1, a|\alpha, q_2) \in T$ a transition of an automaton $\mathcal{A}$, we set :

- $label(f) = a$,
- $tail(f) = q1$,
- $head(f) = q2$,
- $weight(f) = \alpha$.

A path $c = f_1 \cdots f_m$ is an element of T^* sush that for all $i < m$ one has $head(f_i) = tail(f_{i+1})$.

We also have, in accordance with the conventions about valued graphs:

- $label(c) = label f_1) \cdots label(f_m)$ (concatenation)
- $tail(c) = tail(f_1)$,
- $head(c) = head(f_m)$,
- $weight(c) = weight(f_1). \cdots .weight(f_m)$ (product in the semiring).

3.3.2 Behaviours

We will define in a moment the local behaviour of an automaton $\mathcal{A}$, but the philosophy is as follow. For a path, the weight is the product of the weights of the arrows (see above), for a set of paths with same origin (tail), target (head) and label, the weight is the sum of the weights of the individual paths. The local behaviour of $\mathcal{A}$ between two states $p, q \in Q$ for the label $w \in A^*$ is the product of the initial weight $\lambda(p)$, the total weight of the set of paths between p and q with label w and the final weight $\gamma(q)$, it reads

$$\mathcal{A}_{p,q}(w) = \sum_{tail(c)=p;\ head(c)=q;\ label(c)=w} weight(c) \tag{3.20}$$

and the global behaviour is

$$\mathcal{A}(w) = \sum_{p,q \in Q} \mathcal{A}_{p,q}(w) \tag{3.21}$$

one defines the behaviour as

$$Behaviour(\mathcal{A}) = \sum_{w \in A^*} \mathcal{A}(w)w \tag{3.22}$$

Example 5 *With the data of figure (3.2), one has*

$$\lambda = (1,0,0);\ \gamma = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix};\ \mu(a) = \begin{pmatrix} 3 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix};\mu(b) = \begin{pmatrix} 3 & 3 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 4 \end{pmatrix}$$

3.3.3 Computation of $\mathcal{A}(w)$ by transfer matrices

Now the link with the usual matrix computation will become more transparent and also the reason why we had to extend it to semirings. The one to one correspondence, familiar to Scientists working in Markov processes

$$\text{valued graph} \leftrightarrow \text{matrix}$$

has been implemented by means the sparse representation given by the set of transitions (see remark (5) i). One calls the triplet (λ, μ, γ) (which characterises $\mathcal{A}$) the linear representation of $\mathcal{A}$. The matrices $\mu(a)$ are sometimes called transfer matrices.

One can prove (using a recurrence on the length of w)

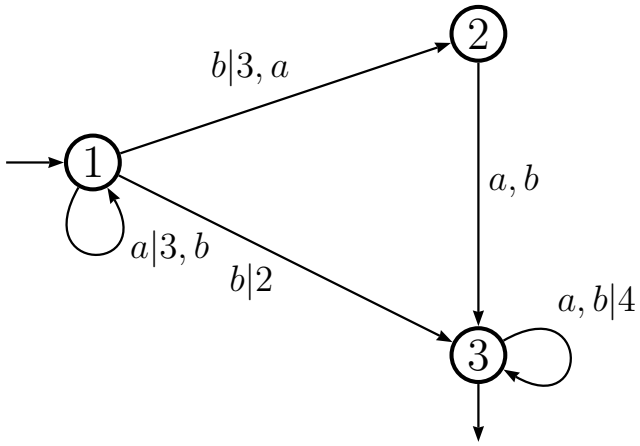


Figure 3.2: A $\mathbb{N}$-automaton. A simple letter x stands for $x|1$.

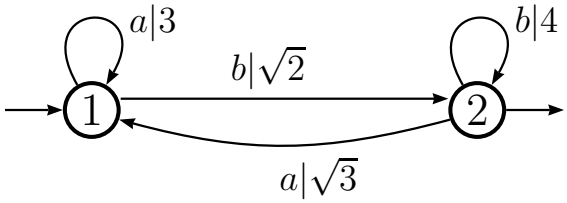


Figure 3.3: k=k automaton

Proposition 5 *Let (λ, μ, γ) the linear representation of an automaton and still denote $\mu : A^* \mapsto k^{Q\times Q}$ the morphism which extends μ to the free monoid i.e.*

$$\mu(a_1 a_2 \cdots a_n) = \mu(a_1)\mu(a_2)\cdots\mu(a_n);\ \mu(\epsilon) = I_{Q\times Q} \text{ (identity matrix)}$$

then, for $w \in A^*$

$$\mathcal{A}(w) = \lambda\mu(w)\gamma \tag{3.23}$$

Example 6 *If $k = \mathbb{R}$, then with the automaton (see figure 3.3) :*
The word $w = aab$, corresponds to the product of the corresponding matrices.

$$\mu(w) = \mu(a)\mu(a)\mu(b) = \begin{pmatrix} 3 & 0 \\ \sqrt{3} & 0 \end{pmatrix}\begin{pmatrix} 3 & 0 \\ \sqrt{3} & 0 \end{pmatrix}\begin{pmatrix} 0 & \sqrt{2} \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 0 & 9\sqrt{2} \\ 0 & 3\sqrt{6} \end{pmatrix}$$

3.3.4 Operations over automata

Let $\mathcal{AUT}_k(A)$ denote the class of automata with multiplicities in k and alphbet A (due to the ralabelling possibility, this is indeed a class and not a set). One then has an arrow $Behaviour : \mathcal{AUT}_k(A) \to k\langle\langle A\rangle\rangle$ (in a moment, we will see that the image of this arrow is not the whole set of series but a subalgebra called $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$. We define formally this set

Definition 7 *Let k be a semiring and A an alphabet. We denote $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ the set of series that are the behaviour or some automaton.*

One can here address the following questions
Q$_1$ Is $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ a semiring ?
Q$_2$ Is i closed by the star, at least of series with no constant term ?
Q$_3$ Can the operation for which $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ is closed be lifted to the level of automata ?

The answer of all these questions is positive, moreover $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ is a sub-bimodule of $k\langle\langle A\rangle\rangle$ that is to say that it is closed for the operations of right and left scalings (and some well-known others as the Hadamard, shuffle and infiltration products [67]).
For the sake of clarity (and with no loss of generality, thanks to relabelling) we will suppose that the sets of states are of the form $\{1, 2, \cdots n\}$. Let us state.

Proposition 6 *[67] Let R (resp. S) be a rational series and $\mathcal{A}_r = (\lambda^r, \mu^r, \gamma^r)$ (resp. $\mathcal{A}_s = (\lambda^s, \mu^s, \gamma^s)$) be a K-automaton which recognizes R (resp. S). Let n (resp. m) be the dimension of $\mathcal{A}_r$ (resp. $\mathcal{A}_r$) . The linear representations of the*

sum, the concatenation and the star are respectively
$R + S$:

$$\mathcal{A}_r \boxplus \mathcal{A}_s = \left(\begin{pmatrix} \lambda^r & \lambda^s \end{pmatrix}, \left(\begin{array}{c|c} \mu^r(a) & 0_{n\times m} \\ \hline 0_{m\times n} & \mu^s(a) \end{array} \right)_{a\in A}, \begin{pmatrix} \gamma^r \\ \gamma^s \end{pmatrix} \right) \ , \tag{3.24}$$

$R.S$:

$$\mathcal{A}_r \boxdot \mathcal{A}_s = \left(\begin{pmatrix} \lambda^r & 0_{1\times m} \end{pmatrix}, \left(\begin{array}{c|c} \mu^r(a) & \gamma^r\lambda^s\mu^s(a) \\ \hline 0_{m\times n} & \mu^s(a) \end{array} \right)_{a\in A}, \begin{pmatrix} \gamma^r\lambda^s\gamma^s \\ \gamma^s \end{pmatrix} \right) \ , \tag{3.25}$$

If $\lambda^s\gamma^s = 0$, S^* :

$$\mathcal{A}_s{}^{\boxtimes} = \left(\begin{pmatrix} 0_{1\times m} & 1 \end{pmatrix}, \left(\begin{array}{c|c} \mu^s(a) + \gamma^s\lambda^s\mu^s(a) & 0_{m\times 1} \\ \hline \lambda^s\mu^s(a) & 0 \end{array} \right)_{a\in A}, \begin{pmatrix} \gamma^s \\ 1 \end{pmatrix} \right) \ . \tag{3.26}$$

Proof. —
The formula for the sum (3.24) is straightforward.
To prove the product formula (3.25), let $(\lambda, \mu, \gamma) = \mathcal{A}_r \boxdot \mathcal{A}_s$. One proves by induction that

$$\mu(w) = \begin{pmatrix} \mu^r(w) & \sum\limits_{\substack{uw=w \\ v\neq 1}} \mu^r(u)\gamma^r\lambda^s\mu^s(v) \\ 0_{m\times n} & \mu^s(w) \end{pmatrix},$$

and then $\lambda\mu(w)\gamma = \sum\limits_{uv=w} \lambda^r\mu^r(u)\gamma^r\lambda^s\mu^s(v)\gamma^s = \sum\limits_{uv=w} \langle R|u\rangle\langle S|v\rangle$.

Concerning the star (3.26), let $(\lambda^*, \mu^*, \gamma^*) = \mathcal{A}_s{}^{\boxtimes}$. Again,

$$\mu^*(w) = \begin{pmatrix} M & 0_{m\times 1} \\ \sum_{n=1}^{|w|} \sum\limits_{\substack{u_1\cdots u_n=w \\ u_i\neq 1}} (\lambda_s\mu_s(u_1)\gamma_s)\cdots(\lambda_s\mu_s(u_{n-1})\gamma_s)(\lambda_s\mu_s(u_n)) & 0 \end{pmatrix},$$

where $M \in K^{m\times n}$. We then have

$$\begin{aligned} \lambda^*\mu^*(w)\gamma^* &= \sum_{n=1}^{|w|} \sum\limits_{\substack{u_1\cdots u_n=w \\ u_i\neq 1}} (\lambda_s\mu_s(u_1)\gamma_s)\cdots(\lambda_s\mu_s(u_n)\gamma_s) \\ &= \sum_{n=1}^{|w|} \langle S^n|w|\rangle = \sum_{n\geq 0} \langle S^n|w|\rangle = \langle S^*|w\rangle. \end{aligned}$$

□

Remark 6 *1. Formulas (3.24) and (3.25) provide associative laws on triplets. They can be found explicitly in [55].*

2. Formula (3.26) makes sense even when $\lambda^s\gamma^s \neq 0$ *(this fact is used for proving a density result in [67]).*

3. *Of course if* $S : (\lambda, \mu, \gamma)$ *and* $\alpha \in K$ *then* $\alpha \times S : (\alpha\lambda, \mu, \gamma)$ *and* $S \times \alpha : (\lambda, \mu, \gamma\alpha)$.

4. *For the sum* $(\mathcal{A}_r \boxplus \mathcal{A}_s)$, $\mathcal{A}_r$ *and* $\mathcal{A}_s$ *are just placed side by side.*

   The product $\mathcal{A}_r \boxdot \mathcal{A}_s$ *has the following components*

   - **States:** *The union of the sets of states of* $\mathcal{A}_r$ *and* $\mathcal{A}_s$.

   - **Inputs:** *Inputs of* $\mathcal{A}_r$.

   - **Transitions:** *Transitions of* $\mathcal{A}_r$ *and* $\mathcal{A}_s$ *and, for each letter* a, *each state* r_i *of* $\mathcal{A}_r$ *and each state* s_j *of* $\mathcal{A}_s$, *a new arc* $r_i \xrightarrow{a} s_j$ *is added with the coefficient* $(\gamma_r)_i(\lambda_s\mu_s(a))_j$.

   - **Outputs:** *The scalar product* $\lambda_s\gamma_s$ *is computed once for all and there is an output on each* q_i *with the coefficient* $(\gamma_r)_i\lambda_s\gamma_s$, *the outputs of* $\mathcal{A}_S$ *being unchanged.*

   For $\mathcal{A}^{\circledast}$, *one adds a new state* q_{n+1} *with an input and an output bearing coefficient* 1, *every coefficient* $\mu_{i,j}(a)$ *is multiplied by* $(1 + \gamma_i\lambda_j)$ *and new transitions* $q_{n+1} \xrightarrow{a} q_i$ *with coefficient* $\sum_k \lambda_k\mu_{k,i}(a)$ *(i.e. the "charge" of the state* q_i *after reading* a*) are added.*

   In case $K = \mathbb{B}$, *one recovers the classical non deterministic boolean constructions implemented in softwares such as Automate [44], AMoRE [122], Grail [142] and a part of the great project MuPAD-Combinat [111].*

As a corollary, we can state.

Corollary 1 *Let* $k,\ A$ *be as above. Then the space* $k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ *is closed under the operations* $(+, \alpha(?), (?)\alpha, ., *)$.

3.4 Kleene-Schützenberger, the jewel of Theoretical Computer Science

We are now in position to state and sketch one of the most beautiful and useful result in Theoretical Computer Science : Kleene-Schützenberger's theorem (KS).

The philosophy of this theorem is that we have now three ways to decribe some special series : rational expressions which are expressions contructed from the letters using the operations $(+, \alpha(?), (?)\alpha, ., *)$, automata with multiplicities and some specifications on the words (length, partial degrees etc..), KS theorem says us not only that it is possible to pass from one representation to the other by how to perform it.
First, we have to lay a solid frame for rational expressions.

3.4.1 Rational expressions

The universal algebra $\mathcal{E}^{cf}(A, K)$ and the constant term function

Let A be an alphabet and k a semiring. The completely free formulas for the laws $(+, \alpha(?), (?)\alpha, ., *)$ is the universal algebra generated by $A \cup \{0_{\mathcal{E}}\}$ as constants and the five preceding laws ($1_{\mathcal{E}}$ will be constructed as $0_{\mathcal{E}}^*$ and still be denoted ϵ). These expressions, by a standard argument form a set which will be denoted $\mathcal{E}^{cf}(A, K)$.

Note 1 *For example $(a^*)^* \in \mathcal{E}^{cf}(A, K)$. However, we will see later that this expression is not to be considered as valid in our setting.*

Now, we construct a pull-back of the "constant term" mapping of the series.

Definition 8 *i) The function $const : \mathcal{E}^{cf}(A, K) \to K$ is (partially) recursively defined by the rules:*

1. *If $x \in A \cup \{0_{\mathcal{E}}\}$ then $const(x) = 0_K$.*
2. *If $E, E_i \in \mathcal{E}^{cf}(A, K),\ i = 1, 2$ then*

$$const(E_1 + E_2) = const(E_1) + const(E_2),$$
$$const(E_1 \cdot E_2) = const(E_1) \times const(E_2),$$
$$const(\lambda E) = \lambda const(E), const(E\lambda) = const(E)\lambda.$$

3. *If $const(E) = 0_K$ then $const(E^*) = 1_K$.*

ii) The domain of $const$ (i.e. the set of expressions for which $const$ is defined) will be denoted $\mathcal{E}(A, K)$ or $\mathcal{E}$, for short (we then have $(0_K)^* = \epsilon \in \mathcal{E}$).
These expressions are called rational expressions.

Remark 7 *i) The set $\mathcal{E}(A, \mathbb{B})$ is a strict subset of the set of free regular expressions, but due to the (Boolean) identity $(X + \epsilon)^* = X^*$, the two sets have the same expressive power.*
ii) The class of rational expressions is a small set (in the sense of Mc Lane [119]), its cardinal is countable if A and K are finite or countable.

Sticking to our philosophy of "following the Boolean track", we must be able to evaluate rational expressions within the algebra of series. It is a straightforward verification to see that, given a mapping $\phi : A \to A^+$, there exists a unique (poly)morphism $\bar{\phi} : \mathcal{E} \to K\langle A\rangle$ which extends ϕ. In particular, let $\phi : A \to A^+$ be the inclusion mapping, then the image of $\bar{\phi}$ will be denoted $K_{\mathrm{rat}}\langle\langle A\rangle\rangle$. Notice here that $\bar{\phi}(1_{\mathcal{E}}) = \epsilon$.

3.4.2 The equivalence of Kleene-Schützenberger

Matrices of series and series of matrices

For k a semiring and A, Q a finite sets, we have a straightforward isomorphism $k\langle\langle A\rangle\rangle^{Q\times Q} \simeq k^{Q\times Q}\langle\langle A\rangle\rangle$ provided by the arrow $\Phi : k\langle\langle A\rangle\rangle^{Q\times Q} \mapsto k^{Q\times Q}\langle\langle A\rangle\rangle$

$$\langle \Phi(M)|w\rangle = (\langle M_{[q,r]}|w\rangle)_{q,r\in Q} \tag{3.27}$$

Now, let $\mathcal{A} = (Q, A, \mu, \lambda, \gamma)$ be an automaton, one has

$$\begin{aligned} Behaviour(\mathcal{A}) &= \sum_{w\in A^*} (\lambda\mu(w)\gamma)w = \\ \lambda(\sum_{w\in A^*} \mu(w))w)\gamma &= \lambda(\sum_{a\in A} \mu(a)a)^*\gamma \end{aligned} \tag{3.28}$$

Now, we recall that we had the inclusion $k_{\mathrm{rat}}\langle\langle A\rangle\rangle \subset k_{\mathrm{rec}}\langle\langle A\rangle\rangle$ as the latter contains A and is closed under $(+, \alpha(?), (?)\alpha, ., *)$. To get the converse, is suffices to prove that the coefficients of $(\sum_{a\in A} \mu(a)a)^*$ are in $k_{\mathrm{rat}}\langle\langle A\rangle\rangle$, which will be a consequence the following contruction.

Recursive computation of the star

In this paragraph, we have followed [70]

Let $M \in k^{Q\times Q}$ be given by

$$M = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

where $a_{11} \in k^{Q_1\times Q_1}$, $a_{12} \in k^{Q_1\times Q_2}$, $a_{21} \in k^{Q_2\times Q_1}$ and $a_{22} \in k^{Q_2\times Q_2}$ such that $Q_1 + Q_2 = Q$. Let $N \in k^{Q\times Q}$ given by

$$N = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}$$

with

$$A_{11} = (a_{11} + a_{12}{a_{22}}^* a_{21})^* \tag{3.29}$$
$$A_{12} = {a_{11}}^* a_{12} A_{22} \tag{3.30}$$
$$A_{21} = {a_{22}}^* a_{21} A_{11} \tag{3.31}$$
$$A_{22} = (a_{22} + a_{21}{a_{11}}^* a_{12})^* \tag{3.32}$$

We have the following theorem.

Theorem 3 *If the right hand sides of formulas (3.29), (3.30), (3.31) and (3.32) are defined, the matrix M admits N as a right star.*

Proof. Suppose, without loss of generality, that $Q = [1, n]_{\mathbb{N}};\ n = p + q;\ Q_1 = [1, p]_{\mathbb{N}};\ Q_2 = [p+1, p+q]_{\mathbb{N}}$. We have to show that N is a solution of the equation $My + 1_{n\times n} = y$. By computation, one has

$$MN + 1 = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} + \begin{pmatrix} 1_{p\times p} & 0_{p\times q} \\ 0_{q\times p} & 1_{q\times q} \end{pmatrix}$$

$$= \begin{pmatrix} a_{11}A_{11} + a_{12}A_{21} + 1_{p\times p} & a_{11}A_{12} + a_{12}A_{22} \\ a_{21}A_{11} + a_{22}A_{21} & a_{21}A_{12} + a_{22}A_{22} + 1_{q\times q} \end{pmatrix}$$

where $0_{p\times q}$ is the zero matrix in $k^{p\times q}$. We verify the relations (3.29), (3.30), (3.31) and (3.32) by:

$$\begin{aligned} a_{11}A_{11} + a_{12}A_{21} + 1_{p\times p} &= a_{11}A_{11} + a_{12}a_{22}^* a_{21}A_{11} + 1_{p\times p} = \\ A_{11}(a_{11} + a_{12}{a_{22}}^* a_{21}) + 1_{p\times p} &= A_{11} \end{aligned}$$

$$\begin{aligned} a_{11}A_{12} + a_{12}A_{22} &= a_{11}{a_{11}}^* a_{12}A_{22} + a_{12}A_{22} = \\ (a_{11}{a_{11}}^* + 1)a_{12}A_{22} &= {a_{11}}^* a_{12}A_{22} = A_{12} \end{aligned}$$

$$\begin{aligned} a_{21}A_{11} + a_{22}A_{21} &= a_{21}A_{11} + a_{22}{a_{22}}^* a_{21}A_{11} = \\ (1 + a_{22}{a_{22}}^*)a_{21}A_{11} &= {a_{22}}^* a_{21}A_{11} = A_{21} \end{aligned}$$

$$\begin{aligned} a_{21}A_{12} + a_{22}A_{22} + 1_{q\times q} &= a_{21}{a_{11}}^* a_{12}A_{22} + a_{22}A_{22} + 1_{q\times q} = \\ (a_{22}a_{21}{a_{11}}^* a12)A_{22} + 1_{q\times q} &= A_{22} \end{aligned}$$

□

Remark 8 *i) In [94] and [143], similar formulas are expressed for the computation of the inverse of matrices when k is a division ring (this can be extended to the case of rings).*
It must be emphasized that the converse of Theorem (3), of course, does not hold as shows the example below (the coefficients are taken in a ring and here the star is unique).

$$M = \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix} \quad M^* = \begin{pmatrix} 2 & -1 \\ 1 & 0 \end{pmatrix} \tag{3.33}$$

However, if the formulas are defined at each step of the computation (see below for a formalization of this), it provides a recursive way to compute a star of a matrix.
ii) Similar formulas can be stated in the case of a left star. The matrix N is the left star of M with

$$\begin{aligned} A_{11} &= (a_{11} + a_{12}{a_{22}}^* a_{21})^* \\ A_{12} &= A_{11} a_{12} {a_{22}}^* \\ A_{21} &= A_{22} a_{21} {a_{11}}^* \\ A_{22} &= (a_{22} + a_{21}{a_{11}}^* a_{12})^* \end{aligned}$$

iii) When all their terms are defined, formulas above (as well as (3.29), (3.30), (3.31) and (3.32)) are valid with matrices of any size with any block partitionning. Matrices of even size are often, in practice, partitionned into square blocks but, for matrices with odd dimensions, the approach called dynamic peeling is applied. More specifically, let $M \in k^{n \times n}$ a matrix given by

$$M = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

where $n \in 2\mathbb{N} + 1$. The dynamic peeling [99] consists of cutting out the matrix in the following way: a_{11} is a $(n-1) \times (n-1)$ matrix, a_{12} is a $(n-1) \times 1$ matrix, a_{21} is a $1 \times (n-1)$ matrix and a_{22} is a scalar.

If desired, formulation of Theorem (3) can be seen as recursive in essence. In this respect, it implies that stars of submatrices could be already computed by the same scheme. This type of computation will be formalized by the notion of *admissible tree of computation* which we describe below.
Let Q be a finite set and $\mathcal{A}[Q]$ be the set of binary trees with leaves in Q. It can be defined by the grammar

$$\mathcal{A}[Q] = Q + (\mathcal{A}[Q], \mathcal{A}[Q]) \tag{3.34}$$

or, if one prefers a graded version

$$\left\{ \begin{array}{lcl} \mathcal{A}_1[Q] & = & Q \\ \mathcal{A}_n[Q] & = & \sum_{i+j=n}(\mathcal{A}_i[Q], \mathcal{A}_j[Q]) \text{ if } n \geq 2. \end{array} \right.$$

Now, the list of leaves of a tree $T \in \mathcal{A}_n[Q]$ is a word $lv(T) \in Q^n$ defined by

$$\left\{ \begin{array}{lcl} lv(T) & = & T \text{ if } T \in \mathcal{A}_1[Q] \\ lv(T) & = & lv(T_1)lv(T_2) \text{ (concatenation) if } T = (T_1, T_2). \end{array} \right.$$

The set of leaves of T is then $alph(lv(T))$, where $alph(w)$ is classically the alphabet of the word w.
We will say that $T \in \mathcal{A}[Q]$ is an admissible tree of computation for $M \in k^{Q\times Q}$ if

1. the word $lv(T)$ is standard (with no repetition) and contains all the indices (i.e. $|lv(T)| = |Q|$ and $alph(lv(T)) = Q$)

2. if $|Q| = 1$ (thus $M \in k^{Q\times Q} \simeq k$ is a scalar), M admits a star in $k^{Q\times Q}$

3. - if $|Q| \geq 2$, set $T = (T_1, T_2)$ and $Q_i = lv(T_i)$ $(i = 1, 2)$ then T_1 is admissible for the submatrix $M\Big|_{Q_1\times Q_1}$ and T_2 is admissible for the submatrix $M\Big|_{Q_2\times Q_2}$.
   - formulas above (3.29), (3.30), (3.31) and (3.32) are defined for the partitionning $Q = Q_1 + Q_2$.

The conditions above assures that the recursive computation of Theorem (3) is defined at each step and, in this case, we will say that the star of M is computed along the (admissible) tree of computation T.

Formulas (3.29), (3.30), (3.31) and (3.32) allow to prove the rationality of the star function on matrices (see [69] for deep consequences of this fact).

Proposition 7 *Let $M \in k_{\mathrm{rat}}\langle\langle A\rangle\rangle^{Q\times Q}$ a matrix of proper series. Then the coefficients of M^* are rational functions of the coefficients of M.*

Now, we have the

Theorem 4 *(Kleene-Schützenberger's Theorem). —*

$$k_{\mathrm{rat}}\langle\langle A\rangle\rangle = k_{\mathrm{rec}}\langle\langle A\rangle\rangle$$

3.5 Tables

The following is intended to be a contribution in the area of what could be called *efficient algebraic structures* or *efficient data structures*. In fact, we define and construct a new data structure, the tables, which are special kinds of two-raws arrays. The first raw is filled with words (or with elements taken in a semigroup) and the second with some coefficients. This structure generalizes the (finite) k-sets of Eilenberg [72], it is versatile (one can vary the letters, the words and the coefficients), easy implemented and fast computable. Varying the scalars and the operations on them, one can obtain many different structures and, among them, semirings. Examples will be provided and worked out in full detail.

Here, we present a new semiring (with several semiring structures) which can be applied to the necessity of automatic processing multi-agents behaviour problems. The purpose of this account is to present also the basic elements of this new structures from a combinatorial point of view. These structures present bunch of properties. They will be endowed with several laws namely : Sum, Hadamard product, Cauchy product, Fuzzy operations (min, max, complemented product). Two groups of applications are presented.

The first group is linked to the process of "forgetting" information in the tables and then obtaining, for instance, a memorized semiring.The latter is specially suited to solve the *shortest path with addresses* problem by repeated squaring over matrices with entries in this semiring.

The second, linked to multi-agent systems, is announced by showing a methodology to manage emergent organization from individual behaviour models.The bases of this methodology will be developed in the following chapter from automata structures.

3.5.1 Tables and operations on tables

The input alphabet being set by the automaton under consideration, we will here rather focus on the definition of semirings providing transition coefficients. For convenience, we first begin with various laws on $\mathbb{R}_+ := [0, +\infty[$ including

1. $+$ (ordinary sum)
2. $\times$ (ordinary product)
3. min (if over $[0, 1]$, with neutral 1, otherwise must be extended to $[0, +\infty]$ and then, with neutral $+\infty$) or max

4. $+_a$ defined by $x +_a y := log_a(a^x + a^y)$ $(a > 0)$
5. $+_{[n]}$ (Hölder laws) defined by $x +_{[n]} y := \sqrt[n]{x^n + y^n}$
6. $+^s$ (shifted sum, $x +^c y := x + y - 1$, over whole $\mathbb{R}$, with neutral 1)
7. $\times^c$ (complemented product, $x + y - xy$, can be extended also to whole $\mathbb{R}$, stabilizes the range of probabilities or fuzzy $[0, 1]$ and is distributive over the shifted sum)

A table T is two-rows array, the first row being filled with words taken in a given free monoid (see [68, 117]). The set of words which are present in the first row will be called the *indices* of the table ($I(T)$) and for the second row the *values* or (*coefficients*)of the table. The order of the columns is not relevant. Thus, a table reads

$$\begin{cases} indices & \text{set of words } I(T) \\ values & \text{bottom row } V(T) \end{cases} \tag{3.35}$$

The laws defined on tables will be of two types :
pointwise type (subscript $_p$) and convolution type (subscript $_c$).
Now, we can define the pointwise composition (or product) of two tables.
Let us consider, two tables $T_1,\ T_2$ and a law $*$

$$T_1 = \begin{array}{c|c|c|c} u_1 & u_2 & \cdots & u_k \\ \hline p_1 & p_2 & \cdots & p_k \end{array} \quad \text{and} \quad T_2 = \begin{array}{c|c|c|c} v_1 & v_2 & \cdots & v_l \\ \hline q_1 & q_2 & \cdots & q_l \end{array}$$

then $T_1 \boxtimes_p T_2$ is defined by $T_i[w]$ if $w \in I(T_i)$ and $w \notin I(T_{3-i})$ and by $T_1[w] * T_2[w]$ if $w \in I(T_1) \cap I(T_2)$

In particular one has $I(T_1 \boxtimes_p T_2) = I(T_1) \cup I(T_2)$.

Note 2 *i) At this stage one do no need any neutral. The structure automatically creates it (see* algebraic remarks *below for full explanation).*
ii) The above is a considerable generalization of an idea appearing in [42], aimed only to semirings with units.

For convolution type, one needs two laws, say $\oplus, \otimes$, the second being distributive over the first, i.e. identically

$$\begin{array}{rcl} x \otimes (y \oplus z) & = & (x \otimes y) \oplus (x \otimes z) \text{ and} \\ (y \oplus z) \otimes x & = & (y \otimes x) \oplus (z \otimes x) \end{array} \tag{3.36}$$

(see http://mathworld.wolfram.com.Se miri ng.ht ml).

The set of indices of $T_1 \boxtimes_c T_2$ ($I(T_1 \boxtimes_c T_2)$) is the concatenation of the two (finite) langages $I(T_1)$ and $I(T_2)$ i.e. the (finite) set of words

$$I(T_1)I(T_2) = \{uv\}_{(u,v)\in I(T_1)\times I(T_2)}. \tag{3.37}$$

then, for $w \in I(T_1)I(T_2)$, one defines

$$T_1 \otimes_c T_2[w] = \bigoplus_{uv=w} \Big(T_1[u] \otimes T_2[v] \Big) \tag{3.38}$$

the interesting fact is that the constructed structure (call it $\mathcal{T}$ for tables) is then a semiring $(\mathcal{T}, \oplus_p, \otimes_c)$ (provided $\oplus$ is commutative and - generally - without units, but this is sufficient to perform matrix computations). There is, in fact no mystery in the definition (3.37) above, as every table can be decomposed in elementary bits

$$T_1 = \begin{array}{|c|c|c|c|}\hline u_1 & u_2 & \cdots & u_k \\ \hline p_1 & p_2 & \cdots & p_k \\ \hline \end{array} = \bigoplus_{i=1}^{k} \begin{array}{|c|}\hline u_i \\ \hline p_i \\ \hline \end{array} \tag{3.39}$$

one has, thanks to distributivity, to understand the convolution of these indecomposable elements, which is, this time, very natural

$$\begin{array}{|c|}\hline u_1 \\ \hline p_1 \\ \hline \end{array} \bigotimes_c \begin{array}{|c|}\hline u_2 \\ \hline p_2 \\ \hline \end{array} := \begin{array}{|c|}\hline u_1 u_2 \\ \hline p_1 \times p_2 \\ \hline \end{array} \tag{3.40}$$

3.5.2 Why semirings ?

As was emphasized in the beginning of this chapter, the structure of semiring is the most efficient for performing computations over paths. Indeed, in many applications, we have to compute the weight of paths in wome weighted graphs (shortest path problem, enumeration of paths, cost computations, automata, transducers to cite only a few) and the computation goes with two main rules: multiplication in series (i.e. along a path), and addition in parallel (if several paths are involved).
This paragraph is devoted to showing that in these conditions, the axioms of Semirings are by no means arbitrary and in fact unavoidable. A weighted graph is an oriented graph together with a *weight* mapping $\omega : A \mapsto K$ from the set of the arrows (A) to some set of coefficients K, an arrow is drawn with its weight (cost) above as follows $a = q_1 \stackrel{\alpha}{\to} q_2$.
For such objects, one has the general conventions of graph theory.

- $t(a) := q_1$ (*tail*)
- $h(a) := q_2$ (*head*)

- $w(a) := \alpha$ (*weight*).

Recall that a *path* is a sequence of arrows $c = a_1 a_2 \cdots a_n$ such that $h(a_k) = t(a_{k+1})$ for $1 \leq k \leq n-1$. The preceding functions are extended to paths by $t(c) = t(a_1),\ h(c) = h(a_n)$, $w(c) = w(a_1)w(a_2)\cdots w(a_n)$ (product in the set of coefficients).

For example with a path of length 3 and ($k = \mathbb{N}$),

$$u = p \xrightarrow{2} q \xrightarrow{3} r \xrightarrow{5} s \tag{3.41}$$

one has $t(u) = p,\ h(u) = s,\ w(u) = 30$.

As was stated above, the (total) weight of a set of paths with the same head and tail is the sum of the individual weights. For instance, with

$$\mathbf{q}1 \quad \begin{matrix} \xrightarrow{\alpha} \\ \xrightarrow{\beta} \end{matrix} \quad \mathbf{q}2 \tag{3.42}$$

the weigth of this set of paths est $\alpha + \beta$. From the rule that the weights multiply in series and add in parallel one can derive the necessity of the axioms of the semirings. The following diagrams shows how this works.

Diagram	Identity
$p \begin{matrix} \xrightarrow{\alpha} \\ \xrightarrow{\beta} \\ \xrightarrow{\gamma} \end{matrix} q$	$\alpha + (\beta + \gamma) = (\alpha + \beta) + \gamma$
$p \begin{matrix} \xrightarrow{\alpha} \\ \xrightarrow{\beta} \end{matrix} q$	$\alpha + \beta = \beta + \alpha$
$p \xrightarrow{\alpha} q \xrightarrow{\beta} r \xrightarrow{c\|\gamma} s$	$\alpha(\beta\gamma) = (\alpha\beta)\gamma$
$p \begin{matrix} \xrightarrow{\alpha} \\ \xrightarrow{\beta} \end{matrix} q \xrightarrow{\gamma} r$	$(\alpha + \beta)\gamma = \alpha\gamma + \beta\gamma$
$p \xrightarrow{\alpha} q \begin{matrix} \xrightarrow{\beta} \\ \xrightarrow{\gamma} \end{matrix} r$	$\alpha(\beta + \gamma) = \alpha\beta + \alpha\gamma$

these identities are familiar and bear the following names:

Line	Name
I	Associativity of $+$
II	Commutativity of $+$
III	Associativity of $\times$
IV	Distributivity (right) of $\times$ over $+$
V	Distributivity (left) of $\times$ over $+$

3.5.3 Total mass

The total mass of a table is just the sum of the coefficients in the bottom row. One can check that

$$\begin{aligned} mass(T1 \oplus T2) &= mass(T1) + mass(T2); \\ mass(T1 \otimes T2) &= mass(T1) \dot{m} ass(T2) \end{aligned} \tag{3.43}$$

this allows, if needed, stochastic conditions.

3.5.4 Algebraic remarks

We have confined in this paragraph some proofs of structural properties concerning the tables. The reader may skip this section with no serious harm.
First, we deal with structures with as little as possible requirements, i.e. *Magmas* and *Semirings*. For formal definitions, see [80] and

`http: encyclopedia.thefreedictionar y.co m.Ma gma%2 0cat egory`

`http://mathworld.wolfram.com.Se mirin g.ht ml`

Proposition 8 *(i) Let $(S, *)$ be a magma, Σ an alphabet, and denote $T[S]$ the set of tables with indices in Σ^* and values in S. Define $\boxtimes_p$ as in (3.5.1). Then*
i) The law $\boxtimes$ is associative (resp. commutative) iff $*$ is. Moreover the magma $(\mathcal{T}[S], \boxtimes)$ always possesses a neutral, the empty table (i.e. with an empty set of indices).
ii) If $(K, \oplus, \otimes)$ is a semiring, then $(\mathcal{T}_K, \oplus, \otimes)$ is a semiring.

Proof (Sketch) Let $S_{(1)}$ the magma with unit built over $(S \cup \{e\})$ by adjunction of a unit. Then, to each table T, associate the (finite supported) function $f_T : \Sigma^* \mapsto S_{(1)}$ defined by

$$f_T(w) = \begin{cases} T[w] & \text{if } w \in I(T) \\ e & \text{otherwise} \end{cases} \tag{3.44}$$

then, check that $f_{T_1 \boxtimes_p T_2} = f_{T_1} \boxtimes_1 f_{T_2}$ (where $\boxtimes_1$ is the standard law on $S_{(1)}^{(\Sigma^*)}$) and that the correspondence is a isomorphism. Use a similar technique for the point (ii) with $K_{0,1}$ the semiring with units constructed over K and show that the correspondence is one-to-one and has $K_{0,1}\langle\Sigma\rangle$ as image.

Note 3 *(i) Replacing Σ^* by a simple set, the (i) of proposition above can be extended without modification (see also K-subsets in [72]).*

(ii) Pointwise product can be considered as being constructed with respect to the (Hadamard) coproduct $c(w) = w \otimes w$ whereas convolution is w.r.t. the Cauchy coproduct

$$c(w) = \sum_{uv=w} u \otimes v \tag{3.45}$$

(see [67]).

3.6 Applications of operations on tables

3.6.1 Specialisations and images

1. **Multiplicities, Stochastic and Boolean**. —

   Whatever the multiplicities, one gets the classical automata by emptying the alphabet (setting $\Sigma = \emptyset$). For stochastic, one can use the total mass to pin up outgoing conditions.

2. **Memorized Semiring**. —

   We explain here why the memorized semiring, devised at first to perform efficient computations on the shortest path problem with memory (of addresses) can be considered as an image of a "table semiring" (thus proving without computation the central property of [103]).
   Let $\mathcal{T}$ be here the table semiring with coefficients in $([0, +\infty], min, +)$. Then a table

   $$T = \begin{array}{c|c|c|c|c} u_1 & \cdots & u_k & \cdots & u_n \\ \hline l_1 & \cdots & l_k & \cdots & l_n \end{array} \tag{3.46}$$

   can be written so that $l_1 = \cdots = l_k < l_m$ for $m > k$ (this amounts to say that the set where the minimum is reached is $\{u_1, u_2 \cdots u_k\}$). Then, to such a table, one can associate $\phi(T) := [\{u_1, u_2 \cdots u_k\}, l_1]$ in the memorized semiring. It is easy to check that ϕ transports the laws and the neutrals and obtain the result.

3.6.2 Application to evolutive systems

Tables are structured as semirings and are flexible enough to recover and amplify the structures of automata with multiplicities and transducers. They give operational tools for modelling agent behaviour for various simulations in the domain of distributed artificial intelligence [15]. The algebraic structures associated with tables values are very promising to define automatic computations with respect to the evolution of agents behaviour during simulation as we will explain in the

following chapter from automata structures.

One of our aims is to compute dynamic multiagent systems formations which emerge from a simulation. The use of table operations delivers calculable automata aggregate formation. With the definition of adapted operators coming from genetic algorithms, we are able to represent evolutive behaviors of agents and so evolutive systems [15]. Thus, tables and memorized semiring are promizing tools for this kind of implementation which leads to model complex systems in many domains. Let us consider the bases of the computation of evolutive agent behaviour in the following chapter.

Chapter 4

Evolutive agent behaviour modeling based on genetic automata

Contents

This chapter is the central component of the work presented in this study. We show how algebraic structures described in the previous chapter can be the basis of complex systems modeling within agent based approaches.

4.1 Introduction

In the previous last chapters, we first presented general concepts for complex systems, based on interacting and adaptative components. In the end of this first chapter, we presented some previous works based on formal models and computational structures to show how these concepts can be implemented. In the second chapter, we have presented in detail the part of the theory which is relevent to ower need. This showed the power of such formal models and computational structures. Semiring is the central concept which must be understood as a powerful operating structure.

In this chapter, we deal on modeling aspect for complex systems, in terms of operative vision which must allow to represent the two bases of complex systems entities: their interacting processes which can lead to self-organized structures and their evolutive or adaptive behaviour which can be seen as the feed-back of the system over its components.

We focus our attention on the fact that the self-organization simulation needs to manage dynamics organizations which can be automatically created during the simulation, but which can disappear because of dynamic modifications of some interactions between specific entities.

We are looking for analysis and modelization which helps us to understand such dynamic reorganizations through multiple interacting entities. Determining the essential factors for such phenomena, give us a more precise modelization in terms of micro-macro exchanges. Such exchanges are well known to be essential for the whole system behaviour evolution. Aquatic complex flux is an example of such a phenomenon [18, 134].

So, we need automatic computations which lead to manage dynamic organizations. A adapted formalism must be used and many operations must be defined on it. We suggest an agent description based on automata implementation as described in the following. In this chapter, we follow the formalism initially proposed in [15, 16].

Finite state automata are tools on which many interesting operations can be

defined [147, 45, 67] We suggest in the following, the use of some species of automata, well-adapted to agent processing.

4.2 A general framework for agent description in operating way

We draw a general model for agent behaviour representation, as operating entities which composed a complex system. We first define the general context of the agent description used in our work and we present the different kinds of automata-based models used to describe them.

4.2.1 Basic agent description

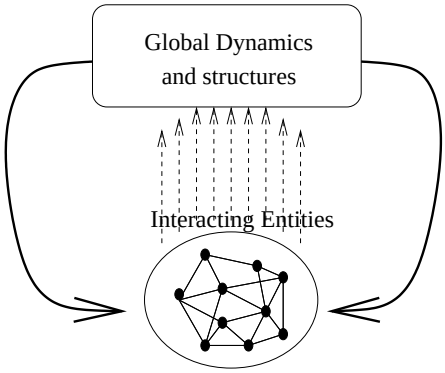
In first chapter, we describe complex systems as composed of entities which can be mainly described by two aspects. The first one concerns the interaction network to whom they believe and which is able to generate emergent dynamics or structures as schematically described in the part (a) of the figure 4.1. The second concerns the evolutive aspect of the dynamicity of the system which lead to generate adaptive behaviours of the entities as a kind of feed-back process of the system on its constitutives components.

The interacting entities network as described in the part (b) of the figure 4.1 lead to each entity to perceive informations or actions from other entities or from the whole system and to act itself.

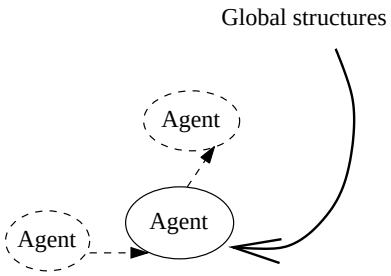
A well-adapted modeling consist of using agent-based representation which is composed of the entity called agent as an entity which perceives and acts on an environment, using a autonomous behaviour as described in the part (c) of the figure 4.1.

To compute a simulation composed of such entities, we need to describe the behaviour of each agent. This one can be schematically describe using internal states and transition processes between these states, as described in the part (d) of the figure 4.1.

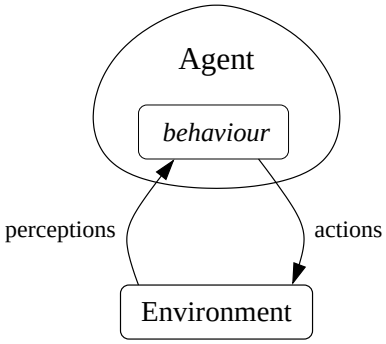
There are several definitions of “agents” or “intelligent agents” according to their behaviour specificities [75, 161]. Their autonomy means that the agents try to satisfy a goal and execute actions, optimizing a satisfaction function to reach it.



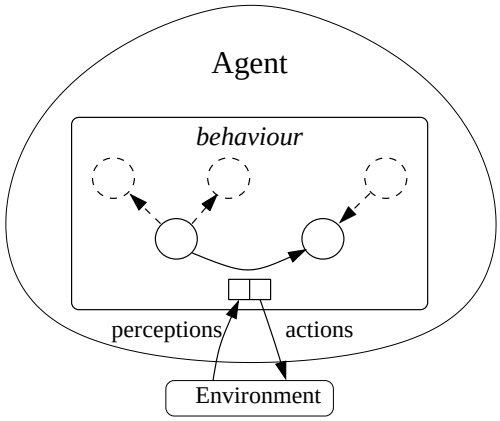
(a) Global complex system



(b) Interacting entities network



(c) Agent−based model for entity



(d) automata−based model for agent behaviou

Figure 4.1: Multi-scale complex system description: from global to individual models

For agents with high level autonomy, specific actions are realized even when no perception are detected from the environment. To represent the processing of this deliberation, different formalisms can be used and a behaviour decomposed in internal states is a effective approach. Finally, when many agents operate, the social aspects must also be taken into account. These aspects are expressed as communications throw 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 that.

4.2.2 A basic review on automata based description for agent modeling

Finite state automata play an important role in the design of many applications and in several areas of computer science: theory of languages [147, 45, 10, 43], systems and networks, parsing, images compression [56, 55], genomic research.

Different representations of agent behaviour, seen as processes, are compared in [75]. Many of them are based on classic automata or on their extensions. In fact, classical representations for agents are Petri nets and register automata like ATN (Augmented Transition Net). Petri nets and ATN are more expressive than boolean finite state automata. Moreover, Petri nets allow the expression of parallel aspects of agent behaviour in a multi-agent system that classic finite states automata are not able to do. An ATN can be defined as a finite state automaton with registers changing when conditions are checked, and with transitions between states labelled by actions [85]. Petri nets and ATN make use of memorization mechanisms and therefore miss some properties of finite state automata. Beside these classical representations, an automaton with multiplicities, as a formal pattern for an agent behaviour, preserves classical rational operations on automata [66] and gives, with the output associated, a formalism to represent the actions of an agent. Moreover, varying the scalars (e.g. change of semiring) one can reach a wealth of an expected innovative outputs.

4.2.3 An agent modeling framework based on automata with multiplicities

The formalism adopted in this work for the representation of the agent behaviour induced by perceptions and actions is automata with outputs. The finite inputs al-

phabet corresponds to the perceptions set. The finite outputs alphabet corresponds to the actions set. From this output alphabet, we build a semiring corresponding to the polynomials over this output alphabet.

As described in chapter 3, an automaton with multiplicities over a finite alphabet Σ and a semiring K is a $5-tuple$ $(\Sigma, Q, I, T, \delta)$, with Q a finite set of states and I, T, δ being mappings such that

- $I : Q \rightarrow K$,
- $T : Q \rightarrow K$,
- $\delta : Q \times \Sigma \times Q \rightarrow K$.

I (resp. T) is the set of initial states (resp. final states) and δ is the transition function.

Such a structure is particularly useful when transitions have outputs: to each input word of Σ^* is associated an output element of K. Thus the behaviour of an automaton with multiplicities is a series $S = \Sigma_{w \in \Sigma^*} \langle S|w \rangle w$ where $\langle S|w \rangle$ is the output element associated to the input word.

4.3 Deterministic agent modeling using transducers

We describe in this section, how a specifical representation of automata with multiplicities can be used to represent a deterministic agent behaviour which is driven by perceptions that induce internal state transitions and can lead to specific action from the agent.

4.3.1 General model description

So, based on this first description, transducers as finite state automata is the adapted formalism.

As far as agents are concerned, we consider the set Σ of the agent perceptions and the set Π of its elementary actions. Let us consider $K = A\langle\Pi\rangle$ as the set of polynomials of these elementary actions with coefficients in A (a commutative semiring), as the set of integers, for example.

To each transition by a given symbol of Σ is associated an output, which is some symbol of K. Thus K is the set of all possible sequences of actions,

equipped with sum and concatenation product operations. An agent can be represented by a tranducer which is a particular form of automaton with multiplicities over the alphabet Σ and the semiring $(K, \oplus, \odot)$. Thus the behaviour of an agent is $S = \Sigma_{w \in \Sigma^*} \langle S|w \rangle w$ where $\langle S|w \rangle$ is the output action induced by the successive perceptions a_1, ..., a_n, such that $w = a_1 ... a_n$.

The formalism described before is really suited for reactive behaviour describing actions induced from perceptions. To take into account some deliberative behaviours, we have to complete this description. We can use meta-automata where the states are themselves some abstract processus. They can be associated to internal automata for exemple, which model deliberative mecanisms from another level of description that the one described by the basic transducer. Multi-level processes are so used, each one is relevant for specific agent signification or interpretation. Automatic computations in terms of organisational or social aspects between agents are defined for each level. In such a way, the organizations deduced from these operations are semantically relevant in the level of description where these automatic computations are made.

In fact, usually, the agent framework able to manage dynamic organisations are often based on reactive architecture for the agent behaviour and not on cognitive architecture.

Another precision has to be given about the automata formalism which uses initial state and final ones. What are the signification for the agent process? In the one hand, this can describe a life cycle where each agent is born in initial state and dies in final one. However, in many problems, agent are represented by a permanent processus, birth and death are irrelevant in this formal descrition. In such model, we have to introduce initial and final states to identify all the actions sequences allowing the agent behaviour to be in specific states (final ones) from another specific one (initial state). We describe in the following a metric tool to compare these different sequences.

4.3.2 Eco-Agent as basic interacting entity behaviour model

Using our formalism, we can show that transition graph of the Eco-problem-solving described in [75] can be expressed as transducer.

An eco-agent is characterized by:

- an internal state which is one of the following: to be satisfied (S), to search satisfaction (SS), to flee (F) or to search to flee (SF); the initial state is

(SS) and the final state is (S); the final state of satisfaction corresponds to the goal of the agent;

- percepting functions which are:
  - to be attacked (by other agents): event denoted A;
  - to perceive some intruder (such as other agents preventing it to be satisfied): event denoted I.
- elementary actions which are:
  - to flee (TF);
  - to satisfy itself (TS);
  - to attack other agents (TA);
  - to do nothing ($N1$).

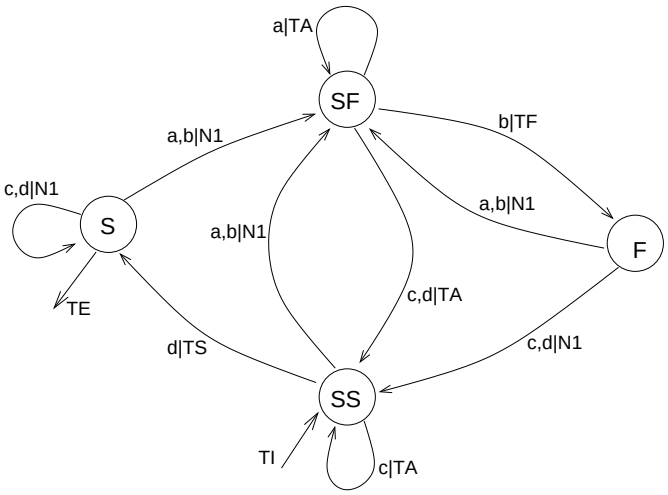


Figure 4.2: Tranducer $\mathcal{A}_x$ for Eco-Agent behaviour

This agent behaviour can so be reprented by the tranducer (see figure 4.2) $\mathcal{A}_x = (\Sigma, \Pi, Q, I, T, \delta)$, such that :

- Σ is the finite alphabet corresponding to transition conditions which are based on agent perception. There we have $\Sigma = \{a, b, c, d\}$, where $a = (A, I)$, $b = (A, \overline{I})$, $c = (\overline{A}, I)$, $d = (\overline{A}, \overline{I})$.
- Q is the set of states: $Q = \{S, SS, F, SF\}$

- Let $\Pi = \{TA, TF, TS, TI, TE, N0, N1\}$ be the set of elementary agent actions described before and completed with TI the agent initialization, TE the end of the agent actions, $N0$ the absence of action (in fact no transition) and $N1$ an action with no incidence (such as the F to SF transition in Fig. 4.2). We consider the semiring $(K, \oplus, \odot)$ as defined above, where $K = \mathbb{N}\langle\Pi\rangle$, the set of polynomials over Π, with integer coefficients. Notice that the action $N0$ (resp. $N1$) is the neutral element of the $\oplus$ (resp. $\odot$) operation.

- $I : Q \to K$ is defined by $I(SS) = TI$ and $I(S) = I(F) = I(SF) = N0$.

- $T : Q \to K$ is defined by $T(S) = TE$ and $T(SS) = T(F) = T(SF) = N0$.

- δ is defined by the transitions of the automaton described in Fig. 4.2.

The Eco-Agent behaviour give an interesting distributed method for problems resolution. In [63], this model is used on a population of agents which collectively solve the N-puzzle problem.

4.3.3 An application to self-organized fluid flow simulation

We present in this section how the eco-resolution automata with multiplicities has been used for a study concerning a simulation leading to implement self-organized processes inside a fluid flow.

Initially, this study concerns a research topic about aquatic ecosystem modeling [21, 22]. The goal is to describe a complex fluid flow using two-scales level:

- Elementary vortex particles evolve following vortex methods which is a discrete formulation in terms of vorticity of the fluid flow based on Navier-Stokes equations [53, 114].

- Emergent vortex structures that appear in the fluid and must be automatically detected during the simulation.

Such multi-scale simulation must deal with two essential aspects:

- The management of the emergent detected structures. How to represent such emergent self-organized systems inside and during the simulation? How to make them involve? How to manage the entropy inside the self-organized structure? Does the entropy or the order have to increase or decrease? All these questions must be treated with pertinent phenomena, dynamically and during the simulation.

- The micro-macro interactions. Multi-scale simulation has to manage this important aspect which is the heart of the flux transfert throught the structures.

In the following, we explain how the structures are detected with geometrical processes and then we explain the management of multi-scale simulation based on an eco-agent automaton which is the heart of the self-organisation evolution.

Self-organized structures detection inside fluid flow

The fluid flow are represented with a set of rotational particles which each have a rotational sense (positive or negative).

The self-organized structures which we deal with in this study is a cluster of same rotational particles whose shape is physically pertinent, e.g. near a ellipsoid shape and which does not contain inside its shape particles of opposite rotation.

The self-organized structures are detected using some steps which are summerized inside the figure 4.3: a Delaunay triangulation is computed and allows to build a minimal spanning tree where non pertinent edges are broken (e.g. the edges which lie between different rotational particles or which are too long). We obtain a set of clusters. The convex hull is then computed for each cluster. The self-organized structures that we try to detect are vortex and their natural physical shape is an ellipsoid. So we try to identify each convex hull with an ellipse, using some specific methods. If some morphological constraints are respected, then the cluster is validated and the self-organized vortex structure is created.

Automata-based modeling for structure evolution and stability simulation

Once the self-organized structures are detected, we have to modelize them and create the associated entities inside and during the simulation. To make that, we represent each structure by an eco-agent which has to manage the coherence of the self-organized structure.

Using eco-agents as behaviour models for a specific problem, consists in the definition of the two elementary perceptions: to be attacked and to perceive intruders, and the four elementary actions: to do nothing, to attack, to escape and to satisfy. We now explain how these notions have to be defined for the structure evolution:

- To perceive intruders means that the structure intercepts another or is on his close trajectory (e.g. inside a specific interaction zone). The figure 4.4

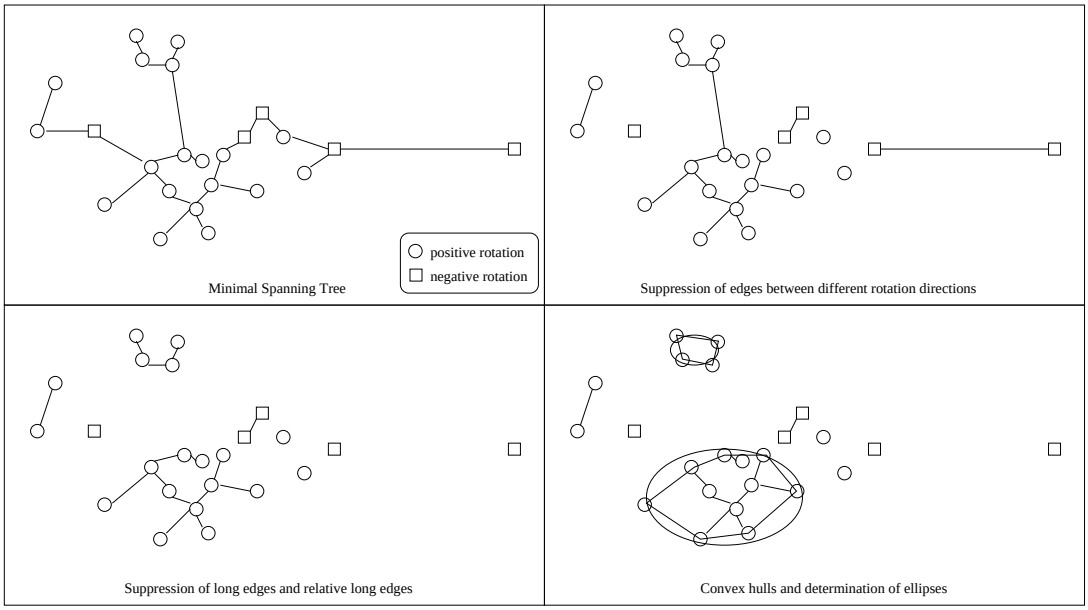


Figure 4.3: Self-organized structures detection

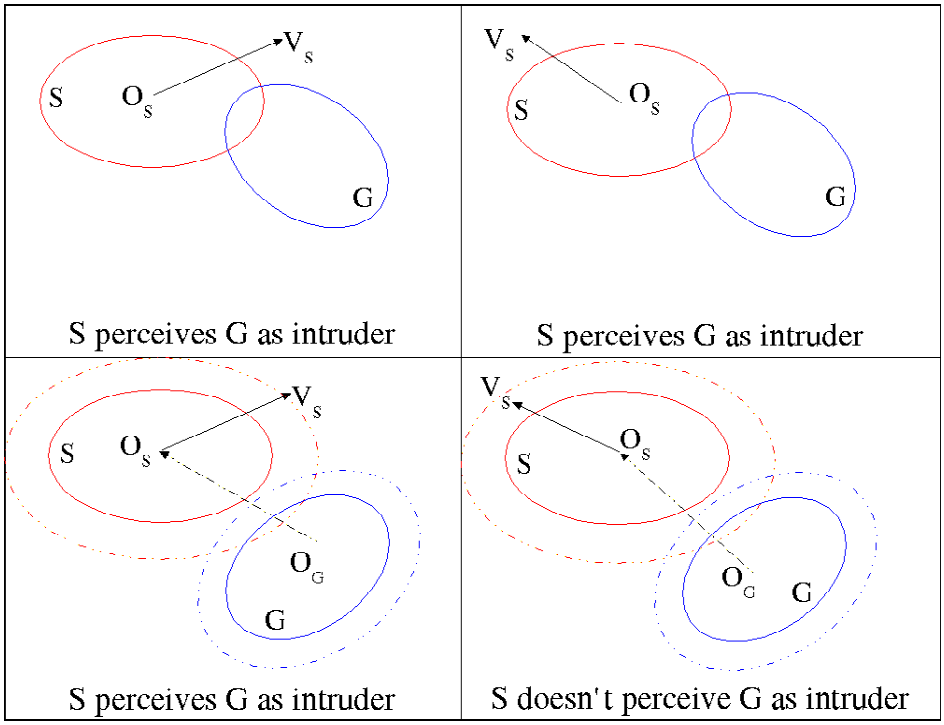


Figure 4.4: Structure perception of intruder

sumerizes the differents cases of opposite rotation structures perceived in a close neighbourhood.

- To attack a structure means to send it a message and to be attacked means to receive a message.
- To escape means to be in unstable stage. In that case, the structure reduces its shape and generate elementary particles on its frontier, preserving the whole rotational value, as described on figure 4.5.
- to satisfy means to increase its stability. In that case, the structure try to aggregate neighbouring particles of same rotational sense, using local spanning tree and new shape computation, as described on figure 4.6.

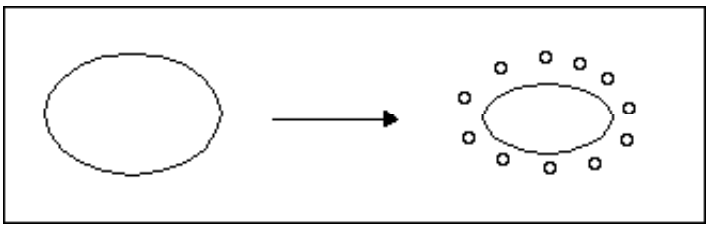


Figure 4.5: Structure escape

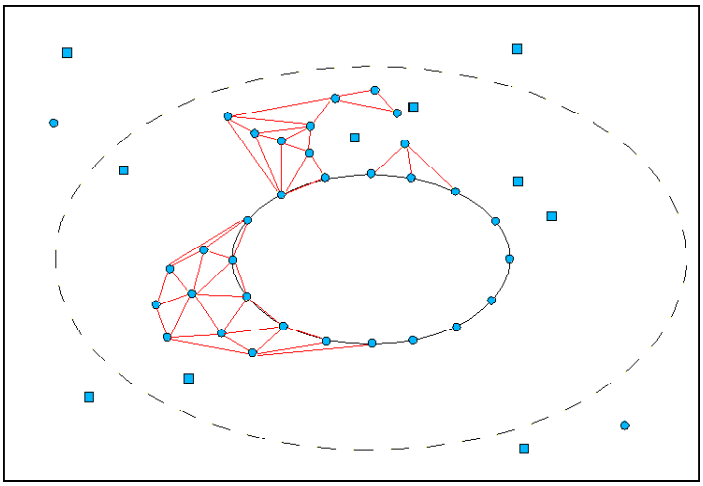


Figure 4.6: Aggregation process as structure satisfaction

As explained, the eco-agent modeling allows to compute the stability of the self-organized process and this modelizes the micro-macro interactions.

4.4 An augmented agent representation for non deterministic behaviour model

In this section, we will focus our attention on probabilistic automata as a model for non deterministic behaviour. Then we finally, propose a generic global formalism which allows to mix the two previous descriptions, transducer and probabilistic automata.

4.4.1 Probabilistic automata for non determinist aspects

An agent behaviour can sometimes be modelled with a probabilistic approach. For example, in prey-predator simulation, a prey, at a given instant, can be in an internal state without the perception of any aggression. So, it can do nothing, or eat, or move itself. This choice can be the result of a probability depending on individual characteristics. Thus, for a given perception, we use probabilistic transitions to go from an initial state to some particular ones, as described in Fig. 4.7.

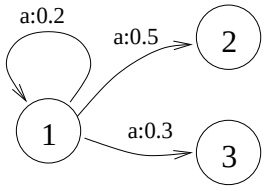


Figure 4.7: Transitions from one node in a probabilistic automata

The framework of automata with multiplicities also allows us to model these probabilistic aspects in agent behaviour: each internal transition produces a probability in output. Indeed, a stochastic (or probabilistic) automaton is a particular case of one with multiplicities. In comparison with the definition given above, the transition function δ defined above becomes $\delta : Q \times \Sigma \times Q \rightarrow [0, 1]$, with the constraint that:

$$\forall a \in \Sigma, \forall q \in Q \quad \sum_{p \in Q} \delta(q, a, p) = 1 \tag{4.1}$$

that is, the sum of probabilities of all transitions for each perception, starting from any state, must be equal to 1.

We use a the linear representation of automata. Such a representation of dimension n (number of states) is a triplet (λ, μ, γ) where $\lambda \in [0, 1]^{1 \times n}$, with

$\sum_{i=1}^{n} \lambda_{1,i} = 1$, a row vector coding the input probabilities, $\gamma \in \{0,1\}^{n\times 1}$ a column vector coding the output probabilities and $\mu : \Sigma^* \to [0,1]^{n\times n}$ a morphism of monoids coding the transition probabilities between states for each letter $a \in \Sigma$, with the row-stochastic matrix $\mu(a)$.

The constraint represented by expression (4.1) is equivalent to

$$\forall a \in \Sigma, \forall q \in Q \quad \sum_{p\in Q} \mu_{q,p}(a) = 1 \tag{4.2}$$

A successful path is a path from an initial state to a final one. Then, for all $w \in \Sigma^*$ corresponding to a successful path, $\lambda\mu(w)\gamma$ is the probability of all successful paths labelled by w (it is the probability that the successful succession of perceptions, w, occurs).

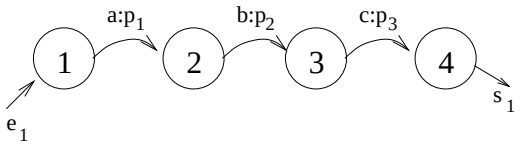


Figure 4.8: A successful path: the probability to perceive abc is $e_1p_1p_2p_3s_1$

We show in Fig. 4.9 an example of probabilistic automaton (Notice that here, the state 6 is not reachable: such a phenomenon can occur after some genetic operations described below). A final state for an agent behaviour represents an internal state which can be perceived by other agents. If a synchronism of the whole system is required, we have to wait that all the agents have reach a final state. That means that a state which are not intitial and not final one is considered as private and is not perceived by the other agents.

From this automaton, the associated linear representation is:

$$\lambda = (0.5, 0, 0, 0, 0.5, 0) \qquad \gamma^t = (0, 1, 0, 1, 0, 0)$$

$$\mu(a) = \begin{bmatrix} 0.2 & 0.5 & 0.3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

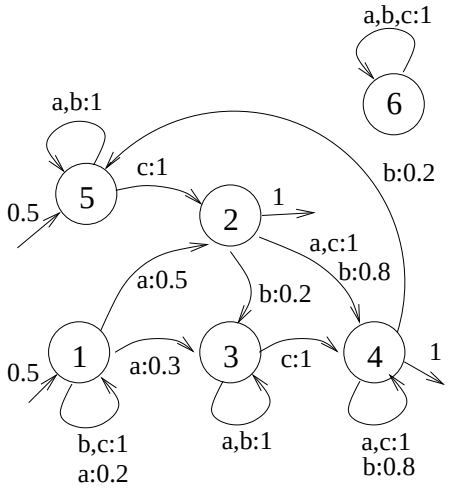


Figure 4.9: A probabilistic automaton $\mathcal{A}$

$$\mu(b) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.2 & 0.8 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mu(c) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

These matrices allow us to obtain a quantitative information for measuring a global agent answer to the perception $p \in \Sigma$. These tools define a metric on the agent behaviours as described in the following.

4.4.2 A global formalism for agent behaviour

We have described previously two formalisms to represent an agent behaviour. Both are based on automata with multiplicities. The first one is a tranducer where the semiring of output data is the set of the polynomials in the elementary actions.

The second one is a probabilistic automaton and the semiring of output data is $\mathbb{R}^+$.

The second representation allows to represent non-deterministic aspects for agent behaviour but it lacks the fact that we do not have agent actions in output. In some case, we can solve this lack with the fact that the reached state by each transition can be characteristic of the action made by the agent during the transition. We can find such kind of model in the strategy models used for the prisoner dilemma in the section 5.2.

We propose here a complete representation which mixes the two previous ones. For that purpose, we simply consider the semiring of output data as the cartesian product of the two previous ones.

The agent behaviour is described with an automaton with multiplicities over a finite alphabet Σ and a semiring K, as a $5 - tuple$ $(\Sigma, Q, I, T, \delta)$, with Q a finite set of states. This automaton with the previous general definition, is fully characterized by:

- The finite alphabet Σ which corresponds to the agent perceptions,
- We consider the set Π of its elementary actions,
- We consider $A\langle\Pi\rangle$ as the set of polynomes of these elementary actions with coefficients in A a commutative semiring, as the set of integers, for exemple.
- The semiring of output data K of the whole model is the cartesian product of the two semirings: $K = (A\langle\Pi\rangle, \oplus, \odot) \times (\mathbb{R}^+, +, \cdot)$ with derivated operations deduced by the cartesian product.

4.5 Agent interaction and evolution modeling using algebraic computation

In this section, we deal with essential aspect of the formalism used for the agent behaviour. Using automata with outputs as semiring, we can make such automata operate with the deduced operations of the semiring. Then we define original operators over the automata, using genetic operators.

4.5.1 Classical algebraic operators for agent aggregation basic models

The advantage of the use of automata-based description for agent behaviour is the feasibility to define classical operators on them [66]. This allows us to implement automatic management on multi-agent systems.

The following of this section, is valid for any semiring $(K, \oplus, \odot)$. The classical rational operations on automata are applied over agent behaviour as follows:

- The Sum of the agent behaviours R and S

$$R + S = \sum_{w \in \Sigma^*} \left(\langle R|w \rangle \oplus \langle S|w \rangle \right) w$$

  corresponds to a new agent having the whole behaviour of the two first ones. Common actions give a composition of the original ones, other ones are simply preserved in the new behaviour, as if only one agent was acting.

- The Cauchy Product of the agent behaviours R and S

$$R.S = \sum_{w \in \Sigma^*} \left(\bigoplus_{uv=w} \langle R|u \rangle \odot \langle S|v \rangle \right) w$$

  allows to obtain a new agent behaviour given by all the possible successive and successful actions of R then S, that is it would modelize a new agent acting exactly as R and then as S (whatever the succession of actions it would have executed "as" R),

- The Star of the agent behaviour S

$$S^* = \sum_{w \in \Sigma^*} \bigoplus_{n \geq 0} \langle S^n|w \rangle w$$

  gives the behaviour of an agent which possibly never terminates and reinitializes itself as long as needed ('loop' agent).

- The Hadamard product of the agent behaviours R and S

$$R \circ S = \sum_{w \in \Sigma^*} \left(\langle R|w \rangle \odot \langle S|w \rangle \right) w$$

  allows the creation of an extractor agent, extracting common behaviours while composing it.

So, it is important to note that to represent collective behaviour, we use the sum to simply aggregate some agent behaviours and we have to use the Cauchy product each time it is necessary to respect sequentialization in the new agent behaviour.

4.5.2 Genetic operators for agent evolution model

We describe in this section the bases of the genetic algorithm [96, 82] used on the probabilistic automata allowing to manage emergent auto-organizations in the multi-agent simulation.

Evaluation of agent behaviour and definition of agent fitness in terms of collective criterium

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 in 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. Notice that the coefficients of this matrix, such as defined, are computed whatever the value of the perception in Σ on each transition on the successful path. 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.

Thus, the evaluation of the automaton $\mathcal{A}$ in the Fig. 4.9, coding the agent x behaviour is $e(x)$ corresponding to the matrix M:

$$M_{i,j} = \begin{cases} \lambda_{1,1}\left(\mu(a)_{1,2} + M_{1,4}\mu(b)_{4,5}\mu(c)_{5,2}\right)\gamma_{2,1} \\ \qquad \text{if } i = 1 \text{ and } j = 2, \\ \lambda_{1,1}\left(\mu(a)_{1,2}(\mu(a)_{2,4} + \mu(b)_{2,4} + \mu(c)_{2,4}) + \right. \\ \quad \left. \mu(a)_{1,2}\mu(b)_{2,3}\mu(c)_{3,4} + \right. \\ \quad \left. \mu(a)_{1,3}\mu(c)_{3,4}\right)\gamma_{4,1} \\ \qquad \text{if } i = 1 \text{ and } j = 4, \\ \lambda_{1,5}\left(\mu(c)_{5,2}\right)\gamma_{2,1} \\ \qquad \text{if } i = 5 \text{ and } j = 2, \\ \lambda_{1,5}\left(\mu(c)_{5,2}(\mu(a)_{2,4} + \mu(b)_{2,4} + \mu(c)_{2,4}) + \right. \\ \quad \left. \mu(c)_{5,2}\mu(b)_{2,3}\mu(c)_{3,4}\right)\gamma_{4,1} \\ \qquad \text{if } i = 5 \text{ and } j = 4, \\ 0 \qquad \text{otherwise} \end{cases}$$

where probabilistic coefficients are not evaluated.

Let x and y two agents and $e(x)$ and $e(y)$ their respective evaluations as described above. We define $d(x, y)$ a distance between the two agents x and y as $\parallel e(x) - e(y) \parallel$, a matrix norm on the difference of their evaluations. Let $\mathcal{V}_x$ a neighbouring 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 :

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

Duplication, Crossover, Mutation and Selection mechanisms

We describe a genetic algorithm which usually manages a population which is here the agent behaviours and which uses individual characteristic representations, named chromosomes. We define the chromosome for each agent as the sequence of all the matrices $\mu(a)$ associated to each perception $a \in \Sigma$. In the following, genetic algorithms are going to generate new agents containing possibly new transitions from the ones included in the initial agents. To authorize only significant behaviour, we have to consider the existence of a family of boolean transition matrices $(\mathcal{T}_a)_{a \in \Sigma}$, associated to each type of agent, and coding its whole possible transitions in accepting this perception. Its effective transition matrix associated to each perception is a "subset" of it (in fact, each transition matrix associated to a given perception $a \in \Sigma$, noted $\mu(a)$, is a matrix of same dimension as $\mathcal{T}_a$, but with probabilistic coefficients).

In the genetic algorithm, each couple of agents follows a reproduction iteration broken up into three steps:

- Duplication where each agent of the couple generates a clone of itself;
- Crossing-over where a sequence of lines of each matrix $\mu(a)$ for all $a \in \Sigma$ is arbitrary chosen. For each of these matrices, a permutation on the lines of the chosen sequence is made between the respective matrices of the two agents corresponding to the reproduction couple.
- Mutation where a line for each matrix $\mu(a)$ is arbitrary choosen and, randomly, a sequence of new values is given to this line, in respect of the probabilistic nature of the matrix represented by the expression (4.2). The new matrix obtained by mutation must respect the authorized transitions given

by the $(\mathcal{T}_a)_{a\in\Sigma}$ family

Finally, the whole genetic algorithm scheduling for a full process of reproduction over all the agents is the evolutionary algorithms:

1. For all couple of agents (x, y), two children are created by duplication, crossover and mutation mechanisms;

2. The fitness, for each agents, is computed;

3. For all 4-tuple composed of parents and children, the two performless agents, in terms of fitness computed in previous step, are suppressed. The two agents, still in live, result of the evolution of the two initial parents.

Some Elementary Operations on Probabilistic Automaton for Coding Genetic Operations

For coding the operations of crossing-over and mutation, we have to define some elementary operations over linear representations.

We will denote $Id_i^{n,0}$ the modified identity matrix of dimension n where the i^{th} row is replace by the null row, and $Id_i^{n,1}$ the modified identity matrix in which all rows have been nulled excepted the i^{th}. Let x (resp. y) an agent verifying the authorized transitions given by the $(\mathcal{T}_a)_{a\in\Sigma}$ family and $(\lambda_x, (\mu_x(a))_{a\in\Sigma}, \gamma_x)$ (resp.$(\lambda_y, (\mu_y(a))_{a\in\Sigma}, \gamma_y)$) a linear representation coding its behaviour. With these notations, the crossing-over between two similar states i of agents of the same $\mathcal{T}$-family gives two new agents x' and y' of respective representations $(\lambda_{x'}, (\mu_{x'}(a))_{a\in\Sigma}, \gamma_{x'})$ and $(\lambda_{y'}, (\mu_{y'}(a))_{a\in\Sigma}, \gamma_{y'})$. It can be written by :

$$\begin{aligned}
&(\lambda_{x'}, (\mu_{x'}(a))_{a\in\Sigma}, \gamma_{x'}) := \\
&\qquad (\lambda_x, (Id_i^{n,0}\mu_x(a) + Id_i^{n,1}\mu_y(a))_{a\in\Sigma}, \gamma_x), \\
&(\lambda_{y'}, (\mu_{y'}(a))_{a\in\Sigma}, \gamma_{y'}) := \\
&\qquad (\lambda_y, (Id_i^{n,0}\mu_y(a) + Id_i^{n,1}\mu_x(a))_{a\in\Sigma}, \gamma_y).
\end{aligned}$$

The mutation of the state i of the agent x, giving an agent x', is applied over a subset $\Sigma' \subset \Sigma$ of perceptions, the input and output probabilities being preserved. Let $R_{[0,1]}$ be a random function in $[0, 1]$. For each perception $a \in \Sigma'$, we compute a new matrix $\mu'_{x'}$ by $\mu'_{x'}(a)_{i,j} = \mathcal{T}_a(i, j)R_{[0,1]}$. We will denote $S_i(a) = \sum_{j=1}^{n} \mu'_x(a)_{i,j}$

the sum of output probabilities, from the state i, for each a. The new agent is then represented by the triplet $(\lambda_{x'}, (\mu_{x'}(a))_{a\in\Sigma}, \gamma_{x'})$ with:

$$(\mu_{x'}(a))_{i,j} := \begin{cases} \frac{\mu'_{x'}(a)_{i,j}}{S_i(a)} & \text{if } S_i(a) > 0 \text{ and} \\ & i \text{ the mutated state,} \\ 0 & \text{if } S_i(a) = 0 \text{ and} \\ & i \neq j \text{ if } i \text{ is the mutated state} \\ 1 & \text{if } S_i(a) = 0 \text{ and} \\ & i = j \text{ if } i \text{ is the mutated state,} \\ (\mu_x(a))_{i,j} & \text{otherwise} \end{cases}$$

The stochastic property is preserved by the crossing-over and the mutation operations defined above.

4.5.3 Evolutive automata in terms of self-organization modeling

In the previous computation, we defined a distance between two agents. This distance is computed using the linear representation of the automata with multiplicities associated to the agent behaviour. This distance is based on successfull paths computation which needs to define initial and final states on the behaviour automata. For specific purposes, we can choose to define in specific way, the initial and final states. That means 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 distance and then the fitness function defined previously. This fitness function is an indicator that returns high value when the evaluated agent is near, in the sense of the behaviour 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 selective process. So during the computation, the genetic algorithm will make evolve the population to 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 to improve their fitness. In that way, the genetic algorithm reinforce the creation of a system which aggregate agents with similar behaviour, 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 ones and we cannot predict what will be the set of these aggregations which depend 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 solving problems, 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 a problem solving evaluation.

Chapter 5

Applications to economic complex modeling

Contents

5.1 Economic complex modeling

This section deals with the use of the simulation approach in analysing and understanding pheomena in the domain of economics. A variety of simulation approaches to analysing economic development and two main streams of models and agent-based approach, are characterized.

5.1.1 Simulation Approach

Computer simulation methods are widely used for operational research and economic science modern modeling. The model representation depends on the aims of our inquiry and on all constraints related to the processes. Having collected records of a real process behaviour for a given input $u(t)$ and an output $y^m(t)$, the modeller tries to adjust the models behaviour to reality either by selecting the proper values of the model's parameters or by changing the model's structure.

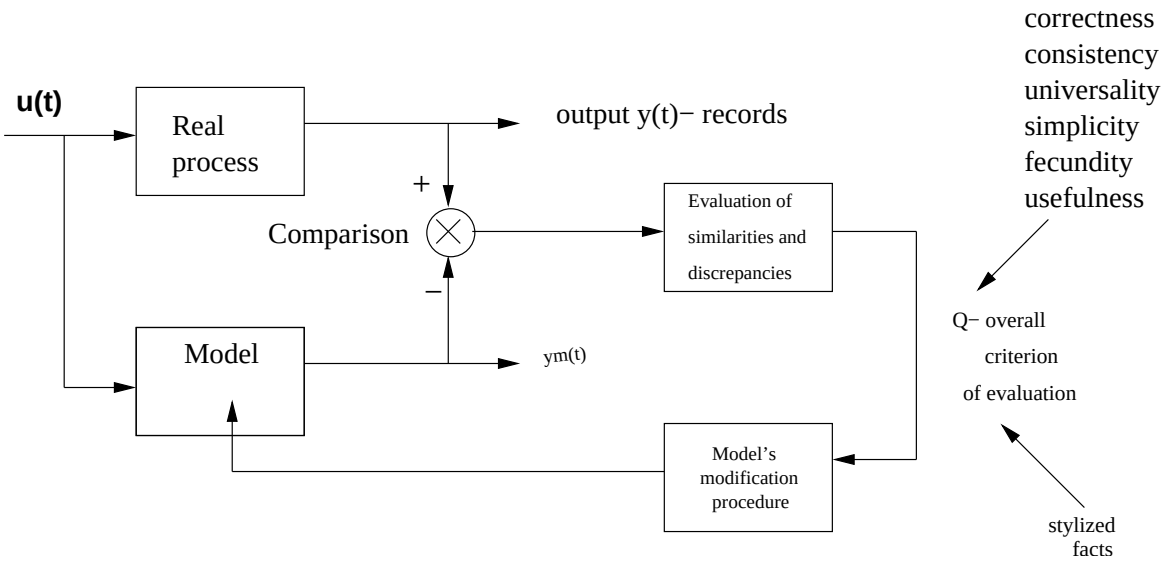


Figure 5.1: Model and reality

A schematic vue of the model adjustment process is represented in Figure 5.1. This kind of adjustment is sometimes called *a behaviour replication test* whose main aim is to compare the model behaviour with the behaviour of the system being modelled.

When historical time series data of the results of a real system development in the factory or laboratory are available, the model must be able to produce similar

data. Socio-economic system are higly interrelated, and disaggregation into semi-isolated subsystems is frequently impossible. We are typically in the description of systems complexity. It seems that in the social sciences and in economics the main aims of building are: better understanding of mechanisms of development of observed phenomena or process; then building different, alternative scenarios of development for a given socio-economic systems. Evaluation of socio-economic models must proceed in systemic approach, that is, following [102]:

1. Isolating a specific sphere of socio-economic reality.
2. Specifying all relations of phenomena within the sphere with the external environment.
3. Building a model which describes all important parameters observed within the chosen sphere , with all essential influences of the external environment included.

The important relations with the external environment, the building of relevant mathematical models and optimizing the choice of suitable policies are almost impossible.

The socio-economic systems differs from the classical engineering ones because of the great different approaches for testing and validating the developed models. The engineering systems compare numerical data (records of development of real systems) with numerical ouputs of the model. In socio-economic systems, collection of reliable set of proper data (records) is frequently impossible to obtain. Therefore, validation of socio-economic models is frequently done on the base of so-called stylized facts and subjective sub-criteria. It seems that the most important and the most popular sub-criteria are :

1. correctness- consequences of the model ought to be very close the results of experiments and/or observations.
2. consistency- the model ought to be consistent not only internally but also with other commonly accepted theories used to describe similar or related phenomena.
3. universality - consequences of the model ought not to be confined to individual cases.
4. simplicity - the model ought to create order in the formerly isolated phenomena.
5. fecundity - the model ought to throw new light on well-know phenomena.

6. usefulness - the practical criterion dominates frequently in sciences, being very close to engineering and industry.

5.1.2 Agent-based Computational Economics (ACE)

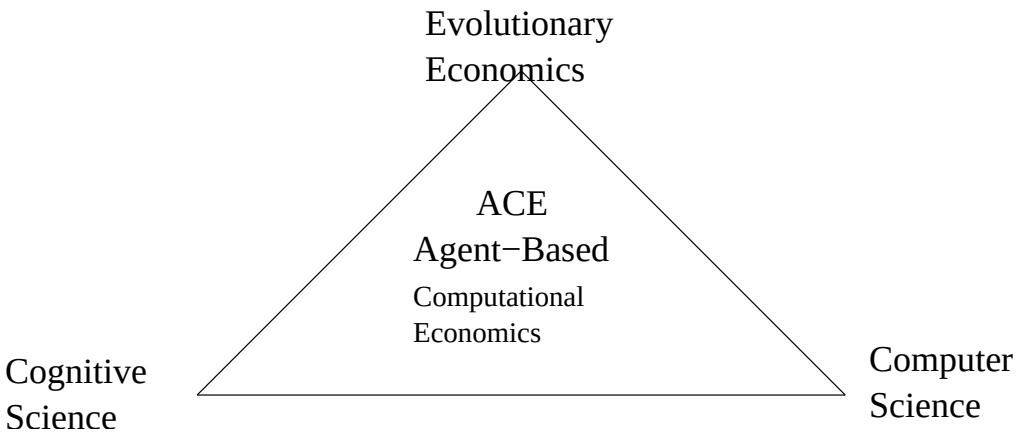


Figure 5.2: Basic ACE representation

Following [102], we can define agent-based Computational Economics (ACE) as the computational study of economics modelled as dynamic and complex systems of interacting *agents*. Here *agent* refers to a bundle of data and behavioral methods representing an entity constituting part of a computationally constructed world.

Artifical life ($Alife$) [108] is the name of a growing field of research that attempts to develop mathematical models and use computer simulations to demonstrate ways in which living organisms grow and evolve. $Alife$ is the one of roots study of basic phenomena commonly associated with living organisms, such as self-replication, evolution, adaptation, self-organization, competition, and social network formation. The study of evolutionary economics has of course been pursed be many researchers, focusing on the potential economic applicability of evolutionary game theory. This consists to define game strategies distributed over a fixed number of strategy types and to reproduce over time in direct proportion to their relative fitness.

ACE researchers have been able to extend previous evolutionary economic works in several directions. First, much greater attention is generally focused on the endogenous determination of agent interactions. Second, a broader range of

interactions is typically considered, with cooperative and predatory associations increasingly taking center stage along with price and quantity relationships. Third, agent actions and interactions are permitting generalizations across specific system applications. Fourth, the evolutionary process is generally expressed in terms of genetic algorithm acting directly on agent characteristics. The evolutionary selection pressures result in the continual creation of new modes of behaviour and ever-changing network of agent interactions.

The central problem for ACE researchers is to understand the apparently spontaneous appearance of regularity in economic processes, such as the unplanned coordination of trading activites in decentralized market economies.

The decentralized market economies consist of large number of economic agents involved in distributed local interactions. These local interactions give rise to macroeconomic regularities such as trading protocols, socially accepted monies, and widely adopted technological innovations which in turn feed back into the determination of local interactions.

The ACE modeler starts by constructing an initial population of agents. These agents can include both economic agents (e.g. consumers, producers, intermediaries,..) and agents representing various other social and environmental phenomena . The ACE modeler specifies the initial state of the economy by specifying the initial attributes of the agents. The initial attributes of any one agent might include type characteristics, internalized behavioral norms, internal modes of behavior (including modes of communication and learning) and internally stored information about itself and other agents. The economy then evolves over time without further intervention from the modeler.

The number of researchers now making use of the ACE methodology is growing, however, and the number of issues being addressed by these researchers is rapidly expanding and other computationally oriented social scientists have used a broad range of representations for learning processes of computational agents. These include reinforcement learning algorithms, neural networks, genetic algorithms, and a variety of other evolutionary algorithms that attempt to capture aspects of inductive learning.

On the other hand, for computational models of economic processes with diverse human participants the salient characteristics of actual human decision-making behavior if predictive power is to be attained. In this case it will generally be necessary to introduce local learning schemes in which individual agents or groups of agents separately evolve their strategies on the basis of their own per-

ceived local benefits.

5.1.3 Bottom-up Modeling of Market Processes

The self-organizing capabilities of specific types of market processes is now one of the most active areas of ACE research. For example financial, electricity, labor, retail, business-to-business, natural resource, entertainment and automated internet exchange systems.

Robert Marks was one of the first researchers to use an ACE framework to address the issue of market self-organization. His research highlighted for economists - in compelling constructive terms - the potential importance of history, interactions, and learning for the determination of strategic market outcomes. One outcome observed by Marks in his experiments was the emergence of globally optimal "joint maximization" pricing across firms without any explicit price collusion[1].

5.1.4 Schumpeterian model

We can start description of Schumpeterian models from the work of Nelson and Winter. Usually the economy is disaggregated into diverse individual firms influencing each other by nonlinear dynamic interactions describing search for innovation, competition and investment.

In most simulation models, agents use boundedly rational behavioural procedures. Learning and searching for innovation is modelled by allowing for mutation and imitation rules operating on the firms operational parameters. Mutations are usually local within the routine space.

The first model presented in Nelson and Winter (1982, ch 9) can be seen as the first evolutionary growth model. The state of the evolutionary process of an industry at any moment t is described by the capital stock and the behavioural rules of each firm. The state in the next moment $t+1$ is determined by the state in a previous moment. In this growth model firms use production techniques which are characterized by fixed labor and capital coefficients. Firms manfacture homogeneous products, so the model that firms produce using a Leontief production function. Therefore substitution between labor and capital is not present in the model. Changes of these both coefficient are not coefficient and not correlated,

[1] See http://www.econ.iastate.edu/tesfatsi/asocnorm.html for pointers to ACE-related resources on the evolution of norms

therefore a phenomenon that ressembles substitution between labor and captial may be observed in the simulated process.

Search activites are determined by satisfying behaviour, in a sense that a new technique is adopted only if the expected rate of return is higher than the firm's present rate of return.

The search process may take two different forms: local search (mutation) or imitation. In the first case, firms search for new techniques, yet not present in the industrial pratice. The term local search indicates that each undiscoverd technique has a probability of being discovered which linearly declines with a suitably defined technological distance from the current techonology.

In the other hand, Schumpeter's model of economic development have stimulated an active programme of research since the publication of his seminar work (Schumpeter 1919, 1939 Schumpeter 1997).

Schumpeter's starting point is the steady state, or rather, a smoothly expanding economy. His population growth was exogenous and his savings rate rather constant. In Schumpeter's view,the driver of "development" (as apposed to boring "growth") were discontinuous punctuated changes in the economic environment. These, he claimed, were brought about by a veriety of things (e.g sudden discoveries of new factor supplies), but entrepreneurial innovation was the central one.

Schumpeter claimed that there were ratchet effects in innovation so that enterpreneurial-driven spurts of economic activity led to progressively higher levels of income and there is no mong run need to slow down. Schumpeter claimed that there were no diminishing returns to innovation. The only reason one may be driven towards slower, steady-state is that all the entrepreurs in a generation might be already "used up".Schumpeter abandoned Say's law and claimed that credit made present activity independent of past activity and thus enabled entrepreneurship. Hence, since entrepreneurial innovation could be arrested by lack of credit, then financial innovation was also an important factor for increasing growth. Schumpeter did have long-run elements in his theory which induced a breakdown in growth. The concept of "steady-state" was still primitive in Schumpeter .

5.1.5 Sugarscape Model

Sugarscape (SSC) model is buid from cellular automata concept. In this model, there is two-dimensional lattice representing an environment, with a particular distribution of various resources called Sugar (see figure 5.3). Agents move in this environment according to rules of behavior, that the agents need to survive and

eat. So they have to search the regions which are rich in sugar, some regions are poor in sugar. Agents can *see* their local metabolism and certain traits like vision and *move* from one side to another searching the food to *eat* the sugar with every movement an agent burns a particular amount of sugar, equal to its metabolic rate. Agents with vision can see α units in four principal lattice directions, north, south, east and west. Agents have no diagonal vision. All agents are given with some initial endowment of sugar, which they carry with them as they move about the Sugarscape lattice.

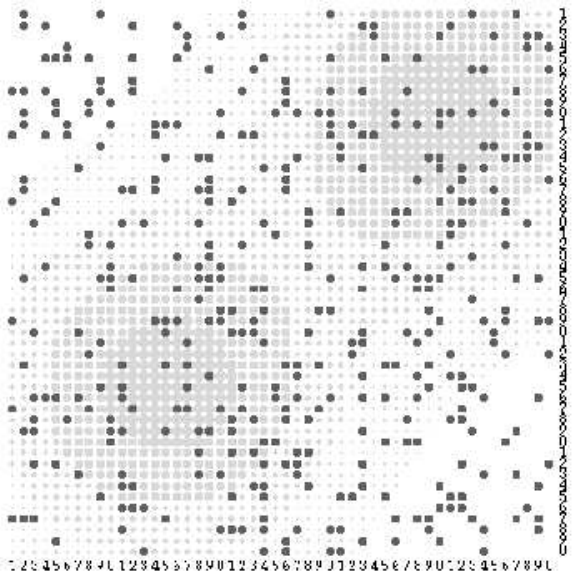


Figure 5.3: Sugarscape lattice with Agents

The Sugarscape model : There are two kinds of rules for environment and agents as we will show.

Initial state variables and fixed parameters may be randomly selected on fixed domains.

Environment Rules: Environment (behavior) governed by a set of rules (RE).

- Two dimensional grid (lattice).
- Each point (x,y) which mean (current sugar level, capacity).
- Distribution: peaks of 4 directions.

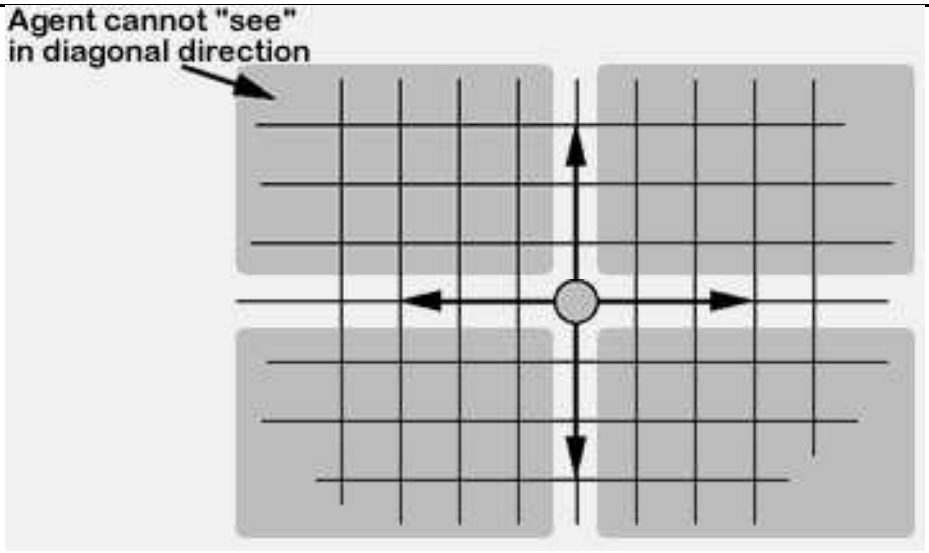


Figure 5.4: Agents vision directions

- Sugar levels.

Sugarscape (SSC) growback rule G_α. At each lattice position, sugar grows back at a rate α units per time interval up to the capacity at that position.

Agents rules: Agents are characterized by a set of fixed and variables states by:

- Active in expression
- Basic internal state (fixed)
  - field of vision: random distance.
  - metabolim: random amount of sugar m burnt per round.
- Basic internal state (variables).
  - Amount of sugar, agents are given some initial endowment of sugar, initial level a.
  - Location random grid position (x,y).

The agent can only see in the direction of the arrows, it does not see the shaded areas , as we have shown in figure 5.4, also agents move according to the following Agents movement rules M:

- Look out as far as vision permits in four lattice directions and identify unoccuped site(s) with max (r) within distance v with random start.

- Move to this site.
- Adjust sugar level $a \leftarrow a + r - m$.
- If a<0 agent dies and is removed from sugarscape so it can not move.

The sugar level of the agent is incremented by the amount of sugar available on the grid point, and decrements by its metabolic rate.

Agents replacement rule R[a,b]

When an agent dies it is replaced by an agent of age 0 having random initial attributes, random position on the (SSC) random initial endowment, and maximum age randomly selected from the range [a,b]. The agents replacement rules can be defined as:

- agents created with maximum age a_m chosen randomly from $[a, b]$ and actual age $a_a = 0$
- at each round $a_a \leftarrow a_a + 1$
- an agent dies when $a_a = a_m$
- the relpacement agent's location, sugar endowment and genetic makeup are random on the appropriate intervals.

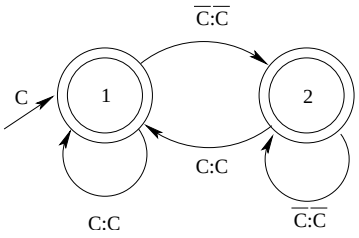
5.2 Prisoner Dilemma : Automata based model for cooperation and competition aspects

In multi-agent systems, interactions correspond to social behaviour. They are often described in terms of cooperation or competition aspects. We focuse our attention with prisoner dilemma [4] which is a model for negociation behaviour, allowing alternance of cooperation and competition between agents of the same system.

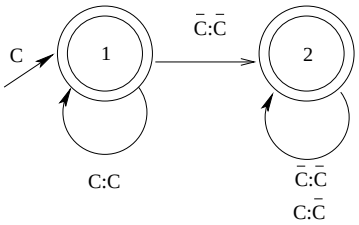
The prisoner dilemma is a two-players game where each player has two possible actions: cooperate (C) with its adversary or betray it ($\overline{C}$). So, four outputs are possible for the global actions of the two players. A relative payoff is defined relatively to these possible outputs, as described 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 5.1: Prisoner dilemma payoff



Strategy A : Tit-for-tat strategy



Strategy B : Vindictive strategy

Figure 5.5: Two prisoner dilemma strategies in term of tranducers

In iterative version of the prisoner dilemma, successive steps can be defined. Each player don't know the action of its adversary during the current step but he knows it for the precedent step. So different strategies can be defined for player behaviour, the goal of each one is to obtain maximal payoff for himself. In the figure 5.5, 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 incomming arrow labeled only by the output. The final states are the states 1 and 2, recognizable with the double circles. In strategy A, the player has systematically the same behaviour as its adversary at the previous step. In strategy B, the player chooses definitively to betray as soon as his adversary does once.

These previous automata represent 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 the figure 5.6.

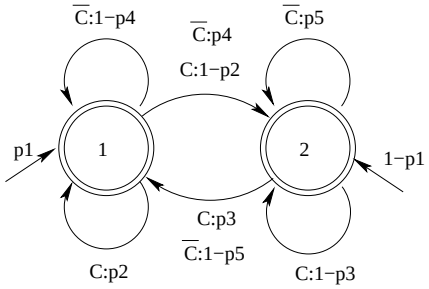


Figure 5.6: Probabilistic multi-strategies two-states automata for prisoner dilemma

This automaton represents all the two-states strategies for cooperation (C) and competitive ($\overline{C}$) behaviour of one agent against another in prisoner dilemma. The transitions are labeled in output by the probabilities (p_i) of their realization. The state 1 is the state reached after cooperation action and the state 2 is reached after betrayal.

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

$$\lambda = (p_1, 1 - p_1) \qquad \gamma^t = (1, 1)$$

$$\mu(C) = \begin{bmatrix} p_2 & 1-p_2 \\ p_3 & 1-p_3 \end{bmatrix}$$
$$\mu(\overline{C}) = \begin{bmatrix} 1-p_4 & p_4 \\ 1-p_5 & p_5 \end{bmatrix}$$

In the following section, we describe a general genetic algorithm on probabilistic automata and we show how it can be applied for modelling an adaptive strategy for the prisoner dilemma based on the previous particular probabilistic automaton.

5.2.1 Genetic algorithms on probabilistic automata

We describe a genetic algorithm managing a population that is the agent behaviours coded with probabilistic automata. Genetic algorithms use individual characteristic representations, named chromosomes. We define the chromosome for each agent as the sequence of all the matrices $\mu(a)$ associated to each perception $a \in \Sigma$. In the following, genetic algorithms will generate new agents containing eventually new transitions from the ones included in the initial agents. To authorize only significant behaviours, we have to consider the existence of a family of boolean transition matrices $(\mathcal{T}_a)_{a\in\Sigma}$, associated to each type of agent, and coding its whole possible transitions for each perception. The effective transitions matrix associated to each perception is a "subset" of it (in fact, each transition matrix associated to a given perception $a \in \Sigma$, denoted $\mu(a)$, is a matrix of same dimension as $\mathcal{T}_a$, but with probabilistic coefficients).

In the genetic algorithm, each couple of agents follows a reproduction iteration broken up into three steps:

- Duplication where each agent of the couple generates a clone of itself;
- Crossing-over where a sequence of lines of each matrix $\mu(a)$ for all $a \in \Sigma$ is arbitrary chosen. For each of these matrices, a permutation on the lines of the chosen sequence is made between the respective matrices of the two agents corresponding to the reproduction couple. In term of automata operator, the crossing over consists in the permutation of all the transitions outgoing from all states/lines selected by the crossed operation.
- Mutation where a line for each matrix $\mu(a)$ is arbitrary choosen and, randomly, a sequence of new values is affected to this line, according to the probabilistic nature of the matrix represented by the expression (4.2). The new matrix obtained by mutation must respect the authorized transitions given by the $(\mathcal{T}_a)_{a\in\Sigma}$ family.

The reproduction steps generate new agents behaviours. Genetic algorithms have to select over these behaviours, some of them which have the best values of a given function called fitness. In dilemma prisoner, the fitness function returns, for a given behaviour automaton, the corresponding payoff value.

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

1. For every couple of agents (x, y), two children are created by duplication, crossing over and mutation mechanisms;

2. The fitness, for every agent, is computed;

3. For every 4-tuple composed of parents and children, the two performless agents, in term of fitness computed in previous step, are removed. The two agents, still alive, result of the evolution of the two initial parents.

5.2.2 Evolutive adaptation for prisonner dilemma: implementation and simulation results

Genetic solvers have yet been experimented on the iterative prisoner dilemma. In [97], the chromosomes of the algorithm is a 64 positions string corresponding to each player memory of the past three outputs. We propose a more generic approach using genetic algorithms for prisoner dilemma as a particular application of the genetic algorithm on general probabilistic automata, as described in previous section. So, this algorithm is applied to the particular probabilistic automaton described in figure 5.6. This allows to simulate adaptive behaviours in term of evolutive strategies. Implementations have been described in [17] using the Madkit platform [89] developped in LIRMM (Montpellier - France). It shows adaptive strategies improving the player payoff.

A first sequence of experimentations has been made. These experimentations consist in building on one hand, an adaptive agents population described by the probabilistic automaton of the figure 5.6 and on the other hand, some agents whith static behaviour. The adversary players in prisoner dilemma are build from these two kinds of agents. Genetic algorithms are used on the adaptive agents population whose fitness is the player payoff. Against the static tit-for-tat strategy, the adaptive behaviour converges, after 250 iterations, to the strategy described by the linear representation:

$$\mu(C) = \begin{bmatrix} 0.99637 & 0.00363 \\ 0.99259 & 0.00741 \end{bmatrix}$$

$$\mu(\overline{C}) = \begin{bmatrix} 0.88216 & 0.11784 \\ 0.98521 & 0.01479 \end{bmatrix}$$

So, the emerging strategy seems to be the one which consists in cooperating whatever the perception. — valeur moyenne du payoff —

Another sequence of experimentations has been made. Two agents populations are opposed. The first one is composed of 12 static strategies. The other one is composed of adaptive agents which evolve genetically to a strategy which must be efficient against many strategies. The first result obtained is a effective convergence, after 250 iterations, to the strategy described by the linear representation:

$$\mu(C) = \begin{bmatrix} 0.08589 & 0.91411 \\ 0.93810 & 0.06190 \end{bmatrix}$$

$$\mu(\overline{C}) = \begin{bmatrix} 0.75001 & 0.24998 \\ 0.37071 & 0.62929 \end{bmatrix}$$

This strategy emerging from genetic algorithm is quite different from the tit-for-tat one. The agent payoff average is 2.4 which is still worth than the tit-for-tat strategy payoff (considered like one of the best one) which is 2.6.

This first results show that the adaptive strategy modelization gives a convergent proces able to improve the payoff. Some parameters have to be fixed like the mutation rate whose value has been fixed to 0.5% in the previous experimentations. Complementary experiments have to be studied to confirm the efficient of the method.

5.3 Cognitives sciences and Decision support systems

In this section, we study an application of evolutive automata in human decision processus, following [48]. Human decision in complex domains like management, medecine, environment is stressed by the time and many emotional or psychological constraints that are seldom included in Decision Support system (DSS).

Some DSS have been proposed to help workers during their job to cope with the steps of the decision making in complex domain like management, medecine and environment. In these DSS, we use available knowledge sources or stored experience (ref....). However these DSS have no representation of each actor's ability to take appropriate decisons. In stressfull domains, for a more realistic system, we need to model as well the impact of emotion and the psychologic structure

		+	-
Consequences of events	For others	Happy for	Resentment
		Gloating	Pity
	For self	Hope	Fear
		Joy	Distress
Actions of Agents	Self Agent	Pride	Shame
		Gratification	Remorse
		Gratitude	Anger
	Other Agent	Admiration	Reproach
		Gratification	Remorse
		Gratitude	Anger
Aspects of Objects		Love	Hate

Figure 5.7: couples of emotions variables from OCC model

of each actor making a decision.

The purpose in this section is to provide a Multi-Agent Decison Support System (MADSS) to model the emotion and the psychological structure of interacting actors making decisions in complex domains and stressfull contexts. This model will be used to build simulators of agent attitude in various situations such as emergency medicine, trading, education, environment and industrial catastrophes, military conflicts? In all these domains, modelling the emotional aspect and the psychological structure of the agents is essential.

5.3.1 A multilayer and agent-based model for decision support system

Most available computationally models of emotion rely on the OCC model developed in 1988 [135]. The OCC model defines events, agents and objects. Events are considered to induce emotional consequences. Agents are able of actions that have effects on the environment. Objects have imputed properties. The OCC model represents emotions as valenced reactions to the perception of the world. That is: one can be pleased about the consequences of an event or not (pleased/ displeased); one can endorse or reject the actions of an agent (approve/disapprove) or one can like or not aspects of an object (like/dislike). Then, the events can have consequences for others or for oneself and on acting agents. Thus, the different emotional balances are depicted by couples of (positive/negative) reactions represented by variables. For short, we do not provide the specialization tree of the OCC model but we just summarize the couples of variables in figure 5.7.

The aim of our proposition about model of decision is to use a multilayer

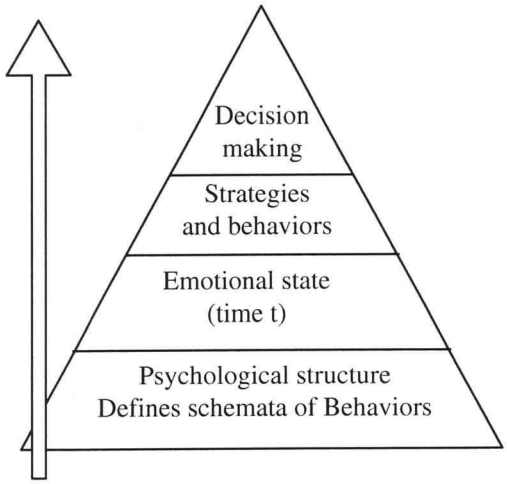


Figure 5.8: a multilayer model of decision

description with 4 layers. The base of this model is the psychological structure as described in figure 5.8.

The psychological structure layer

First, we describe the psychological structure which are able to act on the decision making behavior which is influence by emotional aspects. In figure 5.9, we present how and where the psychological structure intervene in the emotional learning loop.

Psychological structure can be studied through neurotic aspects of the considered personality. We show in the table 2, a description of such personality, while table 3 shows a neurosis classification.

The emotional state layer

The emotional state layer relies on the Ortony Clore Collins (OCC) model of appraisal which is the most widely accepted. The OCC model defines a set of 22 emotion parameters representing a certain positive and negative intensity of 11 items. The emotion quickly changes over time.

We use a sigmoid fuzzy logic function to describe each couple of parameters of the OCC model table1. The figure 5.10 depicts the example of Joy/distress concerning the effect of an event k evaluated by a score [-1,1] that is considered as desirable when is near to 1 or undesirable when it is near toăăă-1, neutral when it is near 0. All other couples of variables of the OCC classification table 1 are represented in a similar way. The item k could represent the consequences of an

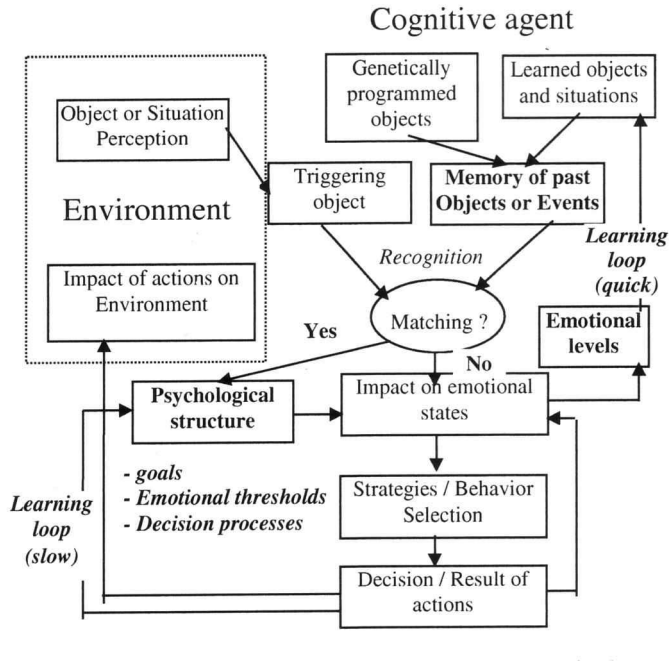


Figure 5.9: The emotion learning loop and psychological learning loop

event, the impact of agent actions for self or others and the aspects of an object.

5.3.2 Evolvable automata based strategies and behaviors layer

This layer has to manage previous layers, psychological level and emotional states and to define behaviors taking into account them.

The formalism proposed is based on finite-state automata composed of:

- a set of internal states characterizing mental situation, for example;
- transitions which describe the way going from one state to another. In the following, the event k evaluated by the score in the interval [-1,1] is the shutter release of the transitions.

We proposed to complete this formalism allowing us to introduce stochastic aspects in behavior descriptors. For this, each transition has to output a probability which indicates the chance that the transition was really done. To achieve this aspect, we base our model on automata with multiplicities or probabilistic automata as a sub-class of them.

Thus, the mathematical formalism for the behavior representation at each time t is a probabilistic automata as the following 5-uple :

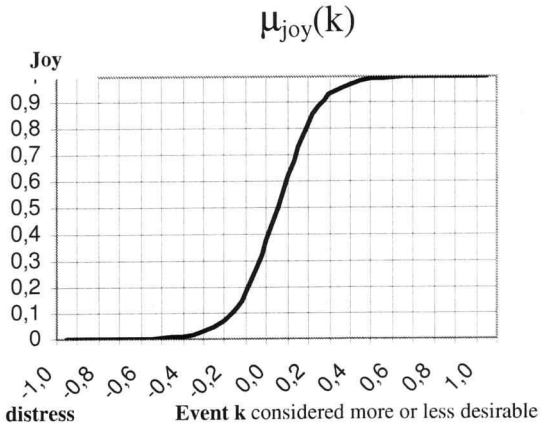


Figure 5.10: a fuzzy logic membership function

- is the finite set of internal behavior states;
- is the set of perceived event k. As previously indicated, each event k is evaluated by the score in the interval [-1,1]
- and are respectively the set of initial and final states from whom a behavior (or specific strategy, for example) can start and finish respectively.
- is a mapping from to [0,1] which describe all transitions associated to a couple of states and a event in input. The output of the mapping is a stochastic value.

We have proposed a computable formalism (automata-based) for modeling the behavior in respect of the emotional aspect. One of the major aspects in this operating automata description is the introduction of the coefficients for each transition j, each emotion parameter I at the instant t. These coefficients are obtained from the psychological structure and a database knowledge about them. So, via these parameters, the psychological structure acts on the behavior model, taking into account nevrotic personality, for example. Another aspect of the efficiency of these parameters is based on the fact that they can evolve, using genetic processus. This aspect can confer to the behavior some adaptive aspects.

5.3.3 The decision making layer

This layer is mainly described in a previous article [10]. We briefly summarize the architecture of a Multi-Agent Decision System which provide the necessary knowledge sources to take appropriate decisions but don't take into account the

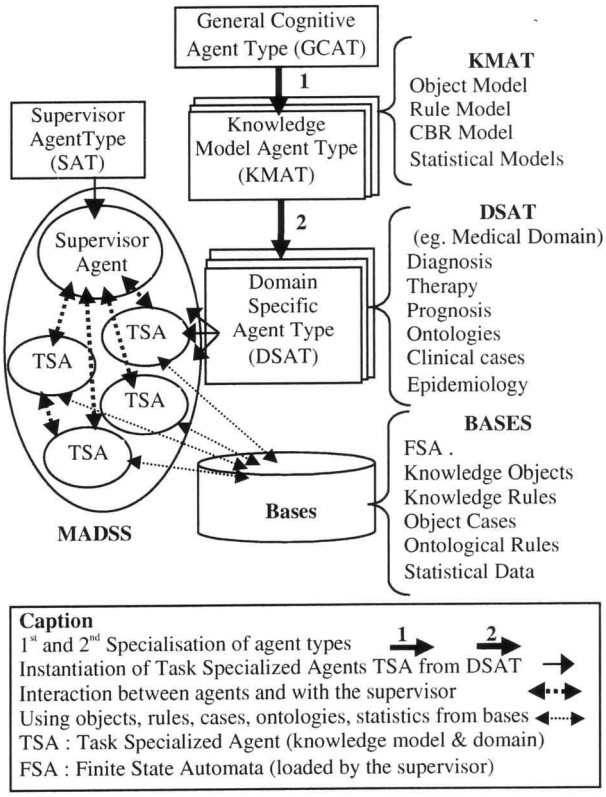


Figure 5.11: Specialisation of agents involved in a Multi-Agent Decision Support System (MADSS)

emotion and the psychological structure of the decision-maker.

A MADSS is using four agent categories that are specialized from a generic type (GCAT) described in [10]. A first specialization provides Knowledge Model Agent Types (KMAT), which are specific according to the knowledge model (rules, evaluation functions, Case-Based Reasoning?) but independent to the application purpose. The second specialization supplies Domain Specific Agent Types (DSAT), which inherit reasoning and knowledge model capabilities from the appropriate KMAT and apply them to the specific domain e.g. in medicine: (infectious disease diagnosis, poisoning prognosis, epilepsy therapy, patient and treatment follow-up, previous clinical case retrieval?)? Then, Task Specialized Agents (TSA) are instantiated from DSAT and commit to tasks elected by the supervisor agent that coordinates the whole decision process. The TSAs access the domain relevant bases containing knowledge and data (represented with the appropriate model) when necessary.

The ontological agent type is a DSAT, which formalizes the terminology, and the entity definitions used during the information transaction between agents [16]ă[17].

The supervisor agent type provides a unique instance named the supervisor agent that make an intensive use of finite state automata which define the necessary tasks to build the appropriate decision. The decision-making layer is consistent with the other layers of the model defining the emotion and the psychological structure of the decision-maker.

Thus, the model both represents the necessary decision steps, the psychological and emotional ability of the decision-maker to actually take the appropriate decision. Such a model is much more suited to realistically simulate the agent attitude and behavior in stressful situations or when the agent's personality hinders to take a decision.

Chapter 6

Conclusions and perspectives

The work presented here deals with the construction of operators coming from algebraic data structures for the modelization of complex systems. Complex systems are presently widely spread in many scientific domains. Emergence and self-organization are central aspects which can be considered as invariants through these numerous domains. We deal in this study about building effective operators for managing individual behaviors which can produce aggregative self-organized systems or which can be controlled by means of feed- back of the self-organized system over its own constitutive elements.

Automata associated to general semirings as sets of outputs are powerful tools for such purposes. Agent-based models are used for modeling constitutive elements of complex systems. They are well-suited to be the nodes of interaction networks. Agent-based modeling needs some autonomous characteristic concerning the behavior of these entities. These behaviors are so simulated by automata with multiplicities. The outputs of these automata are found inside a set which is the cartesian product of the set of (noncommutative) polynomials on elementary actions and the interval [0,1] as probabilistic values. Defining a semiring with this cartesian product, we can operate on actions and on probabilities which can be seen as a non- deterministic representation of the agent behavior. The powerful operational capabilities of semirings allows to define on the one hand some aggegative systems based on the active rôle of the agent and on the other hand adaptative behaviours with the implementation of genetic algorithms over the probabilistic value fitting. We deal in this way with the two aspects of complex system formation: aggregative aspect and adaptative behavior which can be seen as feed-back of the system over its own contituants.

The development of versatile tools based on algebraic data structures lead us to propose a new one that we called tables as a generalization of the k-sets of

Eilenberg. All the power of this generic structure is not completly developped in this work and need some advanced studies for enhancing new relevant operators well-suited for the specific need of the modelization of complex systems.

Concerning the evolutive representation of the behavior, we used in our study genetic algorithms over the automata probabilistic weights which have to fit an objective function on the whole behavior or the whole system. Such a process and representation allows us to define a metric over the agent behaviour but leads to high constraints over the automata topology. Some more sophisticated evolution processes could be considered to improve the expressivity of these evolutive representation. In such way, we could think to investigate the genetic programming which can lead to propose to build during the simulation, some original processes for the agents behavior, breaking the restrictive framework given at present with a fixed automata topology.

Concerning the application domains, we have investigated in this work, the economic modeling. We only defined here a sketch of possible modelling purposes and the studies concerning an advanced analysis for the results in terms of qualitative efficience of the models and in terms of quantitative validation are still to be done. Many further specialized works inside these applications domains could be developped, using the formal bases proposed in this work.

Annexe:
Data structure implementation in MuPAD

In this annexe, we will show the implementation of new data structures (Tables,Memorized semirings and semirings). In fact, we define and construct this new data structure which are a special kind of two-rows arrays. The first row is filled with words and second with some coefficients.

We present a new semiring which can be applied to the necessity of automatic processing multi-agents behaviors problems, and we present the implementation of our semiring to prove all its laws.

```
(1 First we define the two different tables with their
   coefficients,the third table is a void table.

L1 := table([
    x = .5
    yz = .3 ])

L2 := table([
    u = .5
    y = .1
    yz = .5 ])

L3 := table([
    xy = .1
    yz = 1.2
    uu = .5 ])

Lneutral:=table();       {We create a void table}
```

```
Lneutral   := table([])
```

2) The second step is to prove that the adding operation is commutat
over the tables.

```
{Commutativity     of  (+)}

addt(L1,L2);

table([
     u = .5
     x = .5
     y = .1
     yz  = .8  ])

> addt(L2,L1);

table([
     u = .5
     x = .5
     y = .1
     yz  = .8  ])
```

3)In this step we have proved that addition is commutative
over the tables.

```
evalb(op(op(addt(L2,L1)))=op        (op(a ddt( L1,L2 )))) ;

                                          {true}

addt(L3,addt(L1,L2));

table([
     u = .5
     x = .5
     y = .1
     xy  = .1
     yz  = 2.0
     uu  = .5  ])
```

```
addt(L2,L3);

table([
     u = .5
     y = .1
     xy = .1
     yz = 1.7
     uu = .5 ])

addt(L1,addt(L2,L3));

table([
     u = .5
     x = .5
     y = .1
     xy = .1
     yz = 2.0
     uu = .5 ])

addt(L1,Lneutral);      (The  void  table  is  the  neutral  element  of  +)

table([
     x = .5
     yz = .3 ])

addt(Lneutral,L1);(The       void  table  is  the  neutral  element  of  +)

table([
     x = .5
     yz = .3 ])
```

4) In this step we have proved that the addition is associative over the tables.

```
{Associativity     of  (+)}

evalb(op(op(addt(L1,addt(L2,L3        )))) =(op( op(a ddt( L3,ad dt(L 1,L2) )))) );

                                          {true}
```

```
{Neutral   element   of  (+)}


evalb(op(op(addt(Lneutral,L1        )))=( op(o p(add t(Ln eutra l,L1 )))) );

                                          {true}

{Associativity     of  product}
5)In   this   step   we   want   to   prove   that   product    is  associative.

eqtab:=proc(S,T)      evalb(op(op(S))=op(op(T)))         end;

           eqtab   :=  proc(S,    T)  evalb(op(op(S))      =
op(op(T)))    end

eqtab(mm1,mm2);

                                          true

evalb(op(op(mm1))=op(op(mm2)        ));

                                          true
```

After we defined the semiring and its laws over this new data structure (over table T) with two-rows array, the first row being filled with words taken in a given free monoid, the second row are the values or coefficients :

$$\begin{cases} indices & \text{set of words } I(T) \\ values & \text{bottom row } V(T) \end{cases} \tag{6.1}$$

If we consider, two tables T_1, T_2 and law (+) then the results will be as:

```
S1:=table(a=p1,ab=p2,ba=p3);

                                    table(
                                      ba  =  p3,
                                      ab  =  p2,
                                      a  =  p1
                                    )
```

```
S2:=table(a=q1,ba=q2,bb=q3);
```

```
table(
  bb = q3,
  ba = q2,
  a = q1
)
```

```
add(S1,S2);
```

```
table(
  ab = p2,
  bb = q3,
  a = p1 + q1,
  ba = p3 + q2
)
```

If we consider the two tables for product operation we can get this results $(T_1 \times T_2)$:

```
prod(S1,S2);
```

```
table(
  babb = p3 q3,
  baba = p3 q2,
  baa = p3 q1,
  abbb = p2 q3,
  abba = p2 q2,
  abb = p1 q3,
  aba = p1 q2 + p2 q1,
  aa = p1 q1
)
```

Bibliography

[1] H. Abbad and E. Laugerotte. Symbolic computation on weighted automata. In *JICSSE'04*. BAU Jordan, 2004.

[2] F. Alarcón and D. Anderson. Commutative semirings and their lattices of ideals. *Houston J. Math.*, 20, 1994.

[3] J.-L. Austin. *Quand Dire C'est Faire*. Le Seuil, 1970.

[4] R. Axerold. *The evolution of cooperation*. New York Basic Book, 1984.

[5] P. Bak. *How Nature Works - the Science of self-organized criticaly*. Springer Verlag, 1996.

[6] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[7] A.-L. Barabási, R. Albert, and H. Jeong. Mean-field theory for scale-free random networks. *Physica A*, 272:173–197, 1999.

[8] O. Barreteau and F. Bousquet. Jeux de rôles et validation de systèmes multi-agents. In Hermès, editor, *Ingénierie des systèmes multi-agents - JFIADSMA'99*.

[9] N.A. Bass. Emergence, hierarchies and hyperstructures. In C.G. Langton, editor, *Artificial Life III, SFI Studies in the Sciences of Complexity*, volume XVII, pages 775–797. Addison Wesley, 1994.

[10] J. Berstel and C. Reutenauer. *Rational Series and Their Languages*. EATCS, Monographs on Theoretical Computer Science. Springer Verlag, 1988.

[11] C. Bertelle, A. Cardon, and D. Olivier. Modélisation et implémentation des systèmes complexes. Cours du DEA Informatique Théorique et Applications, Ecole Doctorale SPMI Rouen-Le Havre, 184 pages, 2001.

[12] C. Bertelle, A. Dutot, F. Guinand, and D. Olivier. Dimants: a distributed multi-castes ant system for dna sequencing by hybridization. In *Nettab 2002, aamas 2002 conf.*, Bologna (Italy), July 2002.

[13] C. Bertelle, A. Dutot, F. Guinand, and D. Olivier. Distribution of agent based simulation with colored ant algorithm. In *Ess 2002 conf.*, Dresden (Germany), October 2002.

[14] C. Bertelle, A. Dutot, F. Guinand, and D. Olivier. Dna sequencing hybridization based on multi-castes ant system. In *Bixmas 2002, aamas 2002 conf.*, Bologna (Italy), July 2002.

[15] C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty. Automata with multiplicities as behaviour model in multi-agent simulations. In *SCI*, Orlando (USA), 2001.

[16] C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty. Genetic algorithms on automata with multiplicities for adapta tive agent behavior in emergent organisations. In *SCI*, Orlando (USA), 2001.

[17] 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.

[18] C. Bertelle, V. Jay, S. Lerebourg, D. Olivier, and P. Tranouez. Dynamic clustering for auto-organized structures in complex fluid flows. In *ESS 2002 Conf.*, Dresden (Germany), October 2002.

[19] C. Bertelle, V. Jay, and D. Olivier. Une approche multi-agent pour la simulation d'environnement estuarien. In *Colloque seine-aval*, page 40, Rouen (France), November 17-19 1999.

[20] C. Bertelle and D. Olivier. Les simulations multi-agents : concept et outil de modélisation non-linéaire pour l'émergence de systèmes organisés. In *3ème colloque chaos temporel et chaos spatio-temporel*, Le Havre, Septembre 2001.

[21] C. Bertelle, D. Olivier, V. Jay, P. Tranouez, and A. Cardon. A multi-agent system integrating vortex methods for fluid flow computation. In *16th imacs congress 2000*, volume 122-3, Lausanne (Switzerland), August 21-25 2000. electronic edition.

[22] C. Bertelle, D. Olivier, P. Tranouez, and V. Jay. Agent-based simulation of water flow for environment modelling in estuaries. In *Workshop 2000 agent-based simulation*, pages 115–122, Passau (Germany), May 2-3 2000.

[23] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence - from natural to artificial systems*. Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, 1999.

[24] E. Bonabeau, F. Henaux, S. Guérin, D. Snyers, P. Kuntz, and G. Théraulaz. Routing in telecommunications networks with "smart" ant-like agents. In *Proceedings of Intelligent Agents for Telecommunications Applications'98*, 1998.

[25] E. Bonabeau and G. Theraulaz. *Intelligence collective*. Hermès, 1994.

[26] T.R.J. Bossomaier and D.G. Green, editors. *Complex systems*. Cambridge university press, 2000.

[27] N. Bourbaki. *Algebra Ch 1-3*. Springer, 1989.

[28] N. Bourbaki. *Theory of sets*. Springer, 2004.

[29] T. Bouron. *Structures de communication et d'organisation pour la coopération dans un univers multi-agents*. PhD thesis, Pierre et Marie Curie University, 1993.

[30] J.-P. Briot. Modélisation et classification de langages de programmation concurrente objets : L'expérience Actalk. In *LMO'94*, pages 103–125, Grenoble, 1994.

[31] J.-P. Briot and R. Guerraoui. Objets pour la programmation paralléle et rèpartie : Intérets, évolutions et tendances. *TSI*, 15(6):765–800, 1996.

[32] Jean-Pierre Briot and Yves Demazeau, editors. *Principes et architecture des systèmes multi-agents*. Hermès, 2001.

[33] J.P. Briot. Actalk : A testbed for classifying and designing actor languages in the smalltalk-80 environment. pages 3–15, Nottingham, UK, 1989. ECOOP'89.

[34] F. Capra. *The web of life*. Anchor books, 1996.

[35] A. Cardon. Modélisation des systèmes adaptatifs par agents : vers une analyse-conception orientée objet. Rapport de recherche 011, LIP6, 1998.

[36] A. Cardon. Système de gestion de crises coopératif : un processus d'interprétation de points de vues multiples. *Journal of Decision Systems, Hermès*, 1:39–67, 1998.

[37] A. Cardon. *Conscience artificielle et systèmes adaptatifs*. Eyrolles, 2000.

[38] A. Cardon and F. Lesage. Toward adaptive information systems : considering concern and intentionality. In *KAW'98*, Banff, Canada, 1998.

[39] A. Cardon and J.-P. Vacher. Genetic algorithm using multi-objective in a multi-agent system. *Robotics and Autonomous Systems, Elsevier*, 1999.

[40] P. Cariani. Emergence and artificial life. In C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors, *Artificial Life II, SFI Studies in the Sciences of Complexity*, volume X, pages 515–537. Addison Wesley, 1991.

[41] G. Di Caro and M. Dorigo. Antnet: A mobile agents approach to adaptive routing. Technical report, IRIDIA, Université libre de Bruxelles, Belgium, 1997.

[42] J.-M. Champarnaud and G. Duchamp. Brzozowski's derivatives extended to multiplicities. In *Lectures Notes in Computer Science*, volume 2494, pages 52–64, 2001.

[43] J.-M. Champarnaud and G. Duchamp. Derivatives of rational expressions and related theorems. *T.C.S.*, (313):31, 2004.

[44] J.-M. Champarnaud and G. Hansel. Automate, a computing package for automata and finite semigroups. *J. Symbolic Comput.*, 12:197–220, 1991.

[45] J.-M. Champarnaud, J.-L. Ponty, and D. Ziadi. From regular expressions to finite automata. *International Journal of Computer Mathematics*, 72:415–431, 1999.

[46] G. Clergue. *L'apprentissage de la complexité*. Hermès, 1997.

[47] J. Colloc. Un système multi-agents neuronal: vers des systèmes d'information épigénétiques,. *Revue Systèmes d'Information et Management, ESKA*, 5(4):55–71, 2000.

[48] J. Colloc and C. Bertelle. Multilayer agent-based model for decision support system using psychological structure and emotional states. In *ESMc 2004*, Paris, 2004.

[49] J. Colloc and M.-H. Cuvelier. A multi-agent approach of modelling metacognition and emotion in education. In *SEG'04 in JICCSE*, Al-Salt, Jordan, 2004.

[50] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. second Edition, 2003.

[51] D. Costa and A. Hertz. Ant can colour graphs. *JORS*, (48):295–305, 1997.

[52] M. Cotsaftis. *Comportement et contrôle des systèmes complexes: Introduction aux méthodes algébriques, qualitatives et fonctionnelles*. Sciences an actes: mathématiques pour l'ingénieur. Diderot arts et sciences, 1980.

[53] G.H. Cottet and P.D. Koumoutsakos. *Vortex methods - theory and practice*. Cambridge University Press, 2000.

[54] J. Crutchfield. Discovering coherent structures in nonlinear spatial systems. In A. Brandt, S. Ramberg, and M. Shlesinger, editors, *Non linear dynamics of ocean waves*, pages 190–216, Singapore, 1992. World scientific.

[55] K. Culik and J. Kari. Finite state transformations of images. In *Proceedings of ICALP 95*, volume 944 of *Lecture Notes in Comput. Sci.*, pages 51–62. Springer, 1995.

[56] K. Culik and J. Kari. Image compression using weighted finite automata. In G. Rozenberg and A. Salomaa, editors, *Handbook of formal languages*, pages 599–616. Springer, 1997.

[57] J. de Rosnay. *Le macrocosme*. Points - Essais. Editions du Seuil, 1990.

[58] Y. Demazeau. From interactions to collective behaviour in agent-based systems. In *Proceedings of the European Conference on Cognitive Science*, Saint Malo, 1995.

[59] M. Dorigo. *Optimization, learning and natural algorithms*. PhD thesis, Department of Electronics Politecnico di Milano, Italy, 1992. In italian.

[60] M. Dorigo and L.M. Gambardella. Ant colony system: A cooperative learning approach to the traveling saleman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[61] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Trans. Systems Man Cybernet.*, 26:29–41, 1996.

[62] A. Drogoul. *De la simulation multi-agents à la résolution collective de problèmes*. PhD thesis, Université de Paris 6, 1993.

[63] A. Drogoul and C. Dubreuil. Eco-problem-solving: results of the n-puzzle. In Y. Demazeau and E. Werner, editors, *Decentralized Artificial Intelligence III*, pages 283–295. North Holland, 1992.

[64] A. Drogoul and C. Dubreuil. Eco-problem-solving: results of the n-puzzle. In Y. Demazeau and E. Werner, editors, *Decentralized Artificial Intelligence III*, pages 283–295. North Holland, 1992.

[65] Alexis Drogoul. *Systèmes multi-agents situés*. Habilitation à diriger des recherches, Université Pierre et Marie Curie, 2000.

[66] G. Duchamp, M. Flouret, and E. Laugerotte. Operations over automata with multiplicities. In J.-M. Champarnaud, D. Maurel, and D. Ziadi, editors, *WIA'98*, volume 1660 of *Lecture Notes in Computer Science*, pages 183–191. Springer-Verlag, 1999.

[67] G. Duchamp, M. Flouret, E. Laugerotte, and J.-G. Luque. Direct and dual laws for automata with multiplicites. In *Theoret. Comput. Sci*, pages 105–120, 2001.

[68] G. Duchamp, Hatem Hadj Kacem, and Éric Laugerotte. On the erasure of several letter-transitions. In *JICSSE'04*. BAU Jordan, 2004.

[69] G. Duchamp and C. Reutenauer. Un critère de rationalité provenant de la géometrie noncommutative. *Invent. Math.*, 128:613–622, 1997.

[70] G.H.E. Duchamp, H. Hadj Kacem, and E. Laugerotte. Algebraic elimination of ϵ-transitions. *DMTCS*, 7(1):51–70, 2005.

[71] S. Durand, F. Lesage, , and C. Moulin. Utilisation des systèmes multiagents dans la modélisation des systèmes adaptatifs. pages 475–480, Bucarest, Mai 1997. International Symposium of Economics and Informatics.

[72] S. Eilenberg. *Automata, languages and machines*, volume Vol A. Academic Press, 1974.

[73] S. Eilenberg. *Automata, languages and machines*, volume Vol B. Academic Press, 1976.

[74] J. Ferber. *Les systèmes multi-agents : vers une intelligence collective*. InterEditions, 1995.

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

[76] M. Flouret. *Contribution à l'algorithmique non commutative*. Thèse de l'université de rouen, 1999.

[77] S. Forrest and T. Jones. Modeling complex adaptive systems with echo. *Complex Systems : Mechanisms of Adaptation*, pages 3–21, 1994. R.J. Stonier and X.H. Yu, eds.

[78] Stephanie Forrest, editor. *Emergent Computation*. MIT Press, 1991.

[79] S. Gaubert. A few introductive texts on (Max, +) algebra and discrete events systems. http://Amadeus.inria.fr.

[80] J. S. Golan. *Power algebras over semirings with applications in Mathematics and Computer Science*. Kluwer, 1999.

[81] J. S. Golan. *Semirings and affine equations over them: Theory and applications*. Kluwer, 2003.

[82] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[83] M. Gondran and M. Minoux. *Graphes et algorithmes*. Eyrolles, 1979.

[84] D.M. Gordon. The expandable network of ant exploration. *Animal Behaviour*, 50:995–1007, 1995.

[85] Z. Guessoum. *Un environnement opérationnel de conception et de réalisation de systèmes multi-agents*. PhD thesis, Pierre et Marie Curie University, 1996.

[86] Z. Guessoum. Dima, une plate-forme multi-agents en smalltalk. *Objet*, 3(4):393–409, 1997.

[87] Z. Guessoum. A hybrid agent model: reactive and cognitive behavior. In *ISADS'97*, pages 25–32, 1997.

[88] Z. Guessoum and J.-P. Briot. From concurrent objects to autonomous agents. In *8^{th} European Workshop on Modelling Autonomous Agents in a Multi-Agent World*, Ronneby, 1997.

[89] O. Gutknecht and J. Ferber. Madkit: Organizing heterogeneity with groups in a platform for multiple multi-agents systems. Technical report, LIRMM, Montpellier University, http://www.madkit.org, 1997.

[90] H. Haken. *Advanced synergetics*. Springer-Verlag, 1983.

[91] T. Haynes and S. Sen. Evolving behaviour strategies in predators and prey. In G. Weiss and S. Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, LNAI, pages 113–126, IJCAI'95 Workshop, Montreal, Canada, August 1995. Springer.

[92] J.-C. Heudin. *La vie artificielle*. Hermès, 1994.

[93] M. Heusse, D. Snyers, S. Guérin, and P. Kuntz. Adaptive agent-driven routing and load balancing in communication networks. In *Proceedings of the 1st International Workshop on Ant Colony Optimization, Oct. 15-16, Brussels, Belgium*, 1998.

[94] A. Heyting. Die Theorie der Linearen Gleichungen in einer Zahlenspezies mit nichtkommutativer Multiplikation. *Math. Ann.*, 98:465–490, 1927.

[95] J. Holland. *Adaptation in natural and artificila systems*. University of Michigan Press, Ann Arbor, 1975.

[96] J. H. Holland. *Adaptation in natural and artificial systems: an introduction analysis with applications to biology, control and artificial intelligence*. University of Michigan Press, 1975.

[97] John H. Holland. *Hidden Order - How adaptation builds complexity*. Helix Book, 1995.

[98] J.E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to automata theory, languages and computation*. Addison-Wesley, 2001.

[99] S. Huss-Lederman, E.M. Jacobson, J.R. Johnson, A. Tsao, and T. Turnbull. Implementation of Strassen's Algorithm for Matrix Multiplication. In *Proceeding of the ACM/IEEE conference on supercomputing*, Pittsburgh, Pennsylvania, USA, 1996.

[100] M.R. Jean. Emergence et sma. In *JFIADSMA'97*, 1997.

[101] K. Jensen. *Coloured Petri Net - Basic concepts, analysis method and practical use*. Springer Verlag, 1997.

[102] W. Kawasnicki. Evolutionary economics and simulation. In T. Brenner, editor, *Computational techniques for modelling learning in economics*. Kluwer Academic Publisher, 1999.

[103] Khalaf Khatatneh. Construction of a memorized semiring. Master's thesis, University of Rouen DEA ITA Memoir, 2003.

[104] W. Kuich and A. Salomaa. Semirings, Automata, Languages. In *EATCS, Monographs on Theoretical Computer Science*, volume 5. Springer Verlag, 1986.

[105] L. Lamport. *Latex : user's guide and reference manual, 2nd edition*. Addison-Wesley, 1994.

[106] Samuel Landau. *Des AG vers la GA - Sélection darwinienne et Systèmes multi-agents*. PhD thesis, Paris 6, 2003.

[107] Samuel Landau and Sébastien Picault. Developping Agents Populations with Ethogenetics (accepted). In *Proc. of the Workshop on Radical Agent Concepts*, 2001.

[108] C. Langton. Studying artificial life with cellular automata. *Physica D*, 22, 1986.

[109] C.G. Langton, editor. *Artificial Life*. Addison Wesley, 1987.

[110] E. Laugerotte. *Combinatoire et calcul symbolique en théorie des représentations*. Thèse de l'université de rouen, 1997.

[111] E. Laugerotte and H. Abbad. Mupad-Automat. http://mupad-combinat.sourceforge.net/.

[112] J.-L. Le Moigne. *Modélisation des systèmes complexes*. Dunod, 1994.

[113] J.-L. LeMoigne. *La modélisation des systèmes complexes*. Bordas, Paris, 1990.

[114] A. Leonard. Vortex methods for flow simulation. *J. Comp. Phys.*, 37:289–335, 1980.

[115] S. Lerebourg. Clustering dynamique appliqué aux écoulements fluides complexes. Master's thesis, Le Havre University, 2002.

[116] M. Lothaire. *Algebraic Combinatorics on Words*. Cambridge University Press, April 2002.

[117] M. Lothaire. *Combinatorics on words*. Cambridge University Press, January 2002.

[118] J.-G. Luque. *Monoïdes et automates admettant un produit de mélange*. Thèse de l'université de rouen, 1999.

[119] S. Mac Lane. *Categories for the Working Mathematician*. Springer, 1988. 4th ed.

[120] P. Marcenac. Modélisation de systèmes complexes par agents. *T.S.I.*, 16(8):1013–1037, 1997.

[121] Pierre Marcenac. *Modélisation et simulation par agents - application aux systèmes complexes*. Habilitation à diriger des recherches, Université de la Réunion, 1997.

[122] O. Matz, A. Miller, A. Potthoff, W. Thomas, and E. Valkena. Report on the Program AMore. Technical report, Institut für Informatik und Praktische Mathematik, Christian-Albrechts Universität, 1995.

[123] N. Minar, R. Burkhart, Ch. Langton, and M. Askenazi. The swarm simulation system : A toolkit for building multi-agent simulations. Technical report, SantaFe Institute, 1996.

[124] M. Minsky. *La société de l'esprit*. 1960.

[125] M. Mohri. Finite-state transducers in language and speech processing. *Journal of Computational Linguistics*, 23(2):269–311, 1997.

[126] E. Morin. *La méthode I : la nature de la nature*. Seuil, 1977.

[127] J.P. Müller. Vers une méthologie de conception de systèmes multi-agents de résolution de problèmes par émergence. In Editions Hermès, editor, *JFIADSMA'98*, pages 355–371, 1998.

[128] M. E. J. Newman, C. Moore, and D. J. Watts. Mean-field solution of the small-world network model. *Phys. Rev. Lett.*, 84:3201–3204, 2000.

[129] M.E.J. Newman. Models of the small world: a review. *J. Stat. Phys.*, 101:819–841, 2000.

[130] P. Nicopolitidis, M.S. Obaidat, and G.I. Papadimitriou. *Wireless Networks*. John Wiley & Sons, 2003.

[131] M.S. Obaidat and G.I. Papadimitriou, editors. *Applied System Simulation*. Kluwer, 2003.

[132] C. Olivier and C. Bertelle. Lagrangien model of suspended matter in a fluvial out-flow used with a multi-agent system. In *Ess'2001*, Marseilles, France, Octobre 2001.

[133] D. Olivier and C. Bertelle. Modèles d'identification et d'évolution de structures hydrodynamiques dans des flux complexes par des systèmes multi-agents. In *Xxvii colloque de l'union des océanographes de france*, Villeneuve d'Asq, Septembre 2001.

[134] D. Olivier, V. Jay, and C. Bertelle. Distributed multi-agent system used for dynamic aquatic simulation. In D.P.F. Müller, editor, *Ess'2000 congress*, pages 504–508, Hambourg (Germany), September 28-30 2000.

[135] A. Ortony, G.L. Clore, and A. Collins. *The cognitive structure of emotions*. Cambridge University Press, 1988.

[136] G.I. Papadimitriou, P.A. Tsimoulas, and A.S. Obaidat, M.S.and Pomportsis. *Multiwavelength Optical LANs*. Wiley, 2003.

[137] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice Hall, 1981.

[138] Philippe Preux. *Réflexions sur les systèmes complexes comme outil d'optimisation, leur modélisation et leur simulation*. PhD thesis, Université du Littoral Côte-d'Opale, 1999.

[139] I. Prigogine. *La fin des certitudes*. Editions Odile Jacob, 1996.

[140] I. Prigogine and D. Kondepudi. *Thermodynamique, des moteurs thermiques aux structures dissipatives*. Editions Odile Jacob, 1999.

[141] I. Prigogine and I. Stengers. *La nouvelle alliance*. Folio essais, 1979.

[142] D.R. Raymond and D. Wood. Grail: A C++ library for automata and expressions. *J. Symbolic Comput.*, 17:341–350, 1994.

[143] A. R. Richardson. Simultaneous Linear equations over a division ring. *Proc. Lond. Math. Soc.*, 28:395–420, 1928.

[144] J. Sakarovitch. *Eléments de théorie des automates*. 2003.

[145] A. Salomaa and M. Soittola. *Automata-Theoretic Aspect of Formal Power Series*. Springer-Verlag, New York, 1978.

[146] R. Schoonderwoerd, O. E. Holland, and J. L. Bruten. Ant-like agents for load balancing in telecommunications networks. In *Proceedings of the 1st ACM International Conference on Autonomous Agents, Feb. 5-8, Marina del Rey, CA, US*, pages 209–216, 1997.

[147] M.P. Schützenberger. On the definition of a family of automata. *Inform. and Control*, 4:245–270, 1961.

[148] J. R. Searle. *Speech Acts*. Cambridge University, 1969.

[149] R. Sedgewick. *Algorithms in C*. Addison-Wesley Publishing Company, 1990.

[150] Y. Shoham. Agent oriented programming. *Journal of Artificial Intelligence*, 60:51–92, 1993.

[151] G. Theraulaz and F. Spitz, editors. *Auto-organisation et comportement*. Hermès, 1997.

[152] R. Thom. *Stabilité structurelle et morphogénèse*. InterEditions, 1977.

[153] P. Tranouez. *Contributin à la modélisation et à la prise en compte informatique de niveaux de descriptions multiples*. Thèse de l'université du havre, Le Havre University, Le Havre university, 2005.

[154] P. Tranouez, C. Bertelle, and D. Olivier. Changing the level of description of a fluid flow in a agent-based simulation. In *ESS 2001 Conference*, Marseilles (France), October 2001.

[155] P. Tranouez, S. Durand, F. Lesage, and A. Cardon. Représentation par des organisations d'agents des connaissances échangées dans un système d'information. In Hermès, editor, *Proceedings of JFIADSMA'99*, 1999.

[156] F. Varela. *Autonomie et Connaissance, Essai sur Le Vivant*. 1989.

[157] L. von Bertalanffy. *Théorie générale des systèmes*. Dunod, 1973.

[158] K. Walkowiak. Graph coloring using ant algorithms. In *Proceedings of the Conference on Computer Recognition Systems KOSYR, Miłkòw, May 28-31*, pages 199–204, 2001.

[159] B. Walliser. *Systèmes et Modèles*. Seuil, Paris, 1977.

[160] G. Weisbuch. *Complex systems dynamics*. Santa Fe Institute Studies in the sciences of complexity. Addison-Wesley, 1991.

[161] G. Weiss, editor. *Multiagent Systems*. MIT Press, 1999.

[162] E. O. Wilson. *Consilience: the unity of knowledge*. Alfred A. Knopf, 1998.

[163] W. Woods. Transition network grammars for natural language analysis. *Communication of Association for Computing Machinery*, 13(10):591–606, 1970.

[164] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley & Sons, LTD, 2002.

[165] M. Wooldridge and N.R. Jennings. Intelligents agents : theory and practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

Khalaf KHATATNEH

Operators for complex modeling

ABSTRACT: The aim of this work concerns the definition of efficient operators within Computer Science for the automatic treatment of self-organized phenomena which appear in complex systems. For that purpose, we define efficient algebraic data structures as automata with multiplicities and tables which are a generalization of k-sets of Eilenberg. This work shows practical applications concerning economic aspects. A first application class is the use of automata in game theory based on probabilistic automata which evolve with genetic algorithms and can produce models for adaptive strategies. These strategies are a kind of more general models which mixed cooperative and competitive aspects. So the model proposed is versatile and give the bases of a generic framework for modelling many kinds of interacting agents in various systems. With this first kind of development for economic model, we show how to use many simple automata and how they are able to generate by interaction, a kind of self-organization. The second kind of developement uses a more sophisticated model based on cognitive sciences. The aim is to build a framework for decision support system as a complex system where knowledge database interact with decision process and where emotional aspects interact too. This work describes some bases for operators for complex system modeling and sketches some innovative methods for various applications which deal with self-organization and complexity of description for natural and artificial systems.

KEYWORDS: Complex systems, automata with multiplicities, genetic algorithms, decision support systems, self-organization.

Opérateurs pour modéliser la complexité

RESUME : L'objectif de ce travail consiste à définir des opérateurs informatiques efficaces pour le traitement automatique des phénomènes d'auto-organisation qui se développent au sein des systèmes complexes. Pour cela nous proposons des structures de données algébriques efficaces que sont les automates à multiplicités et les tables qui sont une généralisation des k-ensembles d'Eilenberg. Ce travail propose aussi des applications pratiques dans le domaine économique. Une première classe d'application concerne l'utilisation des automates en théorie des jeux basée sur des automates probabilistes qui évoluent avec des algorithmes génétiques et peuvent produire des stratégies adaptatives. Ces stratégies sont une sorte de modèles généraux qui mixtent des aspects coopératifs et compétitifs. Ainsi ce modèle est générique et donne une base pour une plate-forme générique pour modéliser de nombreuses sortes d'agents en interaction dans des systèmes variés. Avec ce premier type de développement de modèles économiques, nous montrons comment utiliser de nombres automates simples et comment ils sont capables de générer par interaction, une sorte d'auto-organisation. Le second type de développement utilise un modèle plus sophistiqué basé sur les sciences cognitives. Le but est de construire une plate-forme pour des systèmes d'aide à la décision en tant que système complexe où une base de données interagit avec des processus de décision et où des aspects émotionnels interagissent aussi. Ce travail décrit quelques bases pour des opérateurs pour modéliser la complexité et dresse quelques méthodes innovantes pour des applications variées qui concernent l'auto-organisation et la complexité de la description des systèmes naturels et artificiels.

MOTS-CLEF : Systèmes complexes, automates à multiplicités, algorithmes génétiques, systèmes d'aide à la décision, auto-organisation.