Università degli Studi di Milano
Dipartimento Scienze dell'Informazione
Dottorato di Ricerca in Informatica (XIII ciclo)
a.a. 2000-2001

PhD Thesis
On the power of classes of splicing systems

ADVISORS

Prof. Giancarlo Mauri

Prof.ssa Clelia De Felice (Univ. di Salerno)

PHD STUDENT

Rosalba Zizza

COORDINATORE:

Prof. Gianni Degli Antoni

Contents

Abstract

"It is hoped that the line of research initiated here will interest biologists who have not previously found reason to be concerned with formal language theory... The results discussed here are certainly only modest formal achievements. It is hoped that future developments will provide much stronger results following the given paradigm and yield wholly new paradigms as well (T. Head, 1987)"

My thesis provides an overview and investigations on Molecular Computing and Splicing Systems. Molecular Computing is a novel branch connecting Computer Science, Mathematics and Biology: computation is carried on by DNA molecules and the operations are performed by using enzymes and other biological processes (the "manifesto" is [1]). Splicing Systems are formal models of a recombinant generative process among molecules which are cut and pasted by a simultaneous action of restriction and ligase enzymes.

Splicing operation is introduced as an operator on strings and systems generating languages are defined, whose generating power is compared with classical results in Formal Languages (the initiator of Splicing Systems Theory is Tom Head [34]). My thesis concerns particular (finite) splicing systems for linear and circular splicing in a perspective which has never been explored so far. Decidability and descriptional complexity questions are answered, various results are provided by using combinatorics on words, classical issues in Formal Languages and novel ideas on automata. Deep connections with Theory of (variable length) Codes have been discovered. Problems and conjectures for future work are also proposed. A flash of biochemical background, thought out of our scope, is provided together with an extensive bibliography.

Foreword

"O tutto o nulla io voglio:
il sole d'oro o l'uragano nero...
e, dallo sguardo tuo, martirio o incanto"
(Antonio Nardacci)

To my Angels...
in Heaven and in Earth

I thank God and all those helped and stimulated me, during the PhD time. I know that no words are enough to express what I feel.

I am sincerely indebted with Clelia De Felice. She introduced me to Theory of Codes, teaching the passion for research, the sense of duty and honesty. During these years, she has continued on attending my studies, as critical proofreader, taking care of me much more than is customary for her role. I thank Giancarlo Mauri for introducing me to Molecular Computing and EMCC group, for his paternal guide, especially during the first times. I have been, and I am, glad to work with Paola Bonizzoni. She has made me welcome in Milano, encouraging and stimulating me during these years, sharing uncertainty and joys.

My work has been supported by DSI of the University of Milano, Disco of the University of Milano-Bicocca and DIA of the University of Salerno, which have given me the technical means during the research for and the writing of this thesis.

There is someone who has made my PhD possible. Dear Marilu', you have opened your home, like a sister, and only thanks to you Milano has been another home for me. I bless God for having met you, for your kindness, your friendship, your nice family, for Paola, who made us met.

Good luck to Brunella, Claudio and Marco, my collegues in PhD course; I thank my catechists and my community, who have prayed for me and have showed me what is important; my sincere friends, a God's present who have encouraged, making me laugh and think, without words... I thank my "big brothers - body guards", theirs wives, sons, my sweet treasues.

I thank Mauro, my blue prince, for his patience, his optimism, his love, for having listened my feelings, sharing these last years with me, together with neverending days in the university, anxiety, joys, successes and failures, for helping me to go on, always thoughtful with me.

Finally I dedicate this work to my parents. They have been patient and braves starting with me the PhD adventure and accepting, with me and more than me, the uncertainty of these years, being happy seeing me happy, worried seeing me tired, but they are always two little angels for me, everywhere I were.

Introduction

Automata Theory has always concerned a deep variety of areas, revealing connections with different topics, such as semigroups algebraic theory, algorithms structure, pattern manipulation. It is perhaps this diversity which has enriched the field to make it one with both interesting applications and significant mathematical problems.

This thesis provides results that can be analyzed from different points of view, since common techniques in Formal Languages together with Combinatorics on Words and properties of Finite Monoids are used.

The first historical reference in Automata Theory is a paper of S. Kleene (1956), in which the equivalence between the notion of regular event-language (recognized or accepted by a finite state automaton) and rational language (a language that belongs to the closure of finite languages with respect to rational operations) was given. Since then, the theory of Finite Automata has developed in order to investigate both languages properties (operations among languages, closure properties) and their interpretations on automata (see [61] for an introduction).

Very recently, Formal Languages Theory has been positively influenced by the introduction of new models of computation, which is contributing to its renewal. In particular, a new mathematical formalism for biological phenomena of recombinant processes has been proposed [34]. Indeed a DNA strand can be viewed as a string over a four letter alphabet and so a natural way to model DNA computation is within the framework of Formal Language Theory.

The DNA (deoxyribonucleic acid) is the storage medium for genetic informations. It is composed by units, called *nucleotides*, distinguished by the chemical group (base) attached to them. The four bases are Adenine, Guanine, Cytosine, and Thymine (denoted respectively A, G, C, T). Each (single stranded) nucleotide is linked end-to-end to form DNA strands, called *oligonucleotides* if there are less than 30 nucleotides. The DNA sequence has a *polarity*, i.e., a sequence of DNA is distinct from its reverse. Thus the two distinct ends of a DNA sequence are known as $5'$ end and the $3'$ end. The four bases are related each other by Watson-Crick complementary rule (A=$\bar{T}$, G=$\bar{C}$) and two complementary (single) strands with opposite polarity will join together, forming a double helix (hybridation process).

Even if the possibility of using DNA and biological tools in order to formulate and solve computer science problems may surprise, the idea behind Molecular Computing is the observation that biological processes and mathematical processes are analogous. In fact, the complex structure of living cells is the result of applying *simple* opeations (recognition, copying, duplication...) to initial informations encoded in a DNA sequence. Moreover, each result of application of a computable function to a given argument can be obtained as the application of combined *simple* functions to the given argument [44]. This similarity is the key of Molecular Computing. Indeed, Biology and Computer Science were two different areas until biologists started to use Computer Science to solve problems in Biology (this is *Computational Biology*) and, few years ago, some computer scientists started to use Biology to approach computer science's problems, underlying idea of *Molecular Computing* (for a quite exhaustive bibliography, see [27]).

The rapidly increasing of interest in Molecular Computing started with Adleman's article [1], where the author showed how to solve (a small instance of) the Hamiltonian Path Problem, that is a NP-complete problem, by manipulating DNA sequences in a lab. A series of papers were soon written, generalizing his approach (for example, see [42] or DNAx Proceendings, $x \in \{1, \ldots, 7\}$). This is commonly called 'wet' molecular computing, to make evident how the computer scientists enter in the laboratory to make algorithms in test tubes.

Let me say that Wet Molecular Computing may enlighten unexpected links with the classical Theory of Codes [3]. Indeed, beyond the real value of Adleman's experiment, his paper opened new interactions among different areas. As Hartmanis observed in [32], in Molecular Computing we are working on a new input and computational model, so we must define a new complexity measure, like the weight of the solution needed for the computation. He showed that repeating the Adleman's experiment on a greater input would mean using a solution whose weight is more than Earth's weight. Then, for the feasibility of molecular algorithms, it is decisive the proposal of alternative techniques during the lab-simulation, trying to avoid PCR, the chief agent of time and weight increasing and hybridization. In fact, PCR (see Appendix) may cause errors in computation, i.e., in a test tube wrong molecules are produced with respect to the problem encoding. Errors and weight are the weak points in the molecular computation feasibility. In the literature there are some different attempts to find good encodings for DNA computing. Giving constraints on the DNA sequences encoding the input, different techniques to avoid errors caused by unwanted hybridizations, a new distance more suitable than Hamming distance allowing the development of encodings to prevent errors are presented in [2, 20, 21, 22, 43] (some suggestions of research have been mentioned in [75]). To avoid unwanted hybridization, another approach, using (variable-length) codes, is initiated in [41], studying deeply the Watson-Crick complementary law. Also circular DNA may be useful. For example, the plasmids (see Appendix) have circular form and autonomously replicate without errors and exponential weight increasing. In fact, a new method of computing, applicable to a wide variety of algorithmic problems, has been very recently introduced by using DNA plasmids: a circular DNA plasmid is specifically constructed step by step, using a scheme of enzymatic treatments [37, 39]. Previously, a circular DNA restriction enzyme is used to encode the transition table of a Turing Machine, through insertion and deletion of fragments into the plasmid [73].

The *molecular algorithm* presented by Adleman has been generalized to other combinatorial problems. An interesting example is the molecular algorithm proposed in [42] for the Road Coloring Problem (RCP), a well-known problem on graphs and up to now, it is unknown whether RCP is decidible (see, for example [60]). A partial answer is given in [62], where the problem has been positively solved with particular additional hypotheses, by using a formulation in Symbolic Dinamics of a theorem on the non commutative factorization of polynomials [68]. This contribution is known to be the best result obtained towards the Factorization Conjecture of Schützenberger [3], a very deeply investigated and open problem in the Theory of Codes. In [42], using the massive parallelism of DNA computation, a polynomial time molecular algorithm, for solving RCP for a given graph, is presented. The procedures involved in solving RCP extensively use the hybridization of DNA.

Thanks to this renewed interest in the Molecular Computing, new models of computation have been suggested and old ones have been rediscovered. Nevertheless, Formal Language Theory already studied a new manner of relating with the study of informational macromolecules. Indeed, in 1987 Tom Head revealed his idea in the paper: *Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviours* [34]. In this work, he introduced the formalism of *splicing system* to observe the generative power of some recombination processes. So, abstract models for the generated languages are defined as a formal counterpart of the DNA recombination under the reaction of restriction and ligase enzymes. Nowadays, this is still one of the growing areas in Molecular Computing, as well as an area of Formal Language Theory. It is commonly called 'dry' molecular computing, to underline how the aim is that of starting from the biological mechanism in order to develop theoretical models of computations. It should be said that splicing systems are only one of systems proposed from DNA to Formal Languages. Indeed in [59] other systems are presented, some of which oriented to double stranded DNA model, such us Sticker Systems, Watson-Crick Automata, Insertion and Deletion Systems ([45, 59]).

Since a DNA strand can be likened to a string consisting of a combination of the four bases A, C, G, T, this means that we have a four letter alphabet $\{A, C, G, T\}$ (the four deoxyribonucleotides) to encode information and so biological operations on DNA strands can be seen as operations on strings. Each DNA molecule is double stranded, but we can treat each such molecules as two strings bounded together in parallel (i.e., our alphabet consists of four symbols $[A/T], [C/G], [G/C], [T/A]$). A nice description of biochemical introduction is given in [64], where a single molecule of double stranded DNA is visualized as a long "ladder", depicted horizontally below, whose steps are the vertical pairs $[A/T], [C/G], [G/C], [T/A]$.

As DNA occurs in both linear and circular form, two kinds of splicing operation were defined to model different biological phenomena, but since the underlying idea is the same, we only summed up the biological interpretation of linear splicing following [34, 44, 38] in this description, without facing here the difficulty of circular representation.

Consider the following two double-stranded DNA molecules:

$$\begin{array}{ll} 5'CCCCCTCGACCCCC3' & 5'AAAAAGCGCAAAAA3' \\ 3'GGGGGAGCTGGGGG5' & 3'TTTTTCGCGTTTTT5' \end{array}$$

Splicing occurs in two steps. First, restriction enzymes (proteins) recognize a specific pattern in a DNA sequence and then they cut the molecule inside the pattern, in a specific point inside the pattern. The site and the shape of the cut ends are specific to each enzyme. Consider the two restriction enzymes TaqI and SciNI, whose recognition sites are, respectively:

$$\begin{array}{ll} 5'TCGA3' & 5'GCGC3' \\ 3'AGCT5' & 3'CGCG5' \end{array}$$

Since DNA is double stranded, the cutting phase may produce *blunt* or *staggered* ends. This is caused by cleavege patterns which are blunt or not. For example, TaqI and SciNI have the following cleavege patterns:

$$\begin{array}{ll} 5'T \quad CGA3' & 5'G \quad CGC3' \\ 3'AGC \quad T5' & 3'CGC \quad G5' \end{array}$$

Other enzymes produce blunt ends: for example, we report below the pattern recognized by AluI and its cleveage pattern

$$\begin{array}{ll} 5'AGCT3' & 5'AG \quad CT3' \\ 3'TCGA5' & 3'TC \quad GA5' \end{array}$$

The effect of $TaqI$'s action on the first molecule and of $SciNI$'s action on the second molecule is to produce the four fragments, with dangling single strands (in [64] and others, the remaining unsplit parts are called the left and the right contexts of the site):

$$\begin{array}{llll} 5'CCCCCT & CGACCCCC3' & 5'AAAAAG & CGCAAAAA3' \\ 3'GGGGGAGC & TGGGGG5' & 3'TTTTTCGC & GTTTTT5' \end{array}$$

It is quite evident that no fragment previously cutted by $AluI$ may be pasted with any of the four fragments written above (no sticky end match). Clearly, if such staggered ends are in the neighborhood of each other, the original two molecules may be reassociated, but this is not very interesting. Anyway, it also possible that two new hybrid molecules will be formed as below:

$$\begin{array}{ll} 5'CCCCCTCGCAAAAA3' & 5'AAAAAGCGACCCCC3' \\ 3'GGGGGAGCGTTTTT5' & 3'TTTTTCGCTGGGGG5' \end{array}$$

Ligase enzymes perform this process, pasting together properly matched fragments, under chemical conditions (related to cleavege patterns structure and complementary lows). Thus, two DNA sequences are cut by two restriction enzymes and the fragments may be recombined so that new sequences are produced

On the other hand, the biological circular splicing occurs in a recombinant mechanism (transposition) between bacteria and plasmids. In this case, restriction enzymes cut two circular DNA sequences while ligase enzymes paste them together, producing a new circular DNA sequence. All details about the steps from the biochemical processes to Formal Languages may be found in [34, 38, 59], and some of them will be surveyed in Chapter **??** and in the Appendix.

After being investigated in a few papers as an operation with double stranded sequences [34, 33, 17], the splicing operation has been considered as an operation with single stranded strings (see Introduction of Part I). Even if different definitions of splicing have been given in the literature (see Chapter **??**), formally, the linear splicing operation concatenates a prefix of one string with a suffix of another string, under some conditions, represented into a (splicing) rule. Splicing operation is applied to two words and clearly two strings may be generated, i.e., splicing performs a changing of prefixes and suffixes. Moreover in [57] it is explicitely noted that only one string is obtained by splicing through a rule, since the other one may be obtained by a simmetric rule. On the contrary, the circular splicing opens two circular strings and pastes them together, obtaining a circular string, under some conditions represented into a (splicing) rule. Thus the set of all possible molecules generated in the test tube is nothing but the language (called *splicing language*) of all possible strings that may be obtained by applying the rules to the strings, starting with an initial language. A splicing system is defined by giving the initial set of strings (or axioms) I and the set of rules R (or triples, according to which definition we use).

Based on this operation, the H-systems were introduced as unconventional models of computation and, more precisely, as a generative device of languages (see [58, 59] for a survey on the different models). An H-system is specified by I and R and, given two families of languages F_1, F_2 in Chomsky hierarchy, $H(F_1, F_2)$ denotes the class of languages generated by H-system where $I \in F_1$ and $R \in F_2$. Indeed, the set of rules, representation of splicing specification (cutting and rejoining sites) by using special symbols, may be seen as a language. The generative power of an H-system, i.e., the type of languages that are generated by it, has been deeply studied and will be surveyed in Chapter **??** as well as different variants of splicing systems (Chapter **??**).

By iterating a finite set of splicing rules starting from a finite (even regular) language, only regular languages may be obtained [17, 66]. Hence the power of finite state automata cannot be increased. However, as soon as we use a regular set of rules (with an additional hypothesis on splicing operation that leads to the definition of Extended Splicing Systems), we reach the power of Turing Machines [56]. The jump from finite automata to Turing Machines has encouraged a lot of works and variants proposal, mainly oriented in how to get rid the infinity in the above-mentioned result, maintaining the universality. Several ideas are explored in the literature, new models are introduced (such as "counting" the number of copies of the used strings, i.e., splicing with multeplicity, permitting or forbidding contexts) and in all cases splicing systems are computationally complete, i.e., every algorithm (effective procedure) can be carried out by a splicing system. Thus, the existence of universal programmable DNA computers based on splicing operation is *theoretically* possible. This may be done by showing that the action of any computable function can be realized by a splicing system [26]. In this sense suitable definition of splicing systems have been formalized, and by using extended splicing systems, it has been proved that every recursive enumerable language can be generated by a splicing system, i.e., the work of any computable function can be carried out by a splicing system, or, in other words, everything is Turing-computable can be computed by this DNA (finite) model of computation (see [38], and bibliography in [44] for more details). Decidability questions have been firstly tackled in [25].

In the Introduction of Part I we will enlighten better the hypotheses under which we work. My thesis concerns H-systems and finite state automaton, precisely regular languages and languages generated by H-systems with both finite initial language and set of rules. We would like to find another characterization of infinite (but regular) languages through the generative mechanism of splicing systems with finite objects, as well as automata. H-systems are extensively presented in Chapter **??** while some digressions on other models are glanced in Chapter **??**.

We have also been involved in the study of (finite) circular splicing power and its relationship with regular languages. The framework of circular splicing is less developed than linear splicing, probably because circular splicing deal with the intricate circular languages, i.e., with languages composed by words in which final ends are pasted together. Each circular word, that can be written in a circle and may be read in clockwise orientation, is in bijection with all the possible words obtained by cutting it in all possible ways (full linearizations of circular words). Circular words have been already treated in Combinatorics on Words through conjugacy relation $\sim$ that is an equivalence relation among words [16, 51]. In short, two words are conjugated if they are the same word read along a circle, i.e., unless a cyclic permutation of their letters. Since no concatenation may be defined (conjugacy relation is not a congruence) splicing operation seems to be like a suitable concatenation of circular words. Indeed, given two equivalence classes (two circular words) it is possible to define the concatenation of two circular words via splicing. Thus, the splicing becomes a particular concatenation (or a particular "circular shuffle" on circular words, constrained by a rule). While the linear splicing loses "pieces" of the input strings, the circular splicing does not (at least in Head and Paun's definitions).

Furthermore, since no characterization of circular regular languages is known, the comprehension of circular splicing power may be another approach to face this problem. Again, different variants of circular splicing have been mentioned by the literature with the aim of proving universality results and are collected in Chapter **??**.

Let me say some final remaks, before sketching an overview of the contents of this thesis. In [57], the authors observed that splicing operation may be viewed as an alternative to the rewriting operation on which the present day computers are based. This is the reason for which the main problem in Splicing theory concerns its computational power and universality results inspired different proposal of splicing systems. Moreover, even if a lot of works remains to be done for its feasible realizations, experimental works on splicing sistems have been successfully done also by the Biomolecular Computing Group at Binghamton [5]. This group proved the vialability of finite Head's model using a single step procedure in lab, making clear the predective capability of splicing [48].

My thesis is purely a language-theoretic oriented investigation and it is organized in two parts. The present introduction is followed two parts dealing in turn with Linear Splicing Systems and Circular Splicing Systems, each handling the state of the art and personal contributions. The splicing operation for both linear and circular strings is formally defined. We will give the state of the art, paying particular attention to the relation existing between splicing systems and regular languages, this being my research area. As to the latter, we will also present contributions already obtained that close open questions. Finally, we will mention very recently discovered links between splicing and codes, referring to a not much known and developed branch in Splicing Systems Theory. Indeed for the first time we approach circular splicing with finite objects and no additional hypothesis. All the results are presented at the beginning of each chapter regarding them.

Part I

Linear Splicing

In the introduction we have described the biological phenomenon underlying splicing systems, observed and formalized for the first time by Head [34]. DNA molecules are cut by restriction enzymes (proteins) leaving two pieces of DNA which are pasted, if some biological conditions on the sticky ends happen, by ligase enzymes, producing new molecules. We have already observed that the translation of this recombinant mechanism is natural in Formal Language Theory: splicing systems are generative mechanisms which start from a given set of strings and a given set of splicing rules in order to generate a language. Indeed, the phenomenon is nothing but a pattern recognition inside a string, cutting the string at a specified point inside the factor and then pasting a prefix of a word with a suffix of another word. The two words are related each other by a rule, representing the combined action of enzymes. As noted in [29], splicing exchanges left and right ends of words. Some considerations are necessary, before giving the overview of the contents of this part, since we choose splicing operation with a few of hypotheses (commonly used in the literature).

Firstly, sites can occur more than once and whenever this happens, the sites are chosen nondeterministically. The rules are applied in a maximal parallel manner: each rule which can be applied should be applied.

Moreover, it is evident that a recombinant mechanism happens with more probability if there is an high number of DNA molecule copies instead of a small number of it. If the number of copies is unbounded or, better, is high, then the probability of DNA molecules"meeting" is higher, since no topology is defined in the test tube. We assume that if we splice x, y obtaining w, w', we may use again x, y as terms of splicing, that is the strings are not consumed by splicing (using notation in Chapter **??**, we work without multeplicity). Similarly, there is no restriction on (the number of the copies of) the newly obtained w, w'. In [25] a consideration is esplicitely done about the assumption that all of the initial strings or DNA strands are available in unbounded quantity. There are phisical reasons to consider this assumption reasonable for modeling purposes. The number of molecules in a few grams of most compounds is on the order of 10^{22} or more. In [25] is shown that this assumption is computationally needed since otherwise there are specifically languages for which some problems are undecidable (such as the membership problem). At the opposite site, others consider this assumption not so realistic.

We consider strings, not double stranded structure, since Formal Languages Theory does not yet deal directly with "linked pairs" of strings. This is the reason why Head considered splicing on strings. Recently, some papers have intestigated bidimensional languages (see, for example [30]). Previously, splicing operation has been approached by using the bidimensional structure of DNA through dominoes languages, already defined in other contexts [17]. Even if this approach is more realistic for modelling splicing behaviour, it seems to be quite difficult. As observed in [34, 57], due to complementary lows ($A - T$, $C - G$), there is no loss of information if the double stranded DNA molecule is identified with one single stranded DNA sequence. Furthermore, we do not restrict the alphabet to four letters. Indeed it is well-known that the symbols of an arbitrary alphabet can be codified on two letters preserving the "place" of a language in the Chomsky hierarchy.

Finally, as done in a series of papers, the splicing rules are considered symmetric, i.e., starting from two words and by using a rule, two words are generated. Clearly, the simmetric case corresponds to considering each rule together with its simmetric rule. In [55], for example, symmetry is not automatically imposed, i.e., only one word is generated at each step.

Let us describe our work on linear splicing. After recalling some basics on ormal languages (Section **??**), we give the three definitions of splicing, with a lot of old and new examples on $H-$systems (Sections **??** and **??**). Thus, in Section **??** we survey some variants of linear splicing models and their power. We focus our investigation on regular languages that can be gerated by linear splicing by using both finite initial language and finite set of rules, since until now it is unknown a characterization of this subclass of regular languages (finite) splicing generated. In Chapter **??** we show a decidability property with respect to this problem. Still considering finite splicing, we compare the three splicing systems giving some considerations and inclusion results (Chapter **??**).

Part II

Circular splicing

Circular splicing has been defined in different ways, but only for single stranded DNA molecules. Let us focus on the behaviour of circular splicing. A restriction enzyme recognizes a pattern inside a circular DNA molecule and then it opens the molecule inside the pattern, i.e., it linearizes a circular DNA molecule at a specific point (linearization). Now two different processes may happen: either ligase enzyme pastes together the (linear) fragments and circularizes the newly obtained molecule (Head and Paun's definition), or before the pasting action, ligase enzyme substitutes the recognized pattern (Pixton's definition). Our results and investigations deal with all these definitions.

As in the linear case, we suppose to manage a nonbounded number of copies of a molecule, or, in other words, before performing splicing, we preliminary copy the terms of splicing.

Moreover, we will only deal with finite splicing systems, i.e., the (circular) initial language and the set of rules are both finite sets. Thus, if a language is "generated", we mean that it is generated by a finite splicing system. Moreover, even if the literature consider hypotheses on splicing systems, we do not take them into account.

In Nature circular splicing occurs in a recombinant mechanism (transposition) between bacterial chromosomes and F plasmids in *E.coli*. In [74] we read that a molecular biological observation reports that *lambda phage* DNA can easily change its form from linear to circular and vice versa. When *lambda phage* DNA is transposed in a bacterial DNA like *E.coli* DNA, it forms a circular that is incorporated into a host DNA by splicing (crossing over). Moreover, the replication process is possible only if DNA has a circular form in the host cell. In the Introduction we have already mentioned the possibility of using circular DNA to perform Adleman's experiment, since plasmids autonomously replicate without errors or exponential weight increase, which are weak points in Adleman's approach [1]. Indeed, in [18] a laboratory experiment implementing a new model for DNA biomolecular computing, previously introduced in [41], involves circular and linear molecules with insertions and deletions operations. These systems are capable of universal computation.

As in the linear case, we have investigated the relationship between finite circular splicing and circular regular languages. This aim leads us to approach circular languages, or, analogously, as we will see later, with formal languages closed under conjugacy relation, a well-known equivalence relation between words (Chapter **??**). All the definitions about circular splicing (Chapter **??**) are followed by chapters containing our results toward a characterization of circular regular languages generated by finite splicing. For the sake of clarity, we have preferred to split in different chapters the presentation of the classes of languages proved to be generated.

A comparison among the three finite splicing systems is provided in Chapter **??**. We focus our attention firstly on Paun's definition, we define a class of regular languages (*star languages*) and we prove that for a subclass of them the corresponding circular languages are Paun generated. Restricting ourselves to regular languages on *one-letter alphabet*, we prove that we can decide whether such a regular language is Paun generated, by using a characterization we have found. We extend this characterization, by considering *uniform languages*. *Cyclic* and *weak cyclic languages*, which will be introduced in Chapter **??**, show that this result does not hold when we increase the size of alphabets, even if we restrict ourselves to regular languages. We conclude our studies on (circular) splicing with the definition of a class of circular regular languages which we prove to be Pixton generated. As a conclusion, no regular circular language generated by finite splicing was found not to belong to this class.

Conclusions

In this thesis we have tried to provide a quite exhaustive overview and some new investigations on Molecular Computing and Splicing Systems. We have approached linear and circular (iterated) splicing systems in depth, and then restricted to the finite case. This has allowed us to compare splicing systems and automata and find a new model able to generate infinite (but regular) languages starting with finite languages. We have provided comparisons of different definitions of splicing and several examples of regular languages, generated or not. The reseach has succeded in defining new families of regular languages by means of automata or combinatorial properties and proving them to be splicing languages. Connections between splicing and codes have also been given. In the case of one-letter alphabet it has been proved to be decidable whether or not a regular language is generated by circular splicing. A circular regular language not generated by splicing has been presented, showing that, at least in the simplest formulation, circular splicing cannot be used to characterize circular regularity, which is still an open problem. The present study highlights that, up to now, no circular regular splicing language exist which is not contained in one of the classes we have defined.

In order to characterize regular languages generated by linear splicing, we have used syntactic monoid, so stating a sufficient condition for being linear splicing generated. We argue that this approach, we are already using, can result effective for bounding the length of splicing rules and to complete the characterization, which is a very difficult problem that we have been studying.

Bibliography

[1] L. M. Adleman (1994), Molecular computation of solutions to combinatorial problems, *Science* **226**, pp. 1021 – 1024.

[2] E.B. Baum (1996), DNA sequences useful for computation, *Proc. of 2nd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 122 – 126.

[3] J. Berstel, D. Perrin (1985), *Theory of codes*, Academic Press, New York.

[4] J. Berstel, A. Restivo (1981), Codes et sousmonoides fermes par conjugaison, *Sem. LITP* **81-45**, 10 pages.

[5] Biomolecular Computing Group at Binghampton, http://www.math.binghamton.edu/dennis/DARPA/index.html

[6] P. Bonizzoni (1998), *private communication.*

[7] P. Bonizzoni, C. Ferretti, G. Mauri (1998), Splicing systems with marked rules, *Romanian J. of Information Science and Technology*, Vol. 1, No. 4, pp. 295 – 306.

[8] P. Bonizzoni, R. Zizza (2000), A characterization of regular splicing languages, *Tech. Rep.* **254-00**, University of Milan, 16 pages.

[9] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), Developments on circular splicing, presented at WORDS01, Palermo (Italy).

[10] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), On the power of circular splicing, *in preparation.*

[11] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001), DNA and circular splicing, *in* "DNA 2000", Lecture Notes in Computer Science **2054**, pp. 117 – 129, Springer-Verlag.

[12] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (2001) Circular splicing and regularity, *submitted.*

[13] P. Bonizzoni, C. De Felice, G. Mauri, R. Zizza (1999), Linear and circular splicing, presented at WORDS99, Rouen (France).

[14] P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza (2000), Separating some splicing models, presented at Grammar Systems 2000, Austria.

[15] P. Bonizzoni, C. Ferretti, G. Mauri, R. Zizza (2001), Separating some splicing models, *Information Processing Letters* **76 (6)**, pp. 255 – 259.

[16] C. Choffrut, J. Karhumaki (1996), Combinatorics on Words, *in* "Handbook of Formal Languages" (G. Rozenberg & A. Salomaa, Eds.), Vol. 1, pp. 329 – 438, Springer-Verlag.

[17] K. Culik, T. Harju (1991), Splicing semigroups of dominoes and DNA, *Discrete Applied Mathematics*, **31**, pp. 261 – 277.

[18] M. Daley, L. Kari, G. Gloor, R. Sirimoney (1999) Circular contextual insertions/deletions with applications to biomolecular computing, *Proc. of SPIRE'99, 6th Int. Symp. on String Processing and Information Retrieval*, Concun, Mexico, IEEE Comp. Science Press, Los Alamos, California, pp. 47 – 54.

[19] J.H.M. Dassen (1996), Molecular Computation and Splicing Systems, *MSc. Thesis*, Leiden Univ.

[20] R. Deaton, R.C. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens (1996), Good encoding for DNA-based solutions to combinatorial problems, *Proc. of* 2*nd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 131 – 140.

[21] R. Deaton, R.C. Murphy, M. Garzon, D.R. Franceschetti, S.E. Stevens (1997), On the encoding problem for DNA Computing, *Proc. of* 3*rd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 230 – 237.

[22] R. Deaton, R.C. Murphy, M. Garzon, D.R. Franceschetti, J.A. Rose, S.E. Stevens (1997), The effect of uniform melting temperatures on the efficiency of DNA Computing, *Proc. of* 3*rd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 35 – 42.

[23] C. De Felice (1996), An application of Hajós factorizations to variable-length codes, *Theoretical Computer Science* **164**, pp. 223 – 252.

[24] A. De Luca, A. Restivo (1980), A characterization of strictly locally testable languages and its application to subsemigroups, *Information and Control*, **44**, pp. 300 – 319.

[25] K.L. Denninghoff, R.W. Gatterdam (1989), On the undecidability of splicing systems, *Intern. J. Computer Math.*, Vol. 27, pp. 133 – 145.

[26] R. Freund, L. Kari, G. Paun (1995), DNA Computing based on splicing: the existence of universal computers, *Journal of the ACM*, to appear.

[27] P. Frisco, *A bibliography of molecular computation and splicing systems*, http://www.wi.LeidenUniv.nl/~pier/dna.html

[28] L. Fuchs (1960), *Abelian groups*, Pergamon Press, Oxford, London, New York and Paris.

[29] R. W. Gatterdam (1992), Algorithms for splicing systems, *SIAM Journal of Computing*, **21:3**, pp. 507 – 520.

[30] D. Giammarresi, A. Restivo (1996), Two-dimensional Languages, *in* "Handbook of Formal Languages" (G. Rozenberg & A. Salomaa, Eds.), Vol. 3, pp. 215 – 267, Springer-Verlag.

[31] E. Goode, D. Pixton (2001), Semi-simple splicing systems, *in* "Where Mathematics, Computer Science, Linguistics and Biology Meet" (C. Martin-Vide & V. Mitrana, Eds.), pp. 343 – 357, Kluwer Academic Publ., Dordrecht.

[32] J. Hartmanis (1995), On the Weight of Computations, *Bull. of EATCS* **55**, pp. 136 – 138.

[33] T. Head (1992), Splicing schemes and DNA, *in* "Lindenmayer Systems: Impacts on Theoretical Computer Science and Developmental Biology", pp. 371 – 383, Springer-Verlag, Berlin.

[34] T. Head (1987), Formal Language Theory and DNA: an analysis of the generative capacity of specific recombinant behaviours, *Bull. Math. Biol.* **49**, pp. 737 – 759.

[35] T. Head (1998), Splicing languages generated with one sided context, *in* "Computing with Bio-molecules: Theory and Experiments" (Gh. Paun, Ed.), Springer-Verlag, Singapore.

[36] T. Head (1998), Splicing representations of strictly locally testable languages, *Discrete Applied Math.*, **87**, pp. 139 – 147.

[37] T. Head (1999), Circular suggestions for DNA Computing, *in* "Pattern Formation in Biology, Vision and Dynamics", (A. Carbone, M. Gromov, P. Pruzinkiewicz, Eds.), World Scientific, Singapore and London, to appear.

[38] T. Head, Gh. Paun, D. Pixton (1996), Language theory and molecular genetics: generative mechanisms suggested by DNA re-

combination, *in* "Handbook of Formal Languages" (G. Rozenberg & A. Salomaa, Eds.), Vol. 2, pp. 295 – 360, Springer-Verlag.

[39] T. Head, G. Rozenberg, R. Bladergroen, C. Breek, P. Lommerse, H. Spaink (2000), Computing with DNA by operating on plasmids, *BioSystems* **57**, pp. 87 – 93.

[40] J.E. Hopcroft, J.D. Ullman (1979), *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Mass.

[41] S. Hussini, L.Kari, S. Koustantinidis (2001) Coding Properties of DNA languages, *Proc. of DNA7.*

[42] N. Jonoska, S. Karl (1996), A molecular computation of the road coloring problem, *Proc. of* 2*nd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 148 – 158.

[43] P.D. Kaplan, G. Cecchi, A. Libchaber (1996), DNA based molecular computation: template-template interactions in PCR, *Proc. of* 2*nd Annual Meeting on DNA Based Computers* (DIMACS Workshop), pp. 159 – 171.

[44] L. Kari (1999), DNA computing: an arrival of biological mathematics, *The Mathematical Intelligencer* **19:2**, pp. 9 – 22.

[45] L. Kari, Gh. Paun, G. Thierren (1997), At the crossroads of DNA Computing and Formal Languages: Characterizing Recursive Enumerable Languages using insertion and deletion systems, *in* Proc. of 3rd DIMACS Int. Workshop in DNA Based Computers, pp. 318 – 333.

[46] S.M. Kim (1997), An Algorithm for Identyfying Spliced Languages, *Proc. of Cocoon97*, Lecture Notes in Computer Science **1276**, pp. 403 – 411.

[47] S.M. Kim (1997), Computational modeling for genetic splicing systems, *SIAM Journal of Computing* **26**, pp. 1284 – 1309.

[48] E. Laun, K. Reddy (1999), Wet Splicing Systems, *Proc. of 3rd Annual Meeting on DNA Based Computers*, (H. Rubin and D. Woods, Eds.) DIMACS Series in Discrete Mathematics and Theoretical Computer Science, **48**, AMS, Providence RI.

[49] Life Science BioTech Dictionary, http://biotech.icmb.utexas.edu /search/dict-search.html

[50] Cellular Biology, http://www.mblab.gla.ac.uk/ julian/Dict.html

[51] M. Lothaire (1983), *Combinatorics on Words*, Encyclopedia of Math. and its Appl., Addison Wesley Publishing Company.

[52] R. McNaughton, S. Papert (1971), *Counter-Free Automata*, MIT Press, Cambridge, Mass.

[53] A. Mateescu, Gh. Paun, G. Rozenberg, A. Salomaa (1998), Simple splicing systems, *Discr. Appl. Math.* **84**, pp. 145 – 163.

[54] On-line Medical Dict., http://www.graylab.ac.uk/omd/index.html

[55] Gh. Paun (1996), On the splicing operation, *Discrete Applied Math.* **70**, pp. 57 – 79.

[56] Gh. Paun (1996), Regular extended H systems are computationally universal, *Journal of Aut., Lang., Combinatorics.* **1:1**, pp. 27 – 36.

[57] Gh. Paun, A. Salomaa (1996), DNA computing based on the splicing operation, *Math. Japonica*, **43:3**, pp. 607 – 632

[58] Gh. Paun, G. Rozenberg, A. Salomaa (1996), Computing by splicing, *Theoretical Computer Science* **168:2**, pp. 321 – 336.

[59] G. Paun, G. Rozenberg, A. Salomaa (1998), *DNA computing, New Computing Paradigms*, Springer-Verlag.

[60] D. Perrin (1995), Symbolics Dynamics and Finite Automata, *Proc. of MFCS95*, Lecture Notes in Computer Science **969**.

[61] D. Perrin, Finite Automata, *in* "Handbook of Theoretical Computer Science" (J. Van Leeuwen, Ed.), Vol. B, pp. 1 – 57, Elsevier.

[62] D. Perrin, M. P. Schützenberger (1992), Synchronizing prefix codes and automata and the road coloring problem, *in* "Symbolic Dynamics and its Applications", (P. Walters editor), Vol. 135 of Contemporary Mathematics, pp. 295 – 318, Amer. Math. Soc.

[63] D. Perrin, M. P. Schützenberger (1976), Codes et sous-monoides possedant des mots neutres, *in* "Theoretical Computer Science", Lecture Notes in Computer Science **48**, pp. 270 – 281.

[64] D. Pixton (2000), Splicing in abstract families of languages, *Theoret. Comp. Science* **234**, pp. 135 – 166.

[65] D. Pixton (1996), Linear and Circular Splicing Systems, *in* "Proc. of 1st Int. Symp. on Int. in Neural and Biological Systems", pp. 181 – 188.

[66] D. Pixton (1996), Regularity of splicing languages, *Discrete Applied Math.* **69**, pp. 101 – 124.

[67] C. Reis, G. Thierren (1979), Reflective star languages and codes, *Information and Control* **42**, pp. 1 – 9.

[68] C. Reutenauer (1985), Non Communtative Factorization of variable length codes, *J. Pure and Applied Algebra* **36**, pp. 157 – 186.

[69] M.-P. Schützenberger (1975), Sur certains operations de fermeture dans les langages rationnels, *Symposia Math.* **15**, pp. 245 – 253.

[70] R. Siromoney (2000), Distributed Circular Systems, presented at Grammar Systems 2000, Austria.

[71] R. Siromoney, K.G. Subramanian, A. Dare (1992), Circular DNA and Splicing Systems, *in* "Proc. of ICPIA", Lecture Notes in Computer Science **654**, pp. 260 – 273, Springer-Verlag.

[72] D. G. Thomas, K. G. Subramanian, V. R. Dare, T. Robinson, B. Le Saec (2000), Learning algorithms for subclasses of circular DNA molecules, Poster session in DNA 2000, Leiden 2000.

[73] P. Wilhelm, K. Rothemund(1996), A DNA restriction enzyme implementation of Turing Machines, *in* "DNA Based Computers", Vol. 27 of *DIMACS* Series in Discrete Mathematics and Theoretical Computer Science, AMS.

[74] T. Yokomory, S. Kobayaski, C. Ferretti (1999), On the power of circular splicing systems and DNA computability, *Proc. of IEEE Int. Conf. Evol. Comp.* ICEC99.

[75] R. Zizza (1999), On the power of classes of splicing systems, *Thesis Proposal.*

Appendix: A flash of biochemical background

For the sake of completeness, we collect here a bit of basically definitions on biochemistry, only with respect to what we have used in the thesis. The material of this section is taken from [54, 49, 50, 19].

- **DNA (deoxyribonucleic acid).** It can abstractly be viewed as a string over a four-letter alphabet (A, C, G, T). It is a large molecule (polymer) composed by repeated four smaller units (monomers), known as nucleotides, consisting of a nitrogenous base (Adenine, Guanine, Cytosine, Thymine), a phosphate molecule and a sugar molecule, that link together in a linear structure forming DNA's bachbone. This has two distint ends, giving a polarity. DNA in a cell is normally double stranded, i.e., it forms a double helix of two intertwined strands, binded by hydrogen bounds (A-T and G-C are Watson-Crick complementary).

- **Circular DNA.** DNA arranged as a closed circle. Characteristic of prokaryotes, but also found in mitochondria and some viral genomes.

- **Gene.** Originally defined as the physical unit of heredity, it is probably best defined as the unit of the inheritance that occupies a specific locus on a chromosome (self-replicating genetic structures of cells containing the cellular DNA that bears in its nucleotide sequence the linear array of genome). They are formed from DNA.

- **DNA replication** The process whereby a copy of a DNA molecule is made and thus the genetic informations it contains is duplicated. The parental double stranded DNA molecule is replicated semi-conservatively, i.e., each copy contains one of the original strands paired with a newly sybthesized strand that is complementary in terms of AT and GC base pairing. Though in this sense conceptually simple, mechanistically a complex process involving a number of enzymes.

- **Polymerase Chain Reaction (PCR).** The first pratical system for *in vitro* amplification of DNA, and as such one of the most important recent developments in molecular biology. It is a reaction in which double-stranded sequences of DNA (template) are replicated, strarting from a short sequence that marks the piece to be replicated (primer). Because the copies of templates can function as templates themselves, repeated PCR can be used to exponentially multiply the template. It is realized by a series of temperature cycles.

- **Hybridization.** DNA sequences may be joined when they begin or end in short overhanging complementary single stranded DNA sequences. The forming of hydrogen bonds between such sequences is known as hybridization. It may happen even if the bases are not all perfectly complementary.

- **DNA Ligase.** Enzymes involved in DNA replication. Ligase enzymes are proteins that paste together pieces of DNA whose sticky ends coincide.

- **Restriction enzymes.** Class of bacterial enzymes (endonucleases) that are capable of cutting DNA at a specific site (base sequence) producing strands with scattered or sticky ends. These enzymes, combined with the previous ones are the inspiration for the splicing operator.

- **Recombinant DNA.** Spliced DNA formed from two or more different sources that have been cleaved by restriction enzymes and joined by ligases.

- **Plasmids.** Autonomously replicating circular DNA molecules, distinct from the normal bacterial genome and nonessential for cell survival under nonselective conditions. Some plasmids are capable of integrating into the host genome. A number of artificially constructed plasmids are used as cloning vectors.

- **Restriction Fragment length polymorphism (RFLP).** Tecnique, also known as DNA fingerprinting, that allows familiar relationship to be estabilished by comparing the characteristic polymorphic patterns that are obtained when certain regions of genomic DNA are amplified (typically by PCR) and cut with certain restriction enzymes. In principle, an individual can be identified unambiguously by RFLP. Similarly if a polymorphism can be identified close to the locus of a genetic defect, it provides a valuable marker for tracing the inheritance of the defect.

Splicing in cartoons

Be careful! The last pages of my thesis are devoted to a nice side effect I have received working on splicing.

I have presented the results obtained in my PhD Thesis step by step, in some exciting conferences, like Words99, DNA6, Words01, and some meetings regarding Italian Murst Project. As youngest in the working group, I have prepared the talks and, trying to explain splicing operation to a not specialized audience, I have used to give an informal description by depicting some "cartoons". I am very happy to include some of them in my PhD Thesis, as a reminder of those marvellous days. Someone told me that, thanks to them, I was able to convince not theoricians that splicing theory and formal language theory may be amusing and coloured!