Dynamic Data Structures for Complex Systems

Luaï Jaff

University Of Le Havre

****

PhD Thesis

Speciality : Computer Science

Presented by

Luai JAFF

Subject :

Dynamic Data Structures for Complex Systems

Defended on March 30, 2007
For the obtention of
PhD degree of Le Havre University - France

Jury composition

Pr. Michel COTSAFTIS	*Referee*
Pr. Younès BENNANI	*Referee*
A.Pr. Eric LAUGEROTTE	*Examiner*
A.Pr. Zaid ODIBAT	*Examiner*
Pr. Gérard DUCHAMP	*Advisor*
Pr. Cyrille BERTELLE	*Co-advisor*

****

January 31, 2007

Université du Havre

UFR Science et Techniques - Ecole Doctorale SPMII

****

Thèse de Doctorat d'Informatique

Présenté et soutenue publiquement le 30 Mars 2007
pour l'obtention du

Doctorat de l'université du Havre

Par

Luai JAFF

Sujet :

Structures de Données Dynamiques pour les Systèmes Complexes

Composition du jury

Pr. Michel COTSAFTIS	***Rapporteur***
Pr. Younès BENNANI	***Rapporteur***
A.Pr. Eric LAUGEROTTE	***Examinateur***
A.Pr. Zaid ODIBAT	***Examinateur***
Pr. Gérard DUCHAMP	***Directeur de thèse***
Pr. Cyrille BERTELLE	***Co-directeur de thèse***

****

Contents

List of Figures

Acknowledgements

I would like to thank many people for helping me during my PhD thesis. I also take this opportunity to thank a number of persons who have actively facilitated and contributed to the development of this work.

My first largest and sincere debt of gratitude goes to professor **Gérard H.E. Duchamp** and professor **Cyrille BERTELLE** for welcoming me into their group, and for their knowledgeable guidance throughout these years. I would also like to express my gratitude for their close supervision and guidance of the work presented herein, along with the preparation of more work and papers. I'm proud to be one of their students.

This work would not have been possible without my dear **Dr. Fuad Hussein**. I thank him for having helped me since many years by his support and friendly advices.

I would like to thank all members of the jury; professor **M. Cotsaftis**, professor **Y. Bennani**, A.professor **Z. Odibat** and A.professor **E. Laugerotte** for their participation in the jury.

Many thanks to my family, for all their love and support which they have given to me, without forgetting my sister and friend **Nasren Jaff** who was very close to me during all the difficult times.

To all my friends in France, It is a pleasure to be friend with you. I would like to thank **Cécile Coursieras** for all things.

Finally, I would like to **thank** all my colleagues in **LITIS** for everything.

Many thanks to all of you

Résumé

Mon travail porte sur la dynamique de certaines structures de données et sur les systèmes complexes. Nous avons présenté une approche de la combinatoire des tableaux et des permutations basée sur la dynamique. Cette approche, que nous appelons **Structures de Données Dynamiques** nous ouvre la porte vers des applications en économie via les systèmes complexes.

Les structures de données que nous avons étudiées sont les permutations qui ne contiennent pas de sous-suite croissante de longueur plus que deux, les tableaux de Young standards rectangles à deux lignes, les mots de Dyck et les codes qui lient ces structures de données.

Nous avons proposé un modèle économique qui modélise le bénéfice d'un compte bancaire dont l'énumération des configurations possible se fait à l'aide d'un code adapté. Une seconde application concerne l'évolution de populations d'automate génétique . Ces populations sont étudiées par analyse spectrale et des expérimentations sont données sur des automates probabilistes dont l'évolution conduit à contrôler la dissipation par auto-régulation. L'ensemble de ce travail a pour ambition de donner quelques outils calculatoires liés à la dynamique de structures de données pour analyser la complexité des systèmes.

Mots Clés : Structures de données dynamiques, Systèmes complexes, Automates génétiques, Modèles économiques

Summary

My work concerns the dynamics of some data structures and complex systems. We presented a combinatorial approach of tableaux and permutations based on dynamics. This approach, which we call **Dynamic Data Structures** opens us the door towards applications in economy via complex systems.

The data structures which we studied are the permutations of n letters that have no increasing subsequences of length more than two, the rectangular standard Young tableaux with two lines, the Dyck words and the codes which link these data structures.

We have proposed an economic model which models the benefit of a bank account whose possible enumeration of the configurations is done using an adapted code. The second application deals with the evolution of genetic automata populations. These populations are studied by spectral analysis. Experiments are given on probabilistic automata whose evolution results in controlling dissipation by auto-regulation. The aim of this work is to give some computation tools related to the dynamics of data structures to analyze the complexity of systems.

Key words : Dynamic data structures, Complex systems, Genetic automata, economic models

Introduction

Our work presented here can be decomposed into three parts. The first one deals with bibliographical studies of complex systems and more particularly the complexity in economy. The second one deals with a *"dynamic combinatorial"* approach of evolutive systems that show emergence properties when applied in the domain of economy. The third part covers genetic automata which are based on biological inspired methods, mainly genetic algorithms which consist of building a suited metaphor of life evolution. We use primitive data (Chromosomes) which are sequences of numbers extracted from linear representations of automata. We present some experiments about indirect control based on spectral analysis leading to adjust the dissipative behavior by an auto-regulated process.

Economic Complex systems

Complex systems, as networks of interactive entities, are studied by means of a rapidly increasing amount of data in all domains. At the same time, these domains share a lot of new and fundamental theoretical questions. This situation is especially favorable for developing the new science of complex systems in an interdisciplinary way. There are two kinds of interdisciplinarity within complex systems. The first begins with a particular complex system and addresses a variety of questions coming from its particular domain and point of view. The second kind addresses issues that are fundamental to complex systems in general. The first kind leads to domain-specific interdisciplinary fields such as cognitive science. The new science of complex systems belongs to the second kind of interdisciplinarity. It starts from fundamental open questions relevant to many domains, and searchers for methods to deal with them. Extract from "Towards a science of complex systems", P. Bourgine, F. Képès and M. Schoenauer, European Conference on complex systems, Paris, November 14-18, 2005.

In our century, complex systems are considered as an innovative concept. The theory of systems extends over a larger expanse in spectrum of application [54].

Evolutive systems, agent-based representation are the new illuminating methodologies that are used for complex modeling.

Dynamic Data Structures

In 1962 Schensted discovered a bijective relation between the permutations on [n] and the pairs of standard Young tableaux of the same shape known under the name of the correspondence of Robinson-Schensted. The study of the combinatorial structures like permutations, standard Young tableaux and Dyck words can sometimes reveal new connections between them. In this PhD, we determinate bijections that allow us to go from codes to permutations of symmetric groups, rectangular Young tableaux of height two and Dyck word by a simple rule.

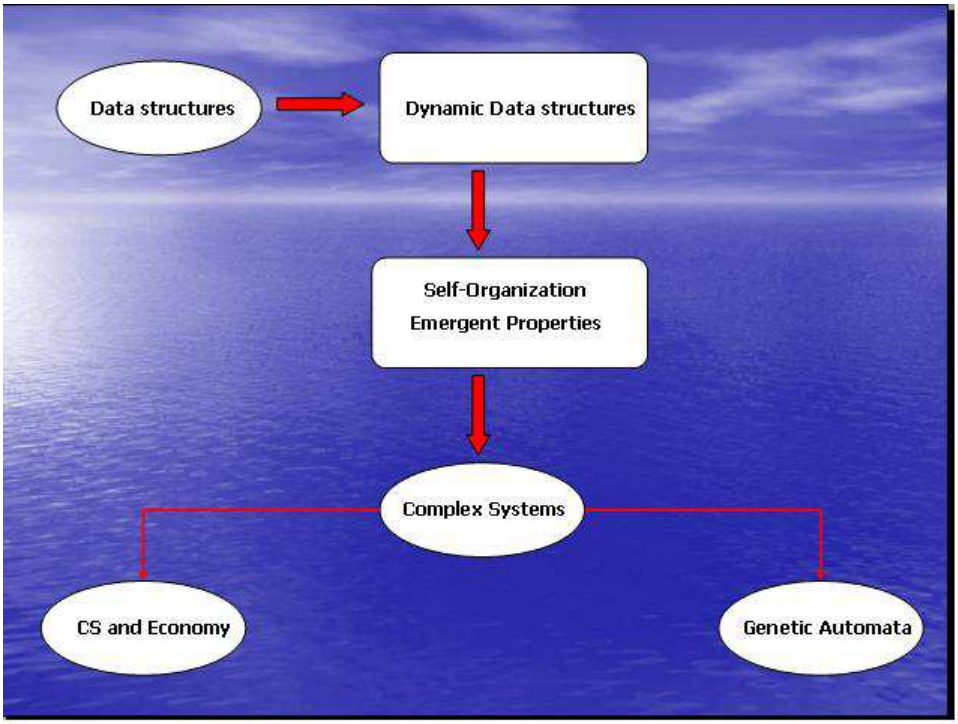
We can present our codes as a family of time series with a simple growth constraint. This family can be the basis of a model to apply to emerging computation in business and micro-economy where global functions can be expressed from local rules. We explicit a double statistics on these series which allows to establish a one-to-one correspondence between three other ballot-like structures mentioned above. We can see our codes as an economic model, this means that the codes are the behavior of a person which earns money every month without loss in his capital.

In this thesis, we are interested with time series with moderate growth but possibly sudden decay. We will focus on a very simple model (a toy model as physicists may say), the combinatorics of which is completely harnessed. This feature is important as one may use simulations and estimates over all the possible configurations, as it is the case, for example, for other combinatorial models (Cox-Ross-Rubinstein, for instance). The model is that of sequences with integer values and growth bounded by a unit (local rule). Surprisingly, there is one-to-one correspondences between the possible configurations and planar combinatorial objects which are endowed with a special dynamics which we describe in this work.

Genetic Automata

The origin of genetic algorithms is credited to J. Holland in the end of 1960s. The aim of genetic algorithms is to obtain solutions to optimization problems. They can be adapted to hard non-linear problems where local optima exist. Genetic operators processes on individual data and selection operators processes on primitive data population. Automata with multiplicities are powerful algebraic structures on

which we can apply various operators such as genetic ones.



The structure of this work is as follows :

- Chapter 1 presents a bibliographic study of complex systems and economy. We introduce the notion of complex systems modeling, self-organization and emergence on the one hand. The different ways to look at the economy as a dynamic complex system and feedback mechanism in economy on the other hand.

- chapter 2 presents some results about some objects of enumerative combinatorics and automata with multiplicities. We establish a new relationship between these objects. We will study Wilf identity which is a kind of generalization of Robinson-Schenested correspondence. We will present the reduction condition of Robinson-Schenested algorithm, that is, for each permutation $\pi_2(n)$ there is one and only one corresponding rectangular standard tableau of height two.

- In chapter 3, we define a dynamic conception of data structures like, dynamic permutations, dynamic standard tableaux, and dynamic Dyck words enumerated by the famous formula of Catalan. We give bijections that relate all these objects.

- Chapter 4 presents an original result about the application of our codes (presented in chapter 2) in economy. We will build combinatorial structures that allow to model and to compute the global behavior of the reserve account **R** by some specific functions. Another original result about the spectral analysis will be presented in this chapter, it concerns the study of genetic automata by means of their transition matrices to control their dissipative behavior.

Chapter 1

Complex Systems and Economy

Contents

One of the basics of science is the use of the scientific method and the ability to establish hypothesis and make predictions which can be tested experimentally (with observed data). Economic theories deal with many parameters and the expression of universal laws is difficult. Confronted with these problems, Economic theorists test some hypothesis using statistical methods such as econometrics, using the data generated by the real world.

Recent advances in complex systems and computational tools are allowing us new approaches to the study of the economic aspects. Two examples of these recent advances are the **Santa Fe approach** (or **Complexity approach**) and the **Agent-Based Computational Economics approach**, that is, the computational study of economic processes modeled as dynamic systems of interacting agents.

Economies are complex dynamic systems covering micro behaviors, interaction patterns and global regularities. In the study of economic systems the question is how to handle difficult real-world aspects such as strategic interaction, imperfect competition and the possibility of multiple solution (equilibrium points).

1.1 A Multi-level description for Economy

Economy is often described as a social science that seeks to analyze and describe the production and distribution of goods and services. That is, economics studies show how individuals, coalitions and societies seek to satisfy needs and wants. However, the vast number of topics to which the methods of economic theory have been applied has caused some to refer to economics as simply that which economists do.

Economics has two branches :

- **Microeconomics :** The unit of analysis is the individual agent such as a firm
- **Macroeconomics :** The unit of analysis is an economy as a whole

According to **Alfred Marshall**, *economics* is the study of mankind in the ordinary business of life, it examines that part of individual and social actions which is most closely connected with attainment and material requisite of well being, Thus it is on one side a study of wealth and the other side a part of study of man (see http://en.wikipedia.org/wiki/Alfred_Marshall).

1.2 Complex Systems

A **system** is an assemblage of entities (or objects), real (or abstract), comprising a whole with each every element interacting or related to another one. Any entity which has no relationship with any other element of the system is not an element of that system. A **sub-system** is a set of elements which is a system itself and a part of the whole system [88].

we can treat as separate the term complex system and the term complicated system:

- **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 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.

A system is defined to be **complex** if it exhibits the following two properties [50]:

- The system is composed of interacting elements
- The system shows **emergent** properties, that is, properties arising from the interaction of the elements that are not properties of the individual elements themselves

Complex system based on the fact that from many applicative areas, we can find similar processes linking emergent global behavior and interaction network of constituents. The general behavior of complex system is generally not accessible using classical analytical methods like differential systems.

In complex system modeling, we have to model the constituents of the system and the interaction network which link these constituents, using a decentralized approach. So the general behavior of the system can not be understood by the description of each constituent. In complex system modeling, the general behavior is an emergent property of the interaction network.

1.2.1 Complex Adaptive Systems

Leigh Tesfatsion [50], consider a complex adaptive systems **(CAS)** as a complex systems whose elements are:

- **Reactive :** That is, capable of exhibiting systematically different attributes in reaction to changed environmental conditions
- **Goal-directed :** That is, they are reactive and at least some of their internal structural changes are directed towards the achievement of goals
- **Planners :** That is, they are goal-directed and they attempt to exert some degree of control over their environment to facilitate achievement of these goals

1.2.2 Complex Systems Modelling

Complex systems are the systems having multiple interacting components and it is difficult to predict the systems behavior from the behavior of its components. Special tools are required for that. The multi-agent simulation techniques are one of them. Modern studies in complexity generated new kinds of modeling paradigms such as multi-agent systems, Boolean networks, and cellular automata. The applications of the study of complex systems are growing with emerging interfaces in every possible branch, from biological macromolecules to ecosystems, to economics, to socio-economic structures. The particular examples being distributed databases, information management systems, software systems, corporate structures and time series analysis and further prediction. Already the paradigm as acquired an emerging vocabulary of complex systems.

1.3 Self-Organizations Classification

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 [32].

We have four broad classes for self-organization each of which include both natural and artificial processes [82]; 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.

1.3.1 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 [24],[23]. The key to a comprehensive 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.

1.3.2 Inherent Computation

We refer with this denomination to systems which evolve with fixed rules. Usually, this rules are computed using automata [40] or discrete events systems or interacting networks of automata or cellular automata [10, 39]. 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.

1.3.3 Non-Linear and Non-Equilibrium Processes

A description of self-organized systems consists in open systems far from equilibrium. 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[31]. 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 [32]. Initially, he studies living organisms capabilities to maintain live process in non-equilibrium conditions. In 1960s, I. Prigogine points out the link between non-equilibrium 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.

In the financial market, it is possible to make face to two different situations; the first one is known as the **stock market bubble** which is a type of economic bubble taking place in stock markets, in which a wave of public enthusiasm, evolving into herd behavior, causes an exaggerated bull market. When such a bubble takes place, market prices of listed stocks rise dramatically, making them significantly overvalued by any measure of stock valuation. Generally stock market bubbles are followed by stock market crashes.
For example, Some of those bubbles are created because of intense and excessive speculation on a new technology or service. The **Dot-com bubble** of the late 1990s is one example [92].

The second situation is known as the **stock market crash** which is a sudden dramatic decline of stock prices across a significant cross-section of a market. Crashes are driven by panic as much as by underlying economic factors. They often follow speculative stock market bubbles such as the dot-com bubble. The most famous crash, the Stock Market Crash of 1929, started on October 24, 1929 (known as Black Thursday) when the Dow Jones Industrial Average dropped 50 percent.

1.3.4 Evolution

The evolution is often associated which adaptive properties. In normal way, a complex system evolves to satisfy its capability to adapt to its environment. Genetic algorithms and genetic programming open new ways in computing, using some similar processes based on biological systems metaphor.

1.4 Economic System

An economic system is a mechanism (social institution) which deals with the production and distribution of goods and services in a particular society. The economic system is composed of people, institutions and their relationships to resources, such as the convention of property. It addresses the problems of economics, like the allocation of resources.

There are several basic questions such as: *what to produce?, how to produce it?, and who gets what is produced?*. An economic system is a way of answering these basic questions.

There is a correlation between certain ideologies, political systems and certain economic systems. For example, consider the meanings of the term communism. Many economic systems overlap each other in various areas, for example, the term mixed economy can be argued to include elements from various systems.

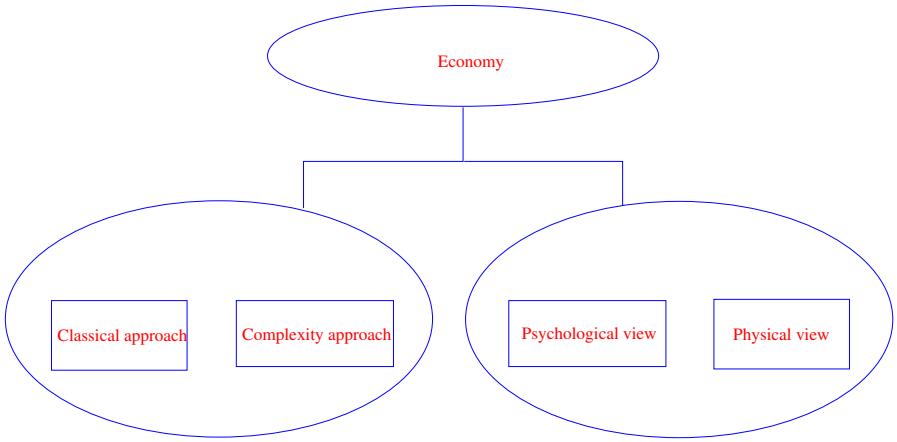
The most basic and general economic systems are; market , mixed , planned , traditional and participatory economics. An economic system can be considered a part of the social system and hierarchically equal to the law system, political system, cultural system, etc [93].

1.4.1 Economic agents

An **agent** is an encapsulated piece of software programs representing physical, individual, social and biological entities that can includes data together with behavioral methods that act on these data.

1.5 Complex System Economics

1.5.1 Complexity in the Economy



We classify the ways to look at the economy in two spaces. The first space is composed of two approaches :

The **classical approach;** based on rational choice of sets of equations representing the state of the economy. There are problems associated with this approach; the unrealistic assumption of rational choice and perfect information; assumptions on homogeneity and lack of distinction between the agent and the aggregate level; inability to account for the emergence of new kinds of entities, patterns, structures, etc.

Complexity perspective approach and **Agent-based Computational Economics approach (ACE);** are the more recent approaches to view economy. These approaches focus on viewing the economy as an evolving complex system (Santa Fe approach). A similar basis is used in Agent-based computational economics **ACE** which uses Holland's complex adaptive systems paradigm, but which concentrates on using computer simulations designed in an object-oriented bottom-up fashion to build a models of the economy (cf.[79],[50]).

With both the **complexity** and **ACE** approach, the aim is to understand how global economic phenomena arise purely from the local interactions and local knowledge

of the agents. This stems from the recognition that there is no central or global control in an economy and that the global regularities which arise are purely due to the local interactions of adaptive autonomous agents. The agents within an economic network may be individuals or institutions. We can consider an institution agent as being composed of many individual agents.

The second space(**standard way**) is composed of two different views :

Psychological (Cognitive) view; Is a collection of beliefs and expectations with decision making upon these beliefs and expectations. Psychological view to the economy is useful because it forces us to think about how beliefs create economic behavior and how economic outcomes create beliefs.

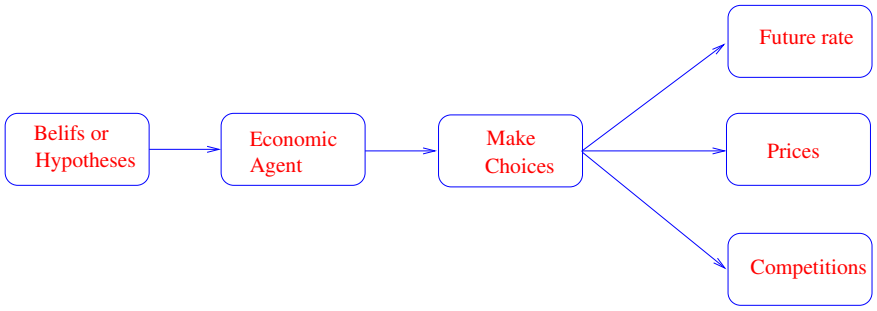


Figure 1.1: Economic agents beliefs are individual, subjective and private

Suppose that we have perfect economic agents, that is, agents who are identical and posses rationality and share beliefs about the situation they face. When these beliefs bring actions that create a world that validates them as hypotheses, they are in equilibrium and are called *Rational beliefs*. The created world may change if some agents deviate [90].

Predictions that an economic agent forms depend on the prediction he believes others economic agents share and so on. The following figure illustrate this fact called also **Self-Referential sequence predictions** as an interacting sequence process.

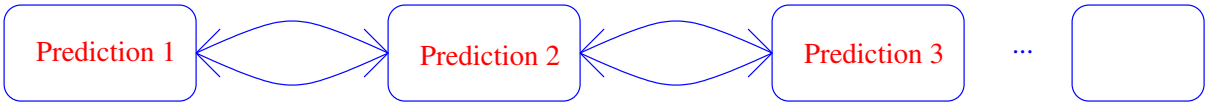


Figure 1.2: Self-Referential sequence predictions

The **guessing game** which is a *toy problem (A toy problem is a simplified version of a more general problem)*, illustrates that *rational beliefs* can unravel easily and lead to a *self-referential sequence predictions* [69].

Physical view; Is a collection of activities and technology, interacting with a market system peopled by decision making agents such as firms, banks, consumers and investors.
There is a relationship between the psychological view and the physical view to the economy as we see in the following figure [90].

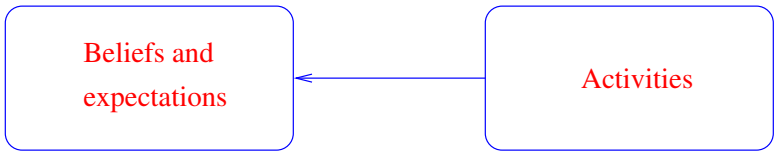


Figure 1.3: Activities follows beliefs and expectations

1.5.2 Feedbacks in the Economy

Recently, **W. Brian Arthur**, professor at the Santa Fe Institute [79], and other economic theorists have been interested to a view of the economy based on feedbacks, more precisely, positive feedback mechanisms. We can say that the modern economies can be divided into two interrelated branches :

- **Diminishing returns**
- **Increasing returns**

These two branches have different economies, they differ in behavior, style, culture and they call for different understandings [89].

Diminishing Returns and Negative Feedback

Definition 1 :

Diminishing returns refers to the notion that the return that a company receives for additional effort decreases as the number of units (outputs) increases. This is typical of industrial goods, but is in contrast to the phenomena of network effects and increasing returns for digital goods. Diminishing returns explains why industrial companies become more inefficient once they grow over a certain size. Thus firms do not compete as effectively when in a large monopolistic market than they do in an oligopolistic market (car company for example) assuming the size of the market is over the scale limit that traditional firms can operate efficiently. With the winner takes all behavior of digital goods, markets can be more efficient when one company supplies the entire market place, especially if the market is governed by proprietary standards versus open standards [14].

Definition 2 :

Negative feedback (**NFB**) is the process of feeding back to the input a part of a system's output, so as to reverse the direction of change of the output. This tends to keep the output from changing, so it is stabilizing and attempts to maintain constant conditions [83].

Diminishing returns assumptions are the basis of the conventional economic theory. The actions of the economy induce a negative feedback which lead to predictable equilibrium for prices and market share. Negative feedback stabilize the economy because any major changes will be cancel out by the reaction they generate. Diminishing returns imply a unique equilibrium point for the economy.

Alfred Marshall believed that we can not applied the increasing returns everywhere in the economy. **Marshall's** remark lead **W. Brian Arthur** [91] to observe that the part of the economy that are **resource-based** (the traditional part) are still subject to diminishing returns, on one hand. And the part of the economy that are **knowledge-based** (the newer part) are still subject to increasing returns, on the other hand as defined in the following.

Increasing Returns and Positive Feedback

Definition 1 :

Increasing returns refers to the notion that the greater the size of the network, the greater the advantage of each participant of the network (network effects). Each participant of the network brings value to the overall network. This is in contrast to diminishing returns which refers to the greater the size (number of users) the less each participant can benefit from participation [15].
In the other word, increasing returns are the tendency for that which ahead to get farther head. For which loses advantage to loss further advantage. Increasing returns generate instability.

Definition 2 :

We call a feedback mechanism positive if the resulting action goes in the same direction as the condition that triggers it. Positive feedback is an open system contain many types of regulatory systems, among which are systems that involve positive feedback and its relative negative feedback [84].

Increasing returns make for multiple solutions (multiple equilibrium points), and it generate instability or criticality stability following Per Bak. Positive feedback economics is parallel to modern non-linear physics [91]. The underlying mechanisms that determine economic behavior have shifted from diminishing returns to increasing returns. Mechanisms of increasing returns exist alongside those of diminishing returns in all industries [89].

Increasing returns mechanisms are important, but the economic theorists have several points of view about the importance of these mechanisms which are some times divergent:

- Some economic theorists found the existence of more than one equilibrium points to the same unscientific problem.
- Economic theorists were perplexed by the question of how a market would select one among several possible solutions.
- Economic theorists could see that increasing returns would destroy their familiar world of unique predictable equilibria.

Example of Positive Feedback Mechanism

Increasing returns are the tendency for that which is a head to get farther ahead. For that which loses to loss further advantage. They are mechanisms of positive feedback that operate. The history of video-cassette recorders **(VCRs)** provides a simple example of positive feedback. When **VCRs** appeared on the market, two competing formats selling at about the same price :

- **VHS**
- **Beta**

Gradually, **VHS** obtained a slight edge in market share. This small lead tended to become self-reinforcing through the following events :

Consumers considering purchasing a **VCRs** noticed slightly more **VHS** format tapes in video stores, because there was slightly more demand for them. This observation increased the probability that consumers would purchase **VHS** machines, increasing their share of the installed product base, further increasing the fraction of **VHS** tapes on the shelves and increasing **VHS** market share even further. The self-reinforcing nature of **VHS** advantage in market share illustrate the economic meaning of positive feedback, as we see in the following figure (1.4). The obvious end result of this process was that **VHS** acquired an overwhelming market share, and the **Beta** format became virtually extinct, a result that tended to be self-perpetuating. The stock of **VHS** format tapes on retail shelves and in consumers home libraries dramatically increased the perceived cost of switchinf to another tape format. this kind of self-perpetuating market dominance is called lock-in.

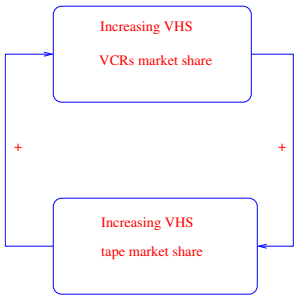


Figure 1.4: Positive feedback in the VCRs market

1.5.3 Introduction to Computational Economics

Another way to look at the economy as dynamic processes is the *Agent-Based Computational Economics* **(ACE)**.

Definition 1 :

(ACE) is a computational study of economic processes as dynamic systems of interaction agents. In the other words, is a computational study of economies modeled as evolving systems of autonomous interacting agents. Thus **ACE** is a specialization to economics of the basic complex adaptive system paradigm.

Definition 2 :

Culture-Dish approach : Is the **ACE** methodology to study the economic system based on complex adaptive systems that include *reactive elements*, *goal-directed elements* and *planner elements*.

The **ACE** modeler construct a virtual economic world composed of multiple interacting agents. The modeler sets initial conditions and then he observe the evolving of the world over time with no imposed rational expectations or beliefs and equilibrium. The world events are driven by agent interactions.

Economic decentralized market as a complex adaptive systems, is composed of large numbers of economic agents involved in parallel local interaction. These interaction which is local give rise to **macroeconomic regularities** such as shared market protocols which in turn feedback into the determination of local interactions.The result is a dynamic complex systems of recurrent causal chains connecting individual behaviors and interaction network.

ACE Research Areas

The topics divide into eight research areas as follows [50]:

- Learning and the embodied mind
- Evolving of behavioral norms
- bottom-up modeling of market processes
- formation of economic networks

- modeling of organizations
- design of computational agents for automated markets
- parallel experiments with real and computational agents
- building **ACE** computational laboratories

1.6 Conclusion

In this chapter, we have made bibliographical studies about complex systems and their use for economic modeling. We have described complexity of feedback mechanism and agent-based computation approach which are recent looks at economy as an evolutive dynamic system.

Chapter 2

Some Data Structures

Contents

This chapter introduces the background material of two terminologies, the first one concerns some combinatorial objects (some Data Structures) and the second one is the notion of automata with multiplicities which generalizes classical Boolean automata.

The purpose of this chapter is to present studied connections diagram. That is, the generalization and the reduction of the Robinson-Schenested algorithm conditions. We can see this chapter as a background material for chapter 3.

2.1 Permutations

A **bi-word** π is a sequence of vertical pairs of some objects. Here we will consider bi-words of positive integers

$$\pi = \begin{pmatrix} i_1 & i_2 & i_3 & ... & i_k \\ j_1 & j_2 & j_3 & ... & j_k \end{pmatrix}$$

with $i_1 < i_2 < i_3 < ... < i_k$. We denote respectively the top and the bottom lines of π by $\hat{\pi} = i_1\ i_2\ i_3\ ...\ i_k$ and $\check{\pi} = j_1\ j_2\ j_3\ ...\ j_k$, also we designate $\pi(i_m) = j_m$ for a pair. For instance

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 2 & 5 & 3 & 1 \end{pmatrix}$$

A **permutation** is a rearrangement of the elements of an ordered list into a one-to-one correspondence with itself. The number of all permutations on $[n] = \{1, 2, ..., n\}$ is given by $n!$. For example, there are $2! = 2.1 = 2$ permutations of $n = 2$, namely 12 and 21, and $3! = 3.2.1 = 6$ permutations of $n = 3$, namely 123, 132, 213, 231, 312 and 321.

There are different ways to represent a permutation [6]:

- array notation or two-line notation : is a notation that explicitly identifies the positions occupied by elements before and after application of a permutation on elements, where the first row is $(1, 2, ..., n)$ and the second row is the new arrangement. For example, the above permutation π is represented by two-line notation.

- Word : providing that the first line be $1\,2\,3\,...\,n$, it is non-ambiguous to represent only the second line
- cycle type : An example of a cyclic decomposition is the permutation 4213 of 1234. This is denoted $(2)(143)$, corresponding to the disjoint cycles (2) and (143) where 2 keeps the same position, but 4 takes the position of 1 and 1 takes the position of 3, finally 3 takes the position of 4, so we have a cycle (143).

The **inverse** of the permutation $\pi = \pi_1\ \pi_2\ \pi_3\ ...\ \pi_n$ is the permutation π^{-1} such that $\pi^{-1}(i) = j \Longleftrightarrow \pi(j) = i$. For example

$$\pi^{-1} = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 2 & 4 & 1 & 3 \end{pmatrix}$$

is the inverse permutation of the above permutation π.

The **mirror permutation** of the permutation $\pi = \pi_1\ \pi_2\ \pi_3\ ...\ \pi_n$ (in the word representation) is the permutation $\pi^* = \pi_n\ \pi_{n-1}\ \pi_{n-2}\ ...\ \pi_1$. For example

$$\pi^* = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 3 & 5 & 2 & 4 \end{pmatrix}$$

is the mirror permutation of the above permutation π.

Involutions

An **involution** π is a permutation which does not contain any permutation cycles of length > 2. It consists exclusively of fixed points and transpositions. Involutions are exactly self-conjugate permutations (i.e, permutations that are their own inverse permutation, $\pi = \pi^{-1}$). For example, the unique permutation involution on 1 element is 1, the two involution permutations on 2 elements are $(1, 2)$ and $(2, 1)$, and the four involution permutations on 3 elements are $(1)(2)(3)$, $(1)(2, 3)$, $(1, 3)(2)$, and $(1, 2)(3)$. We can represent an involution by a graph. For example, the graph representation of

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 2 & 1 & 5 & 4 \end{pmatrix}$$

is showed in the following figure.

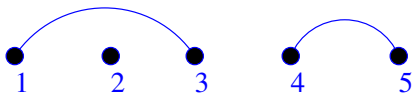


Figure 2.1: Graph representation of (1,3)(2)(4,5)

The generating series of the involutions is

$$\sum_{n,k\geq 0} I(n,k)\frac{x^n}{n!}y^k = e^{y(x+\frac{x^2}{2!})}$$

Where $i(n, k)$ stands for the number of involutions on n objects with k cycles.

2.2 Standard Young Tableaux

2.2.1 Partitions

A **partition** of n, written $\lambda \vdash n$, is a sequence $\lambda = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_k)$ such that the λ_i are decreasing (weakly) and $\sum_{i=1}^{k} \lambda_i = n$.

Let $\lambda = (\lambda_1, \lambda_2, \lambda_3, ..., \lambda_k) \vdash n$. Then the **Ferrers diagram**, or **shape**, of λ is an array of n-squares into k left-justified rows with row i containing λ_i squares for $1 \leq i \leq k$. For example, the partition (4,3,1) is represented as follows

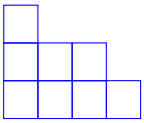


Figure 2.2: Shape of (4,3,1)

The idea of representing a partition by a diagram goes back to Ferrers and Sylvester, and the diagram of a partition is called by a large number of authors the Ferrers diagram or shape [30]. In the literature, we find also a representation different from that of the **French** notation of Ferrers diagram called **anglo-saxon** or **English** notation. For example, the partition (4,3,1) in English notation is represented as follows

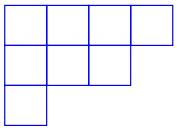


Figure 2.3: Shape of (4,3,1)

In all chapters of this work, we will use the French notation of Ferrers diagrams.

2.2.2 Conjugate Partition

The **conjugate** of a partition λ is the partition λ' whose diagram is the transpose of the diagram λ, that is, the diagram obtained by reflection with respect to the main diagonal (in all cases, $x = y$). Hence λ' is the number of nodes in the i-th column of λ, or equivalently

$$\lambda'_j = Card\{j : \lambda_j \geq i\}$$

For example, the conjugate of the partition (4,3,1) is (3,2,2,1).

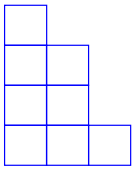


Figure 2.4: Shape of (3,2,2,1)

Let λ be as above. A **standard Young tableau** of shape λ, is an array obtained by filling the squares of the shape λ by the consecutive and not repeated numbers 1,2,...,n, such that, in each row (resp. column), the numbers be increasing from left to right (resp. from bottom to top). For example the following tableau is standard

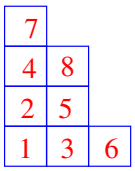


Figure 2.5: Standard tableaux

Tableaux were introduced by A. Young in his series of papers on quantitative substitutional analysis [2].

It is important to note that, in each standard Young tableau of shape $\lambda = (\lambda_1, \lambda_2, ..., \lambda_k) \vdash n$:

- λ_1 is the length of the tableau
- λ_k is the height of the tableau
- n is the size of the tableau

2.2.3 Hook Formula for Standard Tableaux

Is the number of Young tableaux associated with a given Ferrers diagram. In each cell, write the sum of one plus the number of cells horizontally to the right and vertically below the cell (the hook length $h_{(i,j)}$). Let f^λ be the number of standard Young tableaux of shape $\lambda \vdash n$. Then

$$f^\lambda = \frac{n!}{\Pi_{(i,j)\in\lambda} h_{i,j}}$$

The formula above is the general formula called hook formula [6]. We don't prove it here, but we illustrate it by giving the following example :

Let $\lambda = (6, 4, 1) \vdash 11$. The following diagram gives the hook lengths

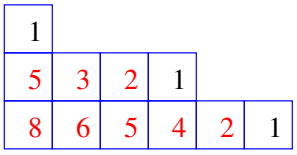


Figure 2.6: Hook Length

The position (1,1) contain the integer 8 which represent the number of cells behind and upon the position (1,1)

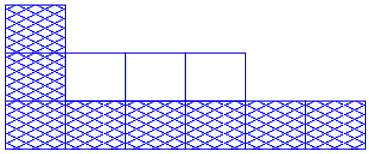


Figure 2.7: 8 cells

So

$$f^\lambda = \frac{n!}{1.2.4.5.6.8.1.2.3.5.1} = \frac{11!}{1.2.4.5.6.8.1.2.3.5.1} = 693$$

2.3 Robinson-Schensted Algorithm

Theorem 1 *There is a bijection between permutations on [n] and the pairs of standard Young tableaux of the same shape [6]*

$$RS : S_n \longrightarrow ST_n \times ST_n$$
$$\pi \longrightarrow (P(\pi), Q(\pi))$$

Therefore, we have

$$\sum_{\lambda \vdash n} (f^\lambda)^2 = n!$$

R-S Algorithm

We will define the map which gives a permutation and produces a pair of standard Young tableaux of the same shape. Let π be the following

$$\pi = \begin{pmatrix} 1 & 2 & . & . & . & n \\ x_1 & x_2 & . & . & . & x_n \end{pmatrix}$$

Then we construct a sequence of tableaux (P_i, Q_i) for $i = 0, 1, ..., n$

$$(P_0, Q_0) = (\emptyset, \emptyset), (P_1, Q_1), ..., (P_n, Q_n) = (P, Q)$$

By inserting the x_i into the P_{i-1} and placing i into the Q_{i-1} at the same place (so that the shape) of P_i is identical to the shape of Q_i by applying the following operations :

Assume P is a partial tableau and x_i an element we wish to insert and Q a partial tableau into which i needs to be placed. Proceed as follows

1. Begin with the first row of P and let $x = x_i$
2. Look for the left most element y in the given row such that $x < y$, substitute x to y, i.e. x bumps y. If no such element is available then place x at the end of the row and i at the corresponding position in Q; reset at 1 by considering the next element x_{i+1}

3. Repeat step 2 with $x = y$ into the next row down of P until there are no bumped elements left

This is repeated for every element of π (see [6]).

Example :

Let

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 4 & 2 & 3 & 6 & 5 & 1 & 7 \end{pmatrix}$$

Then

P :	$\varnothing$	4	$\begin{matrix}4\\2\end{matrix}$	$\begin{matrix}4\\2\ 3\end{matrix}$	$\begin{matrix}4\\2\ 3\ 6\end{matrix}$	$\begin{matrix}4\ 6\\2\ 3\ 5\end{matrix}$	$\begin{matrix}4\\2\ 6\\1\ 3\ 5\end{matrix}$	$\begin{matrix}4\\2\ 6\\1\ 3\ 5\ 7\end{matrix}$
Q :	$\varnothing$	1	$\begin{matrix}2\\1\end{matrix}$	$\begin{matrix}2\\1\ 3\end{matrix}$	$\begin{matrix}2\\1\ 3\ 4\end{matrix}$	$\begin{matrix}2\ 5\\1\ 3\ 4\end{matrix}$	$\begin{matrix}6\\2\ 5\\1\ 3\ 4\end{matrix}$	$\begin{matrix}6\\2\ 5\\1\ 3\ 4\ 7\end{matrix}$

2.3.1 Some Properties due to R-S Algorithm

If we consider the P-tableau and Q-tableau associated to the inverse permutation π^{-1}, we have the following

$$P(\pi^{-1}) = Q(\pi) \ \ and \ \ Q(\pi^{-1}) = P(\pi)$$

Also, if we set π^* to be the reverse (mirror) permutation, we will get the following

$$P(\pi^*) = P^T(\pi) \ \ and \ \ Q(\pi^*) = Q^T_{V.R}(\pi)$$

Where the superscript T means the transpose of the tableau and $V.R$ means that $Q(\pi^*)$ can be computed via the **Vidage-Remplissage** algorithm due to Schutzenberger [56].

Lemma 1 *The number of involutions without fixed point on [2n] is* $2n!! = 1.3.5....2n-1$.

Lemma 2 *[46] The number of standard Young tableaux of size n is equal to the number of involutions on [n]*

$$\sum_{\lambda \vdash n} f^{\lambda} = Inv[n]$$

Proof :

Let π *be an involution, so* $\pi = \pi^{-1}$. *We have* $P(\pi^{-1}) = Q(\pi)$ and $Q(\pi^{-1}) = P(\pi)$, *so we can write*

$$P(\pi) = Q(\pi^{-1}) = Q(\pi)$$

Hence

$$T \Leftrightarrow (T,T) = (P(\pi), Q(\pi)) \Leftrightarrow \{\pi : \pi = \pi^{-1}\}$$

Theorem 2 *Let* π *be a permutation and let* $P(\pi)$ *be P-tableau. Then*

- *The length of the longest increasing subsequence of* π *is the length of the first row of* $P(\pi)$
- *The length of the longest decreasing subsequence of* π *is the length of the first column of* $P(\pi)$

2.4 Studied Connections

In this section, we will study the generalization and the reduction of Robinson-Schensted algorithm represented by the following diagram.

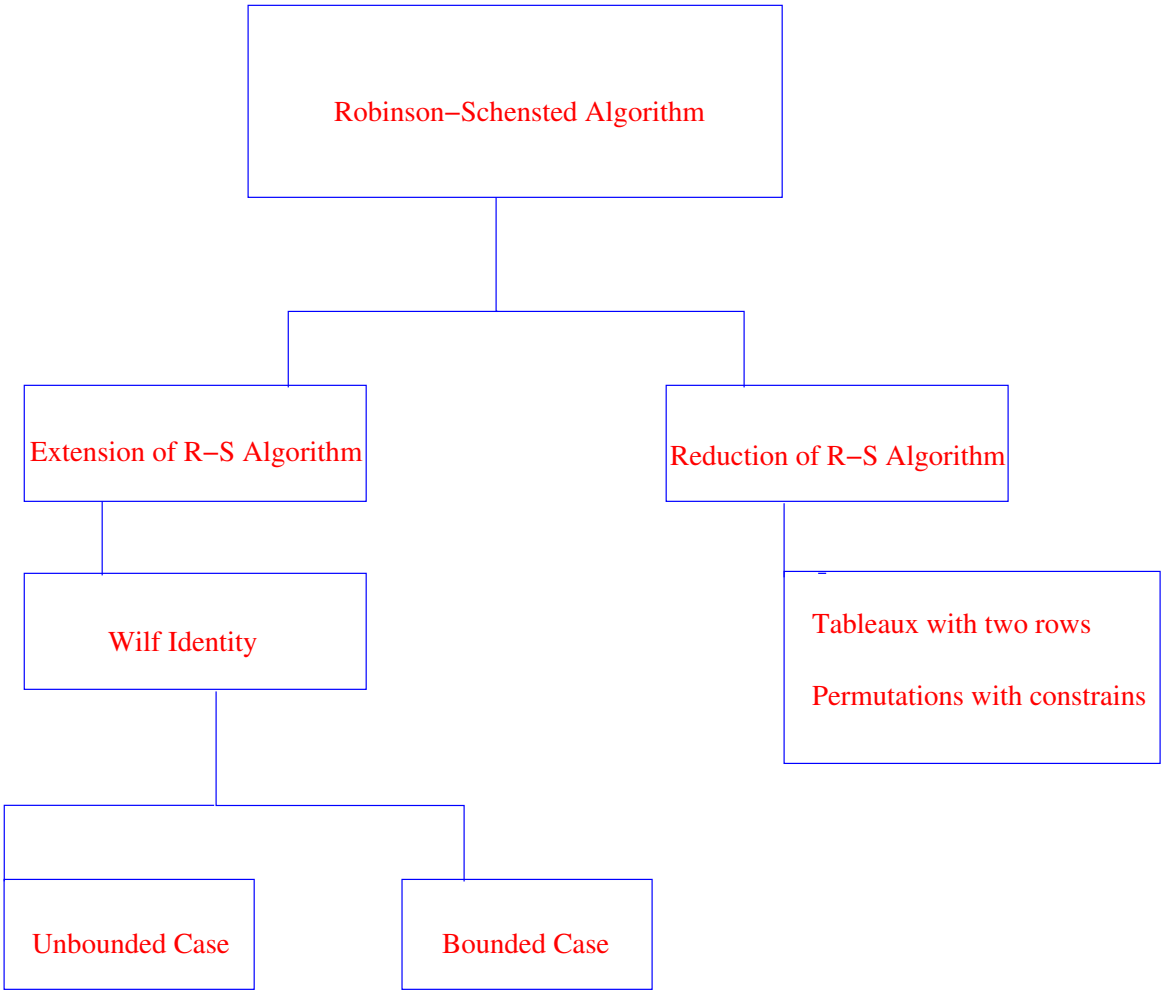


Figure 2.8: Studied Connections

2.5 Extension of R-S : Wilf Identity

In this section, we will introduce the extension relationship between two counting problems in the theory of symmetric functions. Let $\pi_k(n)$ be the number of permutations of n letters that have no increasing (ascending) subsequence of length greater than k. And let $y_k(n)$ be the number of standard Young tableaux of size n whose first row is of length less or equal than k.

We define an exponential generating function of $\pi_k(n)$ as

$$\Pi_k(x) = \sum_{n\geq 0} \pi_k(n) \frac{x^{2n}}{n!^2}$$

Similarly, we define the exponential generating function of $y_k(n)$ as

$$Y_k(x) = \sum_{n\geq 0} y_k(n) \frac{x^{n}}{n!}$$

Theorem 3 *(Proof omitted) We have [28, 29]*

$$\Pi_k(x) = Y_k(x)\, Y_k(-x), \quad for \;\; k = 2, 4, 6, ... \tag{2.1}$$

More explicitly, for $n \geq 0$ and k even numbers. We have

$$\left(\begin{array}{c} 2n \\ n \end{array} \right) \pi_k(n) = \sum_i \left(\begin{array}{c} 2n \\ i \end{array} \right) (-1)^i \;\; y_k(i) \;\; y_k(2n-i) \tag{2.2}$$

By setting the two generating functions above in the equation

$$\sum_{n\geq 0} \pi_k(n) \frac{x^{2n}}{n!^2} = \sum_{n\geq 0} y_k(n) \frac{x^n}{n!} \sum_{n\geq 0} y_k(n) \frac{-x^n}{n!}$$

Now, let $2n = i + j$

$$\sum_{n\geq 0} \pi_k(n) \frac{x^{2n}}{n!^2} = \sum_{i,j}^{2n=i+j} y_k(j) \frac{1}{j!} \; y_k(i) \frac{(-1)^i}{i!} x^{2n}$$

We multiply the right hand side of the equation by $\frac{2n!}{2n!}$

$$\sum_{n\geq 0} \pi_k(n)\frac{1}{n!^2} = \frac{2n!}{2n!}\sum_{i,j}^{2n=i+j} y_k(j)\frac{1}{j!}\ y_k(i)\frac{(-1)^i}{i!}$$

$$\frac{2n!}{n!^2}\pi_k(n) = \sum_i \frac{2n!}{i!j!}(-1)^i\ \ y_k(i)\ \ y_k(j)$$

Since $2n = i + j \longrightarrow j = 2n - i$

$$\frac{2n!}{n!^2}\pi_k(n) = \sum_i \frac{2n!}{i!j!}(-1)^i\ \ y_k(i)\ \ y_k(2n-i)$$

Hence

$$\left(\begin{array}{c} 2n \\ n \end{array}\right)\pi_k(n) = \sum_i \left(\begin{array}{c} 2n \\ i \end{array}\right)(-1)^i\ \ y_k(i)\ \ y_k(2n-i) \tag{2.3}$$

No direct combinatorial proof of (2.3) is known (up to our knowledge). We will now try to find a bijective construction of the identity above and for that we divide the problem into two cases :

- Unbounded case : Where k is infinite
- Bounded case : Where k is finite

2.5.1 Unbounded Case : $K = \infty$

When we set $k = \infty$, we have to find a combinatorial proof of the following identity

$$\left(\begin{array}{c} 2n \\ n \end{array}\right)\pi(n) = \sum_i \left(\begin{array}{c} 2n \\ i \end{array}\right)(-1)^i\ \ y(i)\ \ y(2n-i) \tag{2.4}$$

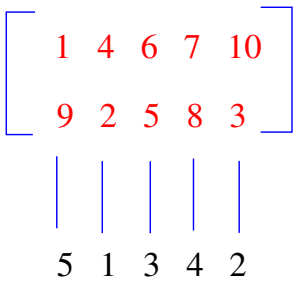
In the left hand side of (2.4) $\pi(n)$ is the number of permutations on [n]

$$\left(\begin{array}{c} 2n \\ n \end{array}\right)\pi(n) = \left(\begin{array}{c} 2n \\ n \end{array}\right)n!$$

$\begin{pmatrix} 2n \\ n \end{pmatrix}$ is the choice of n objects among 2n. For example, let

$$\pi = 51342$$

and let us choose n=5 among 2n=10 like $\{2, 3, 5, 8, 9\}$



Then we will get an involution without fixed point on [2n]. The number of them is equal to (2n-1)!!. Each cycle posses two possible configurations with $i < j$:

$$first\ \ cycle: \begin{bmatrix} i \\ j \end{bmatrix}\ \ or\ \ second\ \ cycle: \begin{bmatrix} j \\ i \end{bmatrix}$$

Hence

$$\begin{pmatrix} 2n \\ n \end{pmatrix} n! = (2n-1)!!2^n$$

The right hand side of (2.4) can therefore be interpreted couples of involution on [i] and [2n-i] respectively (below $Inv[k]$ stands, as in lemma 2, for the number of involutions on k objects)

$$\sum_i \begin{pmatrix} 2n \\ i \end{pmatrix} (-1)^i\ \ y(i)\ \ y(2n-i) = \sum_i \begin{pmatrix} 2n \\ i \end{pmatrix} (-1)^i\ \ Inv[i]\ \ Inv[2n-i]$$

$\begin{pmatrix} 2n \\ i \end{pmatrix}$ is the choice of i among 2n, and $(-1)^i$ is the weight on each point of the first involution Inv[r]

$$w(Inv[i], Inv[2n-i]) = (-1)^{|Inv[i]|}$$

Where w is the weight of the whole involution.

The Construction θ :

Now, we will consider the greatest fixed point of two involutions and we will pass it on the other involution. That is with $(Inver[i])$ the set of involution on $[1...i]$

- If $f \in Inv[i]$, then $Inv[i]^{'} = Inv[i]/f$ and $Inv[2n-i]^{'} = Inv[2n-i] \cup f$
- If $f \in Inv[2n-i]$, Idem

Hence

$$\sum_i \begin{pmatrix} 2n \\ i \end{pmatrix} (-1)^i \ y(i) \ y(2n-i) = \sum_p \begin{pmatrix} 2n \\ 2p \end{pmatrix} \ (2p-1)!! \ (2n-2p-1)!!$$

Which means

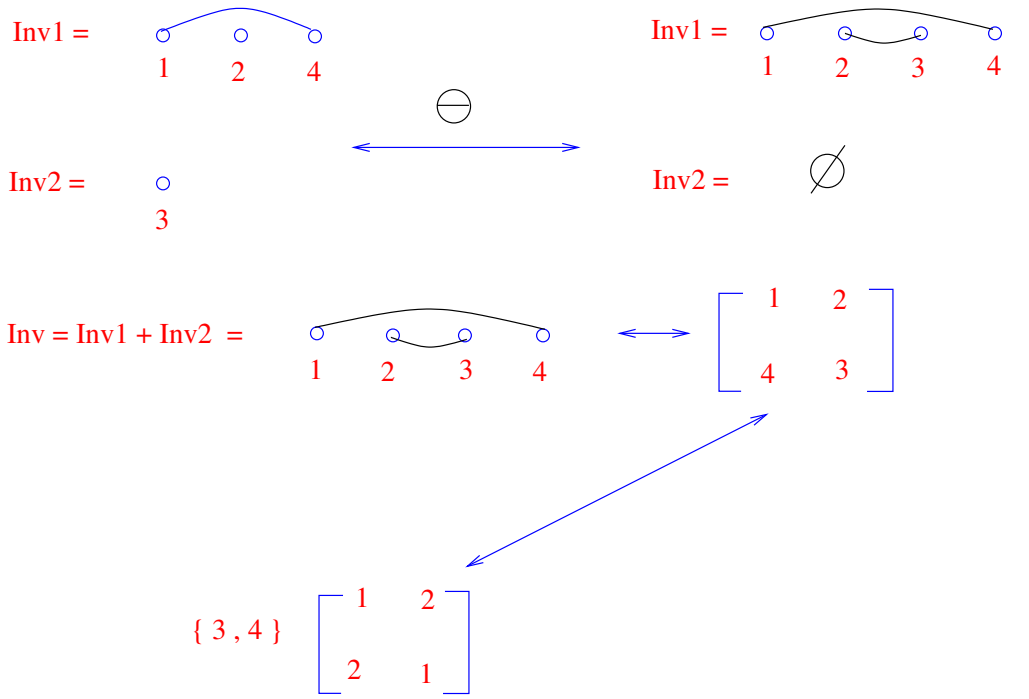
$$\sum_p \begin{pmatrix} 2n \\ 2p \end{pmatrix} \ (2p-1)!! \ (2n-2p-1)!! = (2n-1)!!2^n$$

The proof is complete in this case [46].

Example :

Let $2n = 4$ and $i = 3$, then $2n - i = 1$

$$\sum_i \begin{pmatrix} 2n \\ i \end{pmatrix} (-1)^i \ y(i) \ y(2n-i) = \sum_3 \begin{pmatrix} 4 \\ 3 \end{pmatrix} (-1)^3 \ y(3) \ y(1)$$



2.5.2 Bounded Case : K is finite

Oscillating Tableaux

We define an oscillating tableau of length n and shape λ to be $\emptyset, \lambda_1, \lambda_2, ..., \lambda_n$ of Ferrers diagrams such that λ_1 is a single square, λ_n is λ and for each $i+1$, the shape λ_{i+1} is obtained from λ_i by adding or deleting an admissible square. For example

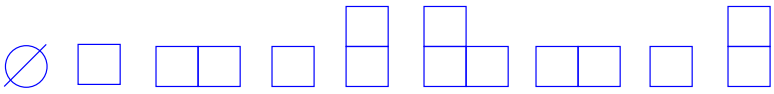


Figure 2.9: Oscillating tableaux

is an oscillating tableau of length 8 and of final shape $(1,1)$. We remark that an oscillating tableau is a standard tableau with insertion and deletion cell.

Some Results About Oscillating Tableaux

We present here some results without proofs, the reader shall see [49] for more details.

- The number of oscillating tableaux of length n is given by

$$O_n = \sum_{k=0}^{n/2} \frac{n!}{k!(n-2k)!}$$

- The number of oscillating tableaux of length $2n + c$ and of final shape $\lambda = (c)$, where $n, c \geq 0$ is given by

$$O_{2n,c} = \begin{pmatrix} 2n+c \\ c \end{pmatrix} (2n)!! = \frac{(2n+c)!}{n!c!2^n}$$

- Let $O_{2n,c}^{=k}$ be the set of all oscillating tableaux of length $2n+c$, of final shape $\lambda = (c)$, and of height k. And let $S_{2n,c}^{=k}$ be the set of all standard tableaux of size $2n + c$, of height k, having c odd columns. Then

$$O_{2n,c}^{=k} = S_{2n,c}^{=2k-1} + S_{2n,c}^{=2k}$$

  More generally, for at most k $(\leq k)$

$$O_{2n,c}^{\leq k} = S_{2n,c}^{\leq 2k} \tag{2.5}$$

2.6 Dyck Words

Let **w** be a word and let **a** be a letter. The length of **w** is denoted by $|w|$, and the number of occurrences of **a** in **w** by $|w|_a$. We denote the empty word by ϵ. If **w = uv** , then **u** is a prefix of **w**.

A Dyck word **w** is a word over the alphabet $\sum = \{a, b\}$ with the following properties :

- For each prefix u of w, $|u|_a \geq |u|_b$
- $|w|_a = |w|_b$

It is clear that the Dyck words are always of even length. The following proposition ensures the correspondence between Dyck words and locally complete binary trees.

Proposition 1 *Each non-empty Dyck word* **w** *can be factorized in a unique way in*

$$\mathbf{w} = a\, w_1\, b\, w_2$$

where w_1 *and* w_2 *are Dyck word.*

A Dyck path is a path in the first quadrant, which begins at the origin $(0, 0)$, ends at $(2n, 0)$. And which consists of steps **North-east** (letter a) and **South-east** (letter b), see the figure below.

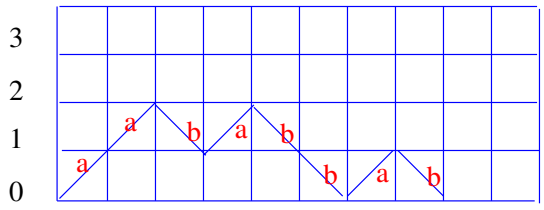


Figure 2.10: Dyck path of length 2n (n=4)

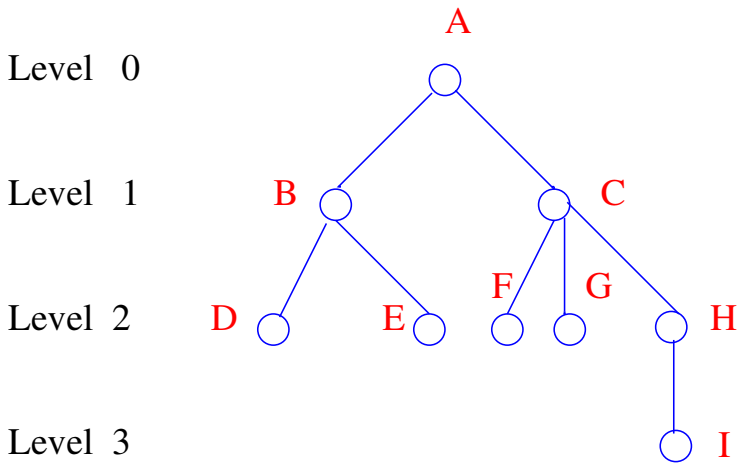
Let D be the set of all Dyck paths and D_n the set of Dyck paths whose length is n. It is well known that $|D_n| = C_n = \frac{1}{n+1}\begin{pmatrix} 2n \\ n \end{pmatrix}$, the n-th Catalan number.

Let $D(n,k)$ be the number of Dyck words of length $2n$ and with k factors. The bi-variate generating series of these numbers is

$$\sum_{n,k} D(n,k)x^n y^k = \frac{1+\sqrt{1-4x}}{1+\sqrt{1-4x}-2xy}$$

2.7 Trees

We will begin our background by introducing some terminology.



Basic Concepts

- **A** is the root node.
- **B** is the parent of **D** and **E**
- **C** is the sibling of **B**
- **D** and **E** are the children of **B**
- **D, E, F, G, I** are external nodes, or leaves

- **A, B, C, H** are internal nodes
- The depth (level) of **E** is 2
- The height of the tree is 3
- The degree of node **B** is 2

Property

- The number of edges equal to the number of nodes minus one

Full Binary Trees

A full binary tree **FBT** is a tree-like structure that is rooted and in which each vertex has zero or two children. Each child of a vertex is designated as its left or right child. The number of binary trees with n nodes are the Catalan number C_n.

A FBT is often defined recursively as either being empty or consisting of a root node together with left and right subtrees, both of which are binary trees.

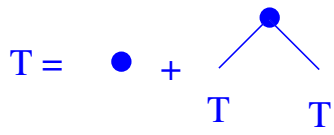


Figure 2.11: Binary tree grammar style

Hence, if one counts by number of leaves (n), one has

$$\sum_n a_n z^n = T = 1 + zT.T \Rightarrow T = \frac{1 - \sqrt{(1-4z)}}{2z} = \sum_n \frac{\binom{2n}{n}}{n+1} z^n$$

Binary trees are extremely useful in computer science as the addresses of the nodes can be given by $0-1$ words.

The number of extended binary trees with n internal nodes and leftmost leaf at level k is

$$\frac{k(2n-k-1)!}{(n-k)!n!}$$

The table of these numbers, shown below, is known as **Catalan's triangle** if the rows are read backwards [81].

n\ k	1	2	3	4	5	6	7
1	1						
2	1	1					
3	2	2	1				
4	5	5	3	1			
5	14	14	9	4	1		
6	42	42	28	14	5	1	
7	132	132	90	48	20	6	1

Notice that the catalan numbers $1, 2, 5, 14, 42, ...$ are the sums of the successive rows.

2.8 Automata Theory

Automata theory is the theory of finite state transition systems with symbolic input and scalar outputs. Beside the fact that it has an impressing number of applications, its *expressive power* is sufficient to allow to solve conjectures in other sciences [26]. The primary application of automata is as symbol processors, that is they process information presented to them as a string of symbols drawn from a fixed alphabet. The simplest of automata is the **finite-state automaton**, which consists of a set of states and a list of rules, at any point the finite state automaton is in one state (called the current state), and responds to one symbol of its input by a transition. Each rule specifies for each state and each possible input symbol which new state the automaton will move to. The finite-state automaton begins in a specified initial state. The current state and the current input symbol are examined.

One of the rules will specify which state to move to. This state becomes the current state, and the next input symbol becomes the current symbol. The computation ends when the last input symbol has been examined and processed. The state that the finite-state automaton has reached when the computation ends is called the final state.

For example, a **CD player** can be modeled as a finite-state automaton with three states stopped, playing and pushed.

- In the **playing state**, the CD player is rotating and music is playing from it.
- In the **paused state**, the CD is rotating but music is not playing.
- In the **stopped state**, the CD is not rotating.

The CD player has input in the form of three buttons labeled **stop**, **play** and **pause**. If the CD player is in the stopped state, the only button that works is the play button, which moves the CD player to the playing state. In the playing state the stop button moves the CD player to the stopped state. The pause button moves it on the paused state. And the play button has no effect. In the paused state the play and paused button move it on the playing state, and the stop button moves it to the stopped state. the CD player being in the stopped state.

In automata theory, an abstract machine that is implemented in hardware is simply called a **machine**. An **abstract machine** is a model of a computer system constructed to allow a detailed and precise analysis of how the computer system

works. Such a model usually consists of input, output, and operations that can be performed (the operation set), and so can be thought of as a processor. An abstract machine implemented in software is termed a **virtual machine**, and one implemented in hardware is called a machine.

2.8.1 Languages, Alphabets and Words

In formal language theory, a language is viewed as a set of **words** or strings composed of letters drawn from some alphabet. An **alphabet** is a finite non-empty set of letters or symbols. We will use the letter $\sum$ to denote a fixed alphabet. A **word** on or over $\sum$ is a finite sequence of letters of $\sum$, the collection of all words of $\sum$ is denoted $\sum^*$. The **empty word** is a word containing no letters and denoted by ϵ. If w_1 and w_2 are two words then the sequence formed by writing the letters of w_2 after those of w_1 is called the **concatenation** of w_1 and w_2 and written as w_1w_2. Note that the concatenation is non-commutative (when the alphabet are more than two letters) but associative , and $\sum^*$ is closed under concatenation that is $w_1w_2 \in \sum^*$. We can define the i**-times iterated concatenation** of a word w as follows : if w is a word then w^i, the i-times iterated concatenation

$$w^0 = \epsilon \; and \; w^{i+1} = w^i w \; \forall i > 0$$

Example 1 *Let* $\sum = a, b$ *be an alphabet. Then* $w_1 = a\,b$ *and* $w_2 = b\,b\,a$ *are words over* $\sum$ *and*

$$w_1\, w_2 = a\,b\,b\,b\,a$$

$$w_1^3 = a\,b\,a\,b\,a\,b$$

A **language** over $\sum$ is a subset $L \subseteq \sum^*$. For example, the English language and the ancient Greek language.

An **alphabet** $\sum$ is a finite set of letters. For example, the set English alphabets $\sum = a, b, c, ..., z$, the set of Greek alphabets $\sum = \alpha, \beta, \gamma, ..., \zeta$, the binary alphabet $\sum$ =0, 1 and the digital alphabet $\sum$ = 0, 1, 2, ..., 9.

2.8.2 Monoids and Semi-Ring

A **Monoid** consists of a set M and a function

$$M \times M \rightarrow M$$

The following axioms are postulated :

- Associativity : For any $m_1, m_2, m_3 \in M \Longrightarrow (m_1 m_2)m_3 = m_1(m_2 m_3)$
- Unit : $\exists\ 1_M \in M\ \ such\ \ that\ \ 1_M m = m 1_M = m,\ \ \forall m \in M$

If M, N are two monoids, a **morphism**

$$\varphi : M \to N$$

is a function satisfying the following conditions

- $(m_1 m_2)\varphi = (m_1\varphi)(m_2\varphi),\ \ \forall m_1, m_2 \in M$
- $1_M\varphi = 1_N$

Let k be a set. A structure of **semi-ring** n k is the data of two internal composition 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 [59, 60]*

Such a semi-ring can be naturally embedded in a semi-ring in our sense.
ii) SR4 is unrelated to SR1..3 as shows the model $(\mathbb{N}, max, +)$.

The semi-rings form a category for which we give the morphism.

Definition 1 *Let* $(k_1, +_1, \times_1)$ *(resp.* $(k_2, +_2, \times_2)$*) be semiring. We say that a mapping* $k_1 \overset{\phi}{\mapsto} k_2$ *is a morphism of semi-rings if and only if* 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 subsemi-ring 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 subsemi-ring k_0 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 2 *Let $(k, +, .)$ be a semiring then:*
1) i) there is a unique morphism 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.

$$\phi_k(\mathbb{N}) = \{0_k, 1_k, 1_k + 1_k, \cdots \underbrace{1_k + 1_k \cdots 1_k}_{n \text{ times}} \cdots\} \tag{2.6}$$

iii) $\phi(\mathbb{N})$ is a subsemi-ring of k, in fact the smallest of all the subsemi-rings of k.
2) (Structure of $\phi(\mathbb{N})$) [22] 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 semi-ring 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 semi-ring $\phi(\mathbb{N})$ is called the* basic semi-ring *of k and denoted $k_{(0)}$ in [36]. In [36], 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 semi-rings (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 semi-ring such that the mapping $r : \mathbb{N} \mapsto \mathbb{K}_{e,p}$ defined by

$$r(n) = \begin{cases} n \text{ si } 0 \leq n \leq e + p - 1 \\ (n - e \ mod \ p) + e \text{ sinon} \end{cases} \tag{2.7}$$

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

Example 1 *We have the following first examples*

- *a1) The boolean semi-ring* $(\mathbb{B}, +, \times)$.
- *a2)*$(\mathbb{N} \cup \{+\infty\}, min, +)$*,*
- *a3)* $(\mathbb{R} \cup \{-\infty\}, max, +)$ *(known as the "Tropical semi-ring") [78].*
- *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 semi-rings (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 semi-ring k is said commutative if and only if the monoid (k,.) is commutative. If k is a semi-ring 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 semi-ring 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

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

2.8.3 Series

Let M be a monoid and K be a semi-ring. We define a **series** S as a function from M to K

$$S : M \to K$$

For any $w \in M$ we have an image $S(w) \in K$ called coefficient of the word w in S. We can write a series like as follows

$$S = \Sigma_{w\in\Sigma^*}(S, w)w$$

The set of all formal series on Σ with coefficient in K is denoted by $k << \Sigma >>$. The support of S is $supp(S) = \{w \in \Sigma^* | (S, w) \neq 0\}$. A polynomial is a formal series with finite support.

We have two operations on series

- The sum of two series

$$R + S = \Sigma_{w \in \Sigma^*}(< R|w > + < S|w >)w$$

- The Cauchy product of two series

$$R.S = \Sigma_{w \in \Sigma^*}(\Sigma_{uv=w} < R|w >< S|v >)w$$

2.8.4 Finite Automata

A formal definition of a **finite automaton** is the following.

Definition 3 *A finite automaton is the 5-tuple* $(\Sigma, Q, I, T, \delta)$ *where*

- Σ *is a set of symbols, called alphabet of the language the automaton accepts, the alphabet is usually, but not necessarily, assumed to be finite*
- Q *is the set of states to be finite*
- $I \subset Q$ *is the set of initial states, that is, the states in which the automaton is when no input has been yet processed*
- $T \subset Q$ *is the set of the final states called accepting states*
- δ *is the transition function, that is* $\delta : Q \times \Sigma \rightarrow Q$ *which from a state S1 and the letter a go to a state S2 if it exist a transition labeled with a from the state S1 to the state S2.*

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

- In deterministic automata, for each state there is at most one transition for each possible input and only one initial state
- In non-deterministic automata, there can be none or more than one transition from a given state for a given possible input

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

Such automata with output give a new classification of machines ; *Transducers; they generate output based on a given input and/or a state using actions. They are used for control applications. Here two types are distinguished* :

- Moore machines : Output depends only on a state, i.e. the automaton uses only entry actions. The advantage of the Moore model is a simplification of the behavior

- Mealy machines: Output depends on input and state, i.e. the FSM uses only input actions. The use of a Mealy FSM leads often to a reduction of the number of states

Finally, we focus our attention on a special kind of automata with outputs which are efficient in operational way. This automata with output are called automata with multiplicities and they are defined in the following.

2.8.5 Automata with Multiplicities

An automaton with multiplicities is based on the fact that the output data of the automata with output belong to a specific algebraic structure, a semi-ring. In that way, we will be able to build effective operations on such automata, using the power of the algebraic structures of the output data. And we are also able to describe this automata in matrix representation with all the power of the linear algebra.

Definition 4 *An* **automaton with multiplicities** *over an alphabet* $\sum$ *and a semi-ring K is the 5-uple* Σ, Q, I, T, δ *where*

- $Q = \{S1, S2, ..., Sn\}$ *is the finite set of state*

- $I : Q \rightarrow K$ *is a function over the set of initial states, which associates to each initial state a value of K, called entry cost, and to non-initial state a null value*

- *$T : Q \to K$ is a function over the set of the final states, which is associated to each final state a value of K, called final cost, and to non-final state a null value*

- *δ is the transition function, that is $\delta : Q \times \Sigma \times Q \to K$ which from a state S1 , a letter a and a state S2 go to a value z of K if it exist a transition labeled with a from the state S1 to the state S2 and with the output z.*

We remark that

- automata with multiplicities is a generalization of finite automata. In fact, finite automata can be considered as automata with multiplicities with for the semi-ring K, the boolean set $B = \{0, 1\}$. To each transition we affect 1 if it exists and 0 if not.

- We have not yet, on purpose, defined what a semi-ring is. Roughly it is the least structure $(K, +, x)$ that allows matrix computation with units (one can think of a ring without the **minus** operation)

2.8.6 Automata with Multiplicities : Linear Representation

The previous automata with multiplicities can be expressed by a linear representation which is a triplet

$$p = (\lambda, \mu, \gamma)$$

with $\lambda \in K^{1 \times n}$ is a row-vector which coefficients are $\lambda_i = I(S_i)$, $\gamma \in K^{n \times 1}$ is a row-vector which coefficients are $\gamma_i = T(S_i)$, and $\mu : \Sigma \to K^{n \times n}$ is a morphism of monoids such that $\forall a \in \Sigma$, the coefficient on the i-th row and j-th column of all transitions labeled with a is $\mu(a)_{ij} = \delta(S_i, A, S_j)$.

In the following, we describe the linear representation of the automata with multiplicities which corresponds to the the following figure.

In this figure, the input states are characterized with input arrows from nothing and the output states are characterized with output arrow to nothing. On each transition, we indicate before the semi-column, the input data and after the semi-column, the output associated for each input data.

The linear representation is the following

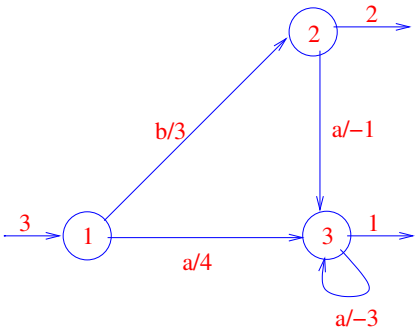


Figure 2.12: Automata with multiplicities

$$\lambda = \begin{pmatrix} 3 & 0 & 0 \end{pmatrix}$$

$$\gamma^t = \begin{pmatrix} 0 & 2 & 1 \end{pmatrix}$$

$$\mu(a) = \begin{pmatrix} 0 & 0 & 4 \\ 0 & 0 & -1 \\ 0 & 0 & -3 \end{pmatrix}$$

$$\mu(b) = \begin{pmatrix} 0 & 3 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

2.9 Conclusion

We have presented Wilf identity which represents a generalization of R-S correspondence. We have divided this identity into two cases : Bounded and unbounded with respect to the height of standard Young tableaux and the permutations of n letters that have no increasing subsequences of length greater than 2.

We reduce R-S algorithm condition to establish an algorithm of passage between $\pi_2(n)$ and $\lambda(l_1, l_2) \vdash n, l_1 \geq l_2 > 0$.

Chapter 3

Dynamic Data Structures

Contents

3.1 From Combinatorics to Dynamic Combinatorics

Young tableaux were introduced by **Alfred Young** a mathematician at Cambridge university, in 1900 in order to compute some idempotents of the symmetric group G_n, as well as to compute the matrices of the representations of irreducible of the symmetric group. They were later applied to the study of symmetric groups by George Frobenins in 1903. The theory was further developed by A. Young, and by other mathematicians including Robinson, Schensted, Richard P. Stanley, ect. The Young tableaux are useful combinatorial objects in representation theory and in every application where permutation of states are relevant like nuclear physics. It provides a convenient way to describe the group representations of the symmetric group and to study their properties. The Young tableaux play an important role in various scientific fields, particularly in combinatorics.

The **Robinson-Schensted algorithm**, first discovered by Robinson in 1937, which established a one-to-one correspondence between permutations of symmetric group S_n and pairs of standard Young tableaux of the same shape $TS \times TS$. It can be viewed as a simple, constructive proof of the combinatorial identity :

$$\sum_{\lambda \vdash n} (f^{\lambda})^2 = n!$$

We will study another approach called **dynamic combinatorics (Dynamic Data Structures)** of some combinatorial (data structures) objects. The fundamental problem of enumerative combinatorics is to determine the number of elements of a set, that is, to enumerate the number of objects of size n inside a given class of objects and more precisely, to enumerate the number of objects using some formal parameters. The two principal tools of enumerative combinatorics are **bijections** (construction a bijection between two or more than two families of objects) and **generating functions**. The word dynamic is a physical concept which means the forces that produce motion. In mathematics a dynamical system is a space (a configuration space) on which there is an evolution operator.

Here, we give a dynamic structure to some class of permutations of symmetric group, standard tableaux, Dyck words and codes and we prove equi-variance of these dynamical systems.

3.2 Some Examples of Dynamic Data Structures

The Catalan sequence was first described in the 18th century by Leonhard Euler, who was interested in the number of different ways of dividing a polygon into triangles. The sequence is named after Belgian mathematician Eugène Charles Catalan (1814-1894), who discovered the connection to parenthesized expressions. The Catalan number play an important role in enumerative theory. They appear as counting a various class of combinatorial objects; parenthesis system, dissection of n-gons into triangles etc ... .

The **Catalan number** which the first numbers are (1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, ...), arise in a number of problems in combinatorics. They can be computed using the following formula

$$C_n = \frac{1}{n+1} \left(\begin{array}{c} 2n \\ n \end{array} \right)$$

Among other things, the Catalan numbers describe the following data structures :

- $\pi_2(n)$: The number of permutations of n-letters without increasing subsequences of length greater than two
- $f^{(n,n)}$: The number of standard Young tableaux of height two with rectangular shape
- The number of Dyck words of length 2n
- The number of incomplete binary trees with n nodes
- The number of finite integer sequences $a_0, a_1, ..., a_n$ such that : $a_0 = 1, a_{i+1} \leq a_i + 1$

The data structures above has the same generating function :

$$\sum_{n \geq 0} C_n \, z^n = 1 + z + 2z^2 + 5z^3 + 14z^4 + 42z^5 + 132z^6 + ...$$

There are several ways of explaining why the Catalan formula solves the problem of enumeration of the data structures listed above, (see http://en.wikipedia.org/wiki/Catalan_number) for the first four data structures and [47] for the five one.

3.3 Dynamic Permutations With Constraints

The object of our study is the symmetric group S_n consisting of all bijections from [n]=$\{1, 2, ..., n\}$ to [n]. The elements $\pi \in S_n$ are called permutations. We will use the composition as the multiplication, and we multiply permutations from right to left [6].

We say that a permutation π of n letters has an increasing subsequence of length k if there are positions $1 \leq i_1 < i_2 < i_3 < ... < i_k \leq n$ such that $\pi(i_1) < \pi(i_2) < \pi(i_3) < ... < \pi(i_k)$. For example

$$\pi = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 5 & 3 & 4 & 1 & 2 \end{pmatrix}$$

has increasing subsequences of length 2, at points $\{2, 3\}$ as well as at position $\{4, 5\}$. Let $\pi_2(n)$ be the number of permutations of n letters that have no increasing subsequences of length > 2. By direct enumeration we obtain the following table.

n	$\pi_2(n)$
0	1
1	1
2	2
3	5
4	14
5	42
6	132
7	429
8	1430

Proposition 3 $|\pi_2(n)| = C_n$, *where* $\pi_2(n)$ *is the number of permutations of n letters that have no increasing subsequences of length greater than 2 and* C_n *is Catalan numbers.*

We can show that for $\alpha, \beta \in \pi_2(n)$, $\alpha \circ \beta$ is not closed under the composition. If we take $\pi_2(3)$, then we have five permutations without subsequences of length 123. Now if we multiply $\alpha = 3\,1\,2$ and $\beta = 2\,3\,1$ then the composition

$$\alpha \circ \beta = 1\,2\,3 \notin \pi_2(3)$$

is not inside $\pi_2(3)$.

3.3.1 Dynamics on Permutations

The **dynamic permutation** population we consider will be $\pi_2(n)$ with a parameter k. Denoting by L(n,k) the number of dynamic permutations $\pi_2(n)$ with a parameter k, and k can be the following :

- $\mathbf{k_1}$: is the position of n in $\pi_2(n)$
- $\mathbf{k_2}$: is the last element of $\pi_2(n)$
- $\mathbf{k_3}$: is the maximal descent of $\pi_2(n)$

If we set $l(n,k) = |L(n,k)|$, we can get the following :

$l(n,0) = l(0,n) = 0\ , \forall n \geq 1, l(0,0) = 1$

$$l(n,k) = \sum_{j \geq k+1} l(n-1,j)$$

Whence the easy computed table of the first values

n\ k	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0
3	0	2	2	1	0	0	0	0
4	0	5	5	3	1	0	0	0
5	0	14	14	9	4	1	0	0
6	0	42	42	28	14	5	1	0
7	0	132	132	90	48	20	6	1

We note the fact that, taking for k_1, k_2, k_3 we do not obtain the same sets but the same statistics.

3.3.2 $\pi_2(n)$ with k_1

We always start with the number 1. When we add (insert) in order the numbers from $[n] = \{2, 3, ..., n\}$, we have to respect the following rule :

The obtained permutation must not contain an increasing subsequence which its length > 2.

Therefore, when we stock the number of positions occupied by the insertion number, we get a sequence of integer numbers $a_0\, a_1\, a_2 \ldots a_n$ with $a_0 = 1$. For example the corresponding code for the permutation **4312** is **1211**.

Let $L(n, k_1)$ be the numbers of $\pi_2(n)$ with the parameter k_1 as a position of the element n. Figure (3.1) show the dynamics on the permutations with k_1.

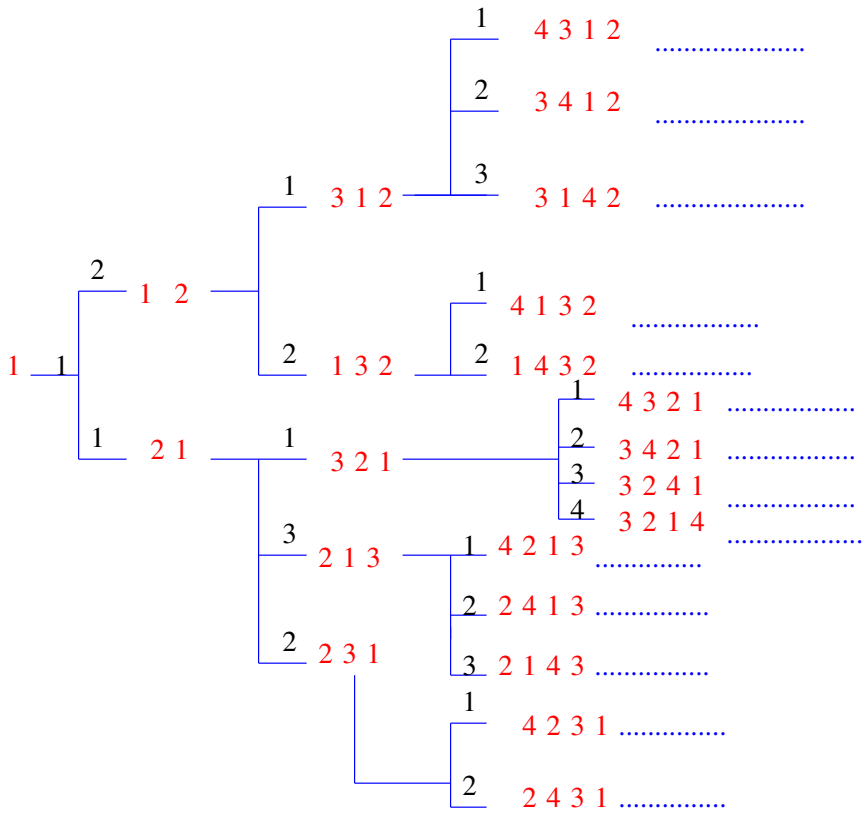


Figure 3.1: Permutations of size n with the parameter k_1

3.3.3 $\pi_2(n)$ with k_2

We always start with the number 1. When we add (insert) in order the numbers from $[n] = \{2, 3, ..., n\}$, we have to respect the following rule :

The obtained permutation must not contain an increasing subsequence which its length > 2.

Therefore, we stock the last number of permutations after inserting the number. We get a sequence of integer numbers, for instance, if n = 4, we have four codes; **1111, 1222, 1114, 1133**. Here, in contrast of (2.2.3), we can get for one code many permutations which is not the case of first one. For example, the corresponding code for the five following permutations **4312, 3412, 3142, 4132, 1432** is **1222**.

Let $L(n, k_2)$ be the number of permutations in $\pi_2(n)$ with the parameter k_2 (last element of the permutation). Figure (3.2) show the dynamics on the permutations with k_2.

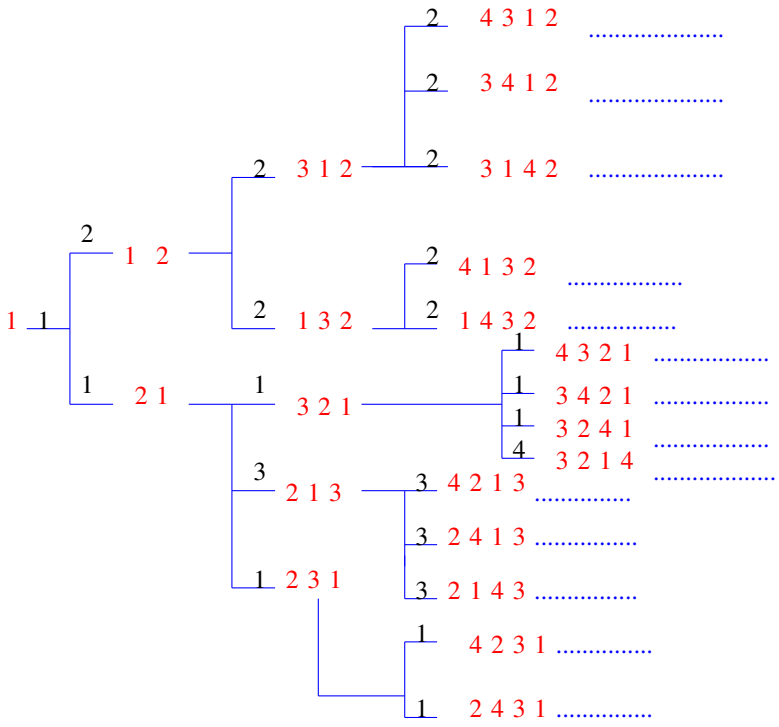


Figure 3.2: Permutations of size n with the parameter k_2

3.3.4 $\pi_2(n)$ with k_3

We always start with the number 1. When we add (insert) in order the numbers from $[n] = \{2, 3, ..., n\}$, we have to respect the following rule :

The obtained permutation must not contain an increasing subsequence which its length > 2.

Therefore, we stock the number of length of the initial maximal descents of the permutation. we get a sequence of integer numbers $a_0\, a_1\, a_2\, ...\, a_n$ with $a_0 = 1$. For example, the corresponding code for the permutation **4312** is **1123**.

Let $L(n, k_3)$ be the numbers of $\pi_2(n)$ with the parameter k_3 (length of the initial maximal descent of the permutation). Figure (3.3) show the dynamics on the permutations with k_3.

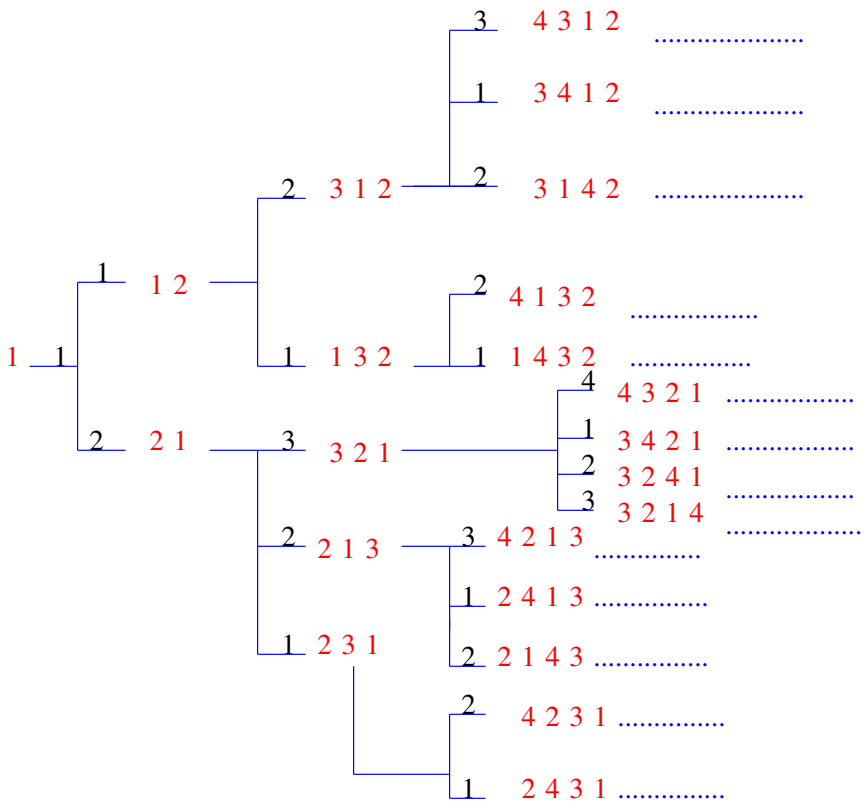


Figure 3.3: Permutations of size n with the parameter k_3

3.4 Dynamic Tableaux

Young's lattice is the set of all number partitions, partially ordered by inclusion of Young diagrams. It is classical to construct the Young lattice by starting from **empty** and adding a new cell at each possible place starting from the right. The Young lattice allows to build and to define some families of tableaux such as *Standard Young Tableaux* [6] and *Oscillating Tableaux* [49]. The Young lattice allows also to draw trajectories, that is a **chain** or a **path** going from the empty to a certain shape.

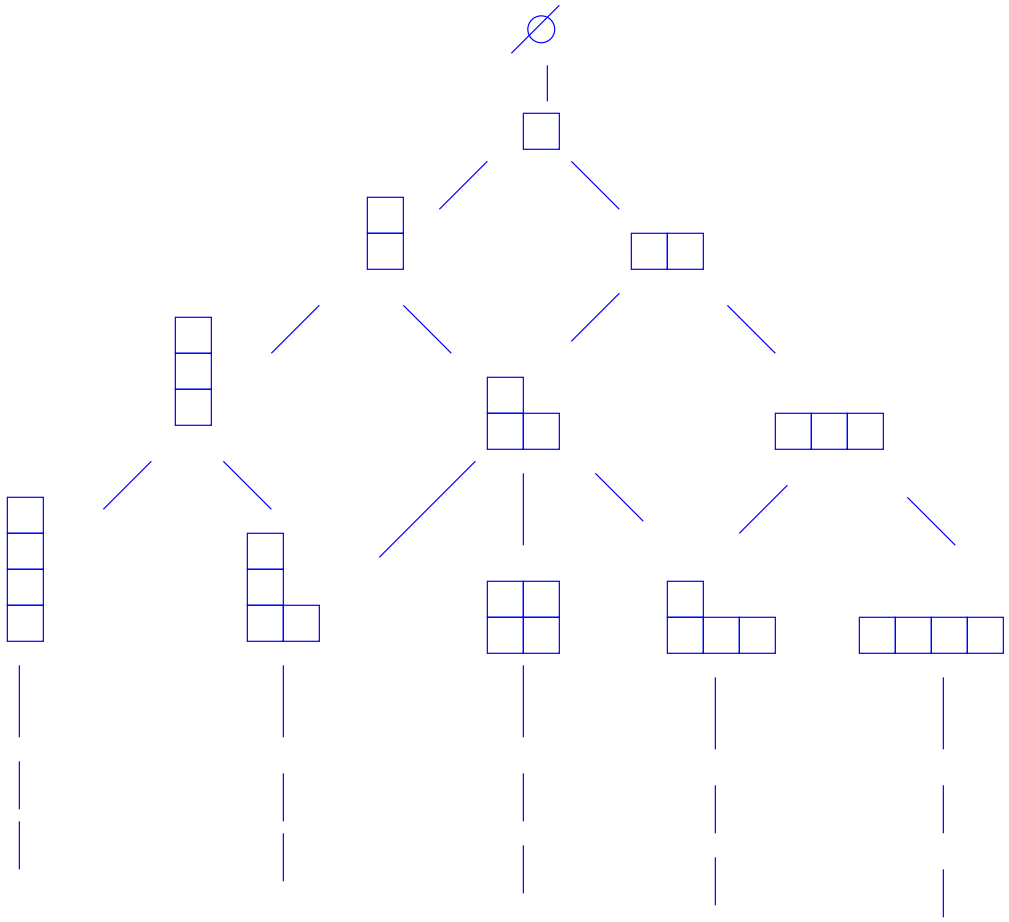
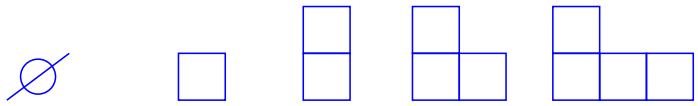
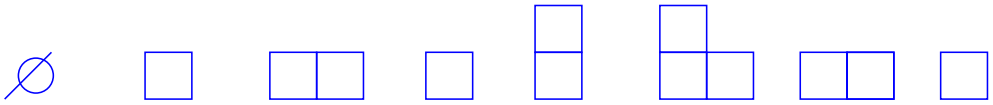


Figure 3.4: Young Lattice

We can interpret a **standard Young tableau** as a **chain** going from the initial shape $\emptyset$ to a certain final shape as we see below



The same interpretation is true for an **oscillating tableau**, it can be interpreted as a **path** going from the initial shape $\emptyset$ to a certain final shape as we see below



Clearly, we have the following inclusions :

$$Standard\ Young\ tableaux \subsetneq Oscillating\ tablraux$$

3.4.1 Dynamics on Standard Tableaux

We are going to study the dynamic standard Young tableaux. In this case, a cell can be added at each possible place starting from the right.

We start with the $empty$. At each cell insertion, we stock the position of the inserting cell, that is, stocking the row number which it take.

This is lead to get a sequence of integer number $a_0\, a_1 \ldots a_n$. Let ST(n,k) be the number of standard Young tableaux of n-cells with the parameter k, and k is the height of the last inserted cell. Figure (3.5) show dynamics on standard tableaux.

If we take the code $0\,1\,0\,2$, we can obtain the corresponding standard Young tableau by considering respectively, 0 as the first row, 1 as the second row and 2 as the third row of the future standard tableau. In the other words, the corresponding Young tableau is of shape $\lambda = (2, 1, 1)$. The codes $a_0\, a_1 \ldots a_n$ are called **Yamanouchi words**.

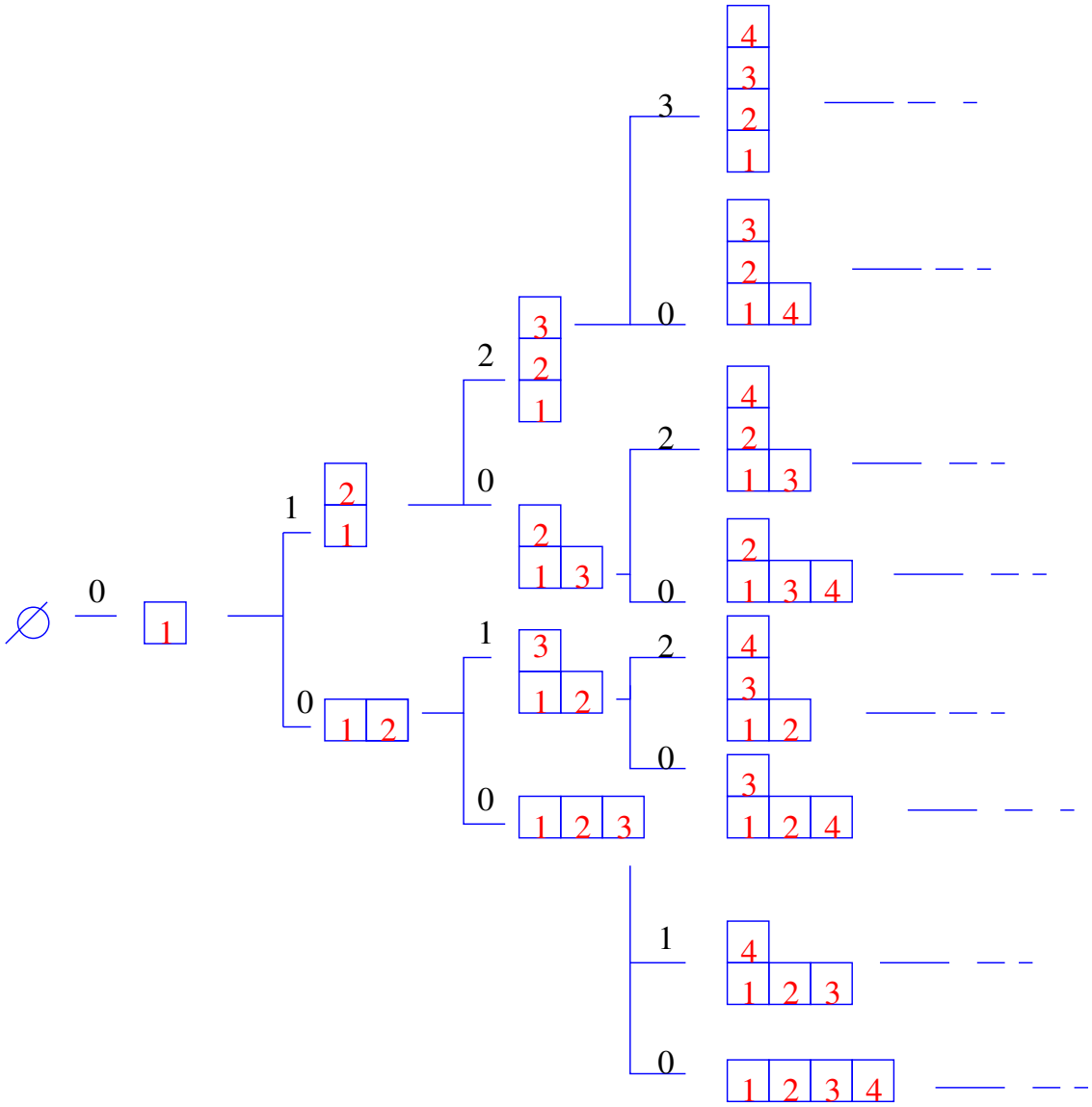


Figure 3.5: Standard tableaux of size n with k rows

3.4.2 Dynamics on Rectangular Tableaux of Height Two

A **standard Young tableau of two lines** is of the shape $\lambda(l_1, l_2) \vdash n$ where $l_1 \geq l_2 > 0$. Let $f^{(l_1,l_2)}$ be the number of standard tableaux of two lines. We have

$$f^{(l_1,l_2)} = \frac{l_1 - l_2 + 1}{l_1 + 1} \begin{pmatrix} l_1 + l_2 \\ l_1 \end{pmatrix}$$

Proposition 4 *If* $l_1 = l_2 = n$ *then*

$$f^{(n,n)} = \frac{1}{n+1} \begin{pmatrix} 2n \\ n \end{pmatrix}$$

which is the n-th numbers of Catalan C_n.

In general, we can represent a standard Young tableaux of two (equal) lines as follows :

Figure 3.6: Standard Young tableaux of size 2n and height two

Then $2n - m = k$ will be the second parameter of the tableau.

Example 2 *A Young tableau of size six and height two*

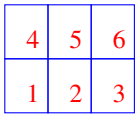


Figure 3.7: Standard Young tableau of size 6 and height 2

Then $6 - 3 = 3$ *we remark that 3 is the last element of* $\pi_2(3) = 213$

Concerning the dynamic on standard Young tableaux of two (equal) lines. we will always start with the shape (1,1). At each insertion, two cells added; they are superimposed at the right.

Let STL(n,k) be the number of standard Young tableaux of two lines (n,n) with the parameter k, and the parameter k is 2n-m. Figure (3.8) show dynamics on standard Young tableaux of two (equal) lines.

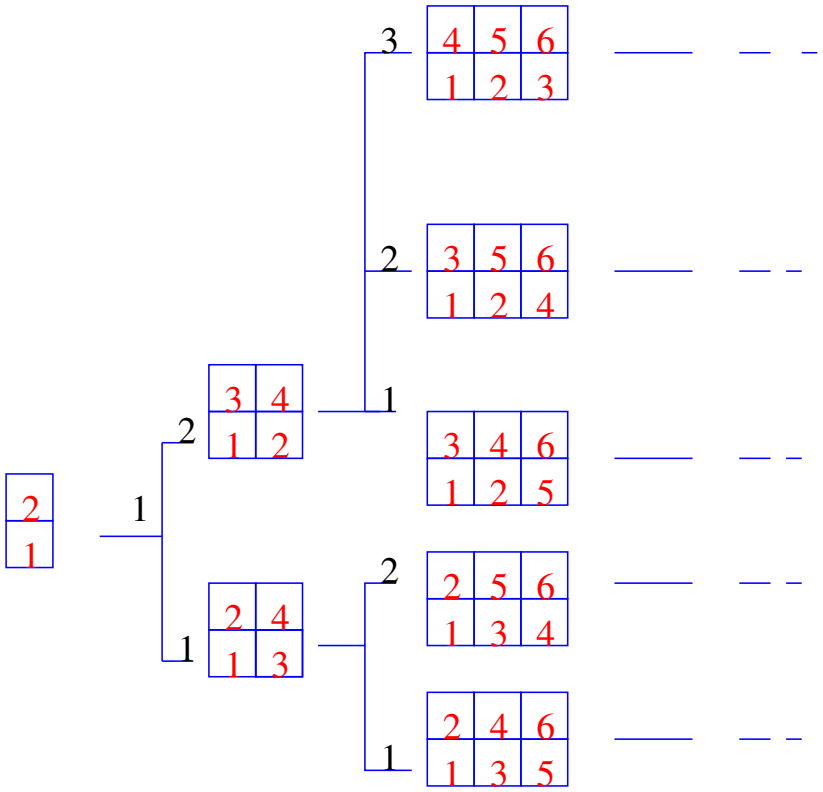


Figure 3.8: Dynamic standard Young tableaux of two lines

3.5 Dynamic Dyck Words

3.5.1 Dynamics on Dyck Words

We start with the word **ab** of length 2 and k=1.
Let $L(n, k)$ be the number of Dyck words with the parameter k, and k is the number of factors. The following figure show the dynamics on the Dyck words with k factors.

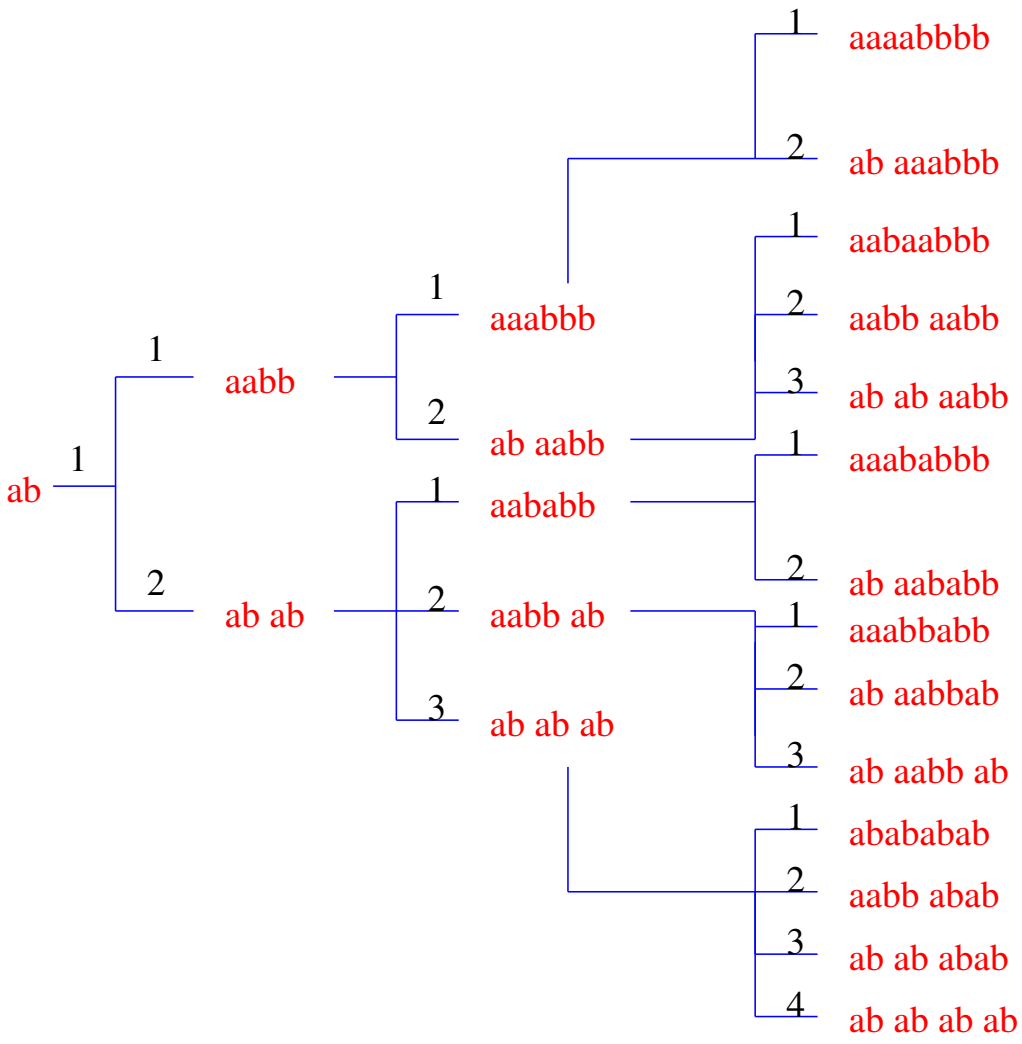


Figure 3.9: Dyck words of length 2n with k factors

3.6 Binary Trees

3.6.1 Binary Tree and Catalan Numbers

Let L(n,k) be the number of binary trees of n nodes with the k edges. Then we have the Catalan numbers C_n, see the following table

n\ k	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	**1**	0	0	0	0	0	0	0
2	0	**2**	0	0	0	0	0	0
3	0	0	**5**	0	0	0	0	0
4	0	0	0	**14**	0	0	0	0
5	0	0	0	0	**132**	0	0	0
6	0	0	0	0	0	**429**	0	0
7	0	0	0	0	0	0	**1430**	0

Binary Tree with Three Vertices

The (fig 3.10) below without external nodes illustrates all possible binary trees that can be created with 3 vertices. If we want to consider binary trees that are structurally different, then trees 1, 2, 4, and 5 are structurally the same. (Tree 3 is different) Binary trees are structurally the same if combinations of mirror images (left to right flips) can convert one version of a tree into another. For example, a mirror image flip taken at the top vertex of Tree 5 will convert it into Tree 1. In the same way, a mirror image flip taken at the middle vertex of Tree 2 (and only using that portion of the tree below this vertex) will convert it into Tree 1.

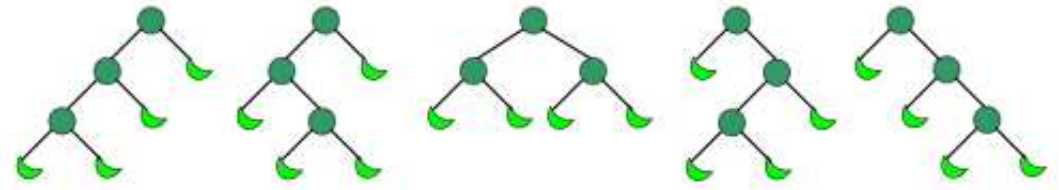


Figure 3.10: Complete Binary trees of n nodes

3.7 A Coding Theorem

In physics, a **toy model** consists of a less complicated collection of objects and equations relating them. In spite of that they can be used to understand a mechanism that is also useful in the full non-simplified theory.
In mathematics, a **toy model** is usually obtained by reducing the number of dimensions or variables or restricting them to a particular symmetric form.
In this section, We will focus ourselves on a toy model to give an economic model from the dynamic combinatorial objects.

3.7.1 Codes

We define these codes as finite sequences
$a_1\, a_2\, a_3 \ldots a_n$ such that

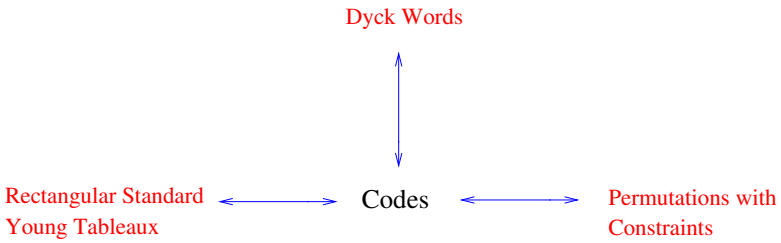
- $a_1 = 1$
- $a_{j+1} \leq a_j + 1$

The first codes (in lexorder) are

n	codes
1	1111
2	1112
3	1121
4	1122
5	1123
6	1211
7	1212
8	1221
9	1222
10	1223
11	1231
12	1232
13	1233
14	1234

3.7.2 Statement of the Theorem

We will describe the links between some combinatorics families and we will try to establish certain properties that help us to understand the connection between them.



Theorem 4 *Let* $\Phi = \{a\} \cup \Phi^+$ *be a data structure with a bi-variate statistics*

$$l : \Phi \longrightarrow N^2$$
$$s \longrightarrow l(s) = (n, k)$$

such that

$$l(\Phi^+) \subset N^+ \times N$$

We suppose that there exist a function $d : \Phi^+ \longrightarrow \Phi$ *such that*

1. $d : \Phi_n \longrightarrow \Phi_{n-1}$ *(*$\Phi_n = (pr_1 \circ l)^{-1}(n)$*),* $n \geq 1$
2. $\phi : \Phi_n \longrightarrow \Phi_{n-1} \times N^+$
   $s \longrightarrow (d(s), k)$ *is injective*
3. *define* $\pi = pr_2 \circ l$*. For all* $s \in \Phi$ *we define his code by*

$$\chi(s) = (\pi(d^{n-1}(s)), \pi(d^{n-2}(s)), \cdots \pi(d(s)), \pi(s))$$

then χ *is injective.*

We will give some examples to illustrate our theorem.

First example :

Suppose that we have a code $1\,1\,2\,1$ and we want to pass it to $\pi_2(4) = 3\,4\,1\,2$.

- $1\,1\,2\,1 \Longrightarrow \pi_2(4) = 3\,4\,1\,2$: The method consists of inserting the word w such that for each a_i, $w \in \pi_2(4)$ is the maximum descent.

$$1 \longrightarrow 1\,2 \longrightarrow 3\,1\,2 \longrightarrow 3\,4\,1\,2$$

The 1 corresponds to maximum descent of length 1, 12 corresponds to maximum descent of length 1, 312 corresponds to maximum descent of length 2 and 3412 corresponds to maximum descent of length 1. Then we have 1121.

- $\pi_2(4) = 3\,4\,1\,2 \Longrightarrow 1\,1\,2\,1$: We delete each time the greatest number among $\pi_2(4) = 3\,4\,1\,2$

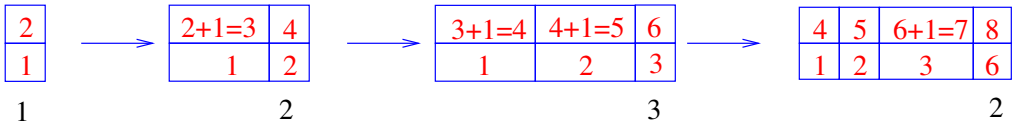
$$3\,4\,1\,2 \longrightarrow 3\,1\,2 \longrightarrow 1\,2 \longrightarrow 1$$

Then, we read the code from right to left $1\,1\,2\,1$.

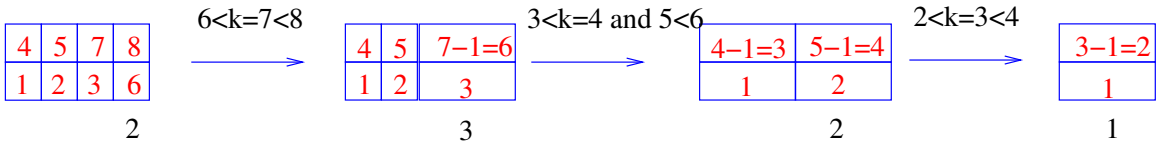
Second example :

Suppose that we have a code $1\,2\,3\,2$ and we want to pass it to $\lambda(l_1, l_2), l_1 = l_2 > 0$, where $l_1 = 1236$ and $l_2 = 4578$.

- $1\,2\,3\,2 \Longrightarrow \lambda(l_1, l_2), l_1 = l_2 > 0$, where $l_1 = 1236$ and $l_2 = 4578$: The method consists of corresponding final couples of tableaux to the a_i



- $\lambda(l_1, l_2), l_1 = l_2 > 0$, where $l_1 = 1236$ and $l_2 = 4578 \Longrightarrow 1\,2\,3\,2$: The method consists of withdrawing the final couples of tableaux from right to left.



Then, we read the code from right to left $1\,2\,3\,2$ corresponding to the final couples of tableaux.

Third example :

- $1\,1\,2\,1 \Longrightarrow$ Dyck word

- Dyck word $\Longrightarrow 1\,1\,2\,1$

3.8 Conclusion

Our purpose is to go from some combinatorial objects to some models which are derived or related directly from the dynamic data structures.

In this chapter, we have studied abstract objects and we have imposed on them some parameters which have transformed these objects into a dynamical system. In the dynamical systems, we will see some complex systems properties like emergence and auto-organization.

Establish a relationship between abstract objects like permutations, standard Young tableaux and Dyck words on one hand, and Complex systems on the other hand has been our aim in this work.

Chapter 4

Dynamic Data Structures for Studying Complex Systems

Contents

This chapter is structured as follows. First of all, in the section (4.1) we will concentrate on our economic model obtained from a simple rule due to dynamic data structures. We will describe a toy-model of the benefit in the economical situation. Finally, we will give a trajectory model of the economical application model.

The ending part of this chapter is about the study of Genetic automata population analysis. An implementation in MuPAD has been developed and experiments leading to diffusion control are proposed.

4.1 Economic Model

4.1.1 Economical Rules and Dynamic Data Structures

Our aim is to describe here a **toy-model** of the benefit in the following situation. A capital owner possesses two accounts, called **P** and **R**, **P** is the account where the principal (untouched) capital is deposited. This capital produces a constant return (one unit per unit of time) which is sent to a reserve **R**. From the account **R** can be withdrawn arbitrary amounts of money and the account must stay positive.

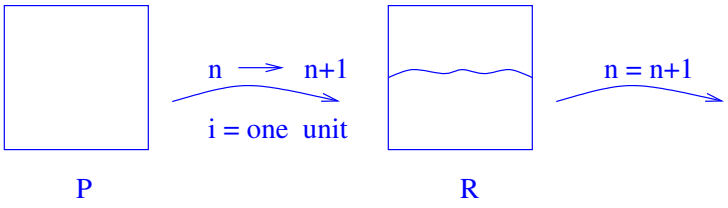


Figure 4.1: Accounts P and R

The possible configurations are described by sequences such that

- $a_1 = 0$
- $a_{i+1} \leq a_i + 1$

Where a_i is the amount of money in **R** at time i. The sequence a_i is the time series and several examples of these series are described in the following figure.

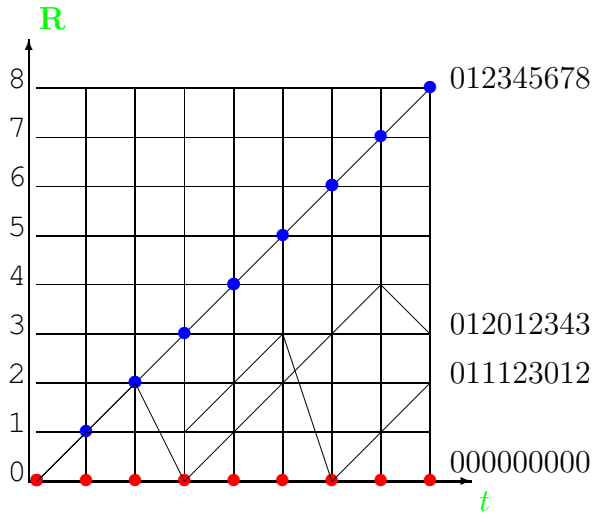


Figure 4.2: Maximal, minimal (dotted) and two intermediate trajectories. Their codes are on the right.

In this work, we will build combinatorial structures that allow to model and to compute the global behavior of the reserve **R** by some specific functions. We can consider this result as an emergent function from the basic local rules.

4.1.2 Trajectories and Codes

We can define the trajectories of our model by sequences (codes) $a_1a_2a_3...a_n$ such that

- $a_1 = 1$
- $a_{j+1} \leq a_j + 1$

Example : For n = 3, we have only the 5 codes who are described in the following table.

n	codes
1	111
2	112
3	121
4	122
5	123

We remark that we have 2 codes which end by 1 or 2, and 1 code ending by 3. Now if one sets $l(n,k)$ to be the number of codes ending by $k-1$, one can check that

- $l(n,0) = l(0,n) = 0$ $(\forall n \geq 1)$
- $l(0,0) = 1$ (the void sequence)
- $l(n,k) = \sum_{j \geq k+1} l(n-1,j)$

Whence the easy computed table of the first values

n\ k	0	1	2	3	4	5	6	7
0	1	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0
3	0	2	2	1	0	0	0	0
4	0	5	5	3	1	0	0	0
5	0	14	14	9	4	1	0	0
6	0	42	42	28	14	5	1	0
7	0	132	132	90	48	20	6	1

The values for $n, k \geq 1$ can be even more easily computed with the (sub-diagonal) local rule described by **West** + **North** = **result**. For instance, we remark that **9** + **5** = 14.

1	0	0	0	0	0	0
1	**1**	0	0	0	0	0
1	2	**2**	0	0	0	0
1	3	5	**5**	0	0	0
1	4	**9**	14	**14**	0	0
1	**5**	**14**	28	42	**42**	0

We remark that the preceding table gives the mirror images of the lines of the previous double statistics.

4.2 Genetic Automata : Experiments and Spectral analysis

Generally, genetic computing **(GC)** used as a synonym for evolutionary computing **(EC)**. Evolutionary computing is a part of computer science which is inspired from biology. The field of **EC** can be separate into two component parts, i.e. the following two sub-categories :

- **genetic algorithms** which propose bio-inspired algorithm for modeling living system or for solving optimization problems using evolutionary computing.
- **genetic programming** which applies genetic operators on the codes themselves allowing to have evolutionary programming.

Evolutionary computation starts from the end of 1950s. As many research domains, there exists different concurrent approaches that have been developed by several scientists independently. To sum up these different approaches, we can simplify and consider two major classes:

- The **German** school which introduced with Rechenberg [33], the concept of evolution strategies, a method used to optimize real-value parameters, for example, and initially concerning airfoils. Fogel, Owens and Walsh [52] developed then the evolutionary programming which consists in the definition of evolvable machines to code parts of programs.
- The **American** school led by John Holland [42] and his students from the University of Michigan. J. Holland is usually considered as the inventor of genetic algorithms as we define them in the following. The original goal of J. Holland is to study the phenomenon of adaptation as it occurs in nature. From this study concerning natural phenomenon, he wants to design artificial systems that retain the important mechanisms of natural systems [19]. He proposes a original life simulation called **echo** [80].

Genetic algorithms plays an important role in studies of complex adaptive systems, ranging from adaptive agents in economic theory to the use of machine learning techniques in the design of complex devices such as aircraft turbines and integrated circuits.

John Holland, began his work on genetic algorithms at the beginning of the 60s. Holland had a double aim [19]:

- To improve the understanding of natural adaptation process,
- To design artificial systems having properties similar to natural systems.

The basic idea is as follow : the genetic pool of a given population potentially contains the solution, or a better solution, to a given adaptive problem. This solution is not active because the genetic combination on which it relies is split between several subjects. Only the association of different genomes can lead to the solution.

Holland method is especially effective because he not only considered the role of mutation (mutations improve very seldom the algorithms), but he also utilized genetic recombination, (crossover) : these recombination, the crossover of partial solutions greatly improve the capability of the algorithm to approach, and eventually find, the optimum [11].

Genetic algorithm can be presented as an universal model by applying it to economics, physiological psychology, game theory, and artificial intelligence and then outlines the way in which this approach modifies the traditional views of mathematical genetics.

Initially applying his concepts to simply defined artificial systems with limited numbers of parameters, Holland goes on to explore their use in the study of a wide range of complex, naturally occurring processes, concentrating on systems having multiple factors that interact in nonlinear ways. Along the way he accounts for major effects of co-adaptation and co-evolution: the emergence of building blocks, or schemata, that are recombined and passed on to succeeding generations to provide, innovations and improvements.

4.2.1 Genetic Algorithm Basis

Genetic Algorithms and Evolutive Systems

We present here the goal of evolutive systems and genetic algorithms. We try to explain the generalities of the genetic operators and the classical applications to optimization problems.

Evolutive systems take their background in evolutionist theory, synthesis of natural selection theory. We will not assume here these theories whose foundations belong to Darwin, and heredity theory which is due to Mendel. Of course, we do not here enter the debate about the validity of genetic theories of evolution. We just take locally the mechanisms induced by: population genetic modification, reproduction and selection. We can sum up the basis of the principles with the three following concepts:

- Evolution is the result of progressive alteration of living entities along generations.
- Reproduction is based on the genetic character which undergo during generations, some recomposition and some mutation.
- Natural selection consists of keeping the well adapted individuals to the context of their living environment.

One of the characteristics of living is that individuals have not been programmed to answer to a specific problem but they evolve inside their environment on the impulse - among another - of adaptation. The aim of evolutive systems is to build algorithms based on these concepts and models.

The Principle of Variation-Selection

The major aspect of genetic algorithms is the adaptation property. This property can be considered as the search of the optimum of a specific function. Genetic algorithms work on a population of individuals. The individuals are represented with chromosomes which are composed with primitive information called alleles. In term of computable formalization, chromosomes are generally strings or

sequences of information over a finite alphabet. To begin the algorithm, we generate a population of chromosomes.

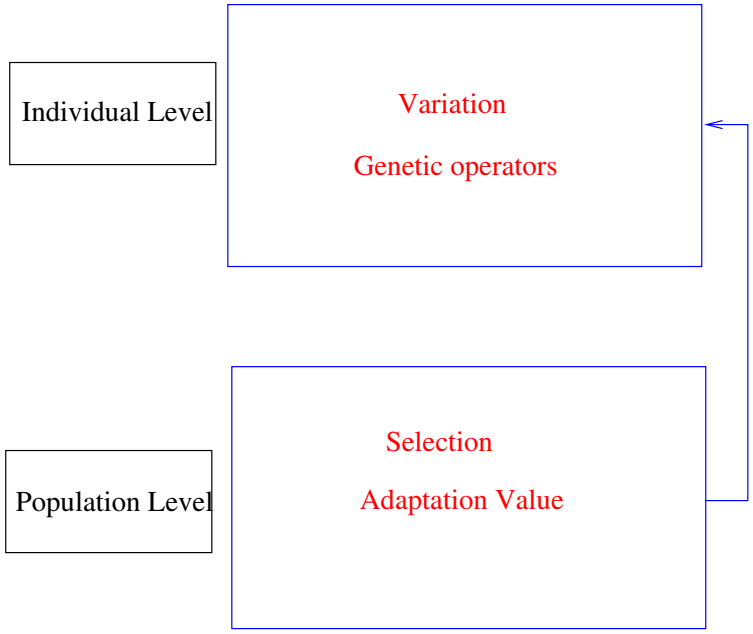


Figure 4.3: GA individual and population levels

The algorithm is based on a variation-selection process (see the Figure above):

- The variation step concerns the basic genetic operators on the individual level and so this step acts on the chromosomes. These basic operators are described in the following section. The result gives another population with a greater number of chromosomes than the initial one
- The selection step concerns the population level inside which a selection function modifies the whole population constitution. This step leads to keep only some of the chromosomes that have been generated during the variation step and which satisfy to specific constraints

The principle of selection is based on an evaluation function, called objective function. This function measures the individual performance. So we can deduce one probability for each individual as its ability to reproduce or to generate some

clones. This probability is called fitness.

The variation-selection process is included in an iterative method which is stopped after a defined number of iterations or when some specific conditions are verified.

To conclude about this principle, this process consists in the production of one population which has better adaptive property and so allows to converge toward an optimum.

Genetic Operations

During the variation step, we usually compute three genetic operators on the chromosomes :

- Reproduction/duplication
- Crossing-over
- Mutation

We describe, in the following, each of these operations.

Reproduction/ duplication

The first operator is the reproductive one. This level consists in the copy of the string or sequence describing chromosomes and generating an identical chromosomes. This process is made by each chromosome itself, in function of the value of the selective function which is a kind of measure of efficiency.

Crossing-over

Crossing-over is a process which, from two chromosomes, is able to generate two others ones. The crossing-over consist in the cutting of the initial chromosomes in the same place chosen at random. We permute the two parts of the chains with a crossing between them. Then we pass the permuted parts to the other part of chromosomes.

Mutation

In this step, we chose a few chromosomes at random to be candidates for the mutation process. Usually, the probability used for this selection is low. Then for each candidate chromosomes, we choose at random, one allele from each chromosome and we modify its value at random. The mutation step allows to go out from a local attractor during the optimization process.

Genetic Algorithm

The general genetic algorithm is described in the following. The genetic algorithm evolution is represented in figure (4.4).

Initialize the population
Calculate the adaptation degree of each individual
While (not finite or not convergent) do

- Reproduction the individual
- Apply the genetic operators
- Select the survivors among the parents and the children
- Calculate the adaptation degree of each individual

Closed.

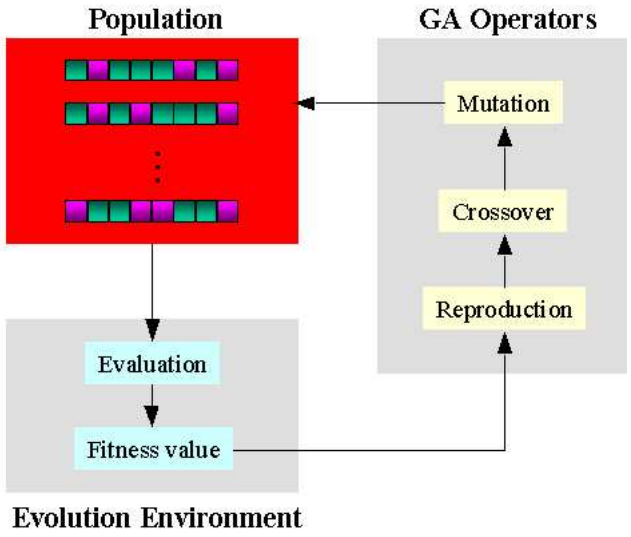


Figure 4.4: Genetic algorithm evolution

4.2.2 MuPAD Implementation of Genetic Automata

The software package **MuPAD** is a computer algebra system that allows to solve computational problems in pure mathematics as well as in applied areas such as the computer science and engineering.

We focus our attention here on the implementation of genetic algorithms for automata with multiplicities. The selected package used is *MuPAD-Combinat* which implements many data structures related to combinatorics.

```
package("Combinat"):
```

```
 +---+
 | T |              MuPAD-Combinat 1.3.2 (devel)
 +---+---+
 | A | K |          an open source MuPAD package for
 +---+---+---+
     | I | N |     research in Algebraic Combinatorics
     +---+---+
```

```
This package provides or extends the following libraries: combinat,
examples, Dom, Cat, output, experimental, IPC

For quick information on a particular library, please type:
info(library)

For the full html documentation, please browse through:
http://mupad-combinat.sf.net/ (project web page)
file:C:\PROGRA~1\SciFace\MUPADP~1.1\packages\Combinat\index.html
(local mirro\ r)
```

The first step is to construct automata with multiplicities. The domain element $DOM :: WeightedAutomaton(S)(n, A, i, t, f)$ represents the weighted automaton with multiplicities in the semi-Ring S given by its linear representation where n is dimension, A is alphabet, i is initial vector, t is the array of matrix transition for each letter of A and f final vector.

Automata Chromosome Coding

We define the chromosome for each automata with multiplicities as the sequence of all the matrices associated to each letter from the (linearly ordered) alphabet. The chromosomes are composed with alleles which are here the lines of the matrices [13].

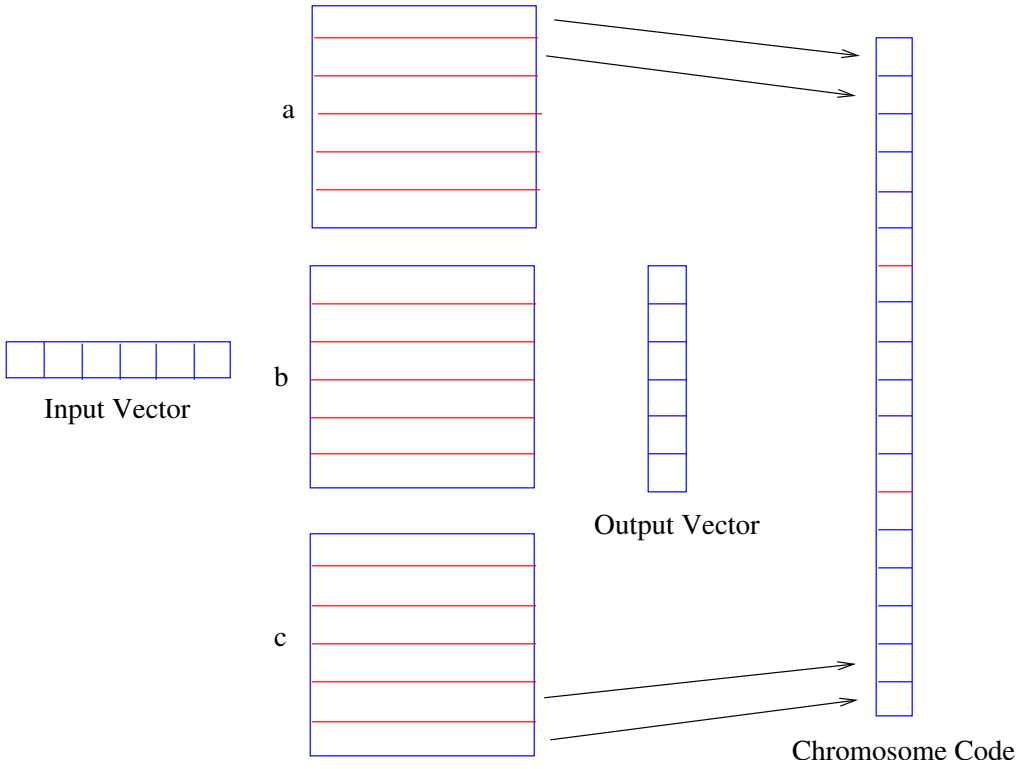


Figure 4.5: Chromosome code

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

The genetic algorithm over the population of automata with multiplicities follows a reproduction iteration broken up in three steps [19, 43, 58]:

- *Duplication*: where each automaton generates a clone of itself.

  The MuPAD associated function is

```
GenericDuplication:=proc(Chrom)
begin
  return(Chrom);
end_proc;
```

- *Crossing-over*: concerns a couple of automata. Over this couple, we consider a sequence of lines of each matrix for all. For each of these matrices, a permutation on the lines of the chosen sequence is made between the analogue matrices of this couple of automata.

  The MuPAD associated function is

```
CrossingOverWa:=proc(Chr1,Chr2,seq)
  local i,j,k,r,m,u,alph,letter,t1,t2;
begin
  alph:=DWA::alphabet(Chr1):
  m:=DWA::dimension(Chr1):
  for i from 1 to nops(alph) do
     letter:=op(alph,i):
     t1[letter]:=DWA::transition(Chr1,letter):
     t2[letter]:=DWA::transition(Chr2,letter):
  end_for:
  for i from 1 to nops(seq) do
     if (op(seq,i) mod m) <> 0 then
       k:=(op(seq,i) div m)+1:
       r:=(op(seq,i) mod m):
     else
       k:=(op(seq,i) div m):
       r:=m:
     end_if:
     letter:=op(alph,k):
     for j from 1 to m do
        u:=(t2[letter])[r,j]:
        (t2[letter])[r,j]:=(t1[letter])[r,j]:
    (t1[letter])[r,j]:=u:
     end_for:
  end_for:
```

```
  return(DWA(m,alph,DWA::initial(Chr1),t1,DWA::final(Chr1)),
  DWA(m,alph,DWA::initial(Chr2),t2,DWA::final(Chr2)));

end_proc;
```

Where $Chr1$ and $Chr2$ are the chromosome codes of the above automata with multiplicities which are involved with the Crossing-Over process and seq is the sequence of matrix lines on which the crossing over deal.

This function return two automata with multiplicities $DOM :: WeightedAutomata()$ corresponding to the result of the crossing-over the two initial chromosomes.

- *Mutation*: where a line of each matrix is randomly chosen and a sequence of new values is given for this line.

  The MuPAD associated function is

```
MutationWa:=proc(Chr,All,seq)
  local i,j,k,r,m,alph,letter,t;
begin
  alph:=DWA::alphabet(Chr):
  m:=DWA::dimension(Chr):
  for i from 1 to nops(alph) do
     letter:=op(alph,i):
     t[letter]:=DWA::transition(Chr,letter):
  end_for:
  if (All mod m) <> 0 then
    k:=(All div m)+1:
    r:=(All mod m):
  else
    k:=(All div m):
    r:=m:
  end_if:
  letter:=op(alph,k):
  for j from 1 to nops(seq) do
     (t[letter])[r,j]:=op(seq,j):
  end_for:
  return (DWA(m,alph,DWA::initial(Chr),t,DWA::final(Chr)));
end_proc;
```

Where Chr is the chromosome code of the automata with multiplicities involved in the mutation process, All is the chromosome allele (matrix line) where the mutation is applied and seq is the new matrix line which will replace All.

This function return the automata with multiplicities $DOM :: WeightedAutomata()$ corresponding to the result of the mutation over the initial chromosome.

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

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

We present in the following the program trace of a basic usage of the genetic operators over automata with multiplicities.

```
DWA:=Dom::WeightedAutomaton(Dom::Real);

DSM:=Dom::SparseMatrix(Dom::Real);

alphabet:=[x,y];

i1:=DSM(1,2,[0.45,0.65]); i2:=DSM(1,2,[0.672,0.328]);
i3:=DSM(1,2,[0.295,0.695]); f1:=DSM(2,1,[0.034,0.966]);
f2:=DSM(2,1,[0.87,0.13]); f3:=DSM(2,1,[0.581,0.419]);
t1[x]:=DSM(2,2,[[0.4,0.6],[0.12,0.88]]);
t1[y]:=DSM(2,2,[[0.796,0.204],[0.3333,0.6667]]);
t2[x]:=DSM(2,2,[[0.456,0.544],[0.789,0.211]]);
t2[y]:=DSM(2,2,[[0.5,0.5],[0.1,0.9]]);
t3[x]:=DSM(2,2,[[0.858,0.142],[0.277,0.723]]);
t3[y]:=DSM(2,2,[[0.011,0.989],[0.0003,0.9997]]);

SeqOfWa1:=[[DWA(2,alphabet,i1,t1,f1)],[DWA(2,alphabet,i2,t2,f2)],
```

```
[DWA(2,alphabet,i3,t3,f3)]];
```

$$\left[(\,0.45\;\;0.65\,),\ x=\begin{pmatrix}0.4 & 0.6\\ 0.12 & 0.88\end{pmatrix},\ y=\begin{pmatrix}0.796 & 0.204\\ 0.3333 & 0.6667\end{pmatrix},\ \begin{pmatrix}0.034\\ 0.966\end{pmatrix}\right]$$

$$\left[(\,0.672\;\;0.328\,),\ x=\begin{pmatrix}0.456 & 0.544\\ 0.789 & 0.211\end{pmatrix},\ y=\begin{pmatrix}0.5 & 0.5\\ 0.1 & 0.9\end{pmatrix},\ \begin{pmatrix}0.87\\ 0.13\end{pmatrix}\right]$$

$$\left[(\,0.295\;\;0.695\,),\ x=\begin{pmatrix}0.858 & 0.142\\ 0.277 & 0.723\end{pmatrix},\ y=\begin{pmatrix}0.011 & 0.989\\ 0.0003 & 0.9997\end{pmatrix},\ \begin{pmatrix}0.581\\ 0.419\end{pmatrix}\right]$$

```
SeqOfWa2:=[[GenericCrossingOver(op(SeqOfWa1,1),op(SeqOfWa1,2),
[1,3,4],CrossingOverWa)],[GenericMutation(op(SeqOfWa1,3),3,
[0.7,0.3],MutationWa)]];
```

$$\left[(\,0.45\;\;0.65\,),\ x=\begin{pmatrix}0.456 & 0.544\\ 0.12 & 0.88\end{pmatrix},\ y=\begin{pmatrix}0.5 & 0.5\\ 0.1 & 0.9\end{pmatrix},\ \begin{pmatrix}0.034\\ 0.966\end{pmatrix},\ (\,0.672\;\;0.328\,),\ x=\begin{pmatrix}0.4 & 0.6\\ 0.789 & 0.211\end{pmatrix},\ y=\begin{pmatrix}0.796 & 0.204\\ 0.3333 & 0.6667\end{pmatrix},\right.$$

$$\left.(\,0.295\;\;0.695\,),\ x=\begin{pmatrix}0.858 & 0.142\\ 0.277 & 0.723\end{pmatrix},\ y=\begin{pmatrix}0.7 & 0.3\\ 0.0003 & 0.9997\end{pmatrix},\ \begin{pmatrix}0.581\\ 0.419\end{pmatrix}\right]$$

4.2.3 Experiments and Spectral Analysis

In this section, we will first give some preliminaries allowing to recall some definitions, properties and theorems on matrix and spectral analysis. Then we present two series of experimentations corresponding to the evolution of a population of genetic automata.

The first series is computed without fitness function and shows some analysis based on spectral analysis outputs. The second series is computed with a specific fitness function based on auto-regulation selection process which uses a fitness function computed with the spectral of the population itself.

Preliminaries

We present here a brief introduction about **matrices** and their calculus.
The **row vector** is a set of numbers $(a_1, a_2, ..., a_m)$ written as

$$\vec{a} = (a_1, a_2, ..., a_m)$$

where $a_i, 1 \leq i \leq m$ is the i-th components of $\vec{a}$. *More simply a row vector is a* $[1 \times m]$ *matrix.*

Similarly, we can define the **column vector** as a set of numbers written as

$$\vec{a} = \begin{pmatrix} a_1 \\ a_2 \\ . \\ . \\ . \\ a_n \end{pmatrix}$$

when $a_i, 1 \leq i \leq n$ is the j-th components of the column vector $\vec{a}$. *More simply a row vector is a* $[n \times 1]$*matrix.*

In the study of matrices we have two things to consider; numbers and arrangements of these numbers. We define a matrix A as a rectangular array (arrangement) of elements (e.g.,numbers) in m-rows and n-columns considered as a single entity such as :

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & . & & . \\ . & . & & . \\ . & . & & . \\ a_{m1} & a_{m2} & ... & a_{mn} \end{pmatrix}$$

We can simply write for this expression by $(a_{ij})_{m \times n}$, $(1 \leq i \leq m$ and $1 \leq j \leq n)$, when a_{ij} are numbers lies in the i-th row and j-th column. The degree of the matrix is $m \times n$ where m is the numbers of rows and n is the numbers of columns of the matrix. We have many kinds of matrices, but the most important among them is the **square matrix** which has the same numbers of rows and columns [63].

A square matrix in which every number over its main diagonal is zero called lower triangular matrix, and if every number below its main diagonal is zero called upper triangular matrix. A square matrix in which every number not on its main diagonal is zero called diagonal matrix. The identity matrix is an example of the diagonal matrix.

The determinant of a matrix is defined only on square matrices and it is a mapping from the set of all square matrices to the set of all real numbers

$$|\ | : M_n \rightarrow R$$

There are many ways to obtain a determinant of a matrix, for example Laplace method to obtain the determinant of those square matrices with degree 3×3 and more [63].

Eigenvalue problem

Let A be a square matrix. The set of (complex) values λ such that $|A - \lambda I| = 0$ is called the set of eigenvalues (or the spectrum)of A. The eigenvalues of an upper or lower triangular matrix are the diagonal elements. If $\lambda_1, \lambda_2, ..., \lambda_n$ are eigenvalues of a square matrix A, then

$$|A| = \lambda_1 \lambda_2 ... \lambda_n$$

Eigenvalues are a set of scalars associated with a polynomial equation (whose coefficients are functions of the entries of the original matrix). Similarly, Eigenvectors are a set of vectors associated with matrix equations. The determination of the eigenvalues and eigenvectors of a system is important in physics and engineering, where it is (most of times) equivalent to matrix diagonalization and arises in such common applications as stability analysis.

The eigenvalues of a matrix A are called its spectrum $\lambda(A)$. The left hand side of the characteristic equation $|A - \lambda I| = 0$ is called characteristic polynomial.

Adjacency Matrix

The adjacency matrix of a simple graph is a matrix with rows and columns labelled by graph vertices, with 1 or 0 in position (v_i, v_j) according to wether v_i and v_j are adjacent or not. For example

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

Permutation Matrix

A permutation matrix is a matrix obtained by permuting the rows of an $n \times n$ identity matrix according to some permutation of the numbers 1 to n. Every row and column therefore contains precisely a single 1 with 0 everywhere else, and every permutation corresponds to a unique permutation matrix. Therefore, there exists $n!$ permutation matrices of size n. For example

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

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

Perron-Frobenius Theorem

For general knowledge about positive matrices and irreducible forms, the reader is referred to [71].
If all elements a_{ij} of an irreducible matrix A are non-negative, then $R = MinM_\lambda$ is an eigenvalue of A and all the eigenvalues of A lie on the disk $|Z| \leq R$, where

$$M_\lambda = Inf\{\mu : \mu\lambda_i > \sum_{j=1}^{n} |a_{ij}|\lambda_j, 1 \leq i \leq n\}$$

And $\lambda = \{\lambda_1, \lambda_2, ..., \lambda_n\}$ is a set of non-negative numbers which are not all zero.

Furthermore, if A has exactly p eigenvalues $(p \leq n)$ on the circle $|Z| = R$, then the set of all its eigenvalues is invariant under rotation by $\frac{2\pi}{p}$ about the origin.

Eigenvalues Calculus with MuPAD

The function call $export(linalg)$ exports all functions of the library $linalg$ (linear algebra) that enable us to define matrices. They offer matrix arithmetic and several functions for matrix manipulation.

The library $linalg$ is based on the domains $DOM :: Matrix$ and $DOM :: SquareMatrix$. A domain created by $DOM :: Matrix$ represents matrices of arbitrary rows and columns over a specified ring.

To calculate the eigenvalues of a matrix A under MuPAD we use $linalg :: eigenvalues(A)$ which returns a list of the eigenvalues of the matrix A. For example

```
export(linalg);

A:= matrix([[1,4,2],[1,4,2],[2,5,3]]);

linalg::eigenvalues(A);
```

The result of the above calculation is $\{0, \sqrt{15}+4, 4-\sqrt{15}\}$.

To calculate the eigenvectors of a matrix A under MuPAD we use

```
linalg::eigenvectors(A);
```

Which compute the eigenvalues and eigenvectors of the matrix A. Finally, to calculate the spectral radius of a matrix A under MuPAD we use

```
numeric::spectralradius[A, s, i];
```

Where s is starting vector parameter which is a one dimensional array or a list of length m. The parameter i is the maximal number of iterations which is a positive integer.

Experiments

Here, two series of experiments are presented. The first one, is computed with out fitness and give histogram evolution of spectral analysis. The second one, uses the results of this spectral analysis over the population itself to implement a fitness and so an auto-regulated process.

First Series of Experiments

- **Step 1 :** We generate a random population of n-probabilistic automata i.e. the function

```
inipop:=proc(n)
begin
  alphabet:=[x];
   wa1:=[];
for j from 1 to n do
    i:=DSM(1,3,[r3seq()]):
    f:=DSM(3,1,[r3seq()]):
    t[x]:=DSM(3,3,[[r3seq()],[r3seq()],[r3seq()]]):
    wa1:=append(wa1,DWA(3,alphabet,i,t,f));
end_for;
  return(wa1);
end_proc;
```

- **Step 2 :** We compute an iteration which make evolve the automata population by applying crosser-over and mutation, i.e. the functions

```
newpop:=proc(wa)
local j,i, nelem, waTemp, seqPermut;
begin
  nelem:=nops(wa)/2;
  waResult:=[];
  for j from 1 to nelem do
    seqPermut:=[];
    for i from 1 to 3 do
  if (frandom()<0.5) then seqPermut:=append(seqPermut,i)
  end_if;
    end_for;
  waTemp:=[GenericCrossingOver(op(wa,j),op(wa,nelem+j),
```

```
    seqPermut, CrossingOverWa)];

    waResult:=append(waResult, op(waTemp,1), op(waTemp,2));
    end_for;
    return(waResult);
  end_proc;


  newpopmut:=proc(wa)
    local i,j, waResult, waTemp;
  begin
    nelem:=nops(wa);
    waResult:=[];
    for j from 1 to nelem do
      waTemp:=op(wa,j);
      for i from 1 to 3 do
      if (frandom()<0.03) then
        waTemp:=GenericMutation(waTemp, i, [r3seq()], MutationWa);

      end_if;
      end_for;
      waResult:=append(waResult,waTemp);
    end_for;
    return(waResult);
  end_proc;
```

At each step, we compute the secondary greatest eigenvalue module because for each probabilistic automata, the greatest one is always known. Its value is 1 associated to the vector

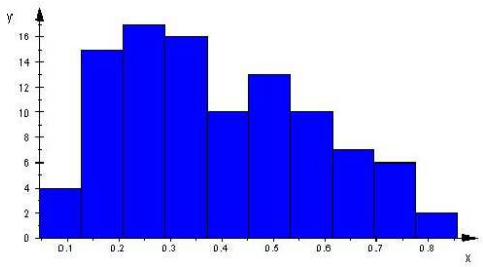
$$\begin{pmatrix} 1 \\ 1 \\ . \\ . \\ . \\ 1 \end{pmatrix}$$

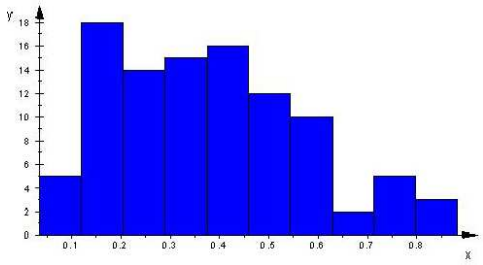
We represent the histogram of the distribution of this secondary eigenvalue module of each automaton over the whole population, i.e. the function

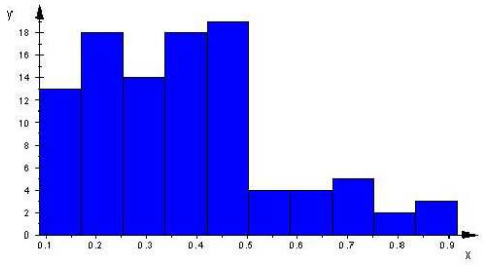
```
spa1:=proc(wa,nb)
```

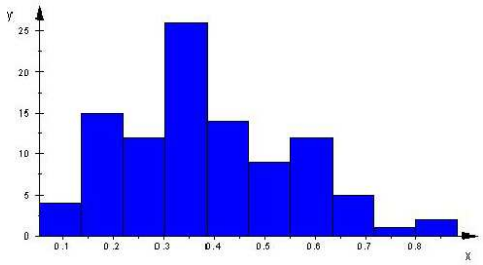
```
 local vpm, ii,s;
 begin
 vpm:=[]:
 for ii from 1 to
nops(wa) do
s:=linalg::eigenvalues(DWA::transition(op(wa,ii),x));
vpm:=append(vpm,maxMoins1(s));
 end_for:
return(plot(plot::Histogram2d(vpm, Cells=[nb],Color=RGB::Blue)));
end_proc;
```

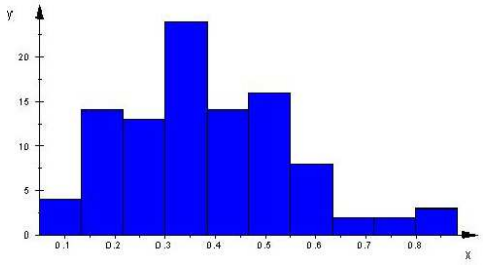
In the following, present an experiment where the series of histograms computed with these previous steps, is obtained

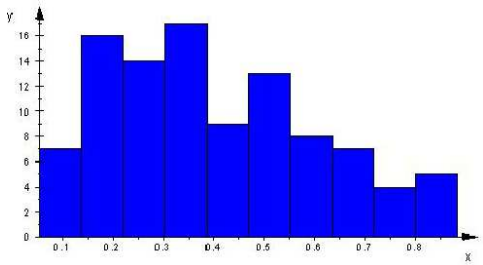





y
18
16
14
12
10
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
x


y
25
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x


y
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x


y
16
14
12
10
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x

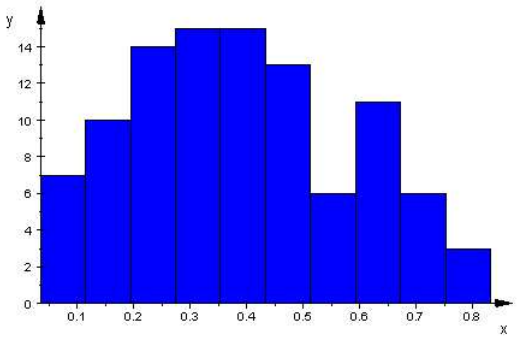
Second Series of Experiments

We generate some similar histograms series with adding a fitness function computation and so a selection step

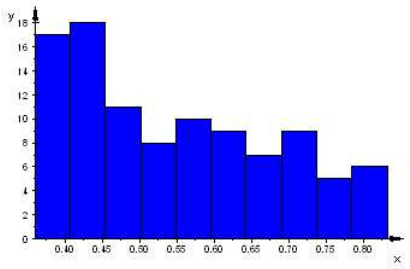
- The fitness function of each automaton corresponds to the value of the secondary greatest eigenvalue module.
- The selection process selects the automata corresponding to the greatest secondary eigenvalue module from two populations of automata.

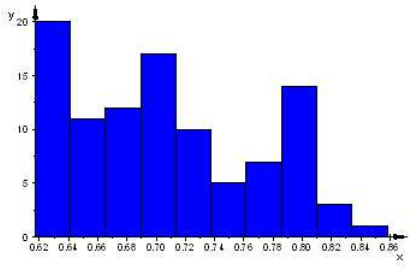
The algorithm is exactly the same that in the first series with the addition of the selection process just after the creation of the new population. The selection process deals with the parent and the new generation and give in output the selected automata population.

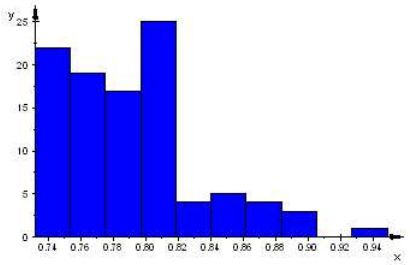
In the following, we present a new experiment integrating the selected process, the abscisse scale is adapted to the values of the secondary greatest eigenvalue module.

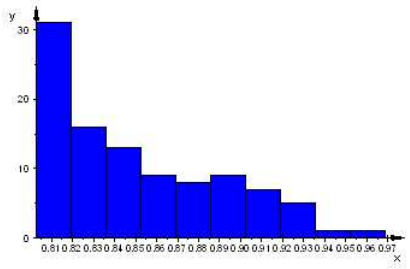


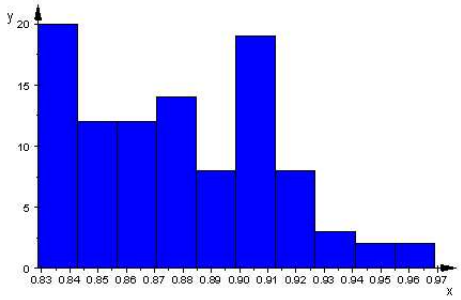
```
wa2:=newpop(wa1): wa1:=selection(wa1,wa2,100,maxvpm1):
spa1(wa1,10);
```

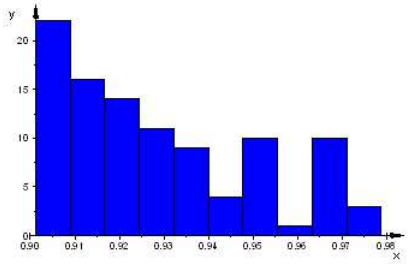


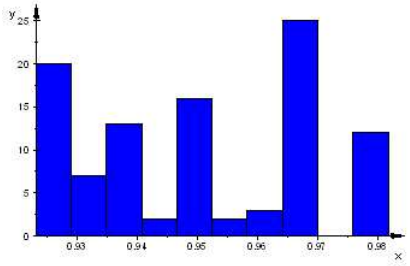


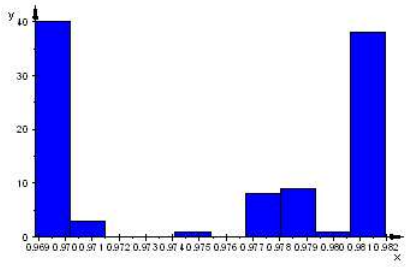


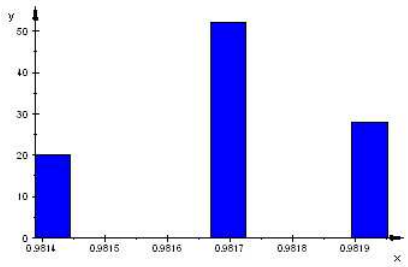

y
30
20
10
0
0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97
x

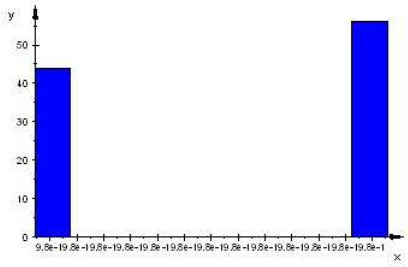

y
20
15
10
5
0
0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97
x


y
20
15
10
5
0
0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98
x


y
25
20
15
10
5
0
0.93 0.94 0.95 0.96 0.97 0.98
x


y
40
30
20
10
0
0.969 0.970 0.971 0.972 0.973 0.974 0.975 0.976 0.977 0.978 0.979 0.980 0.981 0.982
x


y
50
40
30
20
10
0
0.9814
0.9815
0.9816
0.9817
0.9818
0.9819
x


y
50
40
30
20
10
0
x


y
100
90
80
70
60
50
40
30
20
10
0
x

Results Analysis and Interpretation

As expected by the fitness function, we generate successive populations which are characterized by the convergence process which leads to obtain a population where the greatest secondary eigenvalue module of all automata with multiplicities is equals 1.

This processus is an auto-regulation processus on the population itself, managed by a feed-back of the result control on the system itself.

By controlling the generating automata population with eigenvalues module equals 1, we constraint the system to not be dissipative. The spectral analysis and the eigenvalues module give some indications on the dissipative aspect during the evolution. If the eigenvalue modules are less than 1, then we can conclude that a dissipation exists.

The non dissipative evolution process over automata with multiplicities proposed here, is a tool based on the power of the algebraic representation of the automata with multiplicities, using some computation on their matrix representation. So we deal on a dynamic data structure according to the complexity of dissipative process over some self-organized systems. Without managing each matrix coefficient value, we give a kind of control tool according to dissipative process increasing or decreasing.

4.3 Conclusion

This chapter covers two aspects. The first one is to adapt our results about dynamic structures presented in chapter 3 to model a bank account which is composed of two accounts P and R by giving the behavior of R which represent the benefit of P. The second one is to implement genetic automata (probabilistic automata) by applying genetic operators using MuPAD software for studying spectral analysis of the automata population. This analysis allows us to implement an auto-regulation process as an evolution of automata population toward a dissipative process.

Chapter 5

Conclusions and Perspectives

We have presented a toy-model economic behavior based on local rules and we propose some global function expression which can be also described by three combinatorics structures. By this application, we point out a one-to-one correspondence between three other ballot-like structures. The innovative aspect of this work deals with a constructive development of the involved bijections.

The goal of this PhD thesis is to give some combinatorial tools to understand and to model some dynamical evolution of complex systems. We present some applications to economic models and give a genetic tool for self-regulation of dissipative systems implemented by automata population and control by spectral analysis.

For the future, we can continue to develop some applications on complexity

- We have started to study the phenomenology of sand pile model using Young tableaux. The connection between standard Young tableaux and sand pile model avalanches as a most typical example of the notion of self-organized criticality can be treated. Due to these avalanches, it seems possible to obtain some tableaux that respect the standard Young tableaux conditions, that is, tableaux with increasing number from **1** to **n** in each rows and columns. In this project, We will try to obtain a standard Young tableau with some properties by applying a new algorithm called Sand Pile One-Dimensional Avalanche Model (SPODAM).

- The financial market today possed a very large tools to simulate some phenomena which we can not control it by the classical methods. This tools has sometimes advantages and disadvantages. For example, some model describe with precision the option prices, but we prefer Black-Scholes model

to calculate option prices which is not more perfect than other models. It is simple and more near to reality. In the real financial market, we have the agents who are homo sapiens and not the homo economicus, so we have to make face to some difficult situations. The experimentations in the laboratory is useful to give us some predictions and expectations about the phenomena which we try to understand but these experimentations will not be exactly conform to the reality, in particular, in the world of finance. An innovative process could be to use spectral analysis as developed in the previous chapter to control the dissipative phenomena involved in financial market evolution.

Chapter 6

Annexe

Contents

6.1 Genetic Automata under MuPAD

```
package("Combinat");
```

```
 +---+
 | T |             MuPAD-Combinat 1.3.2 (devel)
 +---+---+
 | A | K |          an open source MuPAD package for
 +---+---+---+
     | I | N |    research in Algebraic Combinatorics
     +---+---+

This package provides or extends the following libraries:
 combinat,examples, Dom, Cat, output, experimental, IPC

For quick information on a particular library, please type:
info(library)

For the full html documentation, please browse through:
http://mupad-combinat.sf.net/ (project web page)
file:C:\PROGRA~1\SciFace\MUPADP~1.1\packages\Combinat\
 index.html(local mirro\ r)
```

```
DWA:=Dom::WeightedAutomaton(Dom::Real);

DSM:=Dom::SparseMatrix(Dom::Real);

alphabet:=[x,y];

i1:=DSM(1,2,[0.45,0.65]); i2:=DSM(1,2,[0.672,0.328]);
i3:=DSM(1,2,[0.295,0.695]); f1:=DSM(2,1,[0.034,0.966]);
f2:=DSM(2,1,[0.87,0.13]); f3:=DSM(2,1,[0.581,0.419]);
t1[x]:=DSM(2,2,[[0.4,0.6],[0.12,0.88]]);
t1[y]:=DSM(2,2,[[0.796,0.204],[0.3333,0.6667]]);
t2[x]:=DSM(2,2,[[0.456,0.544],[0.789,0.211]]);
t2[y]:=DSM(2,2,[[0.5,0.5],[0.1,0.9]]);
t3[x]:=DSM(2,2,[[0.858,0.142],[0.277,0.723]]);
```

```
t3[y]:=DSM(2,2,[[0.011,0.989],[0.0003,0.9997]]);



SeqOfWa1:=[DWA(2,alphabet,i1,t1,f1),DWA(2,alphabet,i2,t2,f2),
DWA(2,alphabet,i3,t3,f3)]:

GenericDuplication:=proc(Chrom) begin
  return(Chrom);
end_proc;

GenericCrossingOver:=proc(Chrom1,Chrom2,SeqOfAll,FctCrossingOver)
begin
  FctCrossingOver(Chrom1,Chrom2,SeqOfAll);
end_proc;

CrossingOverWa:=proc(Chr1,Chr2,seq)
  local i,j,k,r,m,u,alph,letter,t1,t2;
begin
  alph:=DWA::alphabet(Chr1):
  m:=DWA::dimension(Chr1):
  for i from 1 to nops(alph) do
     letter:=op(alph,i):
     t1[letter]:=DWA::transition(Chr1,letter):
     t2[letter]:=DWA::transition(Chr2,letter):
  end_for:
  for i from 1 to nops(seq) do
     if (op(seq,i) mod m) <> 0 then
       k:=(op(seq,i) div m)+1:
       r:=(op(seq,i) mod m):
     else
       k:=(op(seq,i) div m):
       r:=m:
     end_if:
     letter:=op(alph,k):
     for j from 1 to m do
        u:=(t2[letter])[r,j]:
        (t2[letter])[r,j]:=(t1[letter])[r,j]:
    (t1[letter])[r,j]:=u:
     end_for:
  end_for:
```

```
  return(DWA(m,alph,DWA::initial(Chr1),t1,DWA::final(Chr1)),
  DWA(m,alph,DWA::initial(Chr2),t2,DWA::final(Chr2)));

end_proc;



GenericMutation:=proc(Chrom,All,SeqOfGen,FctMutation) begin
  FctMutation(Chrom,All,SeqOfGen);
end_proc;

MutationWa:=proc(Chr,All,seq)
  local i,j,k,r,m,alph,letter,t;
begin
  alph:=DWA::alphabet(Chr):
  m:=DWA::dimension(Chr):
  for i from 1 to nops(alph) do
     letter:=op(alph,i):
     t[letter]:=DWA::transition(Chr,letter):
  end_for:
  if (All mod m) <> 0 then
    k:=(All div m)+1:
    r:=(All mod m):
  else
    k:=(All div m):
    r:=m:
  end_if:
  letter:=op(alph,k):
  for j from 1 to nops(seq) do
     (t[letter])[r,j]:=op(seq,j):
  end_for:
  return (DWA(m,alph,DWA::initial(Chr),t,DWA::final(Chr)));
end_proc;

SeqOfWa2:=[GenericCrossingOver(op(SeqOfWa1,1),op(SeqOfWa1,2),
[1,3,4],CrossingOverWa),GenericMutation(op(SeqOfWa1,3),3,
[0.7,0.3],MutationWa)]:


SeqOfWa1;
```

$$\left[(\,0.45\;\;0.65\,),\ x=\begin{pmatrix}0.4 & 0.6\\ 0.12 & 0.88\end{pmatrix},\ y=\begin{pmatrix}0.796 & 0.204\\ 0.3333 & 0.6667\end{pmatrix},\ \begin{pmatrix}0.034\\ 0.966\end{pmatrix}\right]$$

$$\left[(\,0.672\;\;0.328\,),\ x=\begin{pmatrix}0.456 & 0.544\\ 0.789 & 0.211\end{pmatrix},\ y=\begin{pmatrix}0.5 & 0.5\\ 0.1 & 0.9\end{pmatrix},\ \begin{pmatrix}0.87\\ 0.13\end{pmatrix}\right]$$

$$\left[(\,0.295\;\;0.695\,),\ x=\begin{pmatrix}0.858 & 0.142\\ 0.277 & 0.723\end{pmatrix},\ y=\begin{pmatrix}0.011 & 0.989\\ 0.0003 & 0.9997\end{pmatrix},\ \begin{pmatrix}0.581\\ 0.419\end{pmatrix}\right]$$

```
SeqOfWa2;
```

$$\left[(\,0.45\;\;0.65\,),\ x=\begin{pmatrix}0.456 & 0.544\\ 0.12 & 0.88\end{pmatrix},\ y=\begin{pmatrix}0.5 & 0.5\\ 0.1 & 0.9\end{pmatrix},\ \begin{pmatrix}0.034\\ 0.966\end{pmatrix},\ (\,0.672\;\;0.328\,),\ x=\begin{pmatrix}0.4 & 0.6\\ 0.789 & 0.211\end{pmatrix},\ y=\begin{pmatrix}0.796 & 0.204\\ 0.3333 & 0.6667\end{pmatrix},\ \begin{pmatrix}0.87\\ 0.13\end{pmatrix}\right]$$

$$\left[(\,0.295\;\;0.695\,),\ x=\begin{pmatrix}0.858 & 0.142\\ 0.277 & 0.723\end{pmatrix},\ y=\begin{pmatrix}0.7 & 0.3\\ 0.0003 & 0.9997\end{pmatrix},\ \begin{pmatrix}0.581\\ 0.419\end{pmatrix}\right]$$

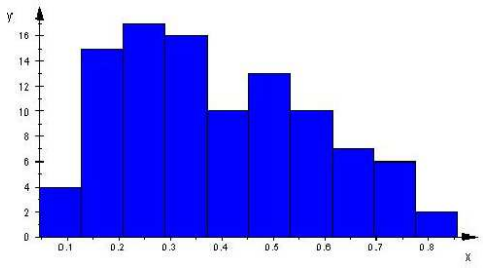
6.2 Spectral analysis programming under MuPAD

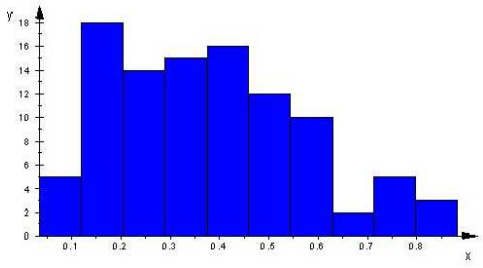
```
r3seq:=proc() begin
  a:=frandom();b:=frandom()*(1-a);c:=1-a-b;
  return (a,b,c);
end_proc;



inipop:=proc(n) begin
  alphabet:=[x];
   wa1:=[];
for j from 1 to n do
    i:=DSM(1,3,[r3seq()]):
    f:=DSM(3,1,[r3seq()]):
    t[x]:=DSM(3,3,[[r3seq()],[r3seq()],[r3seq()]]):
    wa1:=append(wa1,DWA(3,alphabet,i,t,f));
end_for;
  return(wa1);
end_proc;


wa2:=inipop(100):

maxMoins1 := proc(s)
 begin
  maxs:=0.0: eps:=0.00000001:
  for i from 1 to nops(s) do
    value:=abs(op(s,i)):
    if (abs(value-1.0)>eps) then
      if value > maxs then maxs:=value end_if;
    end_if;
  end_for;
  maxs;
end_proc;



newpop:=proc(wa)
local j,i, nelem, waTemp, seqPermut;
```

```
begin
  nelem:=nops(wa)/2;
  waResult:=[];
  for j from 1 to nelem do
    seqPermut:=[];
    for i from 1 to 3 do
  if (frandom()<0.5) then seqPermut:=append(seqPermut,i) end_if;
    end_for;
  waTemp:=[GenericCrossingOver(op(wa,j),op(wa,nelem+j),
  seqPermut, CrossingOverWa)];

  waResult:=append(waResult, op(waTemp,1), op(waTemp,2));
  end_for;
  return(waResult);
end_proc;



newpopmut:=proc(wa)
  local i,j, waResult, waTemp;
begin
  nelem:=nops(wa);
  waResult:=[];
  for j from 1 to nelem do
    waTemp:=op(wa,j);
    for i from 1 to 3 do
    if (frandom()<0.03) then
      waTemp:=GenericMutation(waTemp, i, [r3seq()], MutationWa);
    end_if;
    end_for;
    waResult:=append(waResult,waTemp);
  end_for;
  return(waResult);
end_proc;


spa1:=proc(wa,nb)
local vpm, ii,s;
begin vpm:=[]:
for ii from 1 to
nops(wa) do
```

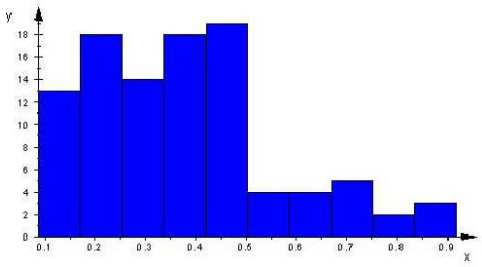
```
s:=linalg::eigenvalues(DWA::transition(op(wa,ii),x));
vpm:=append(vpm,maxMoins1(s));
end_for:
return(plot(plot::Histogram2d(vpm, Cells=[nb],Color=RGB::Blue)));
end_proc;
```

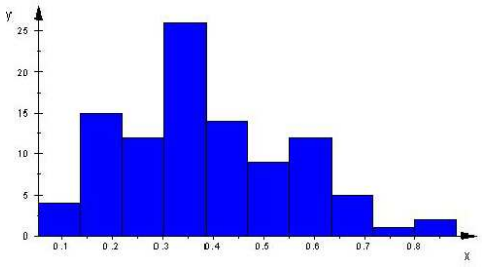
```
for iii from 1 to 10 do
wa2:=newpop(wa2):wa2:=newpopmut(wa2):
spa1(wa2,10);
end_for;
```

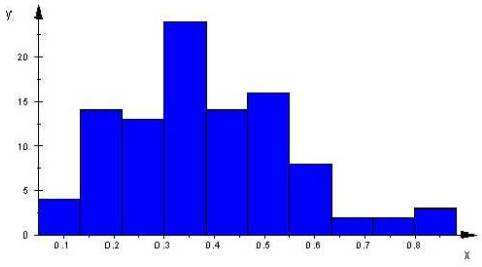


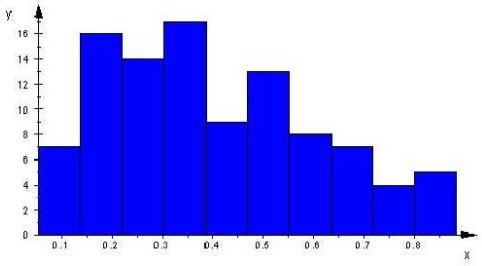


```
inside:=proc(x,l)
 local i; begin
  for i from 1 to nops(l) do
    if (x=l[i]) then return(i) end_if;
  end_for;
  return(0);
```


y
18
16
14
12
10
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
x

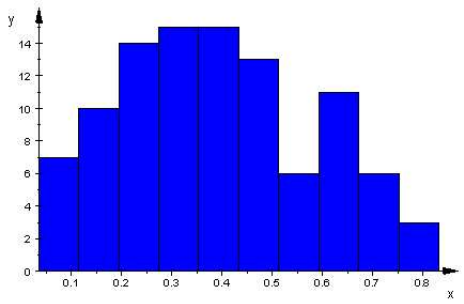

y
25
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x


y
20
15
10
5
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x

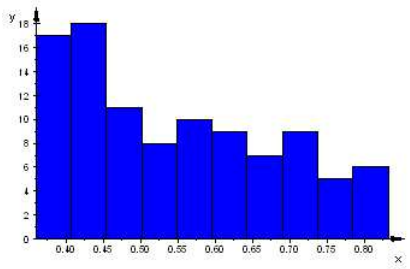

y
16
14
12
10
8
6
4
2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x

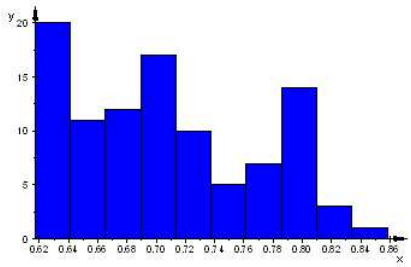
```
end_proc;



maxvpm1:=proc(a)
begin
 print("vp:",linalg::eigenvalues(DWA::transition(a,x)));
return(maxMoins1(linalg::eigenvalues(DWA::transition(a,x))));
end_proc;



selection:=proc(p1,p2,nb,fitness)
local i, p3, tabu1, tabu2, start,
nv, m, mpos, mpop,x, mx; begin
p3:=[];tabu1:=[];tabu2:=[];start:=TRUE; for i from 1 to nb do
  start:=TRUE;
  for i1 from 1 to nops(p1) do
    x:=op(p1,i1);nv:=fitness(x);
    if (inside(i1,tabu1)=0) then
      if start then
        m:=nv; mpos:=i1; mpop:=1; mx:=x; start:=FALSE;
      else
        if (nv>m) then m:=nv; mpos:=i1; mpop:=1; mx:=x;
        end_if;
      end_if;
    end_if;
  end_for;
  for i2 from 1 to nops(p2) do
    x:=op(p2,i2);nv:=fitness(x);
    if (inside(i2,tabu2)=0) then
      if start then
        m:=nv; mpos:=i2; mpop:=2; mx:=x; start:=FALSE;
      else
        if (nv>m) then m:=nv; mpos:=i2; mpop:=2; mx:=x;end_if;
      end_if;
    end_if;
  end_for;
  p3:=append(p3,mx);
  if (mpop=1) then tabu1:=append(tabu1,mpos)
  else tabu2:=append(tabu2,mpos) end_if;
```

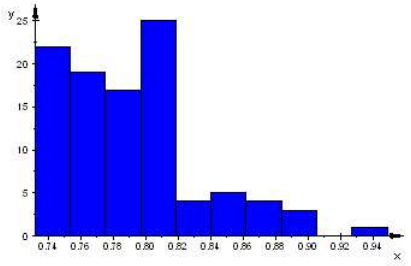
```
end_for; return(p3);
end_proc;


wa1:=inipop(100): spa1(wa1,10);
```

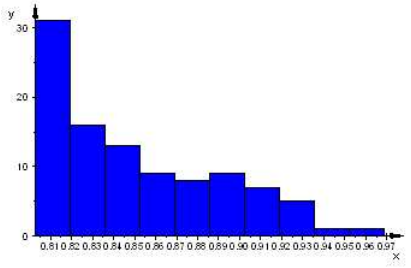


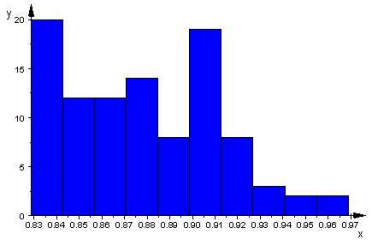

```
wa2:=newpop(wa1): wa1:=selection(wa1,wa2,100,maxvpm1):
spa1(wa1,10);
```

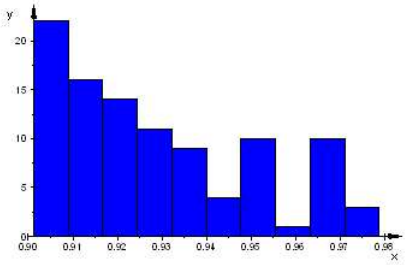


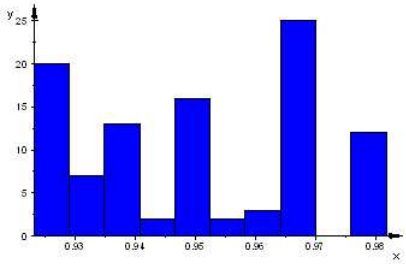

y
20
15
10
5
0
0.62 0.64 0.66 0.68 0.70 0.72 0.74 0.76 0.78 0.80 0.82 0.84 0.86
x

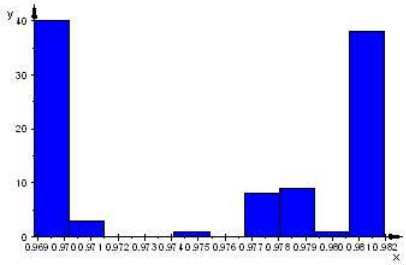

y
25
20
15
10
5
0
0.74 0.76 0.78 0.80 0.82 0.84 0.86 0.88 0.90 0.92 0.94
x

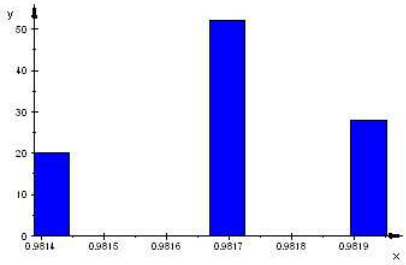

y
30
20
10
0
0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97
x

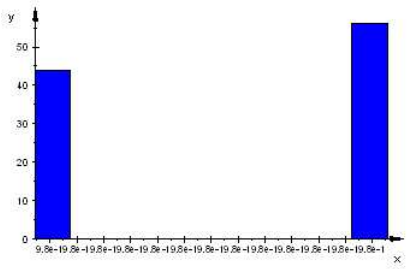

y
20
15
10
5
0
0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.90 0.91 0.92 0.93 0.94 0.95 0.96 0.97
x


y
20
15
10
5
0
0.90
0.91
0.92
0.93
0.94
0.95
0.96
0.97
0.98
x


y
25
20
15
10
5
0
0.93
0.94
0.95
0.96
0.97
0.98
x


y
40
30
20
10
0
x


y
50
40
30
20
10
0
0.9814
0.9815
0.9816
0.9817
0.9818
0.9819
x


y
50
40
30
20
10
0
x


y
100
90
80
70
60
50
40
30
20
10
0
x

Chapter 7

Bibliography

Bibliography

[1] Aziz Alaoui and Cyrille Bertelle, *Emergent Properties in Natural and Artificial Dynamical Systems*, Springer, Understanding Complex Systems series, 2006.

[2] Alfred Young, *Quantitative substitutional analysis*, I-IX Proc. London Math. Soc. 1901-1952.

[3] A. R. Richardson, *Simultaneous Linear Equations Over a Division Ring*, Proc. Lond. Math. Soc., vol 28, 395-420, 1928.

[4] Alfred Marshall, *Principales of Economics*, 459, (8-th Edition, Macmillan, London, 1920).

[5] A. Bechara, H. Damasio and A.R. Damasio, *Emotion, Decision Making and the Orbitofrontal Cortex*, Cerebral Cortex, 10:295-307, march 2000.

[6] B.E. Sagan, *The Symmetric Group*, 1991.

[7] Bernard Pavard and Julie Dugdale, *An Introduction to Complexity in Social Science*, GRIC-IRIT, Toulouse, France.

[8] C. Schensted, *Longest Increasing and Decreasing Subsequences*, Canad. J. Math, 13 (1961), 179-191.

[9] C. Chauve, *Half of the Nodes of Catalan Trees are Leaves*, http://www.lacim.uqam.ca/ chauve/Publications/.

[10] C. Langton, *Studying Artificial Life with Cellular Automata*, Physica D, 22, 1986.

[11] C. Emmeche, *Garden in the Machine: The Emerging Science of Artificial Life*, Princeton University Press, 1994, pp. 114 ss.

[12] 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.

[13] C. Bertelle, M. Flouret, V. Jay, D. Olivier, and J.-L. Ponty *Genetic Algorithms on Automata with Multiplicities for Adaptive Agent Behaviour in Emergent Organizations*, In SCI'2001, Orlando, Florida, USA, 22-25th July 2001.

[14] Definethat, *www.definethat.com/define/296.htm*

[15] Definethat, *www.definethat.com/define/311.htm*

[16] Donald E. Knuth, *The Art of Computer Programming*, Addison-Wesley, Vol 1 and 2 (1997), Vol 3 (1998).

[17] D. R. Raymond and D. Wood, *Grail: A C++ Library for Automata and Expressions*, J. Symbolic Comput., vol 17, 341-350, 1994.

[18] David Kline, *Positive Feedback, Lock-in and Environmental Policy*, Policy Sciences 34:95-107, 2001.

[19] D. Goldberg, *Genetic Algorithms*, Addison Wesley, 1989.

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

[21] E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence*, Oxford University Press, 1999.

[22] F. Alarcón and D. Anderson, *Commutative Semirings and Their Lattices of Ideals*, Houston J. Math., vol 20, 1994.

[23] F. Varela, *Autonomie et Connaissance, Essai sur Le Vivant*, Editions Du Seuil, 1989.

[24] F. Capra, *The Web of Life*, Anchor books, 1996.

[25] G. Duchamp and M. Flouret and E. Laugerotte and J.-G. Luque, *Direct and Dual Laws for Automata With Multiplicites* , Theoret. Comput. Sci. 105-120, 2001.

[26] G. Duchamp, C. Reutenauer, Un critère de rationalité provenant de la géomètrique non-commutative, Invent. Math, 128, 613-622, 1997.

[27] G. Weisbuch, A. Kirman and D. Herreiner, *Market Organisation and Trading Relationships*, 16/02/1998.

[28] Herbert S. Wilf, *Ascending Subsequences of Permutations and the Shape of Tableaux*, J. of Combina. Theory, Series A 60 (1992), 155-157.

[29] Herbert S. Wilf, *The Computer-Aided Discovery of a Theorem About Young Tableaux*, J. Symbolic Computation, Series 20 (1995), 731-735.

[30] I.G. Macdonald, *Symmetric Functions and Hall Polynomials*, Clarendon Press, Oxford, 1979.

[31] I. Prigogine, *La Fin Des Certitudes*, Editions Odile Jacob, 1996.

[32] I. Prigogine and D. Kondepudi, *Thermodynamique, des Moteurs Thermiques aux Structures Dissipatives*, Editions Odile Jacob, 1999.

[33] I. Rechenberg, *Evolution Strategies*, Fromman-Holzboog, 1973.

[34] J. M. Champarnaud and G. Hansel, *Automate, a Computing Package for Automata and Finite Semigroups*, J. Symbolic Comput., 12, 197-220, 1991.

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

[36] J. S. Golan, *Power Algebras Over Semirings With Applications in Mathematics and Computer Science*, Kluwer, 1999.

[37] J. S. Golan, *Semirings and Affine Equations Over them: Theory and Applications*, Kluwer, 2003.

[38] J. Sakarovitch, *Eléments de Théorie des Automates*, Vuibert, 2003.

[39] J. Crutchfield, *Discovering Coherent Structures in Nonlinear Spatial Systems*, Non linear dynamics of ocean waves, A. Brandt and S. Ramberg and M. Shlesinger, 190-216, Singapore, World scientific, 1992.

[40] J.E. Hopcroft and R. Motwani and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 2001.

[41] John Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, 1975.

[42] John H. Holland, *Hidden Order - How Adaptation Builds Coimplexity*, 1995.

[43] John Koza, *Genetic Programming*, Encyclopedia of Computer Sciences and Technology, 1997.

[44] K. Culik and J. Kari, *Finite State Transformations of Images*, Proceedings of ICALP 95, vol 944, Lecture Notes in Comput. Sci., 51-62, Springer, 1995.

[45] K.F. Man, K.S. Tang and S. Kwong, *Genetic Algorithms*, Springer, 1999.

[46] Luaï Jaff, *The Increasing Subsequences and the Shape of Tableaux*, DEA d'Informatique, Labri, University of Bordeaux I.

[47] Luaï Jaff, Gérard H.E.Duchamp, H. Hadj-Kacem and Cyrille Bertelle, *Moderate Growth Time Series for Dynamic Combinatorics Modelisation*, in Proceedings of ICELM-2, pp 42-53, Tirgu-Mures, Romania, May 31- June 3, 2006.

[48] Luai Jaff, Gérard H.E. Duchamp and Cyrille Bertelle *Shift Operators and Complex Systems*, Int. J. of Modeling, Identification and Control, special issue Modelling Complex Systems, 2006.

[49] Luc Favreau, *Combinatoire des Tableaux Oscillants et des Polynômes de Bessel*, Thèse d'Informatique, Labri, University of Bordeaux I, 1991.

[50] Leigh Tesfatsion, *www.econ.iastate.edu/tesfatsi/*

[51] Leigh Tesfatsion, *Agent-Based Computational Economics*, ISU Economics Working Paper No. 1, Economics Department, Iowa State University, July 2002, www.econ.iastate.edu/tesfatsi/acewp1.pdf

[52] L.J. Fogel, A.J. Owens, M.J. Welsh, *Artificial Intelligence Through Simulated Evolution*, John Wiley, 1966.

[53] L. Davis (Ed.), *Genetic Algorithm and Simulated Annealing*, Pitman, 1987.

[54] Michel 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.

[55] M. Lothaire, *Combinatorics on Words*, Cambridge University Press, jan. 2002.

[56] M-P Schutzenberger, Quelques Remarques sur une Construction de Schensted, Math. Scand, 12, 117-128, 1963.

[57] M-P. Schutzenberger,*On the Definitiion of a Family of Automata*, Information and Controm 4, 245-270, 1961.

[58] Memanie Mitchell, *An introduction to Genetic Algorithms*, The MIT Press, 1996.

[59] Mathworld, *http://mathworld.wolfram.com/Semiring.html*

[60] Mathworld, *http://mathworld.wolfram.com/Monoid.html*

[61] Mitpress, *http://mitpress.mit.edu/catalog/item/default.asp*

[62] N. Bourbaki, *Theory of Sets*, Springer 2004.

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

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

[65] O. Brandouy et P. Mathieu, *Marchés Financiers*, Pour la Science, juillet-Septembre 2006.

[66] P.A. MacMahon, *Combinatory Analysis*, Cambridge University Press, 1916.

[67] P. Bak, *How Nature Works - the Science of Self-Organized Criticaly*, Springer Verlag, 1996.

[68] R.P. Stanley, *Enumerative Combinatorics*, Cambridge University Press, Vol. 1, 1997.

[69] R. Nagel, *Reasoning and Learning in Guessing Games*, An experimental investigation. PhD. dissertation, university of Bonn, 1994.

[70] R. Ghnemat, S. Oqeili, C. Bertelle and G.H.E. Duchamp, *Automata-Based Adaptive Behavior for Economic Modelling Using Game Theory*, in Emergent Properties in Natural and Artificial Dynamical Systems by M.A. Aziz-Alaoui and C. Bertelle , pp 173-185, Springer, Understanding Complex Systems series, 2006.

[71] R.A. Horn and C.R. Johnson, *Matrix Analysis*, Cambridge University Press, 1990 (Chapter 8).

[72] R. Ghnemat, C. Bertelle and G.H.E. Duchamp, *Self-Organization Simulation over Geographical Information System Based on Multi-Agent Platform*, The European Simulation and modeling Conference, Octobre 23-25, 2006, Toulouse-France.

[73] R. Axelrod, *The Complexity of Cooperation*, Princeton University Press, 1997.

[74] Rennard, *http://www.rennard.org/alife/english/gavintrgb.html*

[75] S. Eilenberg, *Automata, Languages and Machines*, Academic Press, Vol A, 1974.

[76] S. Eilenberg, *Automata, Languages and Machines*, Academic Press, Vol B, 1974.

[77] S. Eilenberg, *Automata, Languages and Machines*, Vol.A et B, Academic press, 1976.

[78] S. Gaubert, *A few Introductive Texts on (Max, +) Algebra and Discrete Events Systems*, http://Amadeus.inria.fr/.

[79] Santa Fe Institute, *www.santafe.edu/*

[80] Santa Fe, *http://www.santafe.edu/projects/echo/*

[81] The Combinatorial Object Server, *http://theory.cs.uvic.ca/inf/tree/BinaryTrees.html*

[82] T.R.J. Bossolaier and D.G. Green, *Complex Systems*, Camberdge University Press, 2000.

[83] Wikipedia, *http://en.wikipedia.org/wiki/Negative_feedback*

[84] Wikipedia, *http://en.wikipedia.org/wiki/Positive_feedback*

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

[86] W.Brian Arthur, *Increasing Returns and Path-Dependence in the Economy*, University of Michigan Press, Ann Arbor, Mich., 1994.

[87] W.Brian Arthur, *Complexity and the Economy*, Science, 2 April 1999, 284, 107-109.

[88] Wikipedia, *http://en.wikipedia.org/wiki/Sub-system*

[89] W.Brian Arthur, *Increasing Returns and the New World of Business*, Harvard Business Review, July-Aug 1996.

[90] W.Brian Arthur, *Complexity in Economic and Financial Markets*, Complexity, 1, 20-25, 1995

[91] W.Brian Arthur, *Positive Feedbacks in the Economy*, Scientific American, 262, 92-99, Feb. 1990

[92] Wikipedia, *http://en.wikipedia.org/wiki/Financial_market*

[93] Wikipedia, *http://en.wikipedia.org/wiki/Economic_system*