

**A STUDY OF THE APPLICATION OF MULTISET
TO MEMBRANE COMPUTING**

BY

**PETER, CHINEDU MATTHEW B.Sc. (ABU 2004)
M.Sc/SCIEN/10008/2007-2008**

**A THESIS SUBMITTED TO THE POSTGRADUATE
SCHOOL, AHMADU BELLO UNIVERSITY,
ZARIA, NIGERIA**

**IN PARTIAL FULFILMENT FOR THE AWARD
OF MASTER OF SCIENCE IN
MATHEMATICS**

**DEPARTMENT OF MATHEMATICS
AHMADU BELLO UNIVERSITY,
ZARIA, NIGERIA**

MAY, 2011

DECLARATION

I declare that the work in this thesis entitled *A study of the application of multiset to membrane computing* has been performed by me in the Department of Mathematics under the supervision of Prof. D. Singh and Dr. A. M. Ibrahim.

The information derived from the literature has been duly acknowledged in the text and a list of references provided. No part of this thesis was previously presented by me or any other person for another degree or diploma in any other institution.

Peter, Chinedu Matthew

Name of Student

Signature

Date

CERTIFICATION

This thesis entitled “A STUDY OF THE APPLICATION OF MULTISSET TO MEMBRANE COMPUTING” by PETER, Chinedu Matthew meets the regulations governing the award of the degree of Master of Science of the Ahmadu Bello University, Zaria, and is approved for its contribution to knowledge and literary presentation.

Prof. D. Singh
Major Supervisor

Date _____

Dr. A. M. Ibrahim
Minor Supervisor

Date _____

External Examiner

Date _____

Dr. A. A. Tijjani
Head of Department

Date _____

Prof. A. A. Joshua
Dean of Postgraduate School

Date _____

DEDICATION

This work is dedicated to my loving and caring parents, Evang. and Mrs. Peter N. Ogbuehi.

ACKNOWLEDGEMENT

I bless the name of the Lord who gave me life and grace and the enduring spirit to undertake this work.

I sincerely thank my supervisors, Prof. D. Singh of the Department of Mathematics, A.B.U, Zaria, for giving me the discipline, encouragement, direction, advice and some materials which enabled me to start and accomplish this work and Dr. A. M. Ibrahim for his guidance in finalising the work.

I wish to acknowledge the suggestions and comments provided by Prof. Sunday E. Atawodi of Biochemistry Department, A.B.U. Zaria.

I wish to express my thanks and gratitude to the Department of Mathematics, especially the Head of Department, Dr. A. A. Tijjani, and every single member of staff for their suggestions and encouragement throughout my masters program.

My unrelenting appreciation goes to my parents Evang. and Mrs. Peter N. Ogbuehi who supported me financially and otherwise and ensured that I completed this thesis.

Finally, I cannot forget my brothers and sisters, close and distant relations, friends and well wishers who in one way or the other, either directly or indirectly, encouraged me to see to the completion of this work.

ABSTRACT

We present a Venn diagram model, a tree-based model and a multiset-based model for membrane structure and point out some of their limitations. We construct a multiset-based tree model of membrane structure to resolve some of the limitations mentioned earlier. We also construct a saw-like structure to embed into it the multiset-based tree model of membrane structure.

TABLE OF CONTENTS

Title Page	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	i
Declaration	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	ii
Certification	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	iii
Dedication	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	iv
Acknowledgement	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	v
Abstract	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	vi
Table of Contents	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	vii
List of Figures	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xi
List of Tables	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	xii
CHAPTER ONE: GENERAL INTRODUCTION	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	1
1.1 Introduction	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	1
1.2 Statement of the Problem	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	2
1.3 Research Motivation	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	2
1.4 Objective of the Research	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	2
1.5 Organisation of the Thesis	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	3
1.6 Methodology	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	3
CHAPTER TWO: REVIEW OF LITERATURE	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	4
2.1 Cell Biology	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	4
2.2 Some History of the Development of Multisets	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	5
2.3 Some Directions in Developing Membrane Computing	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	9

4.2.2	The Biological Cell Membrane	---	---	---	---	---	---	---	---	---	---	41
4.2.3	Membrane Transportation	---	---	---	---	---	---	---	---	---	---	41
4.3	Membrane Computing	---	---	---	---	---	---	---	---	---	---	45
4.4	Structure of Membrane Computing	---	---	---	---	---	---	---	---	---	---	46
4.5	Basic Variants of P Systems	---	---	---	---	---	---	---	---	---	---	47
4.5.1	Transition P Systems	---	---	---	---	---	---	---	---	---	---	47
4.5.2	P Systems with Active Membranes	---	---	---	---	---	---	---	---	---	---	47
4.5.3	Tissue-like P Systems	---	---	---	---	---	---	---	---	---	---	48
4.5.4	Neural-like P Systems	---	---	---	---	---	---	---	---	---	---	48
4.6	Components of a P System	---	---	---	---	---	---	---	---	---	---	48
4.6.1	The Environment	---	---	---	---	---	---	---	---	---	---	48
4.6.2	Result of Computation	---	---	---	---	---	---	---	---	---	---	49
4.6.3	Membranes	---	---	---	---	---	---	---	---	---	---	49
4.6.4	Catalysts	---	---	---	---	---	---	---	---	---	---	49
4.6.5	Rules	---	---	---	---	---	---	---	---	---	---	50
4.6.6	Computations	---	---	---	---	---	---	---	---	---	---	50
4.7	Aptness for Applying Multisets to Membranes	---	---	---	---	---	---	---	---	---	---	50
4.8	The Structure of Membranes	---	---	---	---	---	---	---	---	---	---	51
4.9	Rules of Evolution	---	---	---	---	---	---	---	---	---	---	52
4.10	A Formal Definition of a Transition P System	---	---	---	---	---	---	---	---	---	---	56
4.11	Defining Computations and Results of Computations	---	---	---	---	---	---	---	---	---	---	58
4.12	Extending the Definition Given in Section 4.10	---	---	---	---	---	---	---	---	---	---	59
4.13	Using Symport and Antiport Rules	---	---	---	---	---	---	---	---	---	---	62

LIST OF FIGURES

Algorithm to Count Tips of Binary Trees -- -- -- -- --	35
Algorithm to Compute McCarthy's 91 Function -- -- -- -- --	36
Algorithm to Compute Ackermann's Function -- -- -- -- --	38
Plant and Animal Cells -- -- -- -- --	42
A Section of a Lipid By-layer -- -- -- -- --	45
A Section of The Cell Membrane -- -- -- -- --	47
A Membrane Structure -- -- -- -- --	53
A Membrane Structure Having the Environment as the Output Region -- -- -- -- --	62
A Binary Tree-- -- -- -- --	77
A Two-element Binary Tree -- -- -- -- --	82
Two Different Binary Trees Generated by the Multiset N -- -- -- -- --	84
A Saw-like Structure Illustrating a Saw-like Permutation -- -- -- -- --	85
A Saw-like Structure Illustrating N_1 -- -- -- -- --	90
A Saw-like Structure Illustrating N_2 -- -- -- -- --	91
A Membrane Structure with Objects and Rules -- -- -- -- --	98
A Tree Representation of the Membrane Structure in Figure 5.5 -- -- -- -- --	99

LIST OF TABLES

Tabular Representation of the Tree in Figure 5.1-- -- -- -- --	79
--	----

CHAPTER ONE

GENERAL INTRODUCTION

1.1 INTRODUCTION

A quasi-generally accepted schematic comprehension of a biological system can be described as a *hierarchical* structure in which deterministic or non-deterministic or stochastic (or random) interactions among its various substructures characterized by a set of basic components take place. It is also presumed that the said interactions do take place cooperatively and competitively leading to an equilibrium (or emergent) or unstable or chaotic state.

Having the aforesaid orientation in mind, a biological system can be viewed as a *multiset object space* that evolves by means of application of *rewriting rules*. Thus, it seems plausible to construct a multiset model to mimic the biological evolution, such as a P system or its variant – a transition P system. Essentially, the interactions between substructures of a bio-system can be mimicked by suitably formulated relations with *bound* multiplicities. In the sequel, the rule based multiset programming paradigm (Krishnamwithy, 2006) has been found of immense importance in the construction of algorithms for Deoxyribonucleic acid (DNA), (more generally, molecular) and membrane computing, augmenting programmable living machines, comprehending evolutionary processes. In the recent years, a good number of researches (Păun 2002, Rogozhin *et al.*, 2004 and Amos 1997) have been undertaken in this direction.

It is difficult to trace the origin of multiset. In the recent years, the notion of multiset has been re-discovered, analyzed and employed in various areas of mathematics, computer science, linguistics and logic.

We will present a brief account of the application of multiset in our literature review. In course of doing this, we will specify the main area of our research in this thesis vis-à-vis: application of multiset to membrane computing.

1.2 STATEMENT OF THE PROBLEM

We propose to study fundamentals of multiset and membrane structure to formulate a multiset-based tree model for membrane computing.

1.3 RESEARCH MOTIVATION

Recently, the subject of membrane computing has become an important area of research. The application of multiset in diverse fields, especially in membrane and molecular computing, motivated our study.

1.4 OBJECTIVE OF THE RESEARCH

The aim of this thesis is as follows:

- i. We propose to present a critical study of the existing multiset models for DNA and membrane computing.
- ii. We propose to study membrane computing specifically by way of providing a multiset-based tree model.
- iii. We also wish to outline constructions of multiset-based biological simulators.

1.5 ORGANISATION OF THE THESIS

The thesis consists of five chapters. In chapter one, a brief explanation of multiset and biological systems, statement of the problem, research motivation, objective of the research and organization of the thesis are presented. In chapter 2, we present literature review. In chapter 3, we present a brief account of fundamentals of multiset with an emphasis on the termination of programs. In chapter 4, we present an overview of various directions the researches have been undertaken in the proposed area. In chapter 5, we design a tree model of membrane structures for membrane computing. We raise some issues that have not been addressed as yet and finally outline construction of multiset-based biological models. The references of all cited works are presented.

1.6 METHODOLOGY

We study the existing models of membrane structure namely: Venn diagrammatic, tree-based, and multiset-based and identify some of their limitations. We construct a multiset based tree model to resolve certain limitations mentioned earlier. We also construct a saw-like structure to embed in it our multiset-based tree model of membrane structure.

CHAPTER TWO

REVIEW OF LITERATURE

In this chapter, we look at some of the various efforts in developing multiset and its application to mathematics, linguistics, statistics, computer science, membrane and molecular (or DNA) computing. Details of the study of some significant applications appear in chapter four; of particular interest being its application to membrane computing.

2.1 CELL BIOLOGY

Throughout time, thoughts and ideas of life have evolved, stretching from biogenesis and spontaneous generation to the modern cell theory. In 1824, Rene Dutrochet discovered that “the cell is the fundamental element in the structure of living bodies, forming both animals and plants through juxtaposition.” However, the first sightings of the internal action of the cell were made by Robert Brown. Schwann created the term “cell theory” and declared that plants consisted of cells. This declaration was made after that of Mathias Schlieden (1804 – 1881) that animals are composed of cells and that living organisms are made up of basic microscopic units called *cells*. All cells fall into one of the two major classifications: *prokaryotes* and *eukaryotes*. The later is more complex and contains *nuclear* materials. Plants and animals are made of cells and that most of the cell organelles are considered identical, yet they do have their differences (Harrison, 1994).

2.2 SOME HISTORY OF THE DEVELOPMENT OF MULTISSETS

The idea of having a repeated element in a set dates back to as far as numbers itself. For example, evidences of representing a number by a collection of tally marks or units are found in the work of the Babylonians in 200 B.C., Egyptians and Greek in 3500 – 1700 B.C. Knuth (1981) notes that enumeration of permutations of a set was known in ancient times and historically the first known document is the Hebrew Book of Creation in 100 A.D., followed by the Indian Classic *Anuyogadvarā-sutra* in 500 A.D.; and the corresponding result for multisets seems to have appeared first in another Indian Classic *Lilāvati* of *Bhāscara Achārya* in 1150. He further notes that Kircher (C. 1650, pp. correctly gave the number of permutations of multiset $\{m.C, n.D\}$ for several values of m and n . A generalization of the rule for enumerating the permutations of multiset appeared in *Prestet's Elémens de mathématiques* (Paris 1675, 351 – 352), and later in John Walli's *Treatise of Algebra 2* (Oxford 1685, pp. 117 – 118). By exploiting Dominique Floata's work done in 1965, Knuth present a good number of significant results on multiset permutations (Singh, 2006).

An early reference to multiset is also found in the work of Marius Nizolius (1498 – 1576). It is not clear whether Nizolius' multitudiness coincides with *class* or *heap*.... But Nizolius' multitudines, might still be heaps in the sense of Quine or Goodman.

Weiestrass defines real numbers as certain msets of rational numbers in which finitely many repetitions are allowed. For example, the quantity $\frac{1}{4} = 3:141:::$ can be identified

with a multiset containing the number 1 with multiplicity 3; the element $\frac{1}{10}$ with multiplicity 1, the element $\frac{1}{100}$ with multiplicity 4 etc. (Blizard, 1991).

The word *multiset* was first suggested by N.G. de Bruijn in private correspondence with Knuth (Knuth, 1981). And is now the accepted name for this concept, replacing *bag*, *heap*, *bunch*, *sample*, *occurrence set*, *weighted set* and *fireset* an acronym for finitely repeated element set. N.G. de Bruijn's interest in multisets grew out of his investigations into the combinatorial properties of the set of divisors of a number. A number or any of its divisors is expressible as a multiset of prime factors, an example can be found in Anderson (1987) and Knuth (1981).

Ars combinatorial is another rich place where msets receive utmost vindication, in theoretical as well as applied directions. Weyl (1949) defines mset as a set with equivalence relation defined on it. Equivalent elements are said to be *in the same state*, *in the same sort*. Weyl (1949) notes: "... electrons may be in this or that position; atoms in a molecule may be N, He, Li ...atoms. ...no artificial difference between elements is introduced by their levels ... and merely the intrinsic differences of states are made use of...". Weyl applies these concepts to a variety of problems in individuals in the same 'complete state' (no further refinement is possible) are indiscernible by any intrinsic characters although they may not be the same thing. Weyl notes: "...one has to distinguish between equal (of the same kind) and identicals". Thus, an aggregate is identified by its distinct states, which is precisely the concept of multiset (Singh, 2006).

Rado (1975) defines an mset to be any cardinal-valued function whose nontrivial domain (the collection of elements not mapped to zero) is a set. The class of msets is called the cardinal module, like a module over the semi-group of all cardinals (p. 135). An mset f represents a family of sets $\mathbb{Q} = \{x_i\}; i \in I$ just in case $f(x_i)$ equals the number of times x_i occurs in $\mathbb{Q}$ (that is, $f(x) = \sum_{i \in I} : x_i = x_j \text{ for all } x$): Rado (1975) also makes use of signed msets (functions which may have negative cardinal values).

Parker-Rhodes observes that "... there exists no branch of mathematics in which a third parity-relation, besides equality and inequality, is admitted". Parker Rhodes develops a theory of *sorts* (collections of indistinguishables), an elaborate mathematical system, and applies it to explicate some fundamental problems of physics. "Copies of elements behave as identicals when they appear as elements of different classes, but as a plurality (each of them contributes to cardinality) when they are elements of the same class" (p. 7). Elements of multisets conform to the Parker-Rhodes principle of indistinguishables. However the system of Parker-Rhodes is a radical departure from classical mathematics because of its tripartitous nature (objects may be identical, distinct or twins). The underlying idea lies in treating equality and identity as different relations. Meyer and MacRobie make use of multisets in the study of relevant implication by way of reckoning how often a premise is repeated in the course of derivations.

Wildberger argues that two physical objects are either different or they are the same (or equal) but separate or they are coinciding (and identical). He takes the example of a water molecule with two hydrogen atoms, say H^1 and H^2 and one oxygen atom say O and

concludes that H^1 and O are obviously different (and, for that matter, H^2 and O are different), however H^1 and H^2 are the same but separate, while H^1 and H^1 are coinciding and identical (Singh, 2006).

Wildberger, by taking some examples from the real world, such as “...to a shopkeeper, any two dollar coins are equal even if their years of minting are different, where as a coin collector would regard them essentially different, etc.”., proposes some very innovative conclusions like “the notion of equality is often a relative one”, “traditional mathematics works with the sense” and “relies heavily on equivalence relation”, etc., which, for the time being, appear to us only a contrived idea.

Since the Pioneer work in Floyd (1967), the use of *wellfounded* sets for proving that programs terminate has been factual. In Dershowitz and Manna (1979), it has been shown that a common tool for proving the termination of programs is the *wellfounded* set: a set ordered in such a way as to admit no infinite descending sequences. The basic approach here is to find a termination function that maps the values of the program variables into some wellfounded set, such that the value of the termination function is continually reduced throughout the computation. The termination functions required are often too difficult to find and an out-of-proportion complexity compared to the program in question. However by providing more sophisticated wellfounded sets, the corresponding termination function can be simplified.

Given a wellfounded set S , we consider multisets over S and define an ordering on all finite multisets over S induced by the given ordering on S . This multiset ordering is shown to be wellfounded. The importance of the multiset ordering is that it permits the use of relatively simple and intuitive termination functions in otherwise difficult termination proofs (Darshowitz and Manna, 1979).

2.3 SOME DIRECTIONS IN DEVELOPING MEMBRANE COMPUTING

Păun (2000) introduces a new computability model, of a distributed parallel type, called P system, based on the notion of a membrane structure. Such a structure consists of several cell-like membranes, recurrently placed inside a unique *skin* membrane. A plane representation is a Venn diagram without intersected sets and with a unique superset. In the regions delimited by the membranes there are placed objects; the obtained construct is called a super-cell. These objects are assumed to evolve: each object can be transformed in other objects, can pass through a membrane, or can dissolve the membrane in which it is placed. A priority relation between evolution rules can be considered. The evolution is done in parallel for all objects able to evolve. In this way, we obtain a computing device and call it a super cell system.

Păun proves that the super-cell systems with the possibility of objects to cooperate characterize the recursively enumerable sets of natural numbers; systems with only two membranes suffice. We use catalysts - specified objects which are present in the evolution rules but are not modified by the rule application. He notes that one catalyst suffices (Păun, 2000).

One of these variants, P systems with symport/antiport, was introduced by Paun *et al.* (2002). These systems have two types of rules: symport rules, when several objects go together from one membrane to another and antiport rules, when several objects from two membranes are exchanged. In spite of a simple definition, we can compute all Turing computable sets of numbers. This result was improved with respect to the number of used membranes and/or the size of symport/antiport rules (Martin-Vide *et al.*, 2002).

Alhazov *et al.* (2005) proves that two classes of Communicative P systems with 3 membranes and with minimal cooperation, namely P systems with symport/antiport rules of size 1 and P systems with symport rules of size 2 are computationally complete: they generate all recursively enumerable sets of vectors of nonnegative integers.

Three Basic types of P systems have been identified in Ibarra *et al.* (2007), namely Cell-like P system, Tissue P system and Neural-like P system. The Cell-like P systems imitate the eukaryotic cells and its basic ingredient is its membrane structure, a hierarchical arrangement of membranes (understood as three dimensional vesicles).

In tissue-like P systems, several one-membrane cells are considered as evolving in a common environment. They contain multisets of objects, while also the environment contains objects. Certain cells can communicate directly (channels are provided between them) and all cells can communicate through the environment. The channels can be given in advance or they can be dynamically established - this latter case appears in so-called population P systems. In the case when the cells are simple, of a limited capacity (as the

number of objects they contain or of rules they can use), we obtain the notion of P colony.

Finally, there are two types of neural-like P systems. One is similar to tissue-like P system in that the cells (neurons) are placed in the nodes of an arbitrary graph and they contain multisets of objects, but they also have a state which controls the evolution. Another promising variant was recently introduced, under the name of spiking neural P systems, where one uses only one type of objects, the spike, and the main information one works with is the distance between consecutive spikes (Ibarra, 2007).

In the first part of their work, Rogozhin *et al.* (2003) expresses ambient calculus in membrane computing. It is an abstract model of the basic features of distribution and mobility of the computing and computation. The central notion of ambient calculus is that of a mobile ambient; which is a bounded place where a computation happens. According to them it was shown that a P system with symbol-objects with membrane dissolution can be expressed in ambient calculus: they show that the Ethernet Network (local electronic computer network) can be expressed in terms of P systems with symbol-objects (Rogozhin *et al.*, 2003).

Arroyo (2002) is where a uniform way of treating objects and rules in P systems is considered: they started with multisets of rules, which are consumed when they are applied, but the application of a rule may also produce rules, to be applied at subsequent steps. They find that this natural and simple feature is surprisingly powerful - systems

with only one membrane can characterize the recursively enumerable languages, both in the case of rewriting and of splicing rules; the same result is obtained in the case of symbol-objects, for the recursively enumerable sets of vectors of natural numbers.

We wish to study some significant applications in chapter three and concentrate our study on the application of multiset to membrane computing which is central in constructing a programming paradigm.

Finally, we elaborate multiset based tree model for membrane computing in Chapter five and examine how multiset based tree structures can help in membrane computing.

CHAPTER THREE

FUNDAMENTALS OF MULTISSET

In this chapter, we intend to explicate the meaning of multiset, show various ways in which a multiset can be represented. We define operations on multiset, submultisets, similar multisets, ordered pair of two multiset terms, power multists, union intersections and sum of multisets, difference and complementation of multisets and function between multisets. We show that Cantor's theorem and Schröder–Bernstein's theorem fail with multisets. Finally, we talk about the Darshowitz–Manna ordering on multisets.

3.1 PRELIMINARY

A *multiset* is a collection of *elements* in which repetition of elements is allowed. A *set* is a multiset in which distinct elements occur only once. The copies of an element in a multiset are called *indistinguishables*. The number of occurrences of an element in a multiset is called its *multiplicity*. The *multiplicity* of an element in a multiset contributes to the *cardinality* of the multiset. That is, the cardinality of a multiset is the sum of the multiplicities of the elements in the multiset. A multiset is *finite* if the distinct elements of the multiset are finite and every element has finite multiplicity. A multiset is therefore *infinite* if the distinct elements of the multiset are infinite or some elements have infinite multiplicities (Blizard, 1991).

The word *multiset* (often shortened to *mset*) which abbreviates the term *multiple-membership set*, is now the commonly accepted name for this concept, replacing *bag*, *bunch*, *weighted set*, *occurrence set*, *heap*, *sample*, and *fireset* – finitely repeated element set.

3.2 REPRESENTATION OF MULTISSETS

3.2.1 Multiplicative form

It is fast becoming a tradition to use square brackets to represent multisets. Thus, an *mset* containing three occurrences of *a*, two occurrences of *b* and one occurrence of *c* is formally written as $[[a; a; a; b; b; c]]$ or $[a; a; a; b; b; c]$ or $[a; b; c]_{3;2;1}$ or $[a^3; b^2; c^1]$ or $[a3; b2; c1]$ as desired.

3.2.2 Linear form

The linear form for a multiset was put forward by Wildberger in which seems quite innovative, especially when negative multiplicities (integral as well as rational) are to be dealt with. For example, the *mset* $M = [a; b; c]_{3;2;1}$ can be written as $M = 3[a] + 2[b] + [c]$: A rational *mset* can similarly be written, for example, as $N = \frac{2}{5}[5] + \frac{1}{2}[8]$: Round brackets are used to accommodate negative multiplicities. Thus (a) stands for negative *a*; for example, $[2; 4; (5); (5); 4] = [2] + 2[4] + 2[5]$: Each individual occurrence of an object *x* in an *mset* *A* is called an *element* of *A*. For example in the *mset* $A = [a; b; c]_{2;4;1}$; *b* is an object appearing four times, and the four occurrences

of b are four elements of A . Thus the distinct elements of an mset are the *objects*. An object is also an element if its multiplicity is 1.

In the following notations are used to represent data structures of sets, ordered sets, multisets and lists. A collection containing $x; y; z; \dots$ is denoted by $\{x\ y\ z; \dots\}$ or $\{x_y_z_ \dots\}$ if it is a set; $\langle x; y; z; \dots \rangle$ if it is an ordered set; $[x\ y\ z\ \dots] = [x_y_z_ \dots]$ if it is a multiset; and $[x; y; z; \dots]$ if it is a list. We note that a list is an ordered sequence of elements with repetitions allowed whereas an mset is a sequence with its ordering excluded.

3.2.3 Multiset as a Sequence

A multiset can also be represented as a sequence in which the multiplicity of an element equals the number of times the element occurs in the sequence. The idea is to construct an mset as a sequence (a function with domain $\mathbf{N}$, the set of natural numbers) and ignore the ordering of its elements, which can be done by taking all permutations of the domain of the sequence.

3.2.4 Multiset as a Family of Sets

A multiset can also be represented as a family of sets, which is altogether a generalization of the idea of a sequence described above. Thus, the family of sets $F = \{F_i; i \in \mathbf{I}\}$; where $F_i = F_j$ if $i = j$; which identifies a repeated element, represents an mset. Clearly,

such a family F is a function: $I \rightarrow \{F_i | i \in I\}$; which in turn is a sequence if $I = \mathbb{N}$. See Singh (2006) for details of the aforesaid definitions.

3.2.5 Multiset as a generalised characteristic function

Similar to the representation of a set by its characteristic function (function whose range is $\{0, 1\}$), a multiset or hybrid set is determined by its generalized characteristic function (whose range is the set of integers, positive, negative or zero), (Singh *et al.*, 2002).

3.3 DEFINITION OF TERMS

Definition 3.1: *Multiset*

Let $D = \{x_1, x_2, \dots, x_j, \dots\}$ be a set. An mset A over D is a cardinal-valued function, i.e. $A : D \rightarrow \mathbb{N} = \{0, 1, 2, \dots\}$ such that $x \in \text{Dom}(A)$ implies $A(x)$ is a cardinal and $A(x) = m_A(x) > 0$; where $m_A(x)$ denotes the number of times an object x occurs in A , i.e. a counting function of A . The set D is called the ground or generic set of the class of all msets containing objects from D .

An mset A can also be represented by the set of pairs as follows:

$$A = \{ \langle m_A(x_1), x_1 \rangle, \dots, \langle m_A(x_j), x_j \rangle, \dots \}$$
 or

$$A = \{ \langle m_A(x_1), x_1 \rangle, \dots, \langle m_A(x_j), x_j \rangle, \dots \}$$

Relatively, an mset is called *regular* or *constant* if all its objects occur with the same multiplicity. Also, an mset is called *simple* if all its objects are the same, for example, $[x]_3$ is a simple mset containing x as its only object. Clearly, the root set of every simple mset contains a single object.

Definition 3.2: Multiset as a Numeric-valued function

Representation of an mset as a numeric-valued or cardinal-valued function can be found, especially in the application areas. Formally, an mset is just a mapping from some ground or generic or universal set into some set of numbers. For example, an mset $A = [x; y; z]_{1;2;3}$ is a mapping from a ground set S to $\mathbb{N}$, the set of natural numbers with zero, defined by

$$\mathbb{R}(t) = \begin{cases} 1; & \text{if } t = x \\ 2; & \text{if } t = y \\ 3; & \text{if } t = z \\ 0; & \text{for all the remaining } t \in S: \end{cases}$$

In general terms, for a given ground set S and a numeric set T , we call a mapping

$$\mathbb{R}: S \rightarrow T;$$

- a set; if $T = \{0; 1\}$;
- a multiset; if $T = \mathbb{N}$; the set of natural numbers;
- a signed multiset (hybrid or shadow set); if $T = \mathbb{Z}$; the set of integers;
- a fuzzy (or hazy) set if $T = [0; 1] \subset \mathbb{R}$; a two-valued Boolean algebra:

Definition 3.3: Dressed epsilon (ϵ_+)

We use the dressed epsilon symbol ϵ_+ as defined in Singh (2006). For any object x occurring as an element of an mset A , i.e. $m_A(x) > 0$, we write $x \epsilon_+ A$, where ϵ_+ (dressed epsilon) is a binary predicate intended to mean ‘belongs to at least once’, as ϵ is ‘belongs to only once’ in the case of sets. Thus, $m_A(x) = 0$ implies $x \not\epsilon_+ A$, and $x \epsilon_+^k A$ implies ‘ x belongs to A at least k times’, however $x \epsilon_+^k A$ means ‘ x belongs k times to A ’. The mset for any ground set D is called empty, denoted by $\emptyset$ or $\mathbf{0}$, if $m_A(x) = 0$ for all $x \in D$.

Definition 3.4: Multisubsets (or msubsets, for short)

Let A and B be two msets, A being an *msubset* or a submultiset of B , written as $A \mu B$ or $B \mu A$, if $m_A(x) \leq m_B(x)$ for all $x \in D$. Also, if $A \mu B$ and $A \neq B$, then A is called a proper subset of B . An mset is called the parent in relation to its msubsets. It follows from the definition of multisets that $A = B$ if and only if for all $x \in D$, $m_A(x) = m_B(x)$. Also, $A = B \Rightarrow A^n = B^n$, but the converse needs not hold.

It is easy to see that μ is not symmetric, i.e. $A \mu B$ and $B \mu A \Rightarrow A = B$; and it is a partial ordering on the class of msets defined on a given generic domain. Clearly, $\emptyset$ is a submultiset of every mset. Note that the term ‘element’ and ‘object’ are being distinguished throughout, and coincide if a generic set is in consideration. We wish to emphasize that the introduction of ϵ_+ greatly enhances the language of msets. For example, $A \mu B$ stands for $\forall x (x \epsilon_+^k A \Rightarrow x \epsilon_+^k B)$:

Relatively, a *whole* msubset of a given mset contains all multiplicities of common elements; while a *full* msubset contains all objects of the parent mset, and accordingly, every mset contains a unique full msubset, its root set. Clearly, for any two msets A and B, if $A \mu B$ and $\text{Dom}(A) = \text{Dom}(B)$; then A is a full msubset of B.

Definition 3.5: Similar multisets

Two msets A and B are said to be *similar* or *cognate* if $\exists x(x \in A \wedge x \in B)$; where x is an object. Thus, similar msets have equal root set but need not be equal themselves.

Definition 3.6: Ordered pair of two multiset terms

Ordered pair of two mset terms u and v, denoted by $[u; v]$; can be defined as follows:
 $[u; v] = f u; v g$ if $u \neq v$; and $[u; v] = f [u]_2 g$ if $u = v$. Here, $[u]$ is written as $f u g$, and $h u; v i$ is actually the ordered pair set, where $\text{Set}(u)$ stands for

$$u = \{ \langle x, n \rangle \mid x \in u, n \in \mathbb{N} \};$$

though x itself may be an mset term.

Definition 3.7: Power multiset

In Cantorian spirit, the power multiset of a given mset A, denoted by $\tilde{\mathcal{P}}(A)$ to distinguish it from the symbol $\mathcal{P}(A)$ used for power set of A, is the multiset of all submultisets of A.

For example, let $A = [x; y]_{2;1} = [x; x; y]$: Then, $\tilde{\mathcal{P}}(A)$

$= [; ; f x g; f x g; [x]_2; f y g; f x; y g; f x; y g; [x; y]_{2;1}]$: In this sense, $C(\tilde{\mathcal{P}}(A)) = 2^{C(A)}$; for any

mset A. However, as has been voiced by many researchers in the area of msets and their

applications (Blizard, 1989), there is not ‘good’ reason for admitting repeated elements into a power multiset. Hence, a power multiset needs to be called a power set only and denoted by $\mathcal{P}(A)$. Accordingly, for $A = [x; y]_{2;1}$;

$$\mathcal{P}(A) = [\emptyset; \{x\}; \{y\}; \{x, y\}]$$

and hence, $C(\mathcal{P}(A)) < 2^{C(A)}$;

which implies that Cantor’s power set theorem: $C(A) < C(\mathcal{P}(A))$ fails.

Definition 3.8: *Union* ($\cup$)

Let A and B be two msets over a given domain set D . $A \cup B$ is the mset defined by $m_{A \cup B}(x) = m_A(x) \cup m_B(x) = \max(m_A(x); m_B(x))$, being the union of two numbers. That is, an object z occurring a times in A and b times in B , occurs $\max(a; b)$ times in $A \cup B$, if such a maximum exists; otherwise the minimum of $(a; b)$ is taken which always exists.

It follows that for any given mset x , there exists an mset y which contains elements of elements of x , where the multiplicity of an element z in y is the maximum multiplicity of z as an element of element of x along with the above stipulation of the existence of such a maximum. We denote this fact by $y = \cup x$.

Clearly, $\text{Dom}(\cup x) = \bigcup \{ \text{Dom}(A) : A \in x \}$ and that the multiplicity of z in y is the maximum of its multiplicities as an element of elements of x , if it exists; otherwise, the minimum is taken. For example, if $A = [2; 3; 4; 4]$; $B = [1; 4; 3; 3]$ then $A \cup B = [1; 2; 3; 3; 4; 4]$.

Also, it follows that for a finite mset x , the maximum multiplicity of elements of elements of x always exists. However, for certain infinite sets like $x = f f y g; [y]_2; [y]_3; :::g$; the maximum multiplicity of elements of elements of x does not exist, and hence $[x = f y g$: It is obvious by definition of the union that multiplicity of any $y \in x$ is irrelevant to $[x$, and hence $[x = [x^a$.

Definition 3.9: Intersection ($\cap$)

Let A and B be two msets over a given domain set D . $A \cap B$ is the mset defined by $m_{A \cap B}(x) = m_A(x) \cap m_B(x) = \min(m_A(x); m_B(x))$ being the intersection of two numbers. That is, an object x occurring a times in A and b times in B , occurs minimum $(a; b)$ times in $A \cap B$, which always exists.

In general, for a given mset x , $\text{Dom}(\cap x) = \{f \text{ Dom}(A) : A \in x\}$ and $z \in \cap x$ implies that the multiplicity of z is the minimum of its multiplicities as element of elements of x . For example, if $A = [3; 3; 3; 4; 4]$; and $b = [1; 4; 3; 3]$; then $A \cap B = [3; 3; 4]$: Note that for any mset x , we have $\cap x \in x$.

Definition 3.10: Addition or sum or merge ($\cup$) or (+)

Let A and B be two msets over a given domain set D . $A \cup B$ is the mset defined by $m_{A \cup B}(x) = m_A(x) + m_B(x)$; direct sum of two numbers.

That is, an object x occurring a times in A and b times in B , occurs $a + b$ times in $A \cup B$:

For example, if

$A = [1; 1; 2; 2; 4; 4; 4]; B = [1; 2; 3; 3];$ then $A \sqcup B = [1; 1; 1; 2; 2; 2; 3; 3; 4; 4; 4]$: Clearly, $C(A \sqcup B) = C(A \sqcap B) + C(A \setminus B)$. We note that if we let x be an infinite mset, then the multiplicity of some mset $z \in x$ may not be finite. In that case, the multiplicity of z in x is used. For example if $x = f f z g; [z]_2; [z]_3; \dots; g$ then $x \sqcup x = f z g$:

Definition 3.11: *Difference*

Let A and B be msets over a domain set D such that $B \sqsubseteq A$. Then $m_{A \sqcap B}(x) = m_A(x) \dot{-} m_{A \setminus B}(x)$; for all $x \in D$. It is sometimes called the arithmetic difference of B from A . We Note that even if B is not contained in A , this definition holds good. It can be seen quickly that some of the consequences of the aforesaid definition are disturbing. For example, if $A = [a; b]_{4;5}; B = [a; b]_{2;3}$ then $A \sqcap B = [a; b]_{2;2} \sqsubseteq B$; contradicting the classical law: $(A \sqcap B) \setminus B = \emptyset$:

Definition 3.12: *Complementation*

In order to define the complement, we follow Petrovsky:

Let $\mathcal{A} = \{A_1; A_2; \dots\}$ be a family of multisets composed of the elements of the generic set D : Then, the maximum multiset Z is defined by $\max_{A \in \mathcal{A}} m_A(x)$ for all $x \in D$ and all $A \in \mathcal{A}$:

Now, the complement of an mset A , denoted by $\hat{A}$, is defined as follows:
 $\hat{A} = Z \sqcap A = \{m_{\hat{A}}(x) \mid m_{\hat{A}}(x) = m_Z(x) \dot{-} m_A(x); \text{ for all } x \in D\}$: It is understood that some new operations like arithmetic multiplication, raising to the arithmetic power,

direct product, raising to the direct power, defined by Petrovsky, can be gainfully exploited for further research.

Definition 3.13: *Function between multisets*

The underlying assumption in defining a function between msets has been invariably not to allow mapping identical elements to non-identical elements and hence, it amounts to defining the function between their root sets, which is just the classical definition of a function.

The function $f : A \rightarrow B$ is an injection iff

- (i) $f : A \rightarrow B$ is an injection, and
- (ii) $\forall z (z \in A \Rightarrow m_A(z) = m_B(f(z)))$:

The function $f : A \rightarrow B$ is a surjection iff

- (i) $f : A \rightarrow B$ is a surjection, and
- (ii) $\forall z (z \in A \Rightarrow m_A(z) \leq m_B(f(z)))$:

The function $f : A \rightarrow B$ is a bijection iff

- (i) $f : A \rightarrow B$ is a bijection, and
- (ii) $\forall z (z \in A \Rightarrow m_A(z) = m_B(f(z)))$:

For example, $f : [x]_3 \rightarrow [x]_{10}$ is an injection, $f : [x]_5 \rightarrow [x]_4$ is a surjection and $f : [x; y]_{6,2} \rightarrow [x; y]_{3,4}$ is neither an injection nor a surjection. We note that some of the

consequences of the above definitions are conflicting with fundamental theorems of the classical set theory.

- Having defined functions between msets as above, it can be proved that Cantor's theorem does not hold, there is no injection from $A \rightarrow \mathcal{P}(A)$.
- Msets of equal cardinality need not have a bijection between them. For example, $[a; b]_{1;2}$ and $[a; b; c]$ both contain three elements, but there can be no bijection between them because the objects and their multiplicities are different in the domain and the range of any such function. In other words there is no bijection between their root sets, viz $f : \{a; b\} \rightarrow \{a; b; c\}$ cannot be a bijection.
- Schröder–Bernstein theorem: Let $A = [x_1; x_2; \dots]_{2;4;6;\dots}$ and $B = [y_0; y_1; y_2; \dots]_{1;3;5;\dots}$. The function $f : A \rightarrow B$ defined as $f(x_n) = y_n$ makes $f : A \rightarrow B$ an injection so that $A \preceq B$. The function $g : B \rightarrow A$ defined by $g(y_n) = x_{n+1}$ makes $g : B \rightarrow A$ an injection so that $B \preceq A$. But there cannot be a bijection $h : A \rightarrow B$ since all multiplicities in A are even and those in B are odd. We note that $\preceq$ is the standard dominance relation.

See Blizard (1989), Singh (2006) and Singh *et al.* (2007) for details of the aforesaid definitions.

Definition 3.14: Partial Ordering

A binary relation $\mid$ on a set X is called a *partial order* on X if $\mid$ satisfies the following axioms:

1. $x \mid x$ for all $x \in X$. (Reflexiveness)
2. $x \mid y$ and $y \mid x \Rightarrow x = y$ for all $x, y \in X$. (Anti symmetry)
3. $x \mid y$ and $y \mid z \Rightarrow x \mid z$ for all $x, y, z \in X$. (Transitivity)

The set X is said to be *partially ordered* with respect to $\mid$. We note that for some pair of elements x, y in X , neither $x \mid y$ nor $y \mid x$ may hold. In the case where $x \mid y$ or $y \mid x$ for all x, y in X then X is said to be *totally ordered* (Dornhoff *et al.*, 1978)).

Definition 3.15: Multiset Ordering

According to Knuth: “A multiset α_1 dominates a multiset α_2 if both α_1 and α_2 contain the same number of elements and the K^{th} largest element of α_1 is greater than or equal to the K^{th} largest element of α_2 for all K .”, (p. 214).

If a and b are multisets of m numbers each, we say that $a \leq b$ iff $a \wedge b = a$ (equivalently, $a \leq b$ iff the largest element of a is less than or equal to the smallest of b).

Thus $a \wedge b \leq a \leq b$ (p. 241).

However, Darshowitz and Manna (1979) is the earliest reference known to introducing multiset ordering and using it for proving termination of programs and terms rewriting system. It has served as a basis for host of orderings introduced in this thesis. The following is an outline of the Dershowitz-Manna multiset ordering:

Let S be a set equipped with a partial ordering $<$ (irreflexive and transitive relation or, equivalently, a transitive but not an equivalence relation). Let $M(S)$ be the set of all finite msets M on S , and let $\leq$ be the mset ordering on $M(S)$ induced by $<$. It is easy to see that each M is an mset with a finite carrier, viz: $\{x \in S : M(x) \neq 0\}$:

Definition 3.16: Darshowitz-Manna Ordering

$M \leq N$ if there exist two msets X and Y in $M(S)$ satisfying:

- (i) $\emptyset \subseteq X \subseteq N$;
- (ii) $M = (N - X) + Y$;
- (iii) $(\forall y \in Y)(\exists x \in X)[y < x]$:

In other words, $M \leq N$ if M is obtained from N by removing none or at least one element (those in X) from N , and replacing each such element x by zero or any finite number of elements (those in Y), each of which is strictly less than (in the ordering $<$) one of the elements x that have been removed. Informally, we say that M is smaller than N in this case. Similarly, A on $M(S)$ with $(S; >)$ can be defined. For example, let $S = \{f, 0, 1, 2, \dots, g, =, @\}$; then under the corresponding multiset ordering A over $\mathcal{C}$, the mset $[3; 3; 4; 0]$ is greater than each of the following msets: $[3; 4]$; $[3; 2; 2; 1; 1; 1; 4; 0]$ and $[3; 3; 3; 3; 2; 2]$: The empty set $\emptyset$ is smaller than any multiset. It is also easy to observe that: $(\forall y \in N \Rightarrow \exists x \in M \wedge x > y) \Rightarrow M \dot{A} N$ (Darshowitz and Manna, 1979).

3.4 PROPERTIES OF MULTISSET OPERATIONS

(i) Commutativity:

$$A \cup B = B \cup A;$$

$$A \cap B = B \cap A;$$

$$A \setminus B = B \setminus A:$$

(ii) Associativity:

$$A \cap (B \cap C) = (A \cap B) \cap C;$$

$$A \setminus (B \setminus C) = (A \setminus B) \setminus C;$$

$$A \cup (B \cup C) = (A \cup B) \cup C;$$

(iii) Idempotency:

$$A \cap A = A;$$

$$A \setminus A = A;$$

$$A \cup A = A:$$

In order to obtain a linear combination of msets, kA may be interpreted to denote the sum of k number of A 's, where k is a natural number.

(i) Identity laws:

$$A \cap \emptyset = A;$$

$$A \setminus \emptyset = A;$$

$$A \cup \emptyset = A:$$

(ii) Distributivity:

$$A \sqcup (B \sqcap C) = (A \sqcup B) \sqcap (A \sqcup C);$$

$$A \sqcup (B \setminus C) = (A \sqcup B) \setminus (A \sqcup C);$$

$$A \setminus (B \sqcap C) = (A \setminus B) \sqcap (A \setminus C);$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

The proofs of all these identities follow from the interpretation of $\sqcup$, $\setminus$ and $\sqcap$ of two natural numbers as maximum, minimum and (direct) sum respectively. It is easy to see that $\sqcup$ is stronger than both $\sqcap$ and $\setminus$ in the sense that neither $\sqcap$ nor $\setminus$ distributes over $\sqcup$, whereas $\sqcup$ distributes over both $\sqcap$ and $\setminus$. Also, $\setminus \times \mu \sqcup \times \mu \sqcup \times$ (Singh, 2006).

3.5 MULTISSET AND THE TERMINATION OF PROGRAMS

The notion behind the involvement of multiset in terminating computer programs cannot be over emphasized. Computer programs run as long as control (similar to the concentration of the computer) is still within the program codes. Normally control leaves the program when the end of the program or a normal exit from the program is reached. In this case there is a normal termination of program. A situation whereby control will never leave a program is usually regarded as a bad programming practice. This is mostly encountered in a loop where control keeps running within the loop and can go on forever. Certain mechanisms are put in place by researchers in order to terminate such programs. One of the recent tools used is the *wellfounded set*.

3.5.1 A Basis for Program Termination

Let S be a set and let $>$ be an ordering defined on S . The pair $(S; >)$ is called wellfounded if $>$ is irreflexive and transitive in such a way that there can be no infinite descending sequence of elements.

The idea is to find a wellfounded set and a termination function that maps the values of the program variables into that set such that the values of the termination function is continually reduced throughout the iteration. Since by the nature of the set, the value cannot decrease indefinitely, the program must terminate.

The wellfounded sets most frequently used for this purpose are the natural numbers under the *greater than* ordering. In practice using these conventional ordering often leads to complex termination functions that are often too difficult to discover. For example the termination proofs of programs involving stacks and production systems are often quite complicated and require much more subtle orderings and termination functions. Finding an appropriate ordering and termination function for such programs is a big challenge. The multiset ordering is a powerful tool that can sometimes make task of proving termination easier (Darshowitz and Manna (1979), for details).

3.5.2 The Multiset Ordering

Given that $(S; >)$ is a partially ordered set, Let $M(S)$ be the set of all finite multisets with elements taken from the set S . Define a partial order A on $M(S)$ as follows:

Let $M; M^0 \in M(S)$: Then $M \dot{\leq} M^0$ if for some multisets $X; Y \in M(S)$; with $\forall x \in X, \exists y \in Y$; we have $M^0 = (M \setminus X) \cup Y$ and $(\forall y \in Y)(\exists x \in X) x > y$: In other words, a multiset is reduced by the removal of at least one element (those in X) and their replacement with any finite number (possibly zero) of elements (those in Y), each of which is smaller than one of the elements that have been removed.

For example, let $S = \mathbb{N}$; the set of natural numbers with the usual ordering $>$, then under the corresponding multiset ordering $\dot{\leq}$ over $\mathbb{N}$, the multiset $\{3; 3; 4; 0\}$ is greater than each of the three multisets $\{3; 4\}$; $\{3; 2; 2; 1; 1; 1; 4; 0\}$ and $\{3; 3; 3; 3; 2; 2\}$:

The multiset ordering is in fact a partial ordering and if $>$ is irreflexive and transitive so also is $\dot{\leq}$ (Darshowitz and Manna, 1979).

Theorem 3.1

The multiset ordering $(M(S); \dot{\leq})$ over $(S; >)$ is wellfounded if and only if $(S; >)$ is wellfounded. Other details and proof of this theorem can be found in Darshowitz and Manna (1979).

3.5.3 The Nested Multiset Ordering

We next consider nested multisets. These are multisets whose elements may be elements of some base sets S or may be multisets containing both elements of S and multisets of elements of S , etc.

More formally let $(S; >)$ be a partially ordered set. Let $M^*(S)$ denote the set of nested multisets over S , and let $\dot{A}^*$ be a nested multiset ordering on $M^*(S)$:

Let $M; M^0 \in M^*(S)$; then $M \dot{A}^* M^0$ if

- $M; M^0 \in S$ and $M > M^0$,

(Two elements of the base set are compared using $>$), or else

- $M \not\in S$ and $M^0 \in S$,

(any multiset is greater than any element of the base set), or else

- $M; M^0 \not\in S$; and for some $X; Y \in M^*(S)$; with $; \in X \subseteq M; M^0 = (M \cap X) \cup Y$ and $(\forall y \in Y)(\exists x \in X) x \dot{A}^* y$:

For example, the nested multiset $\{f\{1; 1g; f\{f\{0g; 1; 2g; 0g\}$ is greater than $\{f\{1; 0; 0g; 5; f\{f\{0g; 1; 2g; 0g\}$; since $f\{1; 1g$ is greater than both $f\{1; 0; 0g$ and 5 . The same nested multiset $\{f\{1; 1g; f\{f\{0g; 1; 2g; 0g\}$ is greater than $\{f\{f\{f\{g; 1; 2g; f\{5; 5; 2g; 5g\}$; since $f\{f\{0g; 1; 2g$ is greater than each of the three elements $f\{f\{g; 1; 2g; f\{5; 5; 2g$ and 5 .

Definition 3.17: The depth of an element of a nested multiset

Let $x_1; x_2; \dots; x_{n-1}$ be elements of some nested multisets and let $x_n = x$ be a multiset such that $x_1 \in x_2 \in \dots \in x_n$. We define the *depth* of x_i as $n - i + 1$ for all $i = 1; 2; \dots; n - 1$.

Denote with $M^i(S)$ the set of all nested multisets of depth i . In other words $M^0(S) = S$; and $M^{i+1}(S)$ contains the multisets whose elements are taken from $M^0(S); M^1(S); M^2(S); \dots; M^i(S)$; with at least one element taken from $M^i(S)$. Thus, the set $M^*(S)$ is the infinite union of the disjoint sets $M^0(S); M^1(S); M^2(S); \dots$. Thus for any two

nested multisets, M and M^c , if the depth of M is greater than the depth of M^c , then $M \dot{\prec} M^c$. In other words, the multisets of $M^i(S)$ are all greater than the multisets of $M^j(S)$ under the ordering $\dot{\prec}$, for any $i > j$. The relation $\dot{\prec}$ is a partial ordering; it can be shown to be both irreflexive and transitive (see details in Darshowitz and Manna (1979)).

Theorem 3.2

The nested multiset ordering $(M^*(S); \dot{\prec})$ over $(S; >)$ is wellfounded, if and only if $(S; >)$ is wellfounded.

This theorem was proved by induction in Darshowitz and Manna (1979). An alternative proof was given in Singh and Singh (2006).

3.5.4 Program Termination with Multiset Ordering

We shall use multiset orderings as the wellfounded sets to demonstrate the termination of certain programs.

3.5.5 Counting tips of binary tree

Consider a simple program to count the number of tips –terminal nodes (without descendents) – in a full binary tree. Each tree y that is not a tip has two subtrees, $\text{left}(y)$ and $\text{right}(y)$: The program is

```

S := (t)
c := 0
loop until S = ()
  y := head (S)
  if tip (y) then
    S := tail (S)
    c := c + 1
  else
    S := left (y):right (y):tail (S)
repeat

```

Figure 3.1: Algorithm to count tips of binary trees

It employs a stack S and terminates when S is empty. At that point, the variable c is 0. The termination of this program may be proved using the wellfounded set $(N; >)$. The appropriate termination function is

$$\zeta(S) = \sum_{s \in S} \text{nodes}(s);$$

where $\text{nodes}(s)$ is the total number of nodes in the sub-tree s - not just the tip nodes. Using the multiset ordering over trees, we can prove termination with the simple termination function

$$\zeta(S) = \sum_{s \in S} |s|;$$

Given the multiset of trees appearing in the stack, the trees themselves are ordered by the natural wellfounded sub-tree ordering, ie. Any tree is greater than its sub-tree. Thus, the removal of a tree from the stack decreases ζ in the multiset ordering by removing an element, and the replacement of a tree with two smaller sub-trees decreases ζ (see Darshowitz and Manna (1979), for details).

3.5.6 McCarthy's 91 function

Consider the following program to compute the simple function

$$f(x) = \text{if } x > 100 \text{ then } x - 10 \text{ else } 91$$

over the set of integers $\mathbb{N}$, in a round-about manner. The program is

```

n := 1
z := x
loop L : assert  $f(x) = f^n(z); n \geq 1$ 
    if  $z > 100$  then
         $n := n - 1$ 
         $z := z - 10$ 
    else
         $n := n + 1$ 
         $z := z + 11$ 
    until  $n = 0$ 
repeat
assert  $z = f(x)$ 

```

Figure 3.2: Algorithm to compute McCarthy's 91 function

The predicates $f(x) = f^n(z)$ and $n \geq 1$ are loop invariants. The loop is exited if control reaches the until clause with $n = 0$; and $f(x) = f^0(z) = z$:

Consider the following wellfounded partial-ordering A on the integers:

$$a \dot{\leq} b \text{ if and only if } a < b < 111:$$

We use the set $(M(Z); \dot{\Delta})$ of all multisets of integers, under the corresponding multiset ordering. The appropriate termination function $\dot{\iota}$ at L yields a multiset in $M(Z)$; and is defined as

$$\dot{\iota}(n; z) = f z; f(z); \dots; f^{n_i-1}(z)g;$$

We must show that for each loop iteration this function decreases. There are three cases:

1. $z > 100$ at L : In this case, the then branch of the conditional is executed and both n and z are decremented. When control returns to L (assuming that the loop has not been exited), we have, in terms of the old values of n and z ,
 $\dot{\iota}(n_i - 1; z_i - 10) = f z_i - 10; f(z_i - 10); \dots; f^{n_i-2}(z_i - 10)g =$
 $f f(z); f^2(z); \dots; f^{n_i-1}(z)g$: Thus, the value of the termination function τ has been decreased by removing the element z from the original multiset $f z; f(z); \dots; f^{(n_i-1)}(z)g$:

2. $90 \leq z \leq 100$ at L : In this case, the else branch is taken and both n and z are incremented, yielding

$$\dot{\iota}(n + 1; z + 11) = f(z + 11); f(z + 11); f^2(z + 11); \dots; f^n(z + 11)g;$$

Either $z + 1 = 101$ or else $z + 1 \leq 100$; in both cases
 $f^2(z + 11) = f(z + 1) = 91 = f(z)$: Thus, we get

$$\dot{\iota}(n + 1; z + 11) = f z + 11; z + 1; f(z); \dots; f^{n_i-1}(z)g: \text{ Since}$$

$z < z + 1 < z + 11 \leq 111$; we have $z \dot{\Delta} z + 11$ and $z \dot{\Delta} z + 1$: Accordingly, the multiset has been reduced by replacing the element z with the two smaller elements, $z + 11$ and $z + 1$:

3. $z < 90$ at L : The else branch is taken and we have
 $\dot{\iota}(n + 1; z + 11) = f z + 11; f(z + 11); f^2(z + 11); \dots; f^n(z + 11)g =$

$f\ z + 11; 91; f(z); ::; f^{(n_i - 1)}(z)g$: Again z has been replaced by two smaller elements (under the A relation), $z + 11$ and 91 : (see Darshowitz and Manna (1979), for details).

3.5.7 Ackermann's function

The following iterative program computes Ackermann's function $a(m; n)$ over pairs of natural numbers:

```

S := (m)
z := n
loop L : assert a(m; n) = a(sk; a(sk-1; ::; a(s2; a(s1; z)); ::))
    y := head(S)
    S := tail(S)
    if y = 0 then
        z := z + 1
    else
        if z = 0 then
            S := (y | 1):S
            z := 1
        else
            S := y:(y | 1):S
            z := z | 1
        -
    =
until S = ()
repeat
assert z = a(m; n)

```

Figure 3.3: Algorithm to compute Ackermann's function

where the stack S has k elements $s_1; s_2; ::; s_k$:

To prove termination, consider the set $N \times N$ of lexicographically ordered pairs of natural numbers and use the corresponding multiset ordering over $N \times N$. Let $y = \text{head}(S) = s_1$. The termination function at L is

$$\zeta(S; z) = f(s_k + 1; 0); (s_{k-1} + 1; 0); \dots; (s_2 + 1; 0); (y; z)g.$$

Thus $\zeta(S; z)$ yields a multiset containing one pair per element in the stack S . We note here that at L , the stack S is nonempty, and all the elements of S as well as z are non negative.

The proof considers three cases, corresponding to the three branches of conditional in the loop:

1. $y = 0$: If the loop is not exited, then the new value of ζ at L is

$$\zeta((s_2; \dots; s_k); z + 1) = f(s_k + 1; 0); \dots; (s_2 + 1; 0); (s_2; z + 1)g.$$

2. $y \neq 0$ and $z = 0$: In this case we obtain

$$\zeta((y-1; s_2; \dots; s_k); 1) = f(s_k + 1; 0); \dots; (s_2 + 1; 0); (y-1; 1)g.$$

Thus, the element $(y; z)$ has been replaced by the smaller element

$$(y-1; 1):$$

3. $y \neq 0$; and $z \neq 0$: Here we have

$$\zeta((y; y-1; s_2; \dots; s_k); z-1) = f(s_k + 1; 0); \dots; (s_2 + 1; 0); (y; 0); (y; z-1)g.$$

The element $(y; z)$ has been replaced by the two smaller elements $(y; 0)$ and $(y; z-1)$, (Darshowitz and Manna 1979).

CHAPTER FOUR

APPLICATION OF MULTISSET TO MEMBRANE COMPUTING

In this chapter, we highlight some biological concepts as relating to cells and membrane structures, we demonstrate how multiset theory can be applied to membrane computing in particular and molecular (DNA computing) in general. Of interest is the representation of the evolving objects in a biological cell as a string of multiset of symbol objects.

4.1 INTRODUCTION

As reflected in the general introduction part of this thesis, as of last three decades or so, the mathematics of multisets has found applications in mathematics and computer science, besides logic, linguistics, physics, biology, economics, etc. Recently, a good number of its applications have been witnessed in DNA (or molecular) and membrane computing.

A more recent branch of natural computing with an enthusiastic beginning and computational yet, unconfirmed applicability is DNA computing, whose birth is related to Adleman's experiment of solving a small instance of the Hamiltonian path problem by handling DNA molecules in a laboratory. DNA structure and processing suggest a series of new data structures and operations that the massive parallelism made possible by the efficiency of DNA as a support of information promises to be useful in solving computationally hard problems in a feasible time. Theoretically speaking, DNA computing started in 1987 when T. Head had proposed a language theoretical model of

what he called the splicing operation (the recombination of DNA molecules, cut by restriction enzymes in fragments pasted together when the sticky ends match). The study of membrane computing (Păun, 2002) has been considered as an extension of DNA computing.

4.2 MEMBRANE STRUCTURE: SOME CONCEPTUAL REFLECTIONS

4.2.1 The Cell

A cell is a complete functional biological unit with many different internal structures. Cells are the building blocks of living organisms. They do not all look the same unlike the bricks in a house. They are specialised to carry out various sorts of distinct jobs. For example, the *brain cells* and *muscle cells*, look quite different and do quite different jobs. Cells are microscopic in size and are connected by membranes. There are cells living alone such as ciliates, bacteria, etc., called *protozoa*. Some organisms are *single-celled* whose family is called *prokaryotes*. *Algae* are not too many cells with organs and yet not one cell. The number of cells determines the complexity of the living organism.

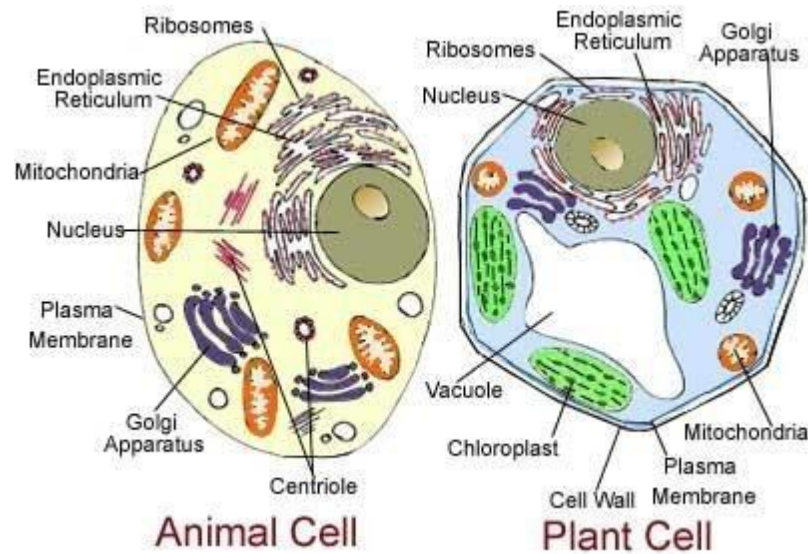


Figure 4.1: Plant and Animal Cells

The following is the hierarchy in which the cells are organized: organelles → cells → tissues → organs → organisms → community of organisms. Every cell has sub units called organelles. Some organelles include the nucleus, Golgi apparatus, mitochondria, etc.

The organelles are free flowing within the cells. Depending on the need, the nucleus can be close to the endoplasmic reticulum. They are already packed together without any interconnectedness except that there is fluid in between them. Organelles communicate but are not interconnected.

The *nucleus* is like the brain of the cell because it controls all the activities within the cell, including the *intra* and *inter-cellular* communications. The nucleus contains the

genetic materials in a cell, including the DNA, in most cells. Cell membranes are in physical contact amongst one another.

4.2.2 The Biological Cell Membrane

The whole cell is covered by a membrane and every organelle is covered by membranes itself. Two cells are connected through their cell membranes. A *membrane structure* is a hierarchically arranged system of membranes. It is made up of a *lipid bi-layer*. Each layer is followed by another and, in between, there is an outburst of proteins which helps in transportation. The *outermost* membrane in a membrane system is called the *skin* or *plasma* membrane. Within the skin membrane, a plant cell has a rigid wall, called a *cell wall*, which determines its shape.

A membrane without any other membrane in it is called an *elementary* membrane. Invariably, in a *biological* membrane system, an elementary membrane is said not to be fully developed. They are at greater risk compared to others in the membrane structure.

The space delimited from above by a membrane and below by another membrane placed inside it is called the *region* of the membrane. There is a one-to-one correspondence between membranes and regions. Thus, membranes and regions are used interchangeably.

4.2.3 Membrane Transportation

Molecules transported from one part of the cell to another are found neither aggressed nor interfering too much. There are two subdivisions of membrane transport: *active* and

passive. Active transport makes use of energy (ATP) and also some other molecules are required in the process. The *mitochondrion*, also known as the *powerhouse* of the cell, is where energy generating activities take place. The nuclear membrane is the one surrounding the nucleus. On the other hand, in a passive transport, no energy is required. *Diffusion* is an example of a passive transport. Diffusion is the passage of molecules from a region of greater concentration to one of lower concentration (Norland, 1983).

Membrane translocation is an injection of particles into a membrane. This can be in the form of a simple selection by size, in the case of small molecules or a much more intricate cycle - a protein assisted transport system.

The *compartments* are the same as organelles. When compartments are too large for the local biochemical activities to take place efficiently, life creates membranes. In Brownian motion, molecules react by moving randomly. Thus space only could be sufficient to allow for a frequent collision of particles.

An *environment* is the space between cells. The compartment of a cell contains substances (*micro molecules*). Micro molecules are *amino acids*, *carbohydrates*, *lipids*, *proteins* etc. Amino acids are *ionic* forms of the metals and non metals, e.g. *calcium*, etc. There is no ordering within a membrane since everything is in the form of a fluid moving at random motion. All reactions take place in a massive parallel way. That is, all reactions take place simultaneously. The reactions are non deterministic, meaning that no reaction determines another reaction.

Membranes in cells typically define enclosed spaces or compartments in which cells may maintain a chemical or biochemical environment that differs externally. Basically, the membrane made up of lipid by layers with the hydrophobic parts of the lipid bi-layer turned inside (see the diagram below) is a selectively permeable structure.

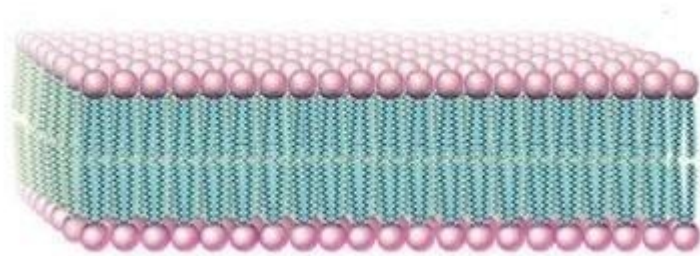


Figure 4.2: A section of a lipid bi-layer

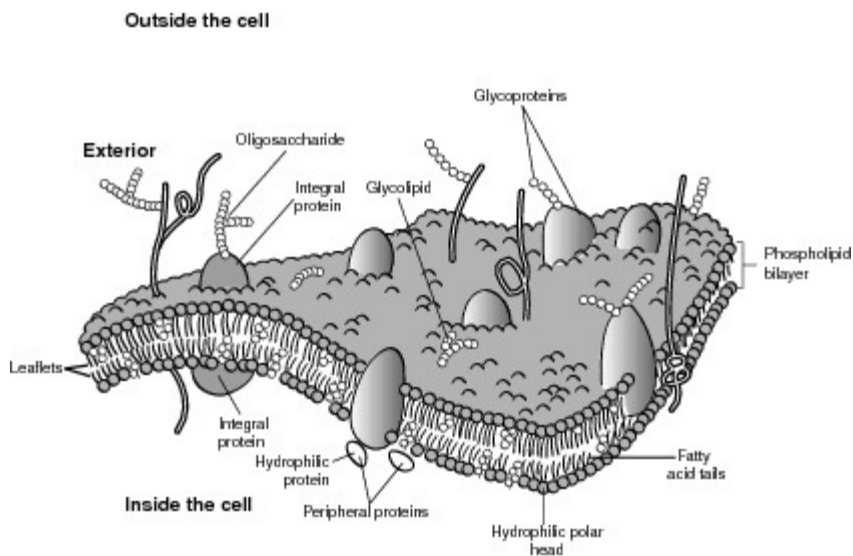
Membranes have many different functions within a typical cell probably the most valuable one being the partitioning of the cell into functional and segregated compartments. Because of the incredible number and often conflicting biochemical reactions occurring in a cell at the same time, the cell must retain order via structural organisation or risk chemical chaos. The internal membranes compartmentalize reactions to prevent interferences. The cell membrane also separates life from the nonlife on its exterior. In this process, an intact and healthy membrane is selectively permeable because it allows substances needed for cell prosperity to enter and attempt to prohibit the penetration of unwanted and unfriendly substances. This means that the size, charge, and other chemical properties of the atoms and molecules attempting to cross it will determine whether they succeed in doing so.

Selective permeability is essential for effective separation of a cell or an organelle from its surroundings. Probably, this is the most important feature of a biological membrane. Unfortunately the system is not always foolproof. Sometimes unwanted substances pass through the membrane and may cause trouble within the cell. This bi-layer phenomenon is also the foundation for the widely upheld *fluid mosaic model* of membrane structure. The phospholipid's molecules have a water-soluble polar *head* and two fat-soluble, non-polar tails. The hydrophobic tail always tries to avoid water and faces the inside of the bi-layer, whereas the hydrophilic head faces the exterior and the interior. Within the phospholipids' bi-layer, there are many different types of embedded proteins and cholesterol molecules whose presence spawned the term *mosaic*.

From scanning electron microscope images, it was observed that the embedded molecules can move sideways throughout the membrane, which means that the membrane is not solid, rather more like a fluid. The membranes also have glycoprotein attached to their surfaces, which help in identifying and controlling food, water, waste and other membrane traffic. Each cell has a particular glycogen structure based on its need to attract or repel membrane traffic.

The proteins embedded in the membranes serve many of the membrane functions, such as holding the membranes in a regular and identifiable structure for easy bonding. They also have a specific and unique shape that allows them to function as receptors and receptor sites for attachment to appropriate raw materials needed for cellular functions. In some cases, the receptor protein is also a signal transducer that begins a series of enzyme-

catalyzed reactions to stimulate a particular reaction or function within a cell. Finally, *the transport proteins*, also called *carrier proteins* help substances move across membranes.-



See Harrison (1994) and Gjorgjieva (2007), for further details.

Figure 4.3: A section of the cell membrane

4.3 MEMBRANE COMPUTING

Membrane computing (also called P system) aims at abstracting computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures. It is concerned with distributed and parallel computing models, processing multisets of symbol-objects in a localized

manner, with an essential role played by the communication among the compartments and with the environment.

A P system is a computational model in the field of computer science that performs calculations using a biologically inspired process. It is based upon the structure of biological cells, abstracting from the way in which chemicals interact first introduced by Păun (1998). Actually, variations on the P system model gave rise to the formation of membrane computing.

4.4 STRUCTURE OF MEMBRANE COMPUTING

As mentioned earlier, membrane computing can be seen as an extension of DNA computing, from the *one processor* level to a distributed computing model. The cell is the main object of research in membrane computing models. The cell is separated from its environment by a membrane, (the external one). Several other membranes enclose *protected reactors*, compartments where specific biochemical processes take place. In particular a membrane encloses the nucleus (of eukaryotic cells), where the genetic material is placed. Through vesicles enclosed by membranes one can transport packages of molecules from a part of the cell to other parts of the cell in such a way that the transported molecules are not *aggressed* during their journey by neighbouring chemicals. Then the membranes allow a selective passage of substances among the compartments delimited by them. This can be a simple selection by size, in the case of small molecules, or a much intricate selection, through protein channels, which not only selects but can also move molecules from a low concentration to a higher concentration, perhaps coupling molecules, through so-called symport and antiport processes.

The membranes of a cell do not only delimit compartments where specific reactions take place in solution. Hence, inside the compartments, but many reactions in a cell develop on the membranes, catalyzed by the proteins bounded on them. It is said that where a compartment is too large for the local biochemistry to be efficient, life creates membranes, both in order to create reactors small enough where any two of the enclosed molecules can collide frequently enough, and to create smaller *reaction surfaces* (Păun, 2002).

4.5 BASIC VARIANTS OF P SYSTEMS

Various variants of P Systems exist. Below are some of them (Păun, 2002).

4.5.1 Transition P Systems

This is a P System based on multiset rewriting rules. A sequence of transition starting from the initial configuration is called a computation.

4.5.2 P Systems with Active Membranes

This is a class of P System, which, together with the basic transition systems and the symport/antiport systems is one of the three central types of cell-like P Systems considered in membrane computing. They start from the observation that membranes play an important role in the reactions which take place in a cell, and, moreover, they can evolve themselves, either changing their characteristics or even getting divided.

4.5.3 Tissue-Like P Systems

We pass now to consider a very important generalization of the membrane structure, passing from the cell-like structure, described by a tree to a tissue-like structure, with the membrane placed in the nodes of an arbitrary graph. In the basic variants of tissue-like P systems, this graph is a virtually total one. What matters is the communication graph, dynamically defined during the computation.

4.5.4 Neural-Like P Systems

This is a bigger step in enlarging the model of tissue-like P systems by considering more complex cells like moving the states from the channels between cells themselves – preserving the network of synapses.

4.6 COMPONENTS OF A P SYSTEM

Several varieties of P system exist, largely sharing the same basic components as listed in the following subsections (see Wikipedia on multiset).

4.6.1 The Environment

The environment is the surroundings of the P system. At the initial state of the P system, it contains only the container-membrane, and while the environment cannot hold rules, it may have objects passed into it during the computation.

4.6.2 Result of Computation

The objects found within the environment at the end of the computation constitute its result. Computation takes place in stages called *transitions*. The computations can be denoted by subscripted letters such as $C_0; C_1; \dots; C_m$ in which case we have $m + 1$ number of transitions and m configurations.

4.6.3 Membranes

Membranes are the main structures within a P system. A membrane is a discrete unit which can contain a set of objects, a set of rules and a set of other membranes contained within. The outermost membrane held within the environment, is often regarded as the container membrane, or skin membrane. Membranes are permeable and symbols (that is objects after computation) resulting from the application of rules may cross them. A membrane (but not the container membrane) may also dissolve, in which case its content, except for rules (which are lost), migrate into the membrane in which it was contained. Some P systems possess a charge, have selective permeability or have varying thickness. A membrane is called elementary if it contains no other membranes inside it.

4.6.4 Catalysts

Catalysts are requirements for a reaction to occur and are never used up. They are represented and used in the same way as symbols. They help speed up reaction processes.

4.6.5 Rules

Rules represent a possible chemical reaction within a membrane, causing it to evolve to a new state. A rule has a required set of input objects upon which the rules are applied.

4.6.6 Computations

A computation works from an initial starting state towards an end state through a number of discrete steps. Each step involves iterating through all membranes in the P system and the application of rules, which occurs in both a maximally parallel and non-determinate manner. A computation stops when no further evolution can take place (i.e., when no rules are able to be applied). At this point, whatever objects passed to the environment are counted as the result of the computation.

4.7 APTNESS FOR APPLYING MULTISSETS TO MEMBRANES

The compartments of a cell contains substances namely ion, molecules, macromolecules, etc, all swimming in an aqueous solution, there is no ordering there, everything is close to everything, the concentration matters, the population and the number of copies of each molecules. This suggests working with sets of objects whose multiplicities matter, and hence, multisets would be more apt than the crisp sets.

As mentioned in chapter three, multisets can be represented in many ways but we shall adopt the multiplicative convention. For example, if each of the objects **a**, **b** and **c** are present in 4, 3 and 7 copies respectively, then we can represent this multiset by the string

$a^4b^3c^7$: We also note that all permutations of this string represent the same multiset. This is because permutations of elements do not change the multiset containing such elements (Păun, 2002).

4.8 THE STRUCTURE OF MEMBRANES

The membrane structure is one of the most fundamental ingredients of a P system. The following is a simple generalized structure of the membrane.

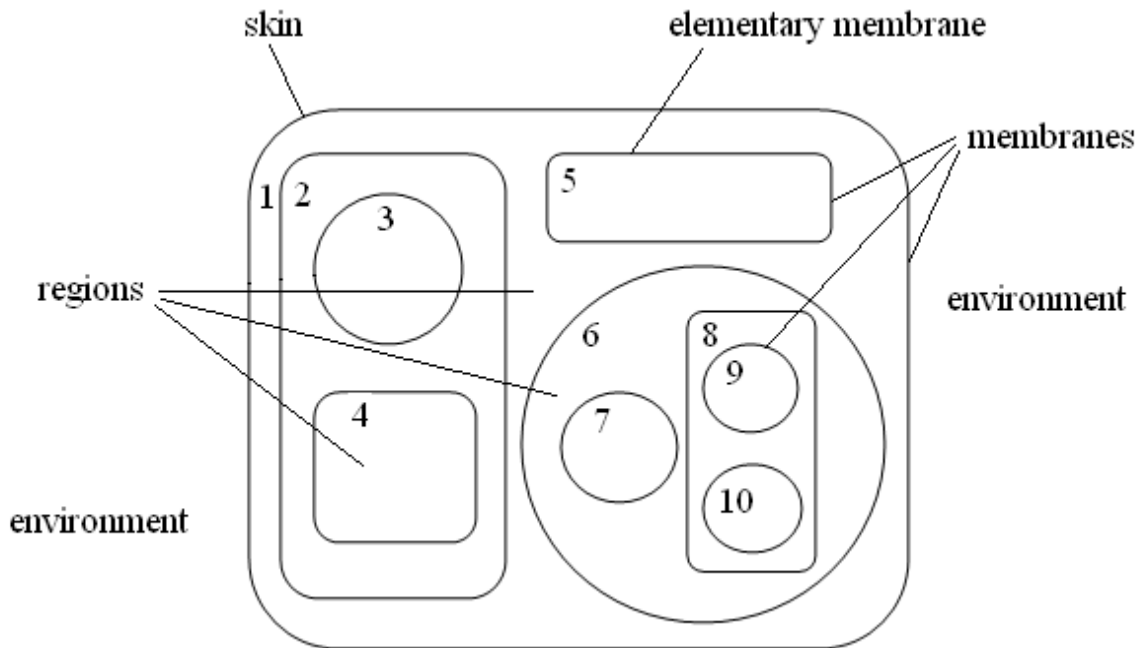


Figure 4.4: A membrane structure

As in Figure 4.4 above, the membrane structure is a hierarchically arranged set of membranes, contained in a distinguished external membrane called the *skin* membrane. It corresponds to the biological plasma membrane. The skin membrane can contain several

other membranes. A membrane is called elementary if it contains no other membrane inside it. Each membrane determines a region or compartment. The space is delimited from above by it and from below by the membranes directly inside it, if any exists.

As in Figure 4.4, the membranes are usually identified by labels from a given set of labels. We use numbers, starting with the number 1 assigned to the skin membrane. The labels are assigned in a one-to-one manner to membranes, but this is possible only in the case of membrane structures which cannot grow indefinitely, otherwise several membranes should have the same label (Păun, 2002).

4.9 RULES OF EVOLUTION

Since the processes to be computed do have biochemical background, where standard chemical reactions are common, it seems prudent to process the multiset from the compartments of our computing device by means of rewriting-like rules. These are rules of the form $u \rightarrow v$; where u and v are multisets of objects. For example, an application of the rule $aab \rightarrow abcc$ says that two copies of the object a and one copy of the object b react to form one copy of the object a , leaves the object b intact and two copies of a new object c : The object b acts as a catalyst in this case. Thus, on applying this rule one time to the multiset $a^5b^2c^6$; we obtain $a^4b^2c^8$; because aab are used up and $abcc$ are produced. Similarly, by applying the rule $bb \rightarrow aac$ to the same multiset $a^5b^2c^6$; we will get the multiset a^7c^7 ; which contains no occurrence of the object b .

There are three main types of rules used in membrane computing: Multiset rewriting rules (otherwise known as evolution rules), communication rules and rules for handling membranes.

In the basic variant of P systems, each region contains a multiset of symbol-objects which correspond to the chemicals swimming in a solution in a cell compartment. Since these chemicals are considered as unstructured, they can be described by symbols from a given alphabet (Păun, 2002).

Evolution rules correspond to the chemical reactions possible in the compartments of a cell. Hence, they are of the form $u \rightarrow v$ where u and v are multisets of objects. However in order to make compartments interact with one another we have to move objects across membranes and to this aim we add target indications to the objects produced by rules. These indications are **here**, **in** and **out**; meaning that an object having associated with the indication **here** remains in the same region, one having the indication **in** goes into an immediate lower membrane, (non-deterministically chosen), and an **out** indicates that the object has to exit the membrane, thus becoming an element of the region surrounding it. For example, the rule $aab! \rightarrow acc$ translates to $aab! \rightarrow (a; \text{here})(b; \text{out})(c; \text{here})(c; \text{in})$: After using this rule in a given region of a membrane structure, two copies of **a** and one copy of **b** are consumed (removed) and one copy of **a**; one of **b**; and two of **c** are produced: the resulting copy of **a** remains in the same region, and the same happens with one copy of **c** (indications **here**), while the new copies of **c** enters one of the child membranes, non-deterministically chosen. If no such child membrane exists, that is, the

membrane with which the rule is associated is elementary, then the indication *in* cannot be followed, and the rule cannot be applied. In turn, if the rule is applied in the skin region, then *b* will exit into the environment of the system (and it is lost there). The indication *here* is not specified (an object without an explicit target indication is supposed to remain in the same region where the rule is applied). This is an example of *cooperative rule* with at least two objects in the left hand side. Particular are the cases of catalytic rules of the form $cu ! \quad cv$ where *c* is an object called a catalyst and it enables the object *u* to evolve into the multiset *v* and of the form $a ! \quad v$; where *a* is an object. Rules of the form $a ! \quad v$; where *a* is an object are called *non-cooperative*.

The rules can also have the form $u ! \quad v\pm$, where $\pm$ denotes the action of membrane dissolving. If this rule is applied, then the corresponding membrane disappears and its contents, objects and membranes alike, are left free in the surrounding membrane. The rules of the dissolved membrane disappear at the same time with the membrane. The skin membrane is never dissolved.

The communication of objects through membranes reminds the fact that the biological membranes contain various channels through which the molecules can pass, in a rather selective manner. Moreover, the fact that the communication of objects from a compartment to a neighbouring compartment is controlled by the *reaction rules* indicates its mathematical formulation is attractive, but not quite realistic from a biological point of view. That is why some variants more developed where the two processes got treated separately: the evolution is controlled by rules as above without target indications, and

the communication is controlled by specific rules (symport/antiport rules). It is also worth noting that evolution rules are stated in terms of names of objects, while their application/execution is done using copies of objects. It can be observed in the example considered above, where the multiset $a^5b^2c^6$ was processed by a rule of the form $aab! \ a(b; out)c(c; in)$; that in the maximally parallel manner, it is used twice for the two possible sub-multisets aab . The key phrase here is: *applying the rules in the maximally parallel manner, non-deterministically choosing the rules and the objects*. More specifically this means that we assign objects to rules, non-deterministically choosing the objects and the rules, until no further assignment is possible. More succinctly, we get to the set of rules and try to find a suitable multiset of rules by assigning multiplicities to rules with two properties:

- (i) The multiset of rules is applicable to the multiset of objects available in the respective region. That is, there are enough objects in order to apply the rules a number of times as indicated by their multiplicities.
- (ii) The multiset is maximal. That is, no further rule can be added to it (because of lack of available objects).

Thus, an evolution step in a given region consists in finding a maximal applicable multiset of rules, removing from the region all objects specified in the left hand of the chosen rules (with the multiplicities as indicated by the rules and by the number of times each rule is used), producing the objects from the right hand side of rules, and then

distributing these objects as indicated by the targets associated with them. If at least one of the rules introduces the dissolving action $\pm$; then the membrane is dissolved and its contents become part of the immediate upper membrane, provided this membrane was not dissolved at the same time; a case where we stop in the first upper membrane which was not dissolved (at least the skin remains intact), (Păun, 2002).

4.10 A FORMAL DEFINITION OF A TRANSITION P SYSTEM

Systems based on multiset-rewriting rules as above are usually called transition P systems, and we will continue to preserve this terminology in this specific content, although *transitions* are present in all types of systems.

Indeed, while presenting a P system one needs to specify: the alphabet of objects (a usual finite non-empty alphabet of abstract symbols identifying the objects), the membrane structure (it can be represented in many ways, but the most used one is by a string of labelled matching parentheses), the multisets of objects present in each region of the system (represented in the most compact way as strings of objects), the sets of evolution rules associated with each region along with the indication about the way the output is defined.

Formally, a transition P system (of degree m) is a construct of the form

$$\Pi = (O; C; {}^1; w_1; w_2; \dots; w_m; R_1; R_2; \dots; R_m; i_0);$$

where:

1. O is the (finite and nonempty) alphabet of objects,
2. $C \subseteq O$ is the set of catalysts,
3. σ is a membrane structure, consisting of m membranes, labelled with $1; 2; \dots; m$;
4. $w_1; w_2; \dots; w_m$ are strings over O representing the multisets of objects present in the regions $1; 2; \dots; m$ of the membrane structure,
5. $R_1; R_2; \dots; R_m$ are finite sets of evolution rules associated with the regions $1; 2; \dots; m$ of the membrane structure,
6. i_0 is either one of the labels $1; 2; \dots; m$; and then the respective region is the output region of the system; or it is $\emptyset$, and then the result of a computation is collected in the environment of the system.

Let $S^?$ denote the power set of the set S of all objects over an alphabet; Let S^+ denote the set $S^? \setminus \{\epsilon\}$. Each element of $S^?$ forms a string of objects; objects which are from S . Here ϵ is the empty string. The rules are of the form $u \rightarrow v$ or $u \rightarrow v\pm$ with $u \in O^+$ and $v \in (O \cup Tar)^?$; where $Tar = \{here; in; out\}$. The rules can be cooperative (with u arbitrary), non-cooperative (with $u \in O \setminus C$), or catalytic (of the form $ca \rightarrow cv$ or $ca \rightarrow cv\pm$ with $a \in O \setminus C; c \in C; v \in ((O \setminus C) \cup Tar)^?$): The catalysts never evolve and never change the region; they only help the other objects to evolve.

A possible reaction about the region i_0 in the case when it is an internal one is to consider only regions enclosed by elementary membranes for output that is, i_0 should be the label of an elementary membrane of γ :

In general, the membrane structure and the multisets of objects from its compartments identify a configuration of a P system. The initial configuration is given by specifying the membrane structure and the multisets of objects available in its compartments at the beginning of computation, viz; $(\gamma; w_1; w_2; \dots; w_m)$: Note that during the evolution of the system, by means of applying the rules, both the multisets of objects and the membrane structure can change (see Păun (2002), for details).

4.11 DEFINING COMPUTATIONS AND RESULTS OF COMPUTATIONS

The transformation of the configuration of the membrane system is called transition. This is done by applying the rules in each region in a non-deterministic and maximally parallel manner. This means that the objects to evolve and the rules governing this evolution are chosen in a non-deterministic way and this choice is *exhaustive* in the sense that, after the choice has been made, no rule can be applied at the same evolution step to the remaining objects.

A sequence of transitions constitutes a computation. A computation is successful if it halts. That is, it reaches a configuration where no rule can be applied to the existing objects and the output region i_0 still exists in the halting configuration. With a successful

computation we can associate a result in various ways. If we have an output region specified and this is an internal region, then we have an internal output. We count the objects present in the output region in the halting configuration and this number is the result of the computation. When we have $i_0 = 0$, we count the objects which leave the system during the computation and this is called external output. In both cases the result is a number. If we distinguish among different objects, then we can have as the result a vector of natural numbers. The objects which leave the system can also be arranged in a sequence according to the moments when they exit the skin membrane and, in this case the result is a string. Non-halting computations provide no output. That is we cannot know when a number is completely computed before halting. Also if the output membrane is dissolved during the computation, then the computation aborts and no result is obtained (this makes sense only in the case of the internal output), (see Păun (2002), for details).

4.12 EXTENDING THE DEFINITION GIVEN IN SECTION 4.10

A possible extension of the definition 4.10 is to consider a terminal set of objects, $T \subseteq O$; and to count only the copies of objects from T ; discarding the objects from $O \setminus T$ present in the output region. It does enhance the construction and *programming* a P system, because we can ignore some auxiliary objects (e.g., the catalysts).

Because of the non-determinism of the application of rules, starting from an initial configuration, we can get several successful computations, hence several results. Thus, a P system computes (one also uses to say generates) a set of numbers. The case when we

get a language is important in view of the qualitative difference between the *loose* data structure we use inside the system (vectors or numbers) and the data structure of the result, string, here we also have a *syntax*, a positional information.

For a given system $\mathcal{P}_i$ we denote by $N(\mathcal{P}_i)$ the set of numbers computed by $\mathcal{P}_i$ in the above way. When we consider the vector multiplicities of objects from the output region, we write $P_s(\mathcal{P}_i)$: In turn, in the case when we take as (external) output the strings of objects leaving the system, then we denote the language of these strings by $L(\mathcal{P}_i)$:

We now illustrate the previous definitions by examining the computations of the system – with the output region being the environment.

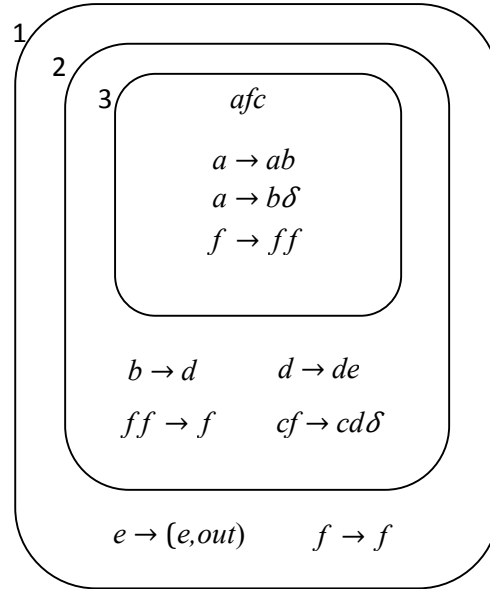


Figure 4.5: A Membrane Structure having the environment as the output region

We have objects only in the central membrane, that with label 3; hence only here we can apply rules. Specifically, we can repeatedly apply the rule $a! \ ab$ in parallel with $f! \ ff$; and in this way the number of copies of b grows each step by one, while the number of copies of f is doubled in each step. If we do not apply the rule $a! \ b\pm$ (again in parallel with $f! \ ff$); which dissolves the membrane, then we can continue in this way forever. Thus, in order to ever halt, we have to dissolve membrane 3. Assume that this happens after $n, 0$ steps of using the rules $a! \ ab$ and $f! \ ff$: When membrane 3 is dissolved, its contents ($n + 1$ copies of b ; 2^{n+1} copies of f , and one copy of the catalyst c) are left free in membrane 2, which now can start using its rules. In the next step, all objects b become d : Let us examine the rules $ff! \ f$ and $cf! \ ad\pm$. The second rule dissolves membrane 2, hence passes the contents of this membrane to membrane 1. If among the objects which arrive in membrane 1 there is at least one copy of f , then the rule $f! \ f$ from region 1 can be used forever and the computation never stops. Moreover, if the rule $ff! \ f$ is used at least once in parallel with the rule $cf! \ cd\pm$, then at least one copy of f is present. Therefore, the rule $cf! \ cd\pm$ should be used only if region 2 contains only one copy of f . This means that the rule $ff! \ f$ was used always for all pairs of f available, that is, in each step the number of copies of f is divided by w . This is already done once in the step when all copies of b become d , and will be done from now on as long as at least two copies of f are present. Simultaneously, in each step each d produces one copy of e : This process can continue until we get a configuration with only one copy of f present; at that step we have to use the rule $cf! \ cd\pm$ (the rule $ff! \ f$ is no longer applicable), hence also membrane 2 is dissolved. Because we have applied the rule $d! \ de$, in parallel for all copies of d (there are $n + 1$ such copies),

during $n + 1$ steps, we have $(n + 1)(n + 1)$ copies of e , $n + 2$ copies of d (one of them was produced by the rule $cf \rightarrow cd$), and one copy of c present in the skin membrane of the system (the unique membrane still present). The objects e are sent out, and the computation halts. Therefore, we compute in this way the number $(n + 1)^2$; for some $n \geq 0$, that is, $N(f) = f n^2$ (see Păun (2002), for details).

4.13 USING SYMPORT AND ANTI-PORT RULES

The multiset rewriting rules correspond to reactions taking place in the cell, inside the compartments. However, an important part of the cell activity is related to the passage of substances through membranes, and one of the most interesting ways to handle this trans-membrane communication is by coupling molecules. The process by which two molecules pass together across a membrane is called *symport*; when the two molecules pass simultaneously through a protein channel, but in opposite directions, the process is called *antiport*.

The operations can be formalized in obvious ways: $(ab; in)$ or $(ab; out)$ are symport rules, stating that a and b pass together through a membrane, entering in the former case and exiting in the latter; similarly, $(a; out; b; in)$ is an antiport rule, stating that a exits and at the same time b enters the membrane. Separately, neither a nor b can cross a membrane – unless we have a rule of the form $(a; in)$ or $(a; out)$; called, for uniformity, *uniport* rule.

We can generalize these types of rules, by considering symport rules of the form $(x; in); (x; out)$; and antiport rules of the form $(z; out; w; in)$; where x, y , and w are

multisets of arbitrary size; one uses to say that $|x|$ is the weight of a symport rule as above, and $\max(|z|; |w|)$ is the weight of the antiport rule.

Such rules can be used in a P system instead of the target indications **here**; **in**; **out** : we consider mutiset rewriting rules of the form $u \rightarrow v$ (or $u \rightarrow v \pm$) without target indications associated with the objects from v ; as well as symport/antiport rules for communicating the objects among compartments.

Formally, a P system with Symport and Antiport rules is a construct of the form

$$\Pi = (O; \sigma; w_1; \dots; w_m; E; R_1; \dots; R_m; i_0)$$

Where:

1. O is the alphabet of objects,
2. σ is the membrane structure (of degree $m \geq 1$; with the membranes labelled in a one-to-one manner with $1; 2; \dots; m$),
3. $w_1; \dots; w_m$ are strings over O representing the multisets of objects present in the m compartments of σ in the initial configuration of the system,
4. $E \subseteq O$ is the set of objects supposed to appear in the environment in arbitrarily many copies,
5. $R_1; \dots; R_m$ are the (finite) sets of rules associated with the m membranes of σ ;
6. $i_0 \in H$ is the label of a membrane of σ ; which indicates the output region of the system.

The rules from $\mathbf{R}$ can be of two types, symport rules and antiport rules, of the form as specified above. The rules are used in the non-deterministic maximally parallel manner. In the usual way, we define transitions, computations, and halting computations.

The number (or the vector of multiplicities) of objects present in region i_0 in the halting configuration is said to be computed by the system along that computation; the set of all numbers (resp., vectors or numbers) computed in this way by $\mathcal{P}$ is denoted by $N(\mathcal{P})$ (resp., by $P_s(\mathcal{P})$).

We remark here a new component of the system, the set E of objects which are present in the environment in arbitrarily many copies; because we only move objects across membranes and because we start with finite multisets of objects present in the system, we cannot increase the number of objects necessary for the computation if we do not provide a supply of objects, and this can be done by considering the set E . Because the environment is supposed inexhaustible, the objects from E are supposed inexhaustible, irrespective of how many of them are brought into the system, arbitrarily many remain outside.

Another new feature is that this time the rules are associated with the membranes, not with the regions, and this is related to the fact that each rule governs the communication through a specific membrane.

The P systems with symport/antiport rules have a set of attractive characteristics: they are fully based on biological types of multiset processing rules; the environment takes a direct part into the evolution of the system; the computation is done only by communication, no object is changed, the objects only move across membranes; no object is created or destroyed, hence the conservation law is observed, (Păun, 2002).

4.14 SUPER-CELLS

Let σ be a membrane structure of degree $n; n \geq 1$; with the membranes labelled in a one-to-one manner. The regions of σ are also similarly identified. If a multiset $M_i : U \rightarrow \mathbb{N}$ is associated with each region i of $\sigma; 1 \leq i \leq n$; we say that we have a *super-cell*.

A super cell can be described by a Venn diagram where both the membranes and the objects are represented.

4.15 TRANSITION SUPER-CELL SYSTEMS

A transition super-cell system of degree $n; n \geq 1$; is a construct

$$\sigma = (V; \sigma; M_1; \dots; M_n; (R_1; \sigma_1); \dots; (R_n; \sigma_n); i_0);$$

where:

- (i) V is an alphabet whose elements are called objects
- (ii) σ is a membrane structure of degree n , with the membranes and regions labelled in a one-to-one manner with elements in a given set Σ :
- (iii) $M_i; 1 \leq i \leq n$; are multisets over V associated with the regions $1; 2; \dots; n$ of σ :

- (iv) $R_i; 1 \leq i \leq n$; are finite sets of evolution rules over V associated with the regions $1; 2; \dots; n$ of σ ; $\leq$ is a partial order relation on R_i : An evolution rule is a pair $(u; v)$ which we will usually write in the form $u \rightarrow v$; where u is a string over V and $v = v^c$ or $v = v^0 \pm$; where v^c is a string over $(V \setminus \{here; out\})$ [$(V \setminus \{in_j; 1 \leq j \leq n\})$]; and $\pm$ is a special symbol not in V : The length of u is called the radius of the rule $u \rightarrow v$:
- (v) i_0 is a number between 1 and n which specifies the output membrane of σ .

Graphically, a super cell can be represented by representing its underlying super cell, and also adding the rules to each region, together with the corresponding priority relation.

4.16 COMPUTATION AND RESULT OF COMPUTATION IN A TRANSITION SUPER-CELL SYSTEM

Consider a rule $u \rightarrow v$ in a set R_{it} : We look to the region of σ^c associated with i_t : If the objects mentioned by u ; with the multiplicities specified by u ; appear in M_{it}^0 ; then these objects can evolve according to the rule $u \rightarrow v$: The rule can be used only if no rule of a higher priority exists in R_{it} and can use the objects in u : A rule can be used only when there are copies of the objects whose evolution it describes and which were not *consumed* by rules of a higher priority. All objects in u are consumed by using the rule $u \rightarrow v$; that is, the multiset identified by u is subtracted from M_{it}^0 :

The result of using the rule is determined by v : If an object appears in v in a pair $(a; here)$; then it will remain in the same region i_t : Sometimes when specifying the rules, the pair $(a; here)$ are simply written a ; (with the indication *here* omitted). If an object appears in v in a pair $(a; out)$; then a will exit the membrane i_t : And will become an element of the region immediately outside it. In this way, it is possible that an object leaves the super-cell itself: if it goes outside the skin of the super-cell then it never comes back. If an object appears in a pair $(a; in_q)$; then a will be added to the multiset M_q^0 , provided that a is adjacent to the membrane q : If $(a; in_q)$ appears in v and the membrane q is not one of the membranes delimiting *from below* the region i_t ; then the application of the rule is not allowed.

If the symbol $\pm$ appears in v ; then the membrane i_t is removed (we say *dissolved*) and at the same time the set of rules R_{i_t} (and its associated priority relation) to the multiset associated with the region which was immediately external to the membrane i_t : We do not allow the dissolving of the skin membrane, because this means that the super-cell is lost.

The operations are done in parallel for all possible application rules $u \vdash v$; for all occurrences of multisets u in the region associated with the rules, for all regions at the same time. No contradiction appears because of multiple membrane dissolving, or because of simultaneous appearance of symbols of the form $(a; out)$ and $\pm$. If at the same step we have $(a; in_i)$ outside a membrane i and δ inside this membrane, then, because of the simultaneity of performing these operations, again no contradiction appears: We

assume that a is introduced in membrane i at the same time when it is dissolved thus a will remain in the region placed outside membrane i ; that is, the effect of $(a; in_i); \pm$ is $(a; here)$:

If there are rules in a super-cell system I with the radius at least two, then the system is said to be *cooperative*; in the opposite case, it is called *non-cooperative*. A system is said to be *catalytic* if there are certain objects $c_1; \dots; c_n$ specified in advance, called *catalysts*, such that the rules of the system are either of the form $a \rightarrow v$; or of the form $c_i a \rightarrow c_i v$; where a is a non-catalysts object and v contains no catalyst.

Example 4.1

Consider the super-cell system of degree 4:

$$I = (V; I^1; M_1; \dots; M_4; (R_1; \frac{1}{4}); \dots; (R_4; \frac{1}{4}); 4);$$

$$V = f a; b; c; dg;$$

$$I^1 = [1[2[3]3]2[4]4]1;$$

$$M_1 = f accg;$$

$$M_2 = f ag;$$

$$M_3 = f cdg;$$

$$M_4 = ;;$$

$$R_1 = f r_1 : c! (c; in_4); r_2 : c! (b; in_4); r_3 : a! (a; in_2)b; dd! (a; in_4)g;$$

$$\frac{1}{4} = f r_1 > r_3; r_2 > r_3 g;$$

$$R_2 = f r_1! (a; in_3); ac! \pm g;$$

$$\frac{1}{2} = ;;$$

$$R_3 = f a ! \pm g;$$

$$\frac{1}{3} = ; ;$$

$$R_4 = f c ! (d; out); b ! b g;$$

$$\frac{1}{4} = ; :$$

In the initial configuration we can apply a rule in membrane 1 and one in membrane 2. If in membrane 1 we use the rule $c ! (b; in_4)$; then the computation will never halt: the rule $b ! b$ can be applied for ever in membrane 4. Thus, we will not use the rule $c ! (b; in_4)$; but the rule $c ! (c; in_4)$: Because both these rules can be applied and they have priority over the rule $a ! (a; in_2)b$; this later rule cannot be used. Thus, a symbol c is sent from membrane 1 to membrane 4 and at the same time a symbol a is sent from membrane 2 to membrane 3:

Now, no c rule in membrane 1 can be applied, hence the rule $a ! (a; in_2)b$ can be used. It has to be used for both copies of a in membrane 1. Hence two copies of a will be sent to membrane 2 and two copies of b will remain in membrane 1. At the same time, the rule $a ! \pm$ will be used in membrane 3, dissolving it, and the rule $c ! (d; out)$ will be used in membrane 4, sending a copy of d to membrane 1: As a result of these operations, membrane 1 will contain the multiset bbd ; membrane 2 will contain $aacd$; while, membrane 4 is empty; membrane 3 does no longer exist (hence, the rule $a ! (a; in_3)$ in membrane 2 is useless from now on).

Two more transitions can be performed. First, the rule $ac! \pm$ can be used in membrane 2; dissolving it and releasing the remaining objects ad . Thus, membrane 1 will contain the multiset $abdd$; which makes possible for the first time the use of the rule $dd! (a; in_4)$ from membrane 1: It consumes the two copies of d and sends a copy of a to membrane 4: No further rule can be applied, the *life* of the super-cell stops here.

A sequence of transitions in a super-cell system $\mathcal{S}$; starting from the initial configuration C_0 ; is called a computation with respect to $\mathcal{S}$:

CHAPTER FIVE

MULTISET-BASED TREE STRUCTURES AND THEIR APPLICATION TO MEMBRANE COMPUTING

In this chapter, we explicate a new paradigm proposed for study - multiset-based tree structures and their applications, especially to membrane computing. We represent a tree in the form of a wellfounded multiset. However, the conventional rule for this representation is not one-to-one from a set of trees to the class of multisets representing such trees. With this in mind, we devise a method of uniquely representing trees by establishing a one-to-one correspondence between trees and suitable permutations of wellfounded multisets, which we call *tree structures*. We give formal definitions of a tree structure and a *subtree structure*. Finally, we represent membrane structures in the form of tree structures. The work contained herein in this chapter has been published; see Singh and Peter (2011).

5.1 INTRODUCTION

A tree is an *acyclic* connected graph (having one source or trunk and several exits or leaves). It can also be defined as a partial order relation over a finite set with the smallest element. Note that if there is only one edge from a source node, then we have only one branch on the node. Formal definitions will be given later in this chapter. Trees have served as handy tools in solving problems involving decisions and the flow of information from one point to another.

Every tree can be represented in the form of a wellfounded multiset - a multiset with an irreflexive and transitive ordering defined on it in order that no infinite descending sequence of elements occurs. We follow the Darshowitz-Manna ordering on multisets over a set of natural numbers which has been proved wellfounded (Darshowitz and Manna, 1979).

In the recent years, the representation of a finitely growing tree in the form of a wellfounded (cardinality-bounded) multiset seems to be a conventional choice. However, the said representation is not one-to-one from a set of trees to the class of multisets representing such trees. In order to achieve uniqueness, various permutations of the wellfounded multiset in consideration along with a suitable rule are adopted. We begin with a binary tree, and generalize the approach to an *n*-nary tree and a *general* tree.

5.2 SOME APPLICATION AREAS

Trees are found in Mathematics, Management, Economics, Commerce, Biology, Computer Science, Statistics and Probability, just to mention but a few.

In Managerial accounting (Hilton, 1996), a tree diagram in the form of an organizational chart was used to illustrate the organization of Aloha Hotels and Rresorts. In Computer Science (Bailey, 1999), the Huffman tree is used to compress bits so as to reduce the amount of storage that is necessary in a storage media. In Biology, Tailor (1996) shows the tree diagram representing the basic characteristics of meiosis showing one chromosome duplication followed by two nuclear and cell divisions. In Probability,

(Kemeny, 1957), we come across the tree measure for a sequence of repeated throws of a coin with either side labelled head or tail. In English language, we represent the classification of nouns in the form of a tree as in Eyisi (2006). As some membranes can contain several other membranes as indicated by Păun (2002), trees can also be used to study membranes in membrane computing.

Note that it may be relevant across the board to count the number of nodes in an n -nary tree - a tree with $n(, 2)$ number of branches emerging from each of the major branches below them (which is called a *tail*) or a general tree for that matter, with n varying from branch to branch.

5.3 APTNESS FOR THE USE OF TREES AND A MULTISSET ENVIRONMENT

Unlike other graphs, trees are quite innovative especially in one of the recently researched areas of computer science - molecular and membrane computing. This is because among all other forms of graphs (e.g., a *loop* or a *multigraph*), only trees can suitably represent membrane structures (without intersections, loops or parallel edges). This singular but essential ability of trees is what has inspired us to study them and to see how they can help us in contributing to the improvement of membrane computing.

There is no provision for a diagrammatic or pictorial input of membrane structures at programming level. Thus, one aim of this thesis is to demonstrate how membrane structures can be represented at programming level by devising a discrete approach for

such a representation. It is in line with this that we have employed multiset as an environment in representing trees, and consequently membrane structures.

5.4 SOME BASIC CONCEPTS

Definition 5.1 Wellfounded multiset

A wellfounded multiset is a multiset with an irreflexive and transitive ordering defined on it, such that every of its submultiset has a minimal element; in other words, no infinite descending chain occurs. We shall follow the Darshowitz-Manna ordering on multisets over a set of natural numbers which has been proved wellfounded by Dershowitz and Manna (1979).

Definition 5.2 A binary tree

A binary tree is a tree which has exactly two branching on each branch of the tree.

Definition 5.3 An n -nary tree

Given a non-negative integer n , an n -nary tree is a tree which has exactly n number of branching on each branch of the tree.

Definition 5.4 A general tree

A *general tree* is a tree which has varying number of branching on each of its branches.

Definition 5.5 A subtree

A part of a tree which is also a tree but smaller than or equal to the tree is called a *subtree* of the tree. A tree is called the parent or a full tree in relation to its subtree.

See Dershowitz and Manna (1979) for details of the aforesaid definitions.

5.5 THE BINARY TREE

Darshowitz and Manna (1979) demonstrated the termination of a program to count the number of tips of a binary tree using a wellfounded multiset ordering. We describe in brief one of the examples they considered.

Consider a simple program to count the number of tips – terminal nodes (without descendents) – in a full binary tree. Each tree y that is not a tip has two subtrees, $left(y)$ and $right(y)$.

Typically, a binary tree can be schematically represented as in the figure below.

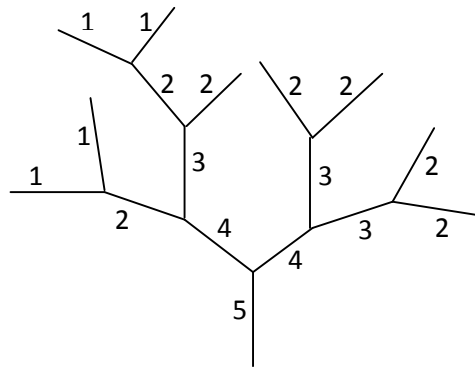


Figure 5.1: A binary tree

In Figure 5.1 above, the *tree trunk* is represented by the largest integer which we write beside it. Every branch has its labelling beside it. The subtree at the lowest part of the tree is called the *base* of the tree. In a y-shaped (or fan-shaped) subtree, one of the branches on top is called an *axis*. The part of the tree which continues from the base to a tip without branching is called a *chain*. The number of straight lines in a chain determines the length of the chain. Let us consider the following pattern drawn from the arrangement of the tree represented in Figure 5.1.

```

5  1:
4  2:      2!  5
3  3:      2!  4      1!  4
2  7:      2!  3      2!  3      2!  3      1!  4
1  4:      2!  2      2!  2

```

In the above pattern, branch 5 appears 1 time on the tree. This is the trunk of the tree. Branch 4 appears 2 times on the tree, both on branch 5. Branch 3 appears 3 times on the tree, twice on one branch 4 and once on the other branch 4. Branch 2 appears 7 times on the tree, twice on each of the 3 branches 3 and once on branch 4. Branch 1 appears 4 times on the tree, twice on each of two of the seven branches 2.

The pattern can further be simplified as in the following pattern by considering the integer representing the multiplicity (or characteristic) of the appearance of each branch.

$5 \rightarrow 1:$
 $4 \rightarrow 2: \quad 2 \rightarrow 5$
 $3 \rightarrow 3: \quad (2 \rightarrow 4), (1 \rightarrow 4)$
 $2 \rightarrow 7: \quad 3(2 \rightarrow 3); (1 \rightarrow 4)$
 $1 \rightarrow 4: \quad 2(2 \rightarrow 2)$

The following is a tabular representation of the aforesaid pattern:

Table 5.1: Tabular representation of the tree in Figure 5.1

n	f_1	$f_2(f_3 \rightarrow n_p)$
5	1	
4	2	$1(2 \rightarrow 5)$
3	3	$1(2 \rightarrow 4), 1(1 \rightarrow 4)$
2	7	$3(2 \rightarrow 3), 1(1 \rightarrow 4)$
1	4	$2(2 \rightarrow 2)$

In the above table, branch n has frequency f_1 on the tree and appears f_3 times on the predecessor branch n_p ; f_2 number of times.

For further application, especially in computer science, this table may prove advantageous in the data entry interface, when writing a program to manipulate a tree.

5.6 CONVENTIONAL APPROACH TO REPRESENTING A TREE BY A WELLFONDED MULTISSET

A *conventional method* of representing a tree by a wellfounded multiset seems to have first appeared in Dershowitz and Manna (1975). The conventional method entails that in a wellfounded multiset, any element-multiset representing a subtree which is built upon another element-multiset representing another subtree needs to be smaller than the one upon which it is built. Moreover, there is no permutation governing the arrangement of the element-multisets of the wellfounded multiset. Rather, one rule inherent in the representation is that the smaller and larger element-multisets have exactly one element in common. This common element is also the largest element in the smaller element-multiset and cannot be the largest element in the larger element-multiset. The action of picking an element-multiset to represent a subtree on the tree is done exhaustively. Though no definition of this method has been given in Dershowitz and Manna (1975), we give the following formal definition to capture the concept.

Definition 5.6

Formally, given a multiset S over a domain set D , a multiset $T(S)$ whose elements are submultisets of S is a *conventional representation of a tree* if and only if it satisfies the following properties:

- i. There exists $z \in T(S)$ such that $z = \max \{y : y \in T(S)\}$,
- ii. For each $y \in T(S)$ where $y \neq \max \{x : x \in T(S)\}$; $\exists z \in T(S)$ with $z \in y$ and $x_0 \in S$ such that $x_0 \in y \setminus z$, $x_0 = \max \{x : x \in y\}$, $x_0 \neq \max \{x : x \in z\}$.

In the above definition, the first condition is called *the base condition* and z is called the *base* of a *tree*. The second condition is called *the join condition* and x_0 is the *join* between two *subtrees*.

5.7 WELLFOUNDED MULTISSET REPRESENTATION OF A BINARY TREE (USE OF THE CHOICE RULE)

A binary tree can be represented in the form of a wellfounded multiset, whose elements are multisets containing only three elements. For example, the wellfounded multiset $\{ \{ \{ 3, 2 \} \}; \{ \{ 1 \} \} \}$ represents the following binary tree:

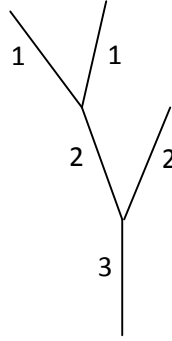


Figure 5.2: A two-element binary tree

Each y-shaped subtree is represented by an element say $\mathbf{a}$ in the multiset; $\mathbf{a}$ itself is a multiset with three elements of the form $\mathbf{f a_1; a_2; a_3g}$. We show that the *conventional* approach to the representation is not injective from a set of trees to the class of wellfounded multisets representing such trees.

Theorem 5.1

The representation of a binary tree by a wellfounded multiset is not unique using the conventional approach.

Proof: Consider $N = \mathbf{f f 544g; f 433g; f 432g; f 322g; f 322g; f 322g; f 211g; f 211gg}$. Each element of N has three elements and represents a y-shaped subtree of the tree in Figure 5.1. It is easy to see that N is ordered according to the Darshowitz-Manna ordering on multisets.

The element $f_{544}g$ represents the first y-shaped subtree with tail 5. The two 4's in $f_{544}g$ show that we can locate two elements in N ; each of which has 4 as its largest element. The two elements are $f_{433}g$ and $f_{432}g$ which give rise to two y-shaped subtrees with tails labelled 4 on the trunk labelled 5.

The three 3's in these two elements show that we can locate three other elements in N each of which has 3 as its largest element. These elements are three $f_{322}g$'s, which give rise to three y-shaped subtrees with tails labelled 3 on the branch labelled 4:

Next are the six 2's in the three elements. But, since there are only two $f_{211}g$'s in N each of which has 2 as its largest element, there can only be two y-shaped subtrees each of whose tail is labelled 2.

Also, the 2 in $f_{432}g$ shows that we can locate an element in N which has 2 as its largest element. This element is $f_{211}g$; and can give rise to a y-shaped subtree with tail labelled 2:

It can be observed that there arises a decidability problem as to how and on which of the two types of branches, one of which is labelled 4 and three of which are labelled 3, should the two subtrees with tail labelled 2 be built upon. If one of the two $f_{211}g$'s is built on any one of the three $f_{322}g$'s while the other $f_{211}g$ is built on

f 432g, we get the binary tree A in Figure 3 below. If on the other hand each of the two f 211g's is built on each of any two of the three f 322g's, we get the binary tree B in Figure 5.3.

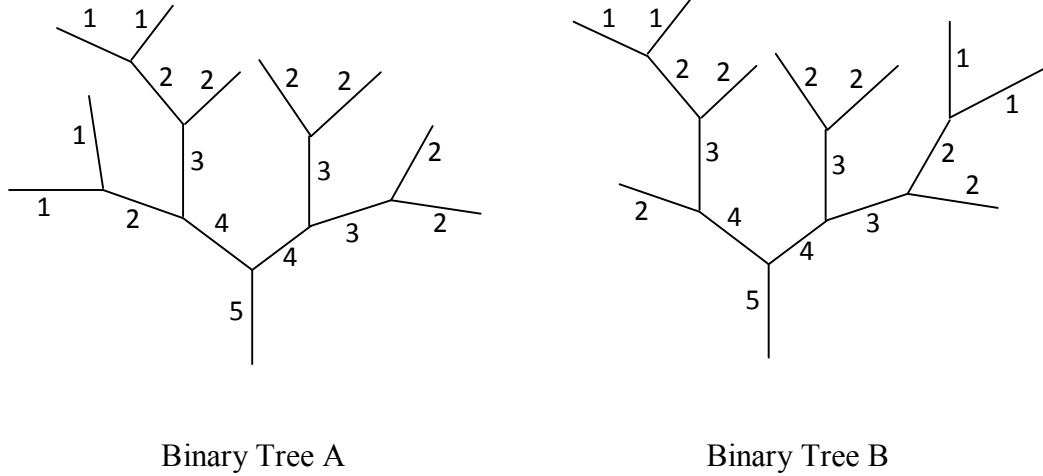


Figure 5.3: Two different binary trees generated by the multiset N

Thus, our ordered multiset N can yield two different binary trees and so the choice rule does not uniquely determine a binary tree as stated. □

This theorem can be generalised in the case of an n -nary tree. The theorem can similarly be proved for a tree with n varying from branch to branch on the tree, called a *general* tree.

5.8 THE SAW RULE

We demonstrate below by considering a suitable permutation of a partially ordered multiset, how to represent a rule in which each submultiset of elements is arranged in an

uphill manner, while at the same time the multiset of all the elements in the system is arranged in a *downhill* manner, called a *saw-like permutation*.

An *uphill* multiset of elements is a permutation of a partially ordered multiset of elements which ends with the largest element in the multiset. A *downhill* multiset of elements is a permutation of a partially ordered multiset of elements which begins with the largest element in the multiset. Thus, all the elements in an uphill multiset of elements or in a downhill multiset of elements may neither be ascending nor descending.

This rule is called a *saw-like permutation* because it creates a resemblance of a wood saw blade when viewed pictorially using vertical bars to represent the elements of a permutation of a partially ordered multiset according to their sizes as in the following figure:

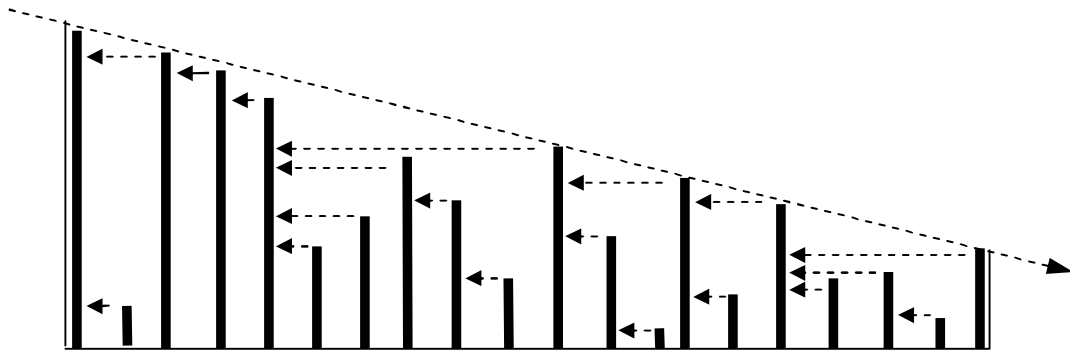


Figure 5.4: A Saw-like structure illustrating a saw-like permutation

The scheme can be interpreted as follows. Each bar in a multiset of ascending bars is attached to the longest bar immediately before the multiset. In other words, any two bars in which one is attached to another must not have any bar longer than the attached bar in

between them. The bars represent the element-multisets of a permutation of a partially ordered multiset. The left arrow pointing from one bar to another indicates attachment from the bar on its right to the bar on its left. In other words, each bar on the right side of an arrow represents a branch while the corresponding bar on the left represents a node on the tree. We shall often refer to the element-multisets of a wellfounded multiset as simply elements.

While going through the elements of the partially ordered multiset, we pick the first and largest element represented by the longest bar. This is followed by the smallest element that can be attached directly to the largest element. This in turn is followed by another element only larger than the smallest element, and can be attached directly to the largest element. Any element that must be attached to an element which has already been attached to the largest element must come after the element upon which it is to be attached, even if it happens to be smaller than the first element attached directly to the largest element.

We continue in this way until we have exhausted all the elements that can be attached to the largest element. In Figure 5.4 above, only two bars have been attached to the first (largest) bar. There is no doubt that these two bars are trivially in an uphill order according to their heights.

This gives us a submultiset of elements immediately following the largest element in an uphill order. Among all the submultisets in an uphill order which have been attached to

an element, this one happens to be the largest. The last but not the least element (the third bar in Figure 5.4 above) in this submultiset turns out to be the second largest element in the partially ordered multiset, and it is the element upon which the second submultiset of elements in an uphill order will be attached starting with the smallest element directly attachable to it. In Figure 5.4, the second largest bar (which is the third bar) has only one bar attached to it. This one bar can be regarded as a submultiset in an uphill order containing a singleton multiset, though a very trivial case.

Next is the third largest submultiset of elements in an uphill order. We continue in this way until all the elements in the partially ordered multiset belong to a group of elements in an uphill order. Note that in Figure 5.4 above, the fifth bar has four bars attached to it, including the eighth bar, which in turn has two bars attached to it. These four bars are clearly in ascending order, whereas the six bars (the four bars with the two bars) make up the multiset of bars attached to the fifth bar in an uphill order. This is a non trivial example of bars arranged in an uphill order.

Every element on the saw-like permutation is called a *bar*; every bar which has an element attached to it is called a *column*; a column which is followed by a non-singleton submultiset of descending bars each of which is less than the column and at least one of which has a bar attached to it, is called a *pillar*. A submultiset of consecutive elements from the same object whose multiplicity is more than one is called a *platform*. The first bar on the saw-like permutation is called its *base*.

Having demonstrated how the *saw-like permutation* is used to arrange the elements of a partially ordered multiset in the above discussion, we now give a formal definition of the representation of a partially ordered multiset *in order of a saw-like permutation* in the following definition:

Definition 5.7

A permutation $P(S)$ of a partially ordered multiset S of order n is said to be *in order of a saw-like permutation* if and only if $y_1 > y_i$ in $P(S)$, for all $i = 2; 3; \dots; n$.

If such a permutation exists, we say that $P(S)$ is arranged in order of a saw-like permutation or $P(S)$ is a saw-like permutation of the elements of S . We shall see later in this thesis that this construction is of immense help in defining a tree structure.

5.9 REPRESENTATION OF A TREE BY A SAW-LIKE PERMUTATION OF A WELLFOUNDED MULTISSET (USE OF THE SAW RULE)

To resolve the aforesaid issue of non-uniqueness of the representation as shown in the theorem above, we demonstrate by considering *saw-like permutations*, how to represent a tree by a permutation of a wellfounded multiset, called *the saw rule*. The following illustration will guide us in understanding how to represent a binary tree using *the saw rule* and also to demonstrate uniqueness using this rule. We consider the following illustrations to bring our point home. Let us consider the following two permutations N_1 and N_2 of N :

$N_1 = [f\ 544g; f\ 423g; f\ 322g; f\ 211g; f\ 433g; f\ 322g; f\ 322g; f\ 211g]$ and

$N_2 = [f\ 544g; f\ 423g; f\ 211g; f\ 322g; f\ 211g; f\ 433g; f\ 322g; f\ 322g]$

The element $f\ 544g$ of N_1 represents the first y-shaped subtree (called the *base*) with tail 5:

The two 4's in $f\ 544g$ show that we can locate at most two elements in N_1 each of which has 4 as its largest element. The two elements are $f\ 423g$ and $f\ 433g$.

The element $f\ 423g$ is smaller than $f\ 544g$ and so can give rise to a y-shaped subtree with tail labelled 4: The 3 in $f\ 423g$ shows that we can locate at most an element in N_1 which has 3 as its largest element, and since the next element $f\ 322g$ is smaller than $f\ 423g$ it can give rise to a y-shaped subtree with tail labelled 3:

The two 2's in $f\ 322g$ show that we can locate at most two elements each of which has 2 as its largest element. Since there is one such element which has 2 as its largest element and smaller than $f\ 322g$ then it can give rise to a subtree with tail labelled 2.

The next element $f\ 433g$ is larger than $f\ 322g$ and so no subtree representing $f\ 433g$ can arise on $f\ 322g$. However, $f\ 544g$ is the smallest element larger than $f\ 433g$, going backwards. Since only one of the two elements having 4 as their largest element has its subtree on $f\ 544g$, there can arise another subtree with tail labelled 4 on $f\ 544g$ (since $f\ 544g$ has two 4's in it).

The two 3's in $f 433g$ show that we can locate at most two elements each of which has 3 as its largest element. Since the next two elements satisfy this condition, and are smaller than $f 433g$, then there can arise two subtrees with tails labelled 3.

The last two 2's in the last $f 322g$ show that we can locate at most two elements each of which has 2 as its largest element. Since we have only one of such element which is $f 211g$ having 2 as its largest element, there can arise only one subtree with tail labelled 2 coinciding with only one of the axes labelled 2.

Our binary tree can be seen to have been uniquely constructed by N_1 and the binary tree A in Figure 5.3 above is the tree in question. Similarly, N_2 uniquely determines the binary tree B in Figure 5.3. We call N_1 and N_2 *tree structures*. The following figures are saw-like structures of N_1 and N_2 equivalent to the one in Figure 5.4, for a clear understanding of the concept.

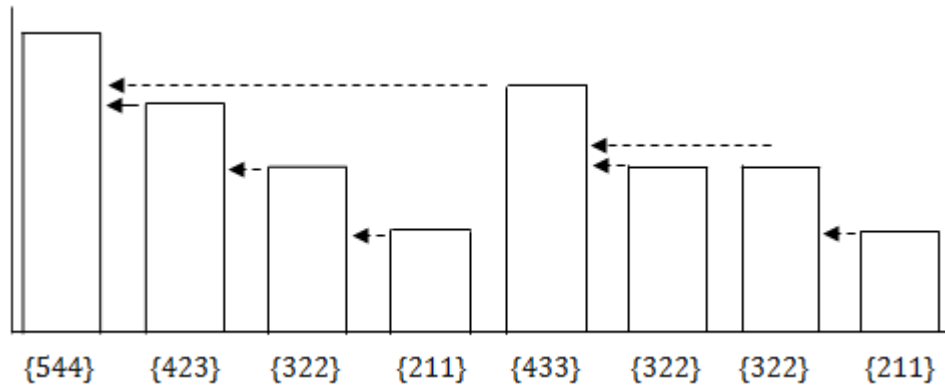


Figure 5.5: A Saw-like structure illustrating N_1 .

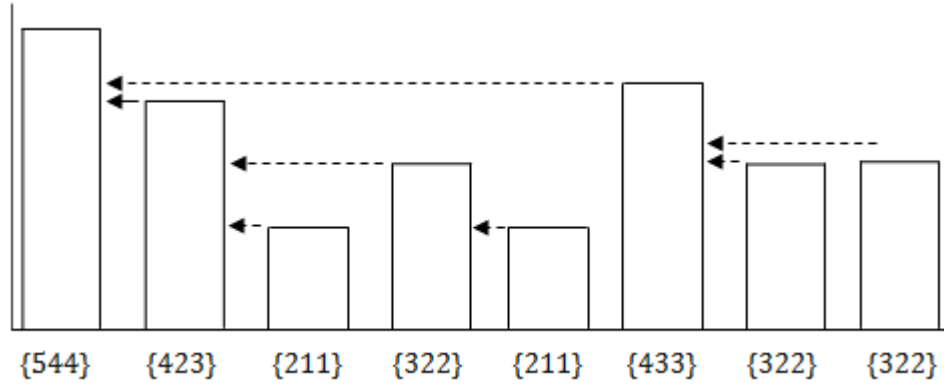


Figure 5.6: A Saw-like structure illustrating N_2 .

Notice in Figures 5.5 and 5.6 above, that a bar (representing a subtree on the tree) can only be attached to a larger bar (representing a node on the tree) where the larger bar is a multiset containing the largest element of the smaller multiset representing the smaller bar. Such largest element must not be the largest in the larger bar. Notice also that a bar has not been attached to another bar of equal size (or height in this case).

Not all wellfounded multisets, even with the *conventional method*, can represent a tree. This is seen from the multiset $[f\ 544g; f\ 432g; f\ 211g; f\ 432g; f\ 211g; f\ 211g]$ since the supposed subtree structure $[f\ 211g]$ of the last element $f\ 211g$ does not have a branch to rest upon. The *saw rule* ensures, firstly, that a wellfounded multiset can suitably represent a tree; secondly, that every tree can be represented in the form of a wellfounded multiset (this is the only property that the conventional method ensures) and thirdly, that this representation is injective from a set of trees to the class of multisets representing such trees.

REMARK 5.1

It is important to observe that in order to construct a tree using a unique permutation of a given wellfounded multiset, the use of *the saw rule* in building an element upon a preceding element is not only that the subsequent element be smaller than the preceding one, but also that its largest element must belong to the preceding element, and that the multiplicity of such subsequent element in an uphill submultiset of elements must not be greater than the number of occurrences of such largest element in the preceding element.

For example, in $[::f\,543g;f\,211g::]$, the subtree representing $[f\,211g]$ cannot be built upon the subtree representing $[f\,543g]$ since the largest element 2 of $f\,211g$ does not belong to $f\,543g$. But in $[::f\,543g;f\,411g::]$, $[f\,411g]$ can be built upon $[f\,543g]$ since the largest element 4 of $f\,411g$ is in $f\,543g$. Again, we cannot build any subtree using $[::f\,543g;f\,411g;f\,411g;f\,411g::]$ since 4 does not appear up to three times in $f\,543g$. However, $[::f\,5444g;f\,411g::]$ represents a subtree. Such an element as 4 in this case is called a *join* in a *tree structure*. We use square brackets for a *tree structure* since a tree structure is a list – an ordered sequence of elements with repetitions allowed.

To avoid misinterpretation, we shall not use a tree and a tree structure interchangeably. As mentioned in the introduction section of this chapter, a tree is an **acyclic connected graph**. On the other hand, a *tree structure* is a **multiset** representation of a tree by exploiting the *saw rule*, e.g., N_1 and N_2 above. Notwithstanding the fact that there is a one-to-one correspondence between the two, this distinction is useful especially when we

shall be applying the concept of tree structures to membrane computing. The same argument goes for a subtree and a subtree structure.

Having discussed the representation of a binary tree in order of a saw-like permutation of a wellfounded multiset (called a *tree structure*), we consider it necessary to generalize our discussion to capture the concepts of an n -nary tree structure (corresponding to a tree with exactly n branches on each node of the tree) and a *general tree* (corresponding to a tree with n varying from branch to branch on the tree). Therefore, based on the discussions above, we now give the following formal definitions:

Definition 5.8

Let S be a partially ordered multiset over a domain set D : A permutation $P(\zeta(S))$ of a wellfounded multiset $\zeta(S)$ of cardinality m ; whose elements are submultisets of S is called an n -nary *tree structure* if and only if it satisfies the following properties:

- i. For all $y \in P(\zeta(S))$ $\exists n \in \mathbb{N}$ such that $C(y) = n$, where $C(y)$ denotes the cardinality of y .
- ii. $y_1 \leq y_i$ in $P(\zeta(S))$ for all $i = 2, 3, \dots, m$.
- iii. For all $i, j \in \mathbb{N}$ with $y_i \leq y_j$ for $i < j$, if $\exists k \in \mathbb{N}$ such that $y_i \leq y_k \leq y_j$ for $i < k < j$ then $\exists x_0 \in \mathbb{N}^1$ $y_i \setminus y_j$ such that $x_0 = \max\{x : x \in y_j\}$ and

$x_0 \in \max\{x : x \preceq y_i\}$, where $y_i, y_k, y_j \preceq P(\zeta(S))$, $i = 1; 2; \dots; m-1$,
 $k = 2; 3; \dots; m-1$ and $j = 2; 3; \dots; m$.

In other words, it says that y_i *yields* y_j via x_0 or y_j is an immediate successor of y_i or y_i is an immediate predecessor of y_j , and we denote this by $y_i \stackrel{f_{x_0}}{\rightarrow} y_j$.

- iv. For each element $z \in P(\zeta(S))$ and for a given $x \preceq S$, the multiset $Y = \{y : z \stackrel{f_x}{\rightarrow} y\}$ is such that $C(Y) = m_z(x)$, where $C(y)$ is the cardinality of Y and $m_z(x)$ is the multiplicity of x in z .

In the above definition, the first condition is known as *the equal cardinality condition*. The second condition is known as *the base condition* (or *the saw rule condition*) and y_1 is *the base* of the tree structure. The third condition is the *join condition* and x_0 is *the join* between two subtree structures. The fourth condition is *the parallelism condition*. The parallelism condition ensures that the multiplicity of a join in a node is greater than or equal to the number of subtree structures (or branches) joinable to the node using such join.

Definition 5.9

Let S be a partially ordered multiset over a domain set D ; a permutation $P(\zeta(S))$ of a wellfounded multiset $\zeta(S)$ of cardinality m , whose elements are submultisets of S ; is called a *general tree structure* if and only if it satisfies the following properties:

- i. $y_1 \wedge y_i$ in $P(\zeta(S))$ for all $i = 2; 3; \dots; m$.

- ii. For all $i, j \in \mathbb{N}$ with $y_i \leq y_j$ for $i < j$, if $\exists k \in \mathbb{N}$ such that $y_i \leq y_k \leq y_j$ for $i < k < j$ then $\exists x_0 \in y_i \setminus y_j$ such that $x_0 = \max\{x : x \leq y_j\}$ and $x_0 \leq \max\{x : x \leq y_i\}$, where $y_i, y_k, y_j \in P(\zeta(S))$, $i = 1; 2; \dots; m-1$, $k = 2; 3; \dots; m-1$ and $j = 2; 3; \dots; m$.

In other words, it says that y_i yields y_j via x_0 or y_j is an immediate successor of y_i or y_i is an immediate predecessor of y_j , denoted by $y_i \xrightarrow{f_{x_0}} y_j$.

- iii. For each element $z \in P(\zeta(S))$ and for a given $x \in S$, the multiset $Y = \{y : z \xrightarrow{f_x} y\}$ is such that $C(Y) = m_z(x)$, where $C(y)$ is the cardinality of Y and $m_z(x)$ is the multiplicity of x in z .

The first condition is *the base condition*. The second condition is *the join condition* and the third condition is *the parallelism condition*.

In the above two definitions, the tree structure $P(\zeta(S))$ is said to be built over the multiset S with D as its domain. We define the root set of the tree structure $P(\zeta(S))$ as the set $R = \{x \in D : x \leq y \text{ } \forall y \in P(\zeta(S))\}$. If there is no confusion about which multiset D is intended, we simply write ζ for $P(\zeta(S))$.

The process by which a subtree structure yields another subtree structure in a tree structure is called a *succession*. That is, if $y_1 \xrightarrow{f_{x_1}} y_2$ and $y_2 \xrightarrow{f_{x_2}} y_3$ then y_3 is a successor

(not an immediate successor) of y_1 and we write $y_1 \xrightarrow{f_{x_1, x_2} g} y_3$ (y_1 yields y_3 via the set $f_{x_1, x_2} g$). There are immediate successions between y_1 and y_2 and between y_2 and y_3 . There is also a succession between y_1 and y_3 , however, this is not an *immediate succession*.

Definition 5.10

Given a tree structure $\mathcal{L}$ over a multiset S , a tree structure $\mathcal{M}$ over S is called a *subtree structure* of $\mathcal{L}$ if and only if $\exists y; z \in \mathcal{M}$ and $x \in S$ such that $z \xrightarrow{f_x g} y; \exists a; b \in \mathcal{L}$ with $y \in a$ and $z \in b$ such that $b \xrightarrow{f_x g} a$.

In other words, a tree structure $\mathcal{M}$ is a subtree structure of a tree structure $\mathcal{L}$ if and only if it *inherits* all its immediate successions from the tree structure $\mathcal{L}$:

The following are some immediate consequences of this definition. A subtree structure of a tree structure may not necessarily be a submultiset of the tree structure and vice versa. There are subtree structures of a tree structure whose members are not elements of the tree structure. For example, for the tree structure

$$\mathcal{L} = [f_{544}g; f_{423}g; f_{322}g; f_{211}g; f_{433}g; f_{322}g; f_{322}g; f_{211}g],$$

the subtree structure $[f_{54}g; f_{43}g; f_{32}g]$ of $\mathcal{L}$ is not a submultiset of $\mathcal{L}$ since all of its elements do not belong to $\mathcal{L}$. The submultiset $[f_{544}g; f_{322}g; f_{211}g]$ of $\mathcal{L}$ does not represent any subtree structure of $\mathcal{L}$, since it does not form a tree structure. The multiset

$[f_{544g}; f_{433g}; f_{21g}]$ is neither a submultiset of ζ nor a subtree structure of ζ while the multiset $[f_{544g}; f_{423g}; f_{322g}]$ is both a submultiset of ζ and a subtree structure of ζ .

5.10 TREE STRUCTURE - BASED REPRESENTATION OF MEMBRANE STRUCTURES

In this section, we apply the aforesaid technique to represent membrane structures. Membrane structures can be represented in the form of a tree as in Păun (2002). Let us consider the schematic representation of the membrane structure in Figure 5.7. It is customary to label the largest membrane with the number 1; the next larger membrane with the number 2 and so on.

In our example and in order to have an intuitively clear representation, we shall identify the membranes by their membrane sizes. For instance, a membrane labelled 2 will be identified by the membrane size M_2 :

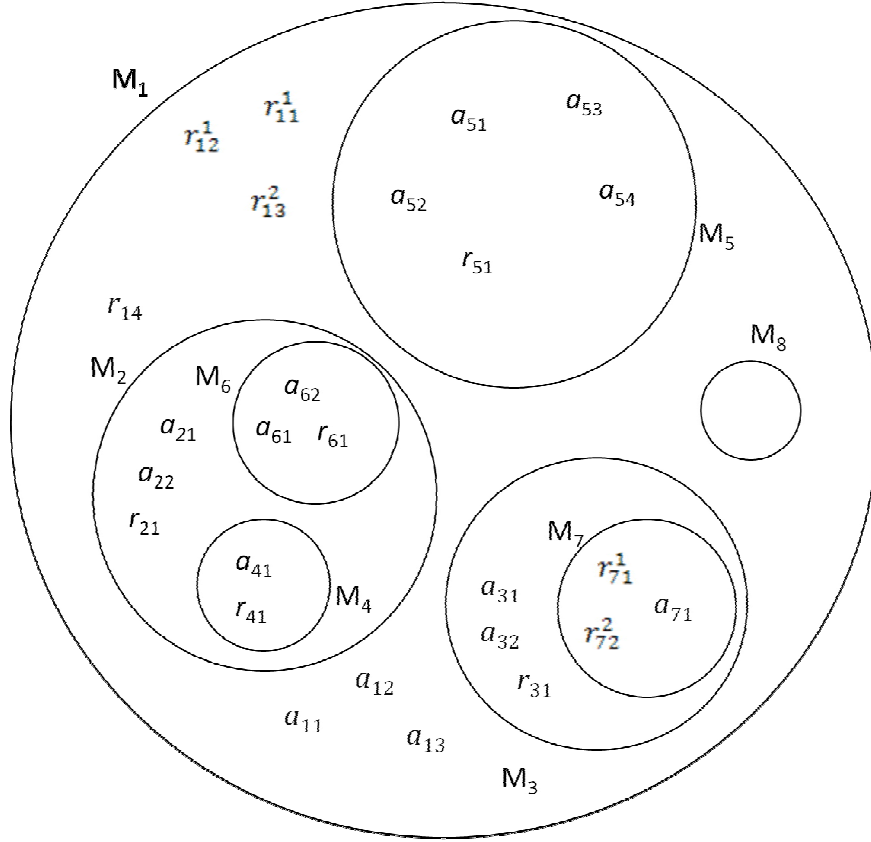


Figure 5.7: A membrane structure with objects and rules

Consider the membrane structure ¹ in Figure 5.7. M_1 contains $M_2; M_3; M_5$ and M_8 ; M_2 contains M_4 and M_6 ; M_3 contains M_7 while M_8 is empty; also the relations $M_2 > M_3 > M_5$ and $M_6 > M_4$ are true.

Let $S = \{M_i : i = 1; 2; \dots; 9\}$. The size of a membrane is defined as the sum of the multiplicities of all the objects in the membrane (Păun, 2000). The domain set of the multisets having the sizes of membranes in a membrane structure as elements is therefore wellfounded with the usual ordering, being a subset of the set $\mathbf{N}$ of natural numbers. Thus, S is wellfounded and it follows that a multiset having elements of S as elements of its elements is also wellfounded (see for proof in Dershowitz and Manna (1979)).

We note that the membrane sizes may change during the process of *transition*. In particular, the above relations may not hold especially for elementary membranes. However, this does not change the fact that our tree structure still remains valid, following its definition and going by the definition of the size of membranes. The following is a tree representing the membrane structure.

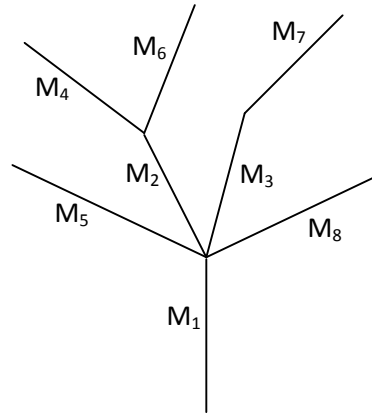


Figure 5.8: A tree representation of the membrane structure in Figure 5.7

The *tree structure* representation of the above tree is

$$^1 = [f M_1 M_2 M_3 M_5 M_8 g f M_3 M_7 g f M_2 M_4 M_6 g]:$$

We now present the above representation in greater details. The contents of the membranes are represented by a subscripted letter such as a_{ij} of the j^{th} element in the i^{th} membrane, while the m^{th} rule in the k^{th} membrane is represented by r_{km}^l , l being the rules' priorities if any. The membrane structure can further be expanded to show the contents of each membrane at the initial configuration of the system.

$$^1 = [f M_1[a_{11}a_{12}a_{13}r_{11}^1r_{12}^2r_{13}]M_2[a_{21}a_{22}r_{21}]M_3[a_{31}a_{32}r_{31}]M_5[a_{51}a_{52}a_{53}r_{51}]M_8[]g \\ f M_3M_7[a_{71}r_{71}^1r_{72}^2]gf M_2M_4[a_{41}r_{41}]M_6[a_{61}a_{62}r_{61}]g]$$

The subscripts used for the labelling of the elements are for illustration purpose and will not appear in the example we shall give below. The contents of membrane M_i have been grouped in the square brackets immediately following the membrane. If a membrane is contained in M_i ; such containment follows the rule governing attachment in the *tree structure*. An empty membrane is denoted by an empty square bracket. Elementary membranes appear only once in the *tree structure*.

The first membrane is the skin membrane and is the only non-elementary membrane which is allowed to appear once in the tree structure. Any other membrane which is neither the skin membrane nor an elementary membrane will appear more than once, since it has to show containment of membranes in separate curly brackets following the rule governing attachment in the tree structure.

5.11 COMPUTATION (AN EXAMPLE)

The following is the example on pages 10 - 11 in Păun (2000) of a transition in a (cooperative) super-cell system. In this example we make a little modification by representing our membrane structure in a *tree structure* (a saw-like permutation of a wellfounded multiset). The subtree structures have both objects and rules (with associated rule priorities) as members, in which case if a membrane contains only rules, such

membrane is said not to be empty. The original tree structure representing the membrane structure is the initial configuration prior to the transitions.

Consider the following super-cell system of degree 4:

$$\mathcal{C}_0 = (V; \mathcal{M}_1; \dots; \mathcal{M}_4; (R_1; \mathcal{M}_1); \dots; (R_4; \mathcal{M}_4); 4);$$

$$V = f a; b; c; dg,$$

$$\mathcal{M}_1 = [f M_1 M_2 M_4 gf M_2 M_3 g],$$

$$M_1 = [aacr_{11}^1 r_{12}^1 r_{13}],$$

$$M_2 = [ar_{21}],$$

$$M_3 = [cdr_{31}],$$

$$M_4 = [r_{41}],$$

$$R_1 = fr_{11} : c_j! \quad (c; in_4); r_{21} : c_j! \quad (b; in_4); r_{13} : a_j! \quad (a; in_2)b; dd_j! \quad (a; in_4)g,$$

$$\mathcal{M}_1 = fr_{11} > r_{13}; r_{12} > r_{13}g,$$

$$R_2 = fr_{21} : a_j! \quad (a; in_3); ac_j! \quad \pm g,$$

$$\mathcal{M}_2 = ; ,$$

$$R_3 = fr_{31} : a_j! \quad \pm g,$$

$$\mathcal{M}_3 = ;_3,$$

$$R_4 = fr_{41} : c_j! \quad (d; out); b_j! \quad bg,$$

$$\mathcal{M}_4 = ;_4.$$

$$C_0 : \mathcal{M}_1 = [f M_1 [aacr_{11}^1 r_{12}^1 r_{13}] M_2 [ar_{21}] M_4 [r_{41}] gf M_2 M_3 [cdr_{31}] g]:$$

In the initial configuration C_0 we can apply a rule in membrane M_1 and one in membrane M_2 : If we use the rule $c! \quad (b; in_4)$ in membrane M_1 ; the rule $b! \quad b$ can be applied non-

stop and the computation will never end. Therefore, we will not use the rule $c! (b; in_4)$; but the rule $c! (c; in_4)$: Since both these rules can be applied and they have priorities over the rule $a! (a; in_2)b$: this latter rule cannot be used. Hence, the object c is sent from membrane M_1 to membrane M_4 and at the same time the object a is sent from membrane M_2 to membrane M_3 :

$$C_1 : ^1 = [f M_1[aar_{11}^1 r_{12}^1 r_{13}] M_2[r_{21}] M_4[\alpha_{41}] gf M_2 M_3[acdr_{31}] g]:$$

No rule can be applied on c in membrane M_1 ; hence the rule $a! (a; in_2)b$ can be used. It will be used for both copies of a in membrane M_1 ; and so two copies of a will be sent to membrane M_2 and two copies of b will remain in membrane M_1 : At the same time, the rule $a! \pm$ will be used in membrane M_3 ; dissolving it, and the rule $c! (d; out)$ will be used in membrane M_4 ; sending a copy of d to membrane M_1 : As a result of these operations, membrane M_1 will contain the string bbd ; membrane M_2 will contain the string $aacd$; while membrane M_4 will contain no string; membrane M_3 no longer exists, therefore the rule $a! (a; in_3)$ in membrane M_2 is useless for now.

$$C_2 : ^1 = [f M_1[b b d r_{11}^1 r_{12}^1 r_{13}] M_2[a a c d r_{21}] M_4[r_{41}] g]:$$

The rule $ac! \pm$ can be used in membrane M_2 ; dissolving it and releasing the remaining objects ad : Thus, membrane M_1 will contain the string $ab b d d$:

$$C_3 : ^1 = [f M_1[a b b d d r_{11}^1 r_{12}^1 r_{13}] M_4[r_{41}] g]:$$

It is now possible for the first time to use the rule $dd! (a; in_4)$ from membrane M_1 : It consumes the two copies of d and sends a copy of a to membrane M_4 : No further rule can be applied, and the *life* of the super-cell stops here.

$$C_4 : ' = [f M_1[abbr_{11}^1 r_{12}^1 r_{13}] M_4[ar_{41}] g]:$$

5.12 CONCLUSION AND FUTURE DIRECTION

This work can be seen as a stepping stone to bringing DNA computing and the structure of membranes to a form in which mathematical tools and methods can be used to analyse the subject. In particular, the communications among membranes can be better interpreted.

As efforts are being made to simulate transition P system on digital computers (Rogozhin and Boian, 2003), it is interesting to observe that a membrane structure represented in the form of a Venn diagram or a tree diagram is not readily amenable to the computer at programming level, the use of multiset-based tree structures can be of great help.

We hope that this work will be helpful in providing tools to explore various applications of multiset-based model in areas of research especially in biology and computer science.

For further application, especially in computer science, table 5.1 proves useful in a data entry interface when writing a program to manipulate trees structures.

The application of the *saw-like permutation* can be exploited in describing various algebraic structures of a *tree structure*.

REFERENCES

1. Alhazov, A., Margenstern, M., Rogozhin, V., Rogozhin, Yu. and Verlan S. (2005), Communicative P systems with minimal cooperation., in (Mauri, G. , Păun, Gh., Prez-Jimenez, M. J., Rozenberg, G., Salomaa, A., eds (2004). *Membrane Computing. International Workshop WMC5, Milan, Italy. Revised Papers*), LNCS 3365, Springer-Verlag, Berlin, 162-178.
2. Amos, M. (1997), *DNA Computation*, A Thesis of the University of Warwick, Coventry CV4 7AL.
3. Arroyo, F. Baranda, A., Castellanos, J. and Păun, Gh., (2002), Membrane Computing: The Power of (Rule) Creation, *Journal of Universal Computer Science*, vol. **8**, no. 3, 369-381.
4. Bailey, D. A. (1999), *Java Structure* McGraw-Hill.
5. Blizard, W. (1989), Multiset theory. *Notre Dame Journal of formal logic*. Vol. **30**, 1, 36-66.
6. Blizard, W. (1991), The Development of Multiset Theory, *Modern Logic*, Vol **1**, 319-352.
7. Dershowitz, N. and Manna, Z. (1975), Proving Termination with Multiset Ordering, *Comm. ACM*, Vol. **22**, 465-476.
8. Dornhoff, L. L. and Hohn, F. E. (1978), *Applied Modern Algebra*, Macmillan Publishing Co.
9. Eyisi, J. (2006.), *A Grammar of English, the Student's Companion*. Rex Charles publications.
10. Floyd R.W. (1967), Assigning meanings to Programs, Proc. Symp. in Applied Mathematics, vol. **19**, *American Mathematical Society*, 19-32.
11. Gjorgjieva J. (2007), Turin patterns of growing spheres: the exponential case, *Discrete and continuous dynamic systems supplement* 436 - 445.
12. Harrison L.G. (1994), Kinetic Theory of Living Pattern, *Endeavor*, **18** (4), 130 - 136.
13. Hilton, R. W. (1996), *Managerial Accounting*, MacGraw-Hill/Irwin.
14. <http://en.wikipedia.org/wiki/Multiset>.
15. Ibarra, O. H. and Paun, Gh., (2007), Membrane computing: A General View, *EAS Publishing House*, 83-101.

16. Kemeny J. G. (1957), *Introduction to finite Mathematics*, Prentice-Hall.
17. Krishnamurthy, E. V. (2005), *Rule-based Multiset Programming Paradigm: Applications to Synthetic Biology*, Computer Sciences Laboratory, Australian National University, Canberra, ACT 0200, Australia.
18. Martin-Vide, C., Păun, A. and Păun, Gh. (2002), On the Power of P systems with Symport and Antiport rules. *Journal of Universal Computer Science* **8**, 295-305.
19. Norland, G. B. (1983), *General Biology, 11th Edition*, St Louis, Missouri.
20. Păun, A. and Păun, Gh. (2002), The Power of Communication: P systems with Symport/Antiport. *New Generation Computing* **20**, 295-305.
21. Păun, Gh. (2000), Computing with Membranes, *Journal of Computer and System Sciences*, **61**, 1, 108–143.
22. Păun, Gh. (2002), Membrane computing, An Introduction. *Springer-Verlag*.
23. Rogozhin, V. and Boian, E. (2004), Simulation of Mobile Ambients by tissue P systems with a dynamic network of membranes, *Sevilla*, 1-6.
24. Rogozhin, V. and Boian, E. (2003), Simulation of Mobile Ambients by P Systems. Part 1, *Preproceedings of Workshop on Membrane Computing, WMC* (Alhazov, A., Martin-Vide, C., Păun, Gh. Eds (2003), Report 28/03 of GRLMC, Rovira Virgili University, Tarragona, 404–427).
25. Singh, D. and Peter, C. M. (2011), Multiset-based Tree Model for Membrane Computing, *Computer Science Journal of Moldova*, vol. **19**, no. 1(55), 3-28.
26. Singh, D. (2006), *Multiset Theory: A New Paradigm of Science*, Ahmadu Bello University Press.
27. Singh, D. and Singh, J. N. (2009), An Alternative Proof of the Well-Foundedness of the Nested Multiset Ordering, *International Mathematical Forum*, **4**, no. 8, 359–362.
28. Singh, D., Ibrahim, A. M., Yohanna, T. and Singh, J. N. (2007), An overview of the applications of multiset, *Novi Sad J. Math*, Vol. **37**, No. 2, 73-92.
29. Tailor, D. J. (1996), *Biological Science*, Cambridge.
30. Wildberger, N. J. (2003), A new look at multiset. *School of Mathematics, UNSW Sydney 2052, Australia*.