Università degli Studi di Udine
Dipartimento di Matematica e Informatica
Dottorato di Ricerca in Informatica

Ph.D. Thesis

Formal Methods of Computer Science for Biology

Candidate:
Elisabetta De Maria

Supervisor:
Angelo Montanari

February 9, 2009

Author's e-mail: elisabetta.demaria@dimi.uniud.it

Author's address:

Dipartimento di Matematica e Informatica
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine
Italia

Courage is going from failure to failure without losing enthusiasm.
Winston Churchill.

Abstract

Formal methods of Computer Science are more and more applied to the study of biological issues. In literature, they are extensively used to model and simulate biological systems and to express formal properties of such systems [110, 105, 69, 19, 56, 75, 85, 22, 46, 106, 66, 10]. In this thesis we show how model checking, game theory, and the combination of simulation and model checking techniques can be exploited to develop effective solutions to three relevant classes of biological problems: the protein structure prediction and protein folding problems, the sequence comparison problems, and the problems of modeling and checking metabolic pathways. As for the protein structure prediction and protein folding problems, we model the space of protein conformations as a finite transition system whose states are all the possible conformations of a protein and whose transitions represent admissible transformations of conformations. Then, we show how meaningful properties of such a transition system can be expressed in temporal logic and we use the model checking machinery to algorithmically check them [35]. As for the sequence comparison problems, we model biological sequences as labeled structures with a "limited order" relation and we define a criterion which allows one to measure their degree of similarity in terms of the number of remaining rounds in an Ehrenfeucht-Fraïssé game played on such structures [36]. Finally, as for the last issue is concerned, we model the mammalian cell cycle and the behaviour of an anti-cancerogenic medicament and of a tumor-suppressor protein by means of a rule-based language [19], we assemble such models into a consensus model, and we exploit model checking to verify some interesting properties of the resulting model. Furthermore, we take advantage of a parameter learning procedure to infer parameter values which make specifications true.

Acknowledgments

To be added before printing the final version.

Contents

List of Figures

List of Tables

Introduction

This thesis aims at substantiating the claim that formal methods of Computer Science may be beneficial to the development of effective solutions to relevant biological problems. We concentrate our attention on three paradigmatic cases: the protein folding/protein structure prediction problem, the sequence comparison problem, and the problem of modeling and checking metabolic pathways. To attack the former problem we exploit on-the-fly model checking techniques; to study the second one we take advantage of game theory; to tackle the last one we use the combination of simulation and model checking techniques.

In literature, the applications of formal methods of Computer Science to biology are numerous and variegated. The main one probably consists in the use of formalisms that allow one to model and simulate biological processes at the computer: this is the case of process calculi [110, 105], rule-based languages [69, 19], and graphical notations [56, 75, 85] for biological networks.

Another main thread is rooted in the use of model checking, technique that, as already said, we massively exploit in this thesis. Model checking is the process of algorithmically verifying whether a given structure is a model for a given logical formula [27]. In computer science, model checking is extensively used to check whether a transition system satisfies some expected properties encoded as temporal logic formulae. When applied to real-world complex systems, such as those that come into play in systems biology, model checking faces the so-called state explosion problem, that is, the combinatorial blow up of the state space. Different approaches to such a problem have been proposed in the literature, including symbolic model checking, that takes advantage of an implicit representation of the states of the transition system [92], and on-the-fly model checking, that, whenever possible, avoids to construct the entire state space of the system [29, 49].

There exist various attempts to apply model checking techniques to biological systems. The use of model checking in systems biology is advocated by Chabrier and Fages in [22], where the authors point out the advantages of such a technique over simulation in validating and querying formal models of biological processes. They also describe some experimental results they obtained by using the symbolic model checker NuSMV and the constraint-based model checker DMC. In a subsequent paper, Fages et al. present a programming language environment, called BIOCHAM, that allows one to model biochemical systems, to check their temporal properties, and to simulate their behavior [46]. An alternative framework, called Simpathica, that supports model building and model checking for biochemical processes is proposed in [4]. Basically, it is a computational tool that, given a representation of a biochemical reaction as a set of differential equations, provides its evolution over time with an automaton-based semantics. In addition, it allows one to qualitatively reason about the properties of such a process using a propositional temporal logic. In [106], Piazza et al. propose semi-algebraic hybrid systems as a natural framework for modeling biochemical networks, taking advantage of the decidability of the model-checking problem for TCTL (Timed Computation Tree Logic) over this large class of systems. In [66], Heath et al. apply the probabilistic model checker PRISM to the study of a complex biological system, namely, the Fibroblast Growth Factor (FGF) signalling pathway. Finally, in order to analyze genetic regulatory networks with parameter uncertainty, Batt et al. develop a modeling framework based on differential equations, where the values of uncertain parameters are given in terms of intervals and dynamical properties of the networks are expressed in temporal logic [10]. Model checking techniques are then exploited to prove that, for every possible parameter value, the modeled systems satisfy the expected properties and to find valid subsets of a given set of parameter values. Such an approach is at the basis of a tool for robust verification of gene networks, called RoVerGeNe.

We conclude the roundup on the applications of formal technique of Computer Science to biology by pointing out that Montanari et al. recently proposed in [95] the novel use Ehrenfeucht-Fraïssé games (*EF-games* for short) [52, 40] for biological sequence comparison, where left-to-right matches often turn out to be inadequate. EF-games are two-player combinatorial games on pairs of structures. The first player (Spoiler) aims at proving that the two structures are different, while the second one wants to show that they are equivalent. The two structures form the playground and at each round players pick elements from them.

EF-games are mainly used to measure the expressive power of a logic. In fact, the rules of such games usually have a logic counterpart, so that the existence of a winning strategy for one of the players corresponds to the ability of formulae of a suitable logic to distinguish the structures used in the game. But EF-games can be also exploited to measure the degree of similarity of structures: given two structures which are not isomorphic, more rounds Spoiler takes to win, more similar the two structures are. Furthermore, the ability to exhibit a winning strategy for the players can lead insight into the similarities and discrepancies of the two structures. It is often complex to provide such characterizations of the playground. In [104], it is proved that the problem of determining the existence of a winning strategy for Duplicator in an m-round EF-game on two finite structures over any vocabulary containing at least a binary and a ternary relation is PSPACE-complete. In [95], the authors model (biological) sequences as structures with a successor relation and a finite number of unary predicates and give a necessary and sufficient condition for Duplicator to win games played on such structures. Such a condition gives rise to a polynomial algorithm to measure the degree of similarity of two structures.

As anticipated before, in this thesis we propose novel approaches to the protein folding/protein structure prediction problem, the sequence comparison problem, and the problem of modeling and checking metabolic pathways. As far as the first problem is concerned, we show how model checking techniques can be successfully exploited to extrapolate meaningful properties of protein conformations and of their relationships to be used to efficiently solve it [35]. Given the molecular composition of a protein, that is, the list of amino acids that defines its *primary structure*, the *protein structure prediction* problem consists in determining the three-dimensional shape (*tertiary structure* or *conformation*) that the protein assumes in normal conditions in biological environments, while the *protein folding* problem consists in predicting the whole folding process that gives rise to such a tertiary structure [28]. To solve both problems, it is crucial to determine the conformations of the amino acid sequences in the three-dimensional space with minimum free energy. According to a commonly accepted belief, a state with minimum energy indeed represents the natural shape, a.k.a. the *native conformation*, of a protein. Although a *definitive energy function* has not been yet identified, the energy of a conformation is usually modeled by means of suitable energy functions, which express the energy level in terms of the interactions between pairs of amino acids [11].

In this thesis, we model the space of protein conformations as a finite transition system whose states are all the possible conformations of a protein and whose transitions represent admissible transformations of conformations. Then, we show how to express relevant properties of such a transition system in temporal logic. Finally, we use the model checking machinery to algorithmically check them. To cope with the combinatorial blow up of the state space, we resort to on-the-fly model checking. We develop an on-the-fly model checker in SICStus Prolog and we experiment it on sample proteins and properties, obtaining quite promising results. It is worth noting that while the solution we propose only relies on information about the primary structure of the protein (*ab initio* calculation), various methods for structure prediction, such as homology modeling and protein threading, make additional assumptions, based on the identification between the input protein and one or more known proteins (obviously, this cannot be done whenever there are no available data for the input protein). This is the case, for instance, with Rosetta [94]. It is thus evident that comparisons between our results and those obtained by such methods are inappropriate.

As for the problem of comparing biological strings, which consists in detecting their degree of similarity to ascertain whether they are evolutionary related or not, we further investigate the possible application of the EF-method along the direction outlined in [95]. This method shows more flexibility than classical algorithms for string alignment [39, 60], which arrange sequences to identify regions of similarity. To model biological sequences, Montanari el. al have considered structures provided with the successor relation s. As an alternative, one may think of replacing s

with the linear order relation $<$; however, $<$ does not preserve locality and thus phenomena such as inversions (some parts may appear in a different order in homologous sequences) cannot be dealt with. In this thesis, we assume a more general point of view by considering a limited order relation that lies in between s and $<$. We give necessary and sufficient conditions for Spoiler/Duplicator to win games played on finite structures with a limited order relation and a finite number of unary predicates. On the basis of such conditions, we outline a polynomial algorithm to compute the "remoteness" of a game, that is, the number of remaining rounds, which can be used to measure the degree of similarity of two strings. Furthermore, we determine the optimal strategies/moves for both players, i.e., the strategy that leads the winning player to win as soon as possible and the loser one to lose as late as possible [36]. From a technical point of view, suitable abstractions are needed for describing winning strategies for EF-games on structures with a limited order relation, e.g., the distinction between rigid and elastic intervals, that make the proofs much more involved than in the case of s.

As far as the the last issue is concerned, we focus our attention on irinotecan, a medicament which shows significant antitumor activity against a variety of solid tumors, including lung, colorectal, and cervical cancers. Scientists are currently trying to optimize its therapy in order to understand how to limit its toxicity and to increase its effectiveness. To study its influence on cellular proliferation, we analyze suitable models of irinotecan, mammalian cell cycle, and p53, a tumor-suppressor protein. We encode such models in the rule-based language of the biochemical abstract machine BIOCHAM [46, 19], which is similar to (and compatible with) the Systems Biology Markup Language [69], a formalism that uses a syntax of reaction rules with kinetic expressions to define reaction models, nowadays supported by a majority of modeling tools [67, 121]. Afterwards, we assemble such models into a consensus model, we express some relevant biological properties of the resulting model in temporal logic, and we automatically check their satisfaction thanks to the use of model-checking. Furthermore, we exploit a procedure for parameter learning given at Biocham user's disposal to find parameter values such that a given specification is satisfied, detecting in such a way interesting aspects of the cellular answer to irinotecan injections.

The thesis is organized as follows. In Chapter 1 we provide the needed biological background. In Chapter 2 we give a detailed description of the protein folding and the protein structure prediction problem. Chapter 3 is devoted to our approach to the latter problems. In Chapter 4 we provide the state of the art relative to the string alignment problem. Our contribution to the string comparison problem is outlined in Chapter 5. A roundup of the most commonly used representation systems for biological networks, along with a description of mammalian cell cycle, irinotecan, and protein p53, are the subjects of Chapter 6. Finally, in Chapter 7 we present our approach to the investigation of the influence of irinotecan on cellular proliferation.

1 Biological background in a nutshell

In this introductory chapter we focus on some fundamental biological notions, such as the ones of DNA, gene, chromosome, and protein, and on some relevant biological computational issues, such as the gene finding problem, the sequence alignment problem, the genome rearranging problem, and the protein structure prediction problem. Classical references are the books *Genomes* [16] and *Molecular biology of the cell* [2].

1.1 Biological foundations

Life is specified by genomes: every organism possesses a genome that contains the biological information needed to construct and maintain a living example of that organism. Most genomes, including those for all cellular lifeforms, are made of DNA (deoxyribonucleic acid) but a few viruses have RNA (ribonucleic acid) genomes.

1.1.1 DNA

DNA is a polymeric molecule made up of linear, unbranched chains of monomeric subunits called nucleotides. Each nucleotide has three parts: a sugar, a phosphate group, and a base. The sugar is 2'-deoxyribose and the bases are Adenine (A), Guanine (G), Cytosine (C), and Thymine (T). As shown in Figure 1.1, the structure of DNA is described as *double helix*, where each helix is a chain of nucleotides held together by phosphodiester bonds. The two helices are held together by hydrogen bonds between the base components of the nucleotides in the two strands. Each base pair consists of one *purine* base (A or G) and one *pyrimidine* base (C or T). The base-pairing rules are that A base-pairs with T, and G base-pairs with C. The two DNA molecules in a double helix therefore have complementary sequences.

The double helix could be imagined as a zipper that unzips, starting at one end; the unwinding of the two strands exposes single bases on each strand. Since the pairing requirements imposed by the DNA structure are strict, each exposed base can only pair with its complementary base. Due to this base complementarity, each of the two single strands acts as a template and begins to re-form a double helix identical to the one from which it was unzipped. Newly added nucleotides are assumed to come from a pool of free nucleotides that must be present in a surrounding micro-environment within the cell. The replication reaction is catalyzed by the enzyme *polymerase*. This enzyme can extend a chain, but can not start a new one. Therefore, DNA synthesis must first be initiated with an oligonucleotide, that is, a short nucleotide chain. The oligonucleotide generates a segment of duplex DNA that is then turned into a new strand by the replication process.

1.1.2 Genes and chromosomes

Each DNA molecule is packaged in a separate *chromosome*, and the total genetic information stored in the chromosomes of an organism is said to constitute its *genome*. The human genome contains about 3×10^9 base pairs (*bp*) organized as 46 chromosomes: 22 different autosomal chromosome

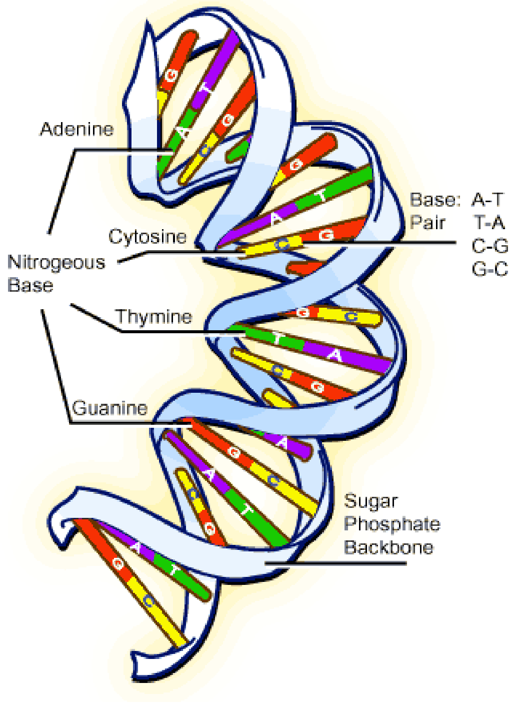


Figure 1.1: The DNA double helix.

pairs, and two sex chromosomes (either XX or XY). The 24 different chromosomes range from 50×10^6 to 250×10^6 bp.

The biological information contained in a genome can be divided into discrete units called *genes*. A gene is a region of DNA that controls a discrete hereditary characteristic, usually corresponding to a single *messenger RNA molecule* (mRNA) carrying the information for constructing a protein. In 1977 molecular biologists discovered that most eukariotic[1] genes have their coding sequences, called *exons*, interrupted by non-coding sequences, called *introns*. In humans, genes constitute approximately the $2-3\%$ of DNA.

1.1.3 The central dogma

The expression of the genetic information stored in DNA involves the translation of a linear sequence of nucleotides into a co-linear sequence of amino acids in proteins. The flow is the following: DNA $\rightarrow$ mRNA $\rightarrow$ Protein (see Figure 1.2).

A segment of DNA is first copied into a complementary strand of RNA. This process, called *transcription*, is catalyzed by the enzyme RNA polymerase. Near most of the genes there is a special DNA pattern called *promotor* which informs the RNA polymerase about where to begin the transcription. This is achieved with the assistance of transcriptional factors that recognize the promotor sequence and bind to it. Although RNA is a long chain of nucleic acids, it is very different from DNA. First of all, RNA is usually single stranded. Secondly, it has a ribose sugar instead of a deoxy-ribose one. Then, it has the pyrimidine based Uracil (U) instead of Thymine. Finally, unlike DNA, which is primarily located in the nucleus, RNA can also be found in the cellular liquid outside the nucleus, which is called *cytoplasm*. After transcription, the RNA obtained molecule moves to the cytoplasm as a messenger RNA molecule, in order to undergo translation.

[1]Living organisms can be divided into two major groups: *prokaryotes*, which are single-celled organisms with no cell nucleus, and *eukaryotes*, which are higher level organisms whose cells have nuclei.

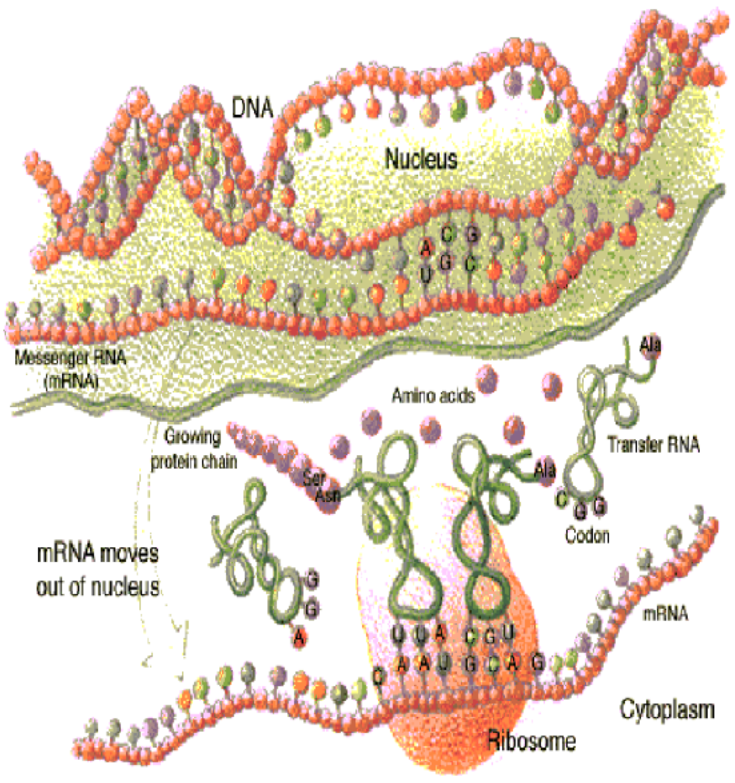


Figure 1.2: From gene to protein.

The rules by which the nucleotide sequence of a gene is translated into the amino acid sequence of the corresponding protein, the so called *genetic code*, were deciphered in the early Sixties. It was found that the sequence of nucleotides in the mRNA molecule can be read in serial order in groups of three. Each triplet of nucleotides, called a *codon*, specifies one *amino acid* (the basic unit of a protein, analogous to nucleotides in DNA). Since RNA is a linear polymer of four different nucleotides, there are $4^3 = 64$ possible codon triplets. However, only 20 different amino acids are commonly found in proteins, so that most amino acids are specified by several codons. In addition, 3 of the 64 codons specify the end of the translation and are called stop codons. The code has been highly conserved during evolution: with a few minor exceptions, it is the same in organisms as diverse as bacteria, plants, and humans.

Translation from RNA to protein

In principle, each RNA sequence can be translated in any one of three *reading frames* in each direction, making a total 6 possible *open reading frames* (ORFs), depending on where the process begins. In almost every case, only one of these reading frames will produce a functional protein.

The *translation* of mRNA into protein depends on adaptor molecules that recognize both an amino acid and a triplet of nucleotides. These adaptors consist of a set of small RNA molecules known as *transfer RNA* (tRNA). The tRNA molecule enforces the universal genetic code logic in the following fashion: one one side tRNA holds an *anticodon*, a sequence of three RNA bases; on the other side, tRNA holds the appropriate amino acid. In eukaryotes, mRNA is formed of *coding* regions flanked by *non-coding* regions. While coding regions are used for the protein creation, non-coding regions are mostly regulatory and are not translated.

Due to the mechanic complexity of ordering tRNA molecules on mRNA, a mediator is required. Such a mediator is *ribosome*, a complex of more than 50 different proteins associated with several structural mRNA molecules. Each ribosome is a large protein synthesizing machine on which tRNA molecules position themselves for reading the genetic message encoded in an mRNA molecule. Ribosomes operate with remarkable efficiency: in one second a single bacterial ribosome adds about 20 amino acids to a growing poly-peptide chain. Many ribosomes can simultaneously translate a single mRNA molecule.

1.1.4 Proteins

A protein is a linear polymer of amino acids linked together by peptide bonds. The average protein size is around 200 amino acids but large proteins can reach over 1000 amino acids. To a large extent, cells are made of proteins, which constitute more than half of their dry weight. Proteins determine the shape and structure of the cell and serve as the main instruments of molecular recognition and catalysis. Proteins have a complex structure, which can be thought of as having four hierarchial structure levels. The amino acid sequence of a protein chain is called its *primary structure*. Different regions of the sequence form local regular *secondary structures*, such as *α-helices*, which are single stranded helices of amino acids, and *β-sheets*, which are planar patches woven from chain segments that are almost linearly arranged. The *tertiary structure* is formed by packing such structures into one or several 3D *domains*. The final, complete protein may contain several protein domains arranged in a *quaternary structure*. The whole complex structure (primary to quaternary) is determined by the primary sequence of amino acids and their physico-chemical interaction in the medium. The structure of a protein determines its functionality.

1.1.5 Mutations

A *mutation* is defined as a heritable change in the DNA nucleotide sequence caused by a faulty replication process. These errors in replication occur often due to exposure to ultra violet radiation or other environmental conditions. There are two different levels at which a mutation may take place. In *gene mutation*, an *allele* (that is, a version) of a gene changes becoming a different allele. Since such a change occurs within a single gene and maps to one chromosomal *locus*, a gene mutation is sometimes called a point mutation. In *chromosomal mutation* or *rearrangements*, a segment interchange takes place, either on the same chromosome or on different ones. In addition, a chromosome may undergo a more global change, such as *reversal*, *deletion*, *duplication*, etc. For example, *Down's Syndrome* is caused by such a chromosomal mutation.

There are several kinds of point mutations:

- *substitution*: a change of one nucleotide in the DNA sequence;
- *insertion*: an addition of one or more nucleotides to the DNA sequence;
- *deletion*: a removal of one or more nucleotides from the DNA sequence.

Point mutations can also be divided according to their influence on the resulting protein:

- *missense*: a mutation that alters the codon so that it encodes a different amino acid;
- *silent*: a mutation that does not alter the codon so that it encodes the same amino acid;
- *nonsense*: a mutation that alters the codon in order to produce a stop codon.

Even though a mutation may change the amino acid sequence of a protein, it does not necessarily affect the protein functionality. This phenomenon can be explained by the fact that chemical similarity between different amino acids may result in small or no impact on the final 3D structure of the protein, therefore preserving its functionality. Furthermore, there are regions in the protein that have a very small influence on the structural functionality of the molecule.

Mutations are important for several reasons. They are responsible for inherited disorders and other diseases. For example, Sickle-cell anemia is a disease caused by a substitution mutation of thymine with adenine, resulting in the codon for valine instead of glutamic acid in the sixth amino acid of the hemoglobin protein. This simple missense mutation causes terminal tissue damage. On a brighter aspect, mutations are the source of the phenotypic variation on which natural selection acts, creating new species and adapting existing ones to changing environmental conditions. Furthermore, gene variation between organisms advances medical research in the on-going search for new and better drugs.

1.2 Computational issues

In the following, we present a (non exhaustive) list of challenging computational issues in biology.

The gene finding problem. *Given a DNA sequence, predict the location of genes (open reading frames), exons and introns.*

A simple solution would be to seek stop codons in regions along the sequence. Detecting a relatively long sequence deprived of stop codons could indicate a coding region. Due to the existence of interleaving exons and introns, the problem becomes more complicate in eukaryotic DNA. Further complications arise from the fact that certain DNA sequences can be interpreted in 6 different ways due to their corresponding open reading frames. In most cases, in eukaryotic organisms, a DNA region will encode only one gene, which is not necessarily true in prokaryote organisms.

The sequence alignment problem. *Given two DNA or protein sequences, find the best match between them.*

In order to do so, it is reasonable to define a set of possible operations and their corresponding penalties. For example, a biological phenomenon such as insertion would be mathematically translated into an open gap action which would carry a penalty. In this manner, it is possible to characterize other features such as deletions, mismatches, frame shifts, etc., each one carrying its own specific penalty according to biological frequency and gravity. The resulting best match is the one with the minimum sum of such penalties. In the more general *multiple sequence alignment problem*, there are more than two sequences.

The genome rearrangement problem *Given two permutations of a set of genomic segments, find the minimal set of operations to transform one permutation into the other.*

Rearrangements events are rare as compared to point mutations. For example, substitutions occur in some organisms about 10 times in each generation, while a non fatal rearrangement event occurs once every 5 to 10 million years. Since the chance of reversal is minute, the lower rate of rearrangements allows to detect a directional evolutionary process. Therefore, by discovering which rearrangements events have occurred and the order of their occurrence, it might be possible to build an evolutionary hypothesis.

The protein structure prediction problem *Given a sequence of amino acids, predict the 3D structure of the protein.*

Since the functionality of the protein is determined by its 3D structure, it is very important to predict the structure of a protein, thus gaining better understanding of its role in the cell. The problem of predicting a protein structure de-novo, i.e., basing on its amino acid sequence and chemical properties, has yet not been solved. Nevertheless, several approaches have been developed to approximate the structure of a protein:

- *homology modeling*: it uses a protein database to search for similar sequences of proteins. If a protein with around 30% of sequence identity is found, it is quite safe to assume that the two proteins have similar structures;

- *threading*: it classifies known structures into families with similar 3D shapes. Given a sequence of amino acids, it is possible to select the family to which the given sequence is most likely to belong to.

2 The protein folding/protein structure prediction problem

Proteins are the machines and building blocks of living cells. If we compare a living body to our world, each cell corresponds to a town, and proteins are houses, bridges, cars, roads, airplanes, etc. There are huge numbers of different proteins, each one performing its specific task.

Since it is already known how to use genetic engineering to produce proteins with a given amino acid sequence, knowledge on how such a protein would fold would allow one to predict its chemical and biological properties. If we were able to solve the so-called protein folding problem, it would greatly simplify the tasks of interpreting the data collected by the human genome project, understanding the mechanism of hereditary and infectious diseases, designing drugs with specific therapeutical properties, and growing biological polymers with specific material properties.

The protein folding problem is undoubtedly one of the most challenging ones in current biochemistry, and is a very rich source of interesting problems in mathematical modeling and numerical analysis, requiring an interplay of techniques in eigenvalue calculations, stiff differential equations, stochastic differential equations, local and global optimization, nonlinear least squares, multidimensional approximation of functions, design of experiment, and statistical classification of data. Consequently, the literature on the various aspects of protein folding is enormous. In order to have a complete picture of the physical and chemical background needed, we mainly referred to the introductory paper of Richards [111] in *Scientific American*, to the books by Brooks et al. [14] and Creighton [30], which contain thorough treatments of the subject, to the survey of Chan And Dill [23], with many additional pointers to the literature related to the physics, chemistry, and biology of protein folding, and to the paper of Pardalos et al. [103] for algorithmic aspects of the optimization problems associated with the problem.

In this chapter we introduce the basic concepts regarding the chemical structure of proteins and the physical process leading to the formation of their native shape (*protein folding*). We proceed reporting the most common simplified models of proteins and computational techniques that are used in the prediction of the structure of a protein (*protein structure prediction*).

2.1 Chemical structure of proteins

From a purely chemical point of view, a *protein* is simply a polymer consisting of a long chain of amino acid residues. More precisely, polymers of this type are called *di-*, *tri-*, *oligo-*, or *polypeptides* if they consist of 2, 3, several, or many residues, respectively. Given a protein, the chain of amino acids is called its *primary structure*, and it is stored in the DNA by means of the genetic code. Proteins in living cells contain 20 different residues, which are typically referred using three (or one) letters, as shown in Table 2.1. Each amino acid is made by several atoms (see Figure 2.1) and its structure can be partitioned into two parts. The first one, which is common to all amino acids, consists of a backbone with a nitrogen (N) and two carbon (C) atoms. One of them is called $C\alpha$ and can be considered as the *center* of the amino acid. These three atoms are bound to two hydrogen (H) atoms and one oxygen (O) atom. The second part, which characterizes each

Alanine	(A)	**Cys**teine	(C)
Aspartic Acid	(D)	**Glu**tamic Acid	(E)
Phenylalanine	(F)	**Gly**cine	(G)
Histidine	(H)	**Ile**ucine	(I)
Lysine	(K)	**Leu**cine	(L)
Methionine	(M)	**As**paragi**n**e	(N)
Proline	(P)	**Gl**utami**n**e	(Q)
Arginine	(R)	**Ser**ine	(S)
Threonine	(T)	**Val**ine	(V)
Tryptophan	(W)	**Tyr**osine	(Y)

Table 2.1: Amino acid full names, three-letter abbreviations (in bold type), and one-letter abbreviations (between brackets).

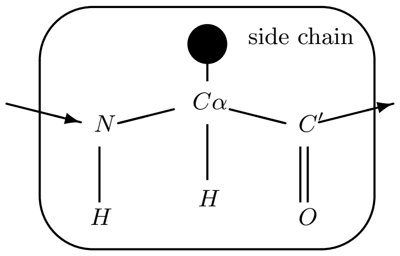


Figure 2.1: Amino acid structure.

amino acid, is known as *side chain* and consists of a number of atoms ranging from 1 (Glycine) to 18 (Arginine and Tryptophan). In Figure 2.2 the structure of the 20 amino acids is graphically depicted.

Each amino acid is linked to the following in the sequence by the incoming and outgoing edges represented by arrows in Figure 2.1. An H atom and an OH group start and close the first and the last amino acid of the sequence, respectively. The bonds joining two residues are called *peptide bonds*. The repeating -NC_αC'- chain of a protein is called its *backbone* (see Figure 2.3). Peptide bonds are geometrically rigid, meaning that their length is fixed, and rotations around them are not possible. On the contrary, the chain can rotate around the bonds between C_α and C' and between C_α and N, which also have a fixed length. Therefore, the *torsional angles* around these bonds, called ϕ and ψ, determine the entire geometry of the chain. The torsional angle ψ is defined as the angle between (the normal to) the plane spawned by $NC_\alpha C'$ and (the normal to) the plane spawned by $C_\alpha C'N$. The torsional angle ψ is defined similarly. Also the side chains of different residues show rotational degrees of freedom, and the torsional angles involved there are denoted by indexed χ letters. Only few discrete configurations of χ angles are allowed, resulting in a finite number of possible configurations of the side chains, called *rotamers*.

Once a protein is formed, it is subject to interatomic forces that bend and twist the chain in a way characteristic for each protein. They cause the protein molecule to fold up into a specific three-dimensional geometric configuration called the *folded state* or the *tertiary structure* of the protein. This spatial disposition determines the chemically active groups on the surface of a folded protein, and therefore its biological function. In Figure 2.4 there is a ball and stick, full-atom 3D representation of an Alanine and a Phenylalanine in a possible spatial configuration. Hydrogen atoms are represented in light color, carbon atoms in dark color, and oxygen and nitrogen are respectively labeled by O (red color) and N (blue color).

The *protein structure prediction problem* is the problem of predicting the tertiary structure of a protein given its primary structure. As a protein in physiologic conditions always reaches its native state, both in vivo and in vitro, it is accepted that the primary structure uniquely determines the tertiary structure. Due to entropic considerations, it is also accepted that the tertiary structure is the one that minimizes the global *free energy* of the protein. This is known

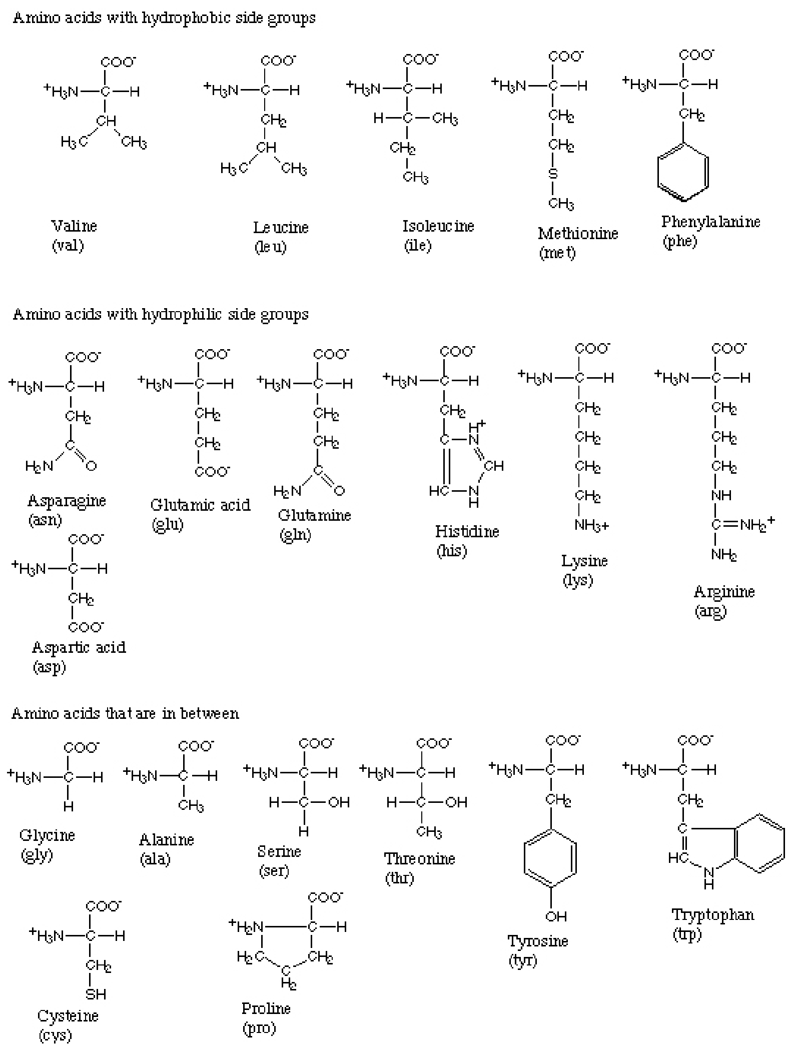


Figure 2.2: Schematic structure of the 20 amino acids.

as *Anfisen thermodynamic hypothesis* [3], which will be treated in the next sections.

As said above, the spatial geometry of a protein is determined by its torsional angles ϕ and ψ. Plotting a ψ versus ϕ diagram (the so called Ramachandran plot, see Figure 2.5) for the angles extracted from known tertiary structure, we can observe that only few regions are allowed. The more populated regions in that diagram correspond to local recurrent structures, known as *Secondary Structure*. There are two main classes of secondary elements: *α-helices* and *β-sheets*.

α-helices are constituted by 5 to 40 contiguous residues arranged in a regular right-handed helix with 3.6 residues per turn (left-handed helices exists, but are extremely rare), see Figure 2.6. β-sheets, instead, are constituted by extended strands of 5 to 10 residues, coupled together in parallel or antiparallel fashion, see Figure 2.7. Each strand is made of contiguous residues, but strands participating in the same sheet are not necessarily contiguous in the sequence. Contiguous strands in the same sheet are usually separated by regions where the chain has an U-shaped turn, called *β-turns*. The plane containing a β-sheets can be slightly bent (*β-pleated sheets*) or closed in a cylinder (*β-barrels*). The presence of secondary structure can be predicted with high accuracy by computer programs [13]. Secondary structure elements are usually combined together to form *motifs*, which are themselves part of *domains*, big super-secondary structures that characterize family of proteins.

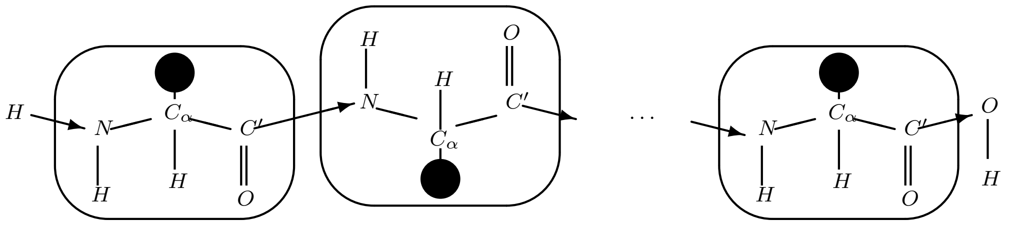


Figure 2.3: Abstract view of the amino acid backbone.

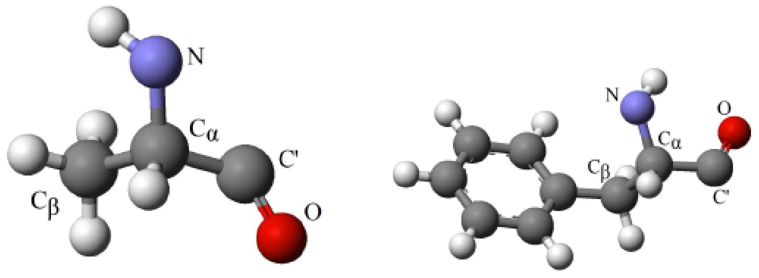


Figure 2.4: Alanine (on the left) and Phenylalanine (on the right).

The smallest proteins, hormones, have about 25-100 residues; typical globular proteins about 100-500; fibrous proteins may have more than 3000 residues. Thus the number of atoms involved ranges from 500 to more than 10000.

2.2 Protein modeling

In this section we look at the physics governing the process of folding of a protein. We will first motivate Anfinsen hypothesis, then introduce a model for the potential energy of a protein, and finally describe the (supposed) structure of the energy landscape.

2.2.1 Molecular mechanics

In this subsection we give an overview of the physics involved in the motion of the atoms in a protein. For every atom i, its position in the three-dimensional space is defined by a vector $\vec{x}_i$, which can be represented by three single coordinate variables x_{i1}, x_{i2}, x_{i3}. A protein, thus, can be described by means of the vector

$$\vec{x} = \begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_N \end{pmatrix} = \begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \\ \vdots \\ x_{N3} \end{pmatrix}$$

where N is the total number of atoms in the molecule. For real proteins the dimension of x is in the range of about 1500–30000.

The force balance within the molecule and the resulting dynamics can be mathematically approximated by the stochastic differential equation of *Langevin dynamics* [74]:

$$M\ddot{x} + C\dot{x} + \nabla V(x) = D\dot{W}(t). \tag{2.2.1}$$

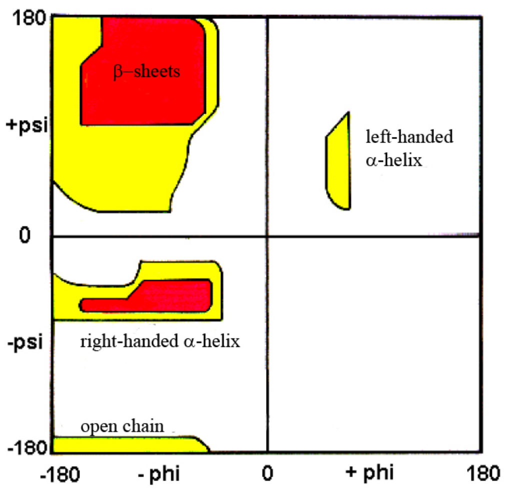


Figure 2.5: Schematic representation of the Ramachandran Plot.

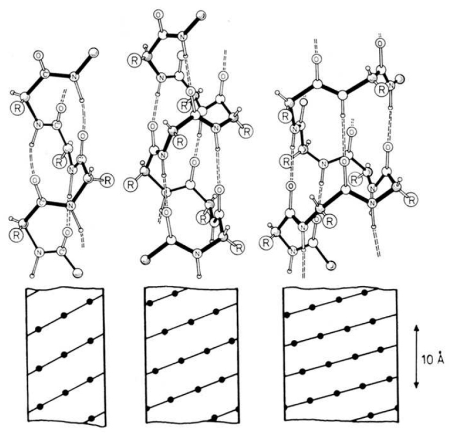


Figure 2.6: Schematic representation of α-helix.

The first term of equation (2.2.1) describes the change of kinetic energy, and is the product of the *mass matrix* M and the *acceleration* $\ddot{x}$. The second term describes the excess energy dissipated to and absorbed by the surrounding, and it is the product of a symmetric, positive definite *damping matrix* C and the *velocity* $\dot{x}$. The third term describes the change in the potential energy, and is expressed as the gradient of a real valued *potential function* V characteristic of the molecule and defined for all x except when two coordinate vectors x_i and x_j coincide. Finally, the right hand side term is a random force accounting for fluctuations due to collisions with the surrounding that dissipate the energy; it is the product of normalized white noise $\dot{W}(t)$ with a suitable matrix D.

Actually, the interaction with the environment is in reality much more complex, the damping and fluctuation terms are only simplified descriptions. Improving such formulations can quickly lead to mathematically intractable problems. More realism is therefore added by introducing an explicit (simplified) representation of the environment where a protein folds up, i.e., by introducing a certain number of water molecules, and extending the potential to account for their interactions with each other and with the protein atoms. More details can be found in the survey of Neumaier [101].

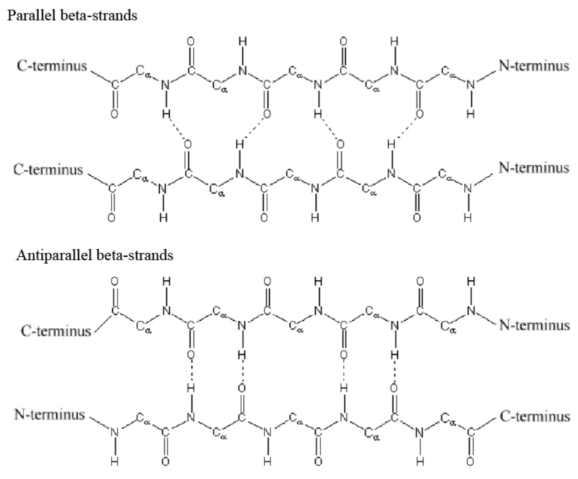


Figure 2.7: Schematic representation of parallel anti-parallel β-strands.

2.2.2 The low temperature limit

When the temperature $T \to 0$, the term $D\dot{W}(t)$ in formula (2.2.1) vanishes. This is equivalent to affirm that, at the limit, there is absence of random forces. The equation (2.2.1) thus becomes the ordinary differential equation

$$M\ddot{x} + C\dot{x} + \nabla V(x) = 0. \tag{2.2.2}$$

In this case, the sum of kinetic and potential energy,

$$E = \frac{1}{2}\dot{x}^T M\dot{x} + V(x), \tag{2.2.3}$$

has time derivative $\dot{E} = -\dot{x}^T C\dot{x}$. Since C is positive definite, this expression is negative, and vanishes only at zero velocity, when all energy is potential energy. This means that, thanks to the dissipation term $C\dot{x}$, the molecule continually loses energy until it comes to rest at a stationary point of the potential ($\nabla V(x) = 0$). This stationary point generally is a *local minimum* of the potential.

Assuming a positive temperature, the random forces will add kinetic energy, so that the molecule will describe random oscillations around the local minimum. Sometimes, it is possible that the protein collects the necessary energy to overcome the local potential barrier and fall into another minimum, associated to a different conformation.

For rigid molecules, characterized by the fact that there is a unique minimizer in the part of state space accessible to the molecules, the minimization of the potential energy surface produces a *stable state* which describes the geometric average shape.

Proteins, however, are not rigid, but can twist along the bonds of the backbone. Therefore, the potential energy surface is complicated and presents a very high number of local minima. It is possible that random forces allow the protein to escape from the neighborhood of one local minimum (*metastable state*) and reach another one. Such transitions are referred as *state transitions.*

The frequency of transitions depends on the temperature and on the energy barrier along the energetically most favorable path between two adjacent local minima. Any such path has its highest point on the energy surface at a peak called *transition state.* Transitions from a state with higher energy to another with lower energy are much more frequent than transitions to a higher energy level.

This implies that, over long time scales, a molecule spends most of the time in the valley close to the global minimizer of the potential.

2.2.3 Modeling the potential

Strictly speaking, the dynamics of atoms in a molecule is governed by the quantum theory of the participating electrons. However, for complex molecules, quantum mechanical calculations are far beyond the computational resources likely to be available in the near future. Hence chemists usually use a classical description of molecules and quantum theoretical calculations are restricted to some properties of some small parts of the molecule.

The interactions of the atoms in proteins can be classified into *bonded* and *non-bonded* interactions. Bonded interaction include *Covalent bonds*, which are considered unbreakable, *Disulfide bonds*, which join together two sulfur atoms and are slow to form and to break, and *Hydrogen bonds*, which connect hydrogen atoms with close oxygen atom, and are fairly easily formed and broken. Non-bonded interactions comprehend the long range *electrostatic* one, which is established between atoms carrying partial charges, and the short range *van der Waals* one, happening between all pairs of atoms.

Hydrogen bonds and non-bonded interactions are particularly relevant for the interaction of the molecule with the atoms of the solvent.

Static forces are fully specified by the potential $V(x)$. Modeling the molecule reduces to specifying the contribution of the various interactions to the potential. The models are called *force fields*. The CHARMM potential [15] is a good example of force field, composed by six kind of terms, given in the table 2.2.

V(x)=	$\sum$ bonds	$c_l(b-b_0)^2$	(b a bond length)
$+$	$\sum$ bond angles	$c_a(\theta-\theta_0)^2$	(θ a bond angle)
$+$	$\sum$ improper torsion angles	$c_i(\tau-\tau_0)^2$	(τ an improper torsion angle)
$+$	$\sum$ dihedral angles	$\mathrm{trig}(\omega)$	(ω a dihedral angle)
$+$	$\sum$ charged pairs	$\frac{Q_iQ_j}{Dr_{ij}}$	(r_{ij} the Euclidean distance from i to j)
$+$	$\sum$ unbonded pairs	$c_w\phi\left(\frac{R_i+R_j}{r_{ij}}\right)$	(R_{ij} the radius of atom i)

Table 2.2: The CHARMM potential.

The Q_i are *partial charges* assigned to the atoms in order to approximate the electrostatic potential of the electron cloud, and D is the dielectric constant. The quantities indexed by 0 are the reference values; different constant apply depending on the type of atom. The trigonometric terms trig(ω) and the *force constants* c depend on the corresponding type of atoms too. Remember that a dihedral angle can be both the angle between three consecutive atoms or the torsional angle of four consecutive atoms.

The van der Waals interactions depend on φ, which is modeled as the *Lennard-Jones potential*:

$$\phi\left(\frac{R_0}{r}\right)=\left(\frac{R_0}{r}\right)^{12}-2\left(\frac{R_0}{r}\right)^{6}. \tag{2.2.4}$$

The first term decreases for small r forcing atoms to repel each other at short distance. The second term slowly increases for large r, causing an attraction of neutral atoms at large distance. There is a minimum for $r=R_0$, which models a correct behavior. Taking the equilibrium distance R_0 between two atoms as the sum $R_0=R_i+R_j$ of the atomic radii is a simple combination rule to reduce the number of parameters.

2.2.4 Parameter estimation

Currently, the determination of the coefficients in a potential energy model is based on data obtained by one of the following methods:

- *X-ray crystallography*, which gives the equilibrium positions of the atoms in crystallized proteins;
- *Nuclear magnetic resonance (NMR) spectroscopy*, which gives position data of proteins in solution;
- *Ab initio* quantum mechanical calculations, which give energies, energy gradients and energy Hessians (i.e., second derivative matrices) at arbitrarily selected positions of the atoms, for molecules in the gas phase;
- Measurements of *energy spectra*, which give rather precise eigenvalues of the Hessian;
- *Thermodynamical analysis*, which gives specific heats, heats of formation, conformational stability information, related to the potential in a more indirect way, via statistical mechanics.

The model parameters are adapted to data from one or several of these sources by using a mixture of least squares fitting and more heuristic or interactive procedures. Starting values for the coefficients come from general knowledge about atomic radii, average bond lengths and angles, and (for the force constants) from the frequencies of the oscillations of these quantities. The resulting rough parameters are then fitted to the data using a least square approach.

2.2.5 Gibbs free energy

The potential relevant for calculations at fixed finite temperature $T > 0$ is the *Gibbs free energy* $G = H - TS$, containing the *enthalpy* H and a correction term involving the *conformational entropy* S of the system.

The conformational entropy is proportional to the logarithm of the number of microscopically distinguishable configurations belonging to an observed macrostate and is thus roughly proportional to the logarithm of the volume of the catchment region of a metastable state. Thus large flat minima (having a large catchment region) or large regions covered by many shallow local minima (corresponding to a glassy regime) are energetically more favorable than a narrow global minimum if this has only a slightly smaller potential.

Currently, entropy considerations are often addressed computationally in a qualitative fashion only, using very simple (e.g., lattice) models together with techniques from statistical mechanics. For semi-empirical potentials fitted to experimental data, these fine points appear somewhat less important in view of the other approximations made: the resulting potentials are always effective potentials adapted to the form of potential used and the experimental conditions from which the data are derived. However, this implies that caution is needed when combining data from different sources.

2.2.6 The energy landscape

We already remarked that the geometry defined by the *global* minimum of the potential energy surface is expected to be the correct geometry describing the conformation observed in folded proteins. The most challenging feature of the protein folding problem is the fact that the objective function has a huge number of local minima, so that a local optimization is likely to get stuck in an arbitrary one of them, possibly far away from the desired global minimum. People working in the field expect an exponential number of local minima. Estimates range from 1.4^n to 10^n for a protein with n residues. However, most of these local minima would have a large potential energy and thus be irrelevant for global optimization; unfortunately, the number of low-lying minima seems also to grow exponentially in n. For very general energy minimization problems, combinatorial difficulty (NP-hardness) can be proved by showing that the traveling salesman problem (TSP) can be phrased as a minimization of the sum of two-body interaction energies [126], though this result does not generalize to more realistic situations.

Running different experiments under physiological conditions, we can observe that most of the proteins always fold in the same native configuration. This strongly suggests the existence of a unique global minimizer with a significantly lower energy than all other local minimizers. This is also supported by experiments that suggest that the approach to the global minimum proceeds in two phases, a rapid phase to reach a nearly folded state, followed by a long period to complete the transition to the final state.

The most natural explanation is the existence of a large barrier with many (but not too many or too deep) saddles around the valley containing the global minimum. This energy landscape goes usually under the name of *folding funnel* [113].

However, quantum mechanical corrections (accounting for vibrational energy) might disfavor the global minimum state [93]. The main effect is that a slightly non-global minimum in a broader valley may be more highly populated than a global minimum in a valley with steep walls (cf. [111]). Therefore the folded state might be a metastable state with high energy barriers, or it might just be the lowest local minimizer that is kinetically accessible from most of the state space. The folded state may also correspond to more extended regions in state space where there are many close local minima of approximately the same energy as at the global minimum. This last situation corresponds to what physicists refer to as *glassy* behavior, and there are some indications from molecular dynamics simulations ([42]) that this might be the situation in typical energy surfaces of proteins.

Recent studies seem to reach an agreement in that the native state is a pronounced global minimizer that is reached dynamically through a large number of transition states by an essentially *random search* through a huge set of secondary, low energy minima (representing a glassy *molten globule* state), separated from the global minimum by a large energy gap.

In simulations with simple lattice models, this scenario appears to be the necessary and sufficient conditions for folding in a reasonable time [37]. Moreover, it also explains the so-called *Levinthal paradox* [86], stating that the time a protein needs to fold is not sufficient to explore even a tiny fraction of all local minima. However, fast collapse to a compact molten globule state, followed by a random search through the much smaller number of low energy minima, could account for the observed time scales, and the energy gap provides the stability of the native state over the molten globule state. Therefore, most of the folding time is spent trying to drive the molten globule through one of the transition states into the native geometry. On the other hand, the studies on the folding process disagree on many of the details, and the simplified drawings of the qualitative form of the potential energy surface are mutually incompatible among them.

From an evolutionary point of view, the hypothesis of a single, well separated global minimum well is also very likely. Indeed, for organisms to function successfully, the proteins performing specific tasks must fold into identical forms. Polypeptides that do not satisfy this requirement lack biological reliability and are not competitive. Thus one expects that at least the polypeptides realized as natural proteins have a single global minimum, separated from nearby local minima by a significant energy gap.

Recently, however, a number of proteins called *prions* were discovered that exist in two different folded states in nature. The normal form appears to be a metastable minimum only, separated by a huge barrier from the sick form in the global minimum. Under ordinary circumstances, only the metastable form is kinetically accessible from random states; but the presence of molecules in sick form acts as a catalyst that reduces the barrier enough to turn the normal form quickly into sick form, too. Substitution of a few crucial amino acids (caused by mutations of the prion-coding genes) also reduces the barrier.

2.3 Simplified models of proteins

Various simplifications of the protein folding problem are studied in the literature in order to understand the global optimization process and to simplify the development and testing of optimization algorithms. The most common simplifications regard either the protein representation, or the representation of the space conformation, or the energy function.

As far as the detail of protein representation is concerned, there are different possibilities. One possibility is dropping information about all the atoms in the system, representing each amino acid in a simplified way. Amino acids can be identified with *one point centered in the C_α atom* [124], or the *side chain* can be described as a *sphere* or an *ellipsoid* [51]. In other models, the entire backbone is represented in full atomic detail. See [119] for a detailed review, and Figure 2.8 for some visual examples.

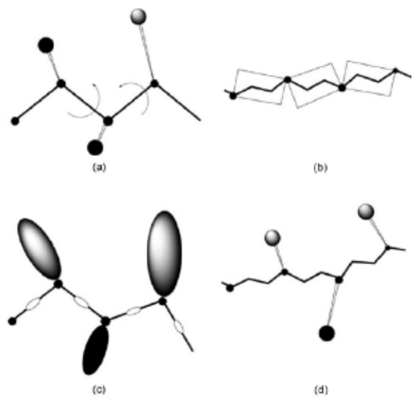


Figure 2.8: Examples of reduced representations of proteins.

As far as the 3D space in which amino acids are positioned is concerned, it can be modeled both as the *real three dimensional space* or as a *discrete lattice*. In this second case, different lattices have been used, ranging from the simple 2D and 3D cubic lattices, to Face-Centered Cubic Lattice, to lattice with an higher degree of coordination, see [59] for a review on lattice models.

As far as the energy representation is concerned, it should be capable of recognizing the native structure in the sense that this structure should ideally be at the global minimum of the energy. The choice of the representation determines also the functional form of the energy, as described hereafter.

Full energy models. Simplified full energy models have fixed bond lengths, bond angles and some torsion angles (e.g., around the peptide bond). The only degrees of freedom are an independent set of torsion angles, which limits the number of variables to $\sim 3n - 5n$, where n is the number of residues. Such models are regarded as highly reliable, but function evaluation is expensive due to the required transformations between angles and Cartesian coordinates.

Statistical backbone potential models. At the level of description of statistical potentials, only the backbone (or the backbone and a side chain center, or even only the set of C_α atoms) is modeled, with fixed bond lengths, bond angles and peptide bond torsion angles (or fixed distant of neighboring C_α's, respectively). In both cases, the number of variables is reduced to $2n$, and the potential has a simple form, determined by assuming that a set of known structures is an equilibrium ensemble of structures, so that the energy can be calculated from Boltzmann's law and statistics on the known structures. In order to obtain a useful statistics, the protein structures used must be carefully selected. The fact that the potential is now directly derived from geometric data implies that it automatically takes account of solvation and entropy corrections; on the other hand, one only gets a mean potential of less resolution. For a survey on statistical potentials at this level of description, see [119].

Lattice models with contact potentials. Under the choice of a discrete space, molecules are forced to have their atoms lying on lattice positions, and the potential is a sum of contact energies taken from tables derived again from statistics on databases, like in [11]. Now function evaluation is extremely cheap (addition of table entries for close neighbors only, resulting in a speedup factor of two order of magnitudes), and the problem has become one of combinatorial optimization. For a survey on lattice models, see [59].

2.4 Computational approaches to Protein Structure Prediction

In this section we give a brief overview of the three most known classes of methods for Protein Structure Prediction. We present them in decreasing order with respect to the information required other than the primary structure.

2.4.1 Homology Modeling

Homology modeling is a technique based on the hypothesis that evolution tend to conserve both the structure of proteins and their sequence, at least in some key amino acids for the folding process. Therefore, proteins with similar sequences are supposed to have similar structures. The first step in homology modeling is to identify some target template structures. This is done by aligning the sequence of the protein versus sequences with known 3D structure in a database, identifying those with higher homology score. For an overview on the most common techniques for sequence alignment, cf. Chapter 4. Then, the corresponding structures become the templates. The sequence under study is somehow fitted in the target structures, taking into account the result of the alignment. In this way, most of the amino acids have 3D coordinates assigned. The missing ones, usually those corresponding to gaps in the alignment, are then fitted by minimizing a suitable energy function, and finally the model is completed with the remaining atoms. Homology modeling produces good results, as long as an alignment with at least 30% of homology has been found. Because of its dependence on large databases, this approach is also termed ‘knowledge-based’. For a survey on homology modeling, see [58] and references therein.

2.4.2 Threading

An approach similar to homology is to try to match amino acid sequences to known folding structures. In this case, no alignment on strings is performed, but the sequence is directly aligned to the structure. This process becomes more interesting as the database of available protein geometries becomes larger and more representative. In fact, the number of different folds observed in nature is much smaller than the number of different proteins, hence there are good chances that a protein has a 3D shape close to one already known.

Various folding structures are tried in turn until one is found that makes some measure of fit (usually a statistical potential) small enough. This matching process, usually involving local optimization techniques, is called *threading.* All reasonable fitting structures are then subjected to a stability test (using molecular dynamics or Monte Carlo simulation) in order to check the correct energetic behavior of the computed structures. The decision whether a fit is reasonable must again be based on statistical potentials.

The main obstacle for successful threading of general sequences resides in irregular coil regions: they can have variable length, hence the structural match must deal with insertions and deletions. Nevertheless, at present, threading seems to be the most efficient way of structural prediction; and it will become more reliable as more and more proteins with known structure become available. For a survey on threading, see [112].

2.4.3 Ab-initio methods

If the protein sequence under study has an unknown structure, both homology modeling and threading will fail in giving a reasonable model of its structure. Their limitation, in fact, resides in the fact that their accuracy depends on the current knowledge available, so new structures are out of their scope. On the contrary, ab-initio methods do not suffer from these limitations, despite having a much smaller accuracy than their knowledge-based competitors.

Ab-initio methods are all based on the Anfisen hypothesis: they search the global minimum of an energy function, for a given (reduced) representation of proteins. In principle, the most precise methods use molecular dynamics simulations [87]. Unfortunately, they are not feasible, due to the high intrinsic complexity of the needed operations. In particular, two to four CPU-days of a supercomputer at Berkeley Labs are needed to simulate a single nanosecond in the evolution of a medium length protein, while the typical folding time is of the order of milliseconds or seconds.

More efficient methods are offered by reduced models, using statistical potential. In this case, the conformational space needs to be searched with effective methods in order to identify the global minimizer. As commented in Subsection 2.2.6, this is an extremely difficult problem, given the fact that all energy functions have an exponential number of local minima to search through. In the exploration phase, different strategies can be used, ranging from local, to stochastic, to global methods (see [50] for an overview).

Among most successful ab-initio predictors (i.e. predictors which do not start from homology with a structurally characterized sequence), the programs Tasser [128, 118] and Fragfold [70] have been performing very well in the recent CASP rounds [99].

3

Model checking approaches to the protein folding/protein structure prediction problem

In this chapter, we apply model checking to the study of protein conformations. First, we model the space of protein conformations as a finite transition system whose states are all the possible conformations of a protein and whose transitions represent admissible transformations of conformations. We distinguish between the states that represent conformations with minimum energy value and the other states. Since the minimum energy value of a given protein is actually known only once the protein structure prediction problem has been solved, we compute an estimate of such a value before model checking the system. Then, we show how meaningful properties of such a transition system can be expressed in temporal logic. As an example, we can write a formula that states the existence of a path from a given conformation to another one, with energy level lower than a certain threshold, whose length is less than or equal to a given value. Finally, we use the model checking machinery to algorithmically check such properties.

An experimental evaluation of the main model checkers proposed in the literature showed us that none of them is well-suited to our application domain. In particular, both SPIN and NUSMV turned out to be (for different reasons) inappropriate. As for the former, the available mechanism for exchanging messages does not fit well with the characteristics of our application domain; as for the latter, finite transition systems modeling protein conformations do not feature those symmetries that only allow symbolic model checkers like NUSMV to speed up the computation in a significant way. As a consequence, to cope with the combinatorial blow up of the state space, we resort to on-the-fly model checking. We decided to build from scratch an on-the-fly model checker in SICStus Prolog and we exploited it to test the protein properties we are interested in. In particular, we experimented it on some proteins belonging to the Protein Data Bank, being able to verify meaningful properties of proteins of significative length.

Our solution presents some similarities with the approaches to protein structure prediction and protein folding based on local search. The set of states of the transition system can indeed be viewed as the search space of local search algorithms and transitions can be interpreted as local search moves that lead to (local) minimum energy states. In the case of local search, however, no information about the paths that connect different protein conformations is provided.

The chapter is organized as follows. In Section 3.1 we introduce some simplified models of proteins. Then, in Section 3.2 we propose a method to estimate the minimum energy value of a given protein in a two-dimensional lattice and we describe a way to parallelize the search for such a value, which is a fundamental input for the model checking algorithm. Next, in Section 3.3 we show how to generate, for any given protein, the corresponding finite transition system and how to express relevant properties of its conformations in temporal logic. In Section 3.4 we describe the model checker we developed in SICStus Prolog to test the properties we are interested in. Some experimental results are reported in Section 3.5. In Section 3.6 we provide an assessment of the work and we outline some future research directions.

Alanine (A) H	**Cys**teine (C) H	**Asp**artic Acid (D) P	**Glu**tamic Acid (E) P
Phenylalanine (F) H	**Gly**cine (G) P	**His**tidine (H) H	**Ile**oleucine (I) H
Lysine (K) P	**Leu**cine (L) H	**Met**hionine (M) H	**Asn**paragine (N) P
Proline (P) P	**Gln**utamine (Q) P	**Arg**inine (R) P	**Ser**ine (S) P
Threonine (T) P	**Val**ine (V) H	**Trp**yptophan (W) H	**Tyr**osine (Y) H

Table 3.1: Amino acid full names, three-letter abbreviations (in bold type), one-letter abbreviations (between brackets), and hydrophobic/polar characterization (blue letter H or red letter P).

3.1 On energy models

In this section we introduce some simplified models of proteins that have been proposed in the literature to cope with the inherent complexity of the protein structure prediction and protein folding problems. As anticipated in the previous section, a common simplification consists in using *lattice space models* to restrict the admissible spatial positions of the amino acids [77]. The energy function can be simplified as well. Two commonly adopted energy models are the 20×20 potential matrix, proposed in [11] and used in [32, 95], and the simpler HP model, proposed in [84] and used in [6, 7]. The 20×20 potential matrix keeps track of the variety of interactions among the 20 kinds of amino acid. The HP model reduces such a 20-letter alphabet $\mathcal{A}$ of amino acids to a two-letter alphabet {H, P}, where H (resp., P) represents a hydrophobic (resp., polar) amino acid [28].

For the sake of simplicity, we will make use of a two-dimensional finite lattice included in $\mathbb{N}^2$; however, the solution we propose can be easily extended to three-dimensional lattices. As for the energy model, we will exploit both the 20×20 model and the simple HP model. While no efficient solutions to the structure prediction problem based on 20×20 model can be found in the literature, there exist various successful approaches based on the HP model. In particular, Backofen and Will outlined a constraint-based approach to the three-dimensional structure prediction problem that exploits the possibility to compute maximally compact sets of points used as hydrophobic cores [8].

The *primary* structure of a protein is a sequence of linked units of reasonable length, called *residues*. Every residue is an amino acid of one of the 20 types reported in Table 3.1, which extends Table 2.1 indicating the hydrophobic/polar characterization of each amino acid.

In nature, amino acids attract/repel each other with different strength levels, which depend on their types. Such a fact can be modeled by associating an energy value with every pair of amino acids. In [11], by applying statistical methods on structures obtained by X-Rays and NMR (Nuclear Magnetic Resonance) experiments, Berrera et al. provide a matrix that, given a pair of amino acids in *contact*, defines the resulting energy value on the basis of their types. The values stored in the matrix can be either positive or negative. In the following, we will refer to this matrix as to the 20×20 potential matrix. A simplified characterization can be obtained by taking advantage of the HP model. The energy function of this model states that the energy contribution of a *contact* between two amino acids is -1 if both of them are H amino acids, 0 otherwise.

Let s be a finite sequence of amino acids. In the 20×20 model, we denote by s_i, with $s_i \in \mathcal{A}$, the i-th element of s. In the HP model, we represent s as an element in $\{0,1\}^*$, where $s_i = 1$ (resp., $s_i = 0$) stands for an occurence of an H (resp., P) amino acid. Furthermore, to restrict the admissible spatial positions of amino acids, we use a two-dimensional discrete lattice, where every conformation of a protein is a self-avoiding walk in $\mathbb{N}^2$ (in [84] Lau and Dill actually consider $\mathbb{Z}^2$ instead of $\mathbb{N}^2$, but this makes no any substantial difference). The subset of admissible protein conformations is defined as follows.

Definition 3.1.1 (Conformation) *A* conformation ω *of a sequence* $s = s_0 \dots s_n$ *is a function* $\omega : [0 \dots n] \to \mathbb{N}^2$ *such that*

(i) $\forall 0 \leq i < n \; (|\omega(i) - \omega(i+1)| = 1)$*, i.e., if* $\omega(i) = (X_i, Y_i)$ *and* $\omega(i+1) = (X_{i+1}, Y_{i+1})$*, then* $|X_i - X_{i+1}| + |Y_i - Y_{i+1}| = 1$*;*

(ii) $\forall i \neq j \; (\omega(i) \neq \omega(j))$ *(*ω *is self avoiding).*

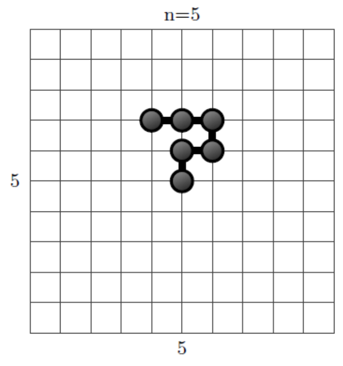


Figure 3.1: The string *rllf* on the 10×10 lattice.

We say that two amino acids s_i and s_j of a given conformation ω are *connected neighbors* if $j = i \pm 1$ and that they are *topological neighbors* if they are not connected neighbors and $|\omega(i) - \omega(j)| = 1$.

In the 20×20 model, the energy of a conformation of a sequence on the 20-letter alphabet $\mathcal{A}$ of amino acids is given by the summation of the energy contributions of topological neighbors.

Definition 3.1.2 (Conformation energy using the 20×20 potential matrix) *Given a sequence $s = s_0 \dots s_n$ on $\mathcal{A}$, the energy of a conformation of s is:*

$$E = \sum_{1 \leq i+1 < j \leq n} M_{s_i,s_j} \cdot \delta(s_i, s_j)$$

where M_{s_i,s_j} equals the energy contribution between s_i and s_j stored in the 20×20 potential matrix and $\delta(s_i, s_j)$ is 1 if s_i and s_j are topological neighbors, 0 *otherwise.*

In the HP model, the energy of a conformation is given by the opposite of the number of topological HH neighbors, e.g., if there exist k topological HH neighbors in ω, then the energy of ω is $-k$.

Definition 3.1.3 (Conformation energy using the HP model) *Given a sequence $s = s_0 \dots s_n$ on $\{0, 1\}$, the energy of a conformation of s is:*

$$E = \sum_{1 \leq i+1 < j \leq n} B_{s_i,s_j} \cdot \delta(s_i, s_j)$$

where B_{s_i,s_j} is equal to -1 whenever both s_i and s_j are 1, 0 *otherwise, and $\delta(s_i, s_j)$ is 1 if s_i and s_j are topological neighbors,* 0 *otherwise.*

According to Definition 3.1.3, we thus have that a conformation has minimum energy if it maximizes the number of HH contacts.

Let $s = s_0 \dots s_n$ be a sequence of amino acids. We define the length of a sequence as the number of "segments" it is made of. Hence, the length of s is n. To represent the possible conformations of a sequence of length n, we use the subset $\mathcal{L} = \{(i, j) : i \in [0, 2n], j \in [0, 2n]\}$ of $\mathbb{N}^2$. Without loss of generality, we assume $\omega(0) = (n, n)$ and, in order to avoid simple symmetries, we fix $\omega(1) = (n, n + 1)$. Once the coordinates of a segment have been fixed, the next segment in the sequence can only assume three possible directions with respect to it: left (l), forward (f), and right (r). As a consequence, a conformation of a sequence of length n can be represented as a string of length $n - 1$ on the alphabet $\{l, f, r\}$[1]. As an example, the sequence of Figure 1 is represented by the string *rllf*.

[1]To avoid symmetries, one can also exclude strings with prefixes of the form f^*l. A perfectly symmetric conformation can indeed be obtained by starting with f^*r.

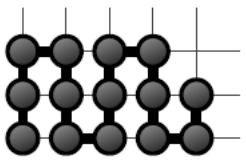


Figure 3.2: The string *frrfllfrrfll*.

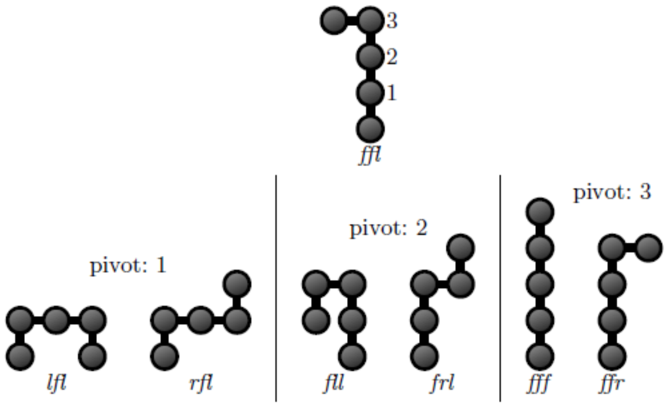


Figure 3.3: Pivot moves from string *ffl*

The number of all possible conformations of a sequence of length n, where the orientation of the first segment is fixed as above, is bounded by 3^{n-1}. It is commonly accepted that the number C_n of self-avoiding walks of length n grows according to the following formula $C_n = B \cdot \mu^n \cdot n^{\gamma-1}$, where $B \sim 1.93$, $\mu \sim 2.63$, and $\gamma = 43/32$ [88][2]. Thus, if the orientation of the first segment is fixed, then the number of self-avoiding walks of length n is $D_n = C_n/4$.

In [31], Crescenzi et al. prove that the decision version of the protein structure prediction problem for the HP model on two-dimensional lattices is NP-complete. It is not difficult to show that the same result holds for the 20×20 model as well.

As explained in the previous chapter, given the native conformation of a protein it is possible to detect the so-called *Secondary Structure Elements* (SSEs) [117], that is, local motifs consisting of short, consecutive parts of the amino acid sequence that present a very regular conformation. Some of them are α-helices, β-sheets, and $\beta\alpha\beta$ turns. In this work, we will restrict our attention to α-helices, which are constituted by 5 to 40 amino acids arranged in a regular right-handed helix, and we will show how their presence can be exploited to speed up the proposed model checker (see Section 3.4). Figure 3.1 provides an example of a helix of the form (*frrfll*)* in the two-dimensional lattice.

Conformation transitions are governed by a set of valid transformations, called *moves*. Various moves have been introduced in the literature. The most well-known are *end*, *corner*, *crankshaft*, and *pivot* moves [28]. End, corner, and crankshaft moves are *local* moves, that is, they explore only a very small subset of the collection of conformations. On the contrary, pivot moves are *global* ones, and we will focus our attention on them. Roughly speaking, given a conformation *fd*, a pivot move consists in randomly selecting a position in *fd* and performing a rotation of the portion of *fd* between this position and the ending one.

Definition 3.1.4 (Pivot move) *Let fd=fd*$_2 \dots$ *fd*$_n$*, with fd*$_i \in$ {*l, f, r*} *for all* $2 \leq i \leq n$*, be a*

[2]As a matter of fact, the value of μ, that determines the asymptotic behavior of C_n, is not known exactly for the two-dimensional square lattice. The most efficient algorithms to estimate μ take advantage of finite state automata [108].

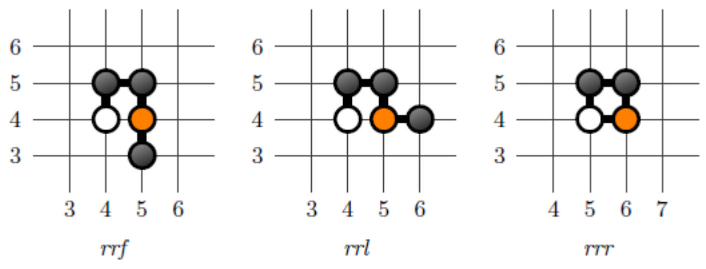


Figure 3.4: The white circle is s_0. Choosing pivot $i = 3$ (the orange circle), the first possible move ($f \to l$) generates a self avoiding walk, the second one ($f \to r$) generates a loop.

conformation of a sequence s of length n. A conformation fd′ of s is obtained from fd through a pivot move *with pivot* $k-1$*, with* $2 \leq k \leq n$*, if* $fd'_i = fd_i$ *for all* $i \neq k$ *and* $fd'_k \neq fd_k$*.*

Since the number of possible pivots of a conformation of a sequence of length n is $n-1$ and each one may give rise to two moves (rotations), the number of successor conformations is at most $2(n-1)$ (there can be conformations that violate the self-avoiding condition). As an example, consider the sequence of length 4 whose conformation is represented by the string *ffl*. The conformations that can be obtained from it by pivot moves are the 6 conformations *lfl*, *rfl*, *fll*, *frl*, *fff*, and *ffr* graphically depicted in Figure 3.3. An example of a pivot move that generates a walk which is not self avoiding is given in Figure 3.4.

In [89, 90] Madras and Sokal show that pivot moves are ergodic, namely, they cover the entire conformation space. More precisely, they prove that, given two conformations fd' and fd'' of a sequence of length n, it is always possible to pass from fd' to fd'' performing at most n^2 pivot moves. They demonstrate this fact by exhibiting a sequence of pivot moves that transform any conformation fd into the all-straight conformation fd_0 (of course, the reverse sequence of moves transforms fd_0 back to fd) and by observing that a transformation from fd' to fd'' can be split into a first stage in which fd' is transformed into fd_0 and a second stage in which fd_0 is transformed into fd''. For each conformation, they take into consideration two parameters, namely, the number of straight internal angles and the difference between the minimum and maximum abscissa, and they propose suitable pivot moves that increase the sum of these two quantities.

3.2 An estimate of the minimum energy value of a protein

As we already pointed out in the introduction, states that represent conformations with minimum energy value play an important role in transition systems modeling the whole set of conformations and of their relations. Unfortunately, to determine such a minimum energy value we should have already solved the protein structure prediction problem. In this section, we show how to compute an estimate of this value to be used in the model checking process.

We proceed as follows. First, we determine a lower bound to the minimum energy value; then, we compute the minimum energy value for proteins of limited length, namely, of length less than or equal to 25, using constraint programming over finite domains; finally, we compute the ratio between the latter and the former value (obviously, for proteins of length less than or equal to 25 only). An estimate of the minimum energy value for any protein (of length greater than 25) will then be given by taking the lower bound to its minimum energy value and multiplying it by the ratio.

3.2.1 The computation of the lower bound

Let us consider the HP model. To compute a lower bound to the minimum energy value, we can take advantage of the solution exploited by Hart and Istrail in their approximation algorithm [63, 64][3]. To start with, they observe that the maximum number of neighbor positions for an amino acid (the so-called *lattice connectivity constant*, denoted by c_L) in both the square and the cubic lattice is $2d$, where d is the dimension of the lattice. Since for every interior (resp., terminal) amino acid, 2 (resp., 1) of the c_L possible neighbor positions are (resp., is) occupied by the neighbors (resp., neighbor) in the HP sequence, it follows that every interior (resp., terminal) amino acid can form at most $c_L - 2$ (resp., $c_L - 1$) contacts. Furthermore, they observe that, again both in the square and in the cubic lattice, if the i-th and j-th amino acids form a contact in a conformation, then i is even and j is odd, or vice versa. Then, they proceed as follows. They label the H amino acids by X and Y, where all amino acids labeled by X (resp., Y) have the same parity, and those labeled by X and Y have opposite parities. Given an HP sequence s, they denote by $\mathcal{N}_X(s)$ (resp., $\mathcal{N}_Y(s)$) the number of H amino acids labeled by X (resp., Y) and by $\#term_X(s)$ (resp., $\#term_Y(s)$) the number of terminal amino acids labeled by X (resp., Y). There are only two possible labelings, which can be obtained one from the other by interchanging X and Y.

Algorithm 1 $LB(Protein)$

```
1: $l \leftarrow length(Protein)$;
2: $ener \leftarrow 0$;
3: for $i = 1$ to $l$ do
4:     $min_3 \leftarrow 0$;
5:     if odd $i$ then
6:         $P1 \leftarrow Remove\_odd\_position(Protein)$;
7:     else
8:         $P1 \leftarrow Remove\_even\_position(Protein)$;
9:     end if
10:    $P2 \leftarrow Remove\_elem\_in\_pos(P1, \lfloor \frac{i}{2} \rfloor)$;
11:    $P3 \leftarrow Remove\_elem\_in\_pos(P2, \lfloor \frac{i}{2} \rfloor)$;
12:    $(min_1, Pos_1) \leftarrow FM(P3, Protein[i])$;
13:    $P4 \leftarrow Remove\_elem\_in\_pos(P3, Pos_1)$;
14:    $(min_2, Pos_2) \leftarrow FM(P4, Protein[i])$;
15:    if $i = 1$ or $i = l$ then
16:        $P5 \leftarrow Remove\_elem\_in\_pos(P4, Pos_2)$;
17:        $(min_3, Pos_3) \leftarrow FM(P5, Protein[i])$;
18:    end if
19:    if $min_1 > 0$ then
20:        $min_1 \leftarrow 0$; $min_2 \leftarrow 0$; $min_3 \leftarrow 0$;
21:    else if $min_2 > 0$ then
22:        $min_2 \leftarrow 0$; $min_3 \leftarrow 0$;
23:    else if $min_3 > 0$ then
24:        $min_3 \leftarrow 0$;
25:    end if
26:    $ener \leftarrow ener + min_1 + min_2 + min_3$;
27: end for
28: return $ener/2$.
```

The authors select the labeling that guarantees that either $\mathcal{N}_X(s) < \mathcal{N}_Y(s)$ or $\mathcal{N}_X(s) = \mathcal{N}_Y(s) \wedge \#term_X(s) \leq \#term_Y(s)$. Since every contact connects an X amino acid with a Y amino acid, it

[3]As a matter of fact, in the case of HP sequences, one can obtain an algorithm to compute the exact minimum energy value by adapting that for three-dimensional lattices proposed in [8]. However, such an algorithm, which is based on the notion of "core", cannot be easily generalized to the case of sequences on the 20-letter alphabet of amino acids, and thus we decided to pursue a different solution.

follows that the maximum number of contacts equals the number of possible topological neighbor positions of X amino acids. Thus, a lower bound to the minimum energy value is given by the opposite of $(c_L - 2)\mathcal{N}_X(s) + (c_L - 1)\#term_X(s)$ $(2\mathcal{N}_X(s) + 3\#term_X(s)$ in the case of the square lattice).

The above algorithm can be generalized to compute a lower bound to the minimum energy value for sequences on the 20-letter alphabet of amino acids over the square lattice. As in the case of the HP model, it can take advantage of the following constraints: (i) every amino acid cannot form contacts with its predecessor and successor amino acids, (ii) every interior (resp., terminal) amino acid can form at most 2 (resp., 3) contacts, and (iii) if two amino acids form a contact, then one of them is in an odd position and the other in an even position of the sequence. As a matter of fact, such constraints do not allow us to exclude all non-admissible contacts. As an example, both the contact between amino acids in positions 1 and 4 and that between amino acids in positions 2 and 5 satisfy the above constraints and thus are admissible; however, it can be easily checked that they cannot occur in the same sequence. Moreover, since the considered 20×20 potential matrix contains both positive and negative energy values, when looking for the minimum energy value that the contacts of a given amino acid can contribute, we can detect a positive value. In the search for a lower bound, we must thus take into account that, in order to minimize the sequence energy value, it is better to avoid a contact than to insert a contact that contributes a positive value. For this reason, we replace every positive value with 0.

Algorithm 2 $FM(P, A)$

```
1: $k \leftarrow length(P)$;
2: $m = energy(A, P[j]) \leftarrow min\{energy(A, P[1]), \dots, energy(A, P[k])\}$;
3: return $m, j$.
```

The proposed algorithm processes the whole input protein (primary structure), searching for, for every interior (resp., terminal) amino acid, the 2 (resp., 3) amino acids that contribute the minimum energy value when forming a contact with it (obviously, we must exclude the predecessor and successor amino acids as well as all amino acids in a position with different parity). The output of the algorithm is the half of the summation of all the energy contributions found in this way (every contact must contribute only once to the summation). A pseudo-code version of such a solution is provided by the algorithm *LB(Protein)*.

Algorithm *LB(Protein)* exploits the function *Remove_odd_position(Protein)* (resp., *Remove_even_position(Protein)*) to remove all the amino acids of the input protein in odd (resp., even) positions. Moreover, it uses the function *Remove_elem_in_pos(P, Pos)* to remove the element of P in position *Pos*. At line 10 (resp., 11), it takes advantage of such a function to remove the predecessor (resp., successor) of the amino acid under consideration. At line 13, the same function is used to remove the amino acid that contributes the minimum energy value when in contact with the considered one, in order to find the amino acid that contributes the minimum energy value among the remaining ones. In case of terminal amino acids (line 15), the function must be applied once more (line 16) to determine the third amino acid that contributes the minimum energy value when forming a contact with the considered one. Energy contributions are determined by the algorithm *FM(P, A)*. Given a list of amino acids P and an amino acid A, *FM(P, A)* returns the minimum energy value that A can generate with the amino acids in P and the position of the amino acid whose energy value generated by a contact with A is minimum.

The complexity of the algorithm *LB(Protein)* is $\mathbf{O}(n^2)$, where n is the length of *Protein*: it calls $2n + 2$ times, that is, $\mathbf{O}(n)$ times, the algorithm *FM(P, A)*, whose complexity is linear in n.

3.2.2 The computation of the minimum energy value

We now provide an algorithm, called OPT, that, given a sequence of amino acids $s = s_0 s_1 \dots s_n$, that is, the primary structure of a protein, calculates the minimum energy value that a possible conformation of s in the square lattice can generate. Once we computed such a value for short proteins, we will be able to give an estimate of the minimum energy value for proteins of any length. Then, we will use this estimate as an input to the model checker.

Algorithm 3 *OPT*

```
1: OPT(Primary,Energy):-
2:     constrain(Primary,Tertiary,Energy),
3:     labeling([ff,minimize(Energy)],Tertiary).
```

The algorithm *OPT* is written in SICStus Prolog and it takes advantage of the library *clpfd* to implement techniques of constraint programming over finite domains. It works as follows.

Algorithm 4 Constraint setting

```
1: constrain(Primary,Tertiary,Energy):-
2:     length(Primary,N),
3:     $M$ $is$ $2 \cdot N$, $N1$ $is$ $N-1$, $Max$ $is$ $2 \cdot N1$,
4:     length(Tertiary,M),
5:     domain(Tertiary,0,Max),
6:     starting_point(Tertiary,N1),
7:     alldifferent(Tertiary),
8:     contiguous(Tertiary),
9:     energy_constraint(Primary,Tertiary,Energy).
```

First of all, we constrain the protein coordinates to take their value within a $Max \cdot Max$ lattice, with $Max = 2 \cdot n$. Then, we force the coordinates of the first two amino acids to be (n, n) and $(n, n + 1)$, respectively, by means of the predicate *starting_point*. Next, we take advantage of the predicates *alldifferent*, that imposes suitable constraints that avoid the presence of loops (in order to obtain a self avoiding walk), and *contiguous*, that constrains the positions of consecutive amino acids in the lattice. Finally, we exploit the predicate *energy_constraint* to define a constraint that stores the energy of the given conformation as a function of its primary and tertiary structures: we constrain two amino acids to give an energy contribution (if and) only if they form a contact (in the case of the HP model, we impose the additional constraint that to contribute to the energy value the two involved amino acids must be two H amino acids).

Once all constraints have been set (constraint setting is accomplished by means of the predicate *constrain(Primary,Tertiary,Energy)*), we take advantage of the SICStus constraint solver to determine the protein conformation (tertiary structure) that minimizes the energy value. To this end, we make use of the predicate *labeling([ff,minimize(Energy)],Tertiary)*, where the variable *Energy* refers to the energy value and the variable $Tertiary$ refers to the tertiary structure of the conformation, that is, to its (X, Y) coordinates.

The experimental evaluation of the algorithm showed that the search becomes rather slow for proteins of length greater than 20. To make it possible to apply the algorithm to longer proteins, we developed a parallel version of it, which exploits the SICStus library *Linda* for process communication [20]. *Linda* distinguishes between a process that runs as a server and one or more other processes that run as clients. Processes communicates with sockets and support networks. The server acts as a blackboard process, that is, it manages a blackboard where clients can write (*out/1*), read (*rd/1*), and remove (*in/1*) data.

Parallelization of the OPT algorithm is achieved by providing a suitable partition of the search space and putting different processes in charge of the exploration of the resulting different portions of the space. Search processes basically have the same structure (that of the original algorithm OPT), but they differ one from the other in the values of a given subset of the variables representing the coordinates of the protein which are fixed in advance. More precisely, we decided to consider all possible ways of fixing the positions of the first 5 amino acids in the lattice. If we ignore conformations which are perfectly symmetric to other ones, such a choice results in 12 essentially different conformations (as an example, fixing the position of the first 6, instead of 5, amino acids would produce 31 different conformations). We assigned each of them to a different process OPT_i, with $i = 1, \dots 12$, and defined the energy value of the protein under consideration as the minimum among the energy values computed by the 12 processes.

Algorithm 5 *Producer*

```
1: produce:-
2:     out(minimum(10000, 0)),
3:     out(counter(0)),
4:     out(region_free).
```

From the point of view of the implementation, besides the server process, we take advantage of various client processes, namely, a producer, a consumer, and the above-mentioned 12 processes OPT_i. Furthermore, we use critical regions in order to prevent processes from either reading or writing inconsistent values. By means of a suitable script, we first launch the producer, then we launch the 12 processes OPT_i in parallel, and, once all of them have completed their execution, we launch the consumer: $Producer; OPT_1 || \dots || OPT_{12}; Consumer$.

The process *Producer* places the tuple *minimum(10000, 0)* in the tuple space of *Linda* to initialize the minimum energy value and the number/identifier of the process that found it. Then, it initializes a counter that keeps track of the number of processes that completed their execution to 0 (the value of such a counter becomes 12 at the end of the execution of all parallel processes). Finally, it puts *region_free* in Linda tuple-space to let processes enter the critical region.

Algorithm 6 OPT_i

```
1: opt(Primary):-
2:     constrain_i(Primary, Tertiary, Energy),
3:     in(region_free),
4:     rd(minimum(X,_)),
5:     out(region_free),
6:     Energy #< X,
7:     labeling([ff,minimize(Energy)],Tertiary),
8:     in(region_free),
9:     in(minimum(Min,Proc)),
10:    comp(Min,Energy,Newmin,Proc,Newproc),
11:    out(minimum(Newmin,Newproc)),
12:    in(counter(C)), C1isC+1, out(counter(C1)),
13:    out(region_free).

14: opt(_):-
15:    in(region_free),
16:    in(counter(C)), C1isC+1, out(counter(C1)),
17:    out(region_free).
```

Each process OPT_i, with $i = 1, \dots, 12$, starts its operation with the constraint setting phase, which is basically the same as that of the sequential process *OPT*, except for the predicate *starting_point* which assigns a value to the coordinates of the first 5 amino acids in the lattice. Next, it enters the critical section, it reads the current minimum energy value X, it exits the critical region, and it sets the constraint $Energy \ \#< X$. If such a constraint cannot be satisfied, it increments by one the counter that records the number of processes that already ended their execution (lines 14–17). Otherwise, it proceeds with the search for the minimum energy value through labeling; at the end of the search, it enters again the critical region. It removes from Linda tuple-space the current minimum energy value *Min* (and the number/identifier *Proc* of the process that determined it) and it compares the energy value it found (*Energy*) with this value (notice that such a value can be different from the one it read before). The predicate *comp* at line 10 returns the minimum between the two values (*Newmin*) and the process identifier/number this minimum comes from (*Newproc*), which is either i (if *Energy #< Min*) or *Proc* (if *Energy #>= Min*). Before exiting the critical region, it puts *Newmin* and *Newproc* in the Linda tuple-space and it increments the counter of the processes that already ended their execution by one.

The process *Consumer* takes from the Linda tuple-space the minimum energy value, the identifier/number of the process that computed it, and the value of the process counter (that must be equal to 12). It concludes its computation by removing $region\_free$ from the Linda tuple-space.

Algorithm 7 *Consumer*

```
1: consume:-
2:     in(minimum(Energy, Process)),
3:     in(counter(C)),
4:     in(region_free).
```

We conclude the section by showing that the lower bound to the minimum energy value we compute for sequences of amino acids over the 20-letter alphabet is effective. For the sake of simplicity, we refer to the sequential algorithm OPT.

Proposition 3.2.1 *Let P be a sequence of amino acids over the 20-letter alphabet $\mathcal{A}$. If Opt (resp., Lb) is the value computed by $OPT(P)$ (resp., $LB(P)$), then Lb $\leq$ Opt.*

Proof. We first prove that, for each amino acid x in P, the energy contribution generated by its contacts computed by the algorithm $LB(P)$ is less than or equal to the one generated by its contacts in the conformation found by the algorithm $OPT(P)$. Without loss of generality, we may assume x to be an interior amino acid. In the following, we write $FM_1(A) \rightarrow B$ (resp. $FM_2(A) \rightarrow B$) to state that if we launch the algorithm $FM(Protein, A)$ for the first (resp. second) time in order to find the amino acid that has best (resp. second best) energy with A, the output amino acid is B.

In the conformation found by $OPT(P)$, x can form 0, 1, or 2 contacts. Such contacts can be either positive or negative. The most interesting case is the one where x forms 2 contacts, say c_1 and c_2, with $c_1 \leq c_2 < 0$. Let us suppose $FM_1(x) \rightarrow y$ and $FM_2(x) \rightarrow z$, with $energy(x, y) = e_1$ and $energy(x, z) = e_2$. Then, by definition of minimum, it follows that $e_1 \leq c_1$ and $e_2 \leq c_2$, and thus $e_1 + e_2 \leq c_1 + c_2$. The other cases are trivial, because, whenever the algorithm $LB(P)$ finds a positive minimum energy value, it replaces such a value with 0.

To complete the proof, it suffices to observe that, in order to compute the total energy value, the algorithm $LB(P)$ inserts 2 (resp., 3) energy values for each interior (resp., terminal) amino acid. It follows that the total energy value is given by $2 \cdot (n-2) + 3 \cdot 2 = 2 \cdot (n+1)$ energy contributions, where n is the number of amino acids in P. On the other hand, since each contact involves a pair of amino acids, in any real conformation where every interior (resp., terminal) amino acid forms 2 (resp., 3) contacts there are $n + 1$ energy contributions. It immediately follows that, even if the algorithm $LB(P)$ divides the total energy value it determines by 2, the resulting value *Lb* is still lower than or equal to *Opt*.

3.2.3 Experimental results

We extensively tested our approach to the estimate of protein minimum energy values on different proteins. In Figure 3.5, we report the results of our tests for sequences over the 20-letter alphabet of amino acids.

We applied the proposed algorithms to 4 sets of 10 proteins of increasing length, namely, 10, 15, 20, and 25. For each protein, we computed both the minimum energy value and the lower bound and we determined the ratio between these two values. Then, for each set of proteins, we computed the average of such a ratio, whose value seems to stabilize at 0.55.

For proteins of length 20 and 25, we took advantage of the parallelized version of algorithm OPT by decomposing the original single process in 12 distinct ones and running the parallel

Protein length	Opt/Lb
10	0.557
15	0.553
20	0.565
25	0.553

Figure 3.5: Ratio between the minimum energy value and the corresponding lower bound.

Process #	Min energy value	Execution time (s)
1	-17743	8746
2	-17757	7481
3	-18001	7976
4	-17784	8033
5	-17576	5295
6	-17553	6734
7	-17784	9569
8	-17500	8135
9	-17500	8198
10	-17601	8468
11	-17757	7165
12	-19176	8983

Figure 3.6: Parallel computation of the minimum energy value for the protein [a,k,l,c,h,t,l,e,-n,i,l,n,k,a,r,n,s,e,i,k,i,t,s,d,l].

search on an AMD Opteron 2.2 GHz Linux machine with 12 processors. In Figure 3.6 we show the outcomes of the execution of such a parallel algorithm on the protein $[a, k, l, c, h, t, l, e, n, i, l, n, k, a, r, n, s, e, i, k, i, t, s, d, l]$. For each process, the table contains the minimum energy value it computed and its execution time. The resulting minimum energy value is -19176, which is returned by *process 12*. As for the execution times of the 12 processes, they turned out to be quite balanced.

3.3 Model checking properties of proteins

In this section, we propose an approach to the formal verification of interesting properties of protein conformations based on model checking [27]. Basically, model checking allows one to verify desirable properties of a system by an exhaustive enumeration of all reachable states. We model the set of protein conformations and their relationships as a finite transition system and we use (linear and branching) propositional temporal logic to specify relevant properties of them [44, 109]. To face the problem of the combinatorial explosion of the state space, we will take advantage of a technique, called on-the-fly model checking, that allows us to avoid to build the entire space state, whenever possible.

3.3.1 Protein transition systems

First, we show how to encode the possible conformations of a protein in terms of the possible states of a transition system. A transition system can be defined as follows.

Definition 3.3.1 (Transition System) *Let AP be a finite set of atomic propositions. A* transition system *over AP is a tuple $M = (Q, T, L)$, where*

- *Q is a finite set of states;*

- *$T \subseteq Q \times Q$ is a total transition relation, that is, for every state $q \in Q$ there is a state $q' \in Q$ such that $T(q, q')$;*
- *$L : Q \to 2^{AP}$ is a labeling function that maps every state into the set of atomic propositions that hold at it.*

The 2D Protein Transition System is defined as follows.

Definition 3.3.2 (2D Protein Transition System) *Let p be a sequence in $\{0,1\}^{n+1}$ (resp., $\mathcal{A}^{n+1}$) and let AP be the set of atomic propositions consisting of the $3 \cdot (n-1)$ predicates 2nd_l, ..., nth_l, 2nd_f, ..., nth_f, 2nd_r, ..., nth_r and the three predicates min_en, inter_en, and max_en. The 2D Protein Transition System for p is a tuple $M_p = (Q, T, L)$, where*

- *Q is the set of all conformations of length n on the $2n \times 2n$ 2D lattice;*
- *$T \subseteq Q \times Q$ contains the pairs of states (q_1, q_2) such that q_2 can be obtained from q_1 by a pivot move;*
- *$L : Q \to 2^{AP}$ is a labeling function such that for every $i = 2, \ldots, n$, the predicate ith_l (resp., ith_f, ith_r) holds at a state q if the i-th segment of q has a* left *(resp.,* forward, right*) orientation and min_en (resp., inter_en, max_en) holds at a state q if the energy of q is minimum (resp., intermediate, maximum).*

It is possible to prove that the 2D Protein Transition System for any given protein has the following properties.

Proposition 3.3.3 (Properties of the 2D Protein Transition System) *Let M_p be the 2D Protein Transition System for a sequence $p \in \{0,1\}^{n+1}$ (resp., $\mathcal{A}^{n+1}$). It holds that:*

1. *M_p is* strongly connected*, i.e., for each pair of states q_1 and q_2, there is a path from q_1 to q_2;*
2. *M_p is* symmetric*, i.e., for each pair of states q_1 and q_2, if (q_1, q_2) belongs to T, then (q_2, q_1) belongs to T;*
3. *the* maximum incidence degree *$D = \max_{q \in Q} |\{(q, q') : (q, q') \in T\}|$ is 2(n-1);*
4. *the number of states is $\boldsymbol{O}(2.63^n)$;*
5. *the diameter, that is, the length of longest path between each pair of states, is $\boldsymbol{O}(n^2)$.*

Proof. Property 1 follows from ergodicity of pivot moves [89, 90]. As for property 2, it is immediate to see that if a state q_2 can be obtained from a state q_1 by executing a pivot move, then q_1 can be obtained from q_2 by executing the opposite pivot move. Property 3 immediately follows from Definition 3.1.4. Property 4 follows from the result in [88], which states that the number of self-avoiding walks of length n in a 2D lattice is $\mathbf{O}(2.63^n)$. Finally, property 5 has been proved by Madras and Sokal in [89, 90].

As far as the energy of a protein is concerned, from our experimental results it turns out that, both in the HP model and in the 20×20 potential matrix, the majority of states has a high energy value and that only a small percentage of states has a minimum energy value.

3.3.2 Temporal logics for protein

Temporal logics are formalisms that allow one to specify (and to check) meaningful properties of systems that evolve over time. In particular, they make it possible to constrain sequences of transitions over states to satisfy specific conditions. In the following, we restrict our attention to two well-known fragments of the *computation tree logic* CTL*, namely, the *branching time* logic CTL and the *linear time* logic LTL [44].

CTL* formulae describe properties of computation trees and they are obtained by (repeatedly) applying Boolean connectives, *path quantifiers*, and *state quantifiers* to atomic formulae. Path quantifiers allow one to state that a property holds over all/some paths starting from a given state (all paths by using the path quantifier **A**, some path by using the path quantifier **E**). State quantifiers allow one to state that a property holds at all/some future states of a given path (all future states by using the "always in the future" operator **G**, some future state by using the "sometimes in the future" operator **F**). As a matter of fact, the operator **E** is the dual of **A** and the operator **F** is the dual of **G**; moreover, **G** (and thus **F**) can actually be defined in terms of the next time operator **X**, that allows one to state that a property holds at the next state (of the given path), and the until binary operator **U**, which holds at the current state (of the given path) if there exists a future state of the path where (the property encoded by) its second argument holds and, at every state in between the current state and such a future state, (the property encoded by) its first argument holds.

While CTL* does not constrain the way in which temporal quantifiers may occur in formulae, CTL (syntactically) forces two path quantifiers to be interleaved with one state quantifier. LTL basically encompasses state quantifiers only and thus it allows one to express properties of single computation paths. More precisely, LTL formulae are of the form $\mathbf{A}\varphi$, where φ does not contain path quantifiers, but it allows the nesting of state quantifiers. From a computational point of view, the complexity of model checking for CTL is linear in the number of states and edges of the transition system, while the model checking problem for LTL is PSPACE-complete. From the point of view of expressiveness, CTL and LTL are incomparable [44]. To benefit from the advantages of both of them, we decided to adopt the one or the other depending on the specific property to be checked (as a matter of fact, most properties of interest of Protein Transition Systems belong to the intersection of CTL and LTL). Finally, from the point of view of existing systems for automatic verification, there exists a number of tools for checking whether a given finite state system satisfis a CTL formula, e.g., SMV [92], while most algorithms for on-the-fly model checking have been developed for LTL [29, 49].

3.3.3 Checking relevant properties of proteins

Given a 2D Protein Transition System $M_p = (Q, T, L)$, a state $q \in Q$, and a temporal logic formula φ expressing some desirable property of the system, the (local) *model checking problem* consists in establishing whether φ holds at q or not, namely, whether $M_p, q \models \varphi$ or not. When a state does not satisfy a formula, model checking algorithms produce a counterexample that falsifies it, thus providing an insight to understand failure causes.

Many meaningful properties of 2D Protein Transition Systems can actually be encoded in CTL and/or LTL. In particular, these logics allow one to specify properties that describe the physical evolution of protein conformations, thus constraining the relationships among different conformations. For every property of interest φ, we impose a bound on the length of any path leading from the initial state to a state that satisfies φ (such a constraint is usually forced in *bounded* model checking [12]). Since in any 2D Protein Transition System any given state can be reached from any other state, as stated by Proposition 3.3.3, if we did not impose such a bound, any (reasonable) property would be trivially satisfied.

Below, we report the encoding of four paradigmatic properties of 2D Protein Transition Systems in CTL and LTL. They are the representatives of a larger set of properties that we took into consideration in our experiments.

F1: Does it exist a path of length at most k that (starting from the current state) reaches a state with minimum energy? Such a condition can be expressed in CTL as follows:

$$min\_en \vee EX min\_en \vee \ldots \vee \underbrace{EX \ldots EX}_{k} min\_en \equiv \bigvee_{i=0}^{k} (EX)^i min\_en$$

The negation of property **F1** can be expressed in LTL as follows:

$$A(\neg min\_en \wedge X \neg min\_en \wedge \ldots \wedge \underbrace{X \ldots X}_{k} \neg min\_en) \equiv A\left(\bigwedge_{i=0}^{k} X^i \neg min\_en\right)$$

F2: Is the energy of the current state the minimum one? If it is not the minimum but the maximum one[4], is it possible to reach in at most k steps a state with minimum energy without passing through any state with intermediate energy? Such a condition can be expressed by the CTL/LTL formula:

$$A(max\_en U_k min\_en),$$

where U_k is a shorthand for a "bounded"version of the until operator that must be satisfied in at most k steps (it can be easily defined by means of the next operator X).

F3: Is it possible to reach in one step a conformation where the first half of the sequence is a helix of the form $rrllrr\ldots$? This property (resp., the negation of this property) can be expressed in CTL (resp., LTL) by taking advantage of the predicates of the form ith_l and ith_r. In both cases, we must distinguish between the case in which $m = \lfloor n/2 \rfloor$ is odd and that in which it is even.
If m is odd, the CTL formula that specifies the condition is the following one:

$$EX\left(\bigwedge_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (ith\_r \wedge i+1th\_r) \wedge \bigwedge_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (ith\_l \wedge i+1th\_l)\right),$$

while in LTL the negation of the considered property can be expressed as follows:

$$AX\left(\bigvee_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (\neg ith\_r \vee \neg i+1th\_r) \vee \bigvee_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (\neg ith\_l \vee \neg i+1th\_l)\right)$$

If m is even, we must distinguish two cases. If $m = 2+4\cdot j$ (for some $j \geq 0$), the condition can be expressed in CTL as follows:

$$EX\left(\bigwedge_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (ith\_r \wedge i+1th\_r) \wedge \bigwedge_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (ith\_l \wedge i+1th\_l) \wedge mth\_r\right),$$

while the LTL encoding of its negation becomes:

$$AX\left(\bigvee_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (\neg ith\_r \vee \neg i+1th\_r) \vee \bigvee_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (\neg ith\_l \vee \neg i+1th\_l) \vee \neg mth\_r\right)$$

If $m = 4+4\cdot j$ (for some $j \geq 0$), the CTL formula that expresses the condition is:

$$EX\left(\bigwedge_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (ith\_r \wedge i+1th\_r) \wedge \bigwedge_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (ith\_l \wedge i+1th\_l) \wedge mth\_l\right),$$

while the LTL formula for its negation is:

$$AX\left(\bigvee_{j\geq 0, i=2+4\cdot j}^{i\leq m-1} (\neg ith\_r \vee \neg i+1th\_r) \vee \bigvee_{j\geq 0, i=4+4\cdot j}^{i\leq m-1} (\neg ith\_l \vee \neg i+1th\_l) \vee \neg mth\_l\right)$$

[4]In the HP model, the maximum energy value is 0; in the 20×20 model, an approximation of such a value can be computed by applying the same method we used to compute the minimum one (see Section 3.2).

F4: Has any state which is at most k steps far from the current one the maximum energy value? In CTL such a condition can be expressed as follows:

$$max\_en \wedge AX max\_en \wedge \ldots \wedge \underbrace{AX \ldots AX}_{k} max\_en \equiv \bigwedge_{i=0}^{k} (AX)^i max\_en,$$

while it can be expressed in LTL as follows:

$$A(max\_en \wedge X max\_en \wedge \ldots \wedge \underbrace{X \ldots X}_{k} max\_en) \equiv A\left(\bigwedge_{i=0}^{k} X^i max\_en\right)$$

3.4 A model checker to verify properties of protein conformations

In this section, we describe the on-the-fly model checker we developed to detect significant properties of protein conformations to be exploited in solving the protein structure prediction and protein folding problems. As a first step, we tried to adapt traditional model checking techniques and tools to our application domain. Unfortunately, the various alternative solutions we explored turned out to be impracticable. To overcome their limitations (in particular to cope with their inefficiency), we resort to an ad-hoc on-the-fly model checker.

3.4.1 Some unsatisfactory solutions

As a first step, in order to verify the feasibility of the model checking approach to the analysis of the properties of protein conformations, we developed some prototypes of CTL and LTL model checkers in SICStus Prolog and we experimented them on some simple test cases. More precisely, we developed an algorithm to encode the set of possible conformations of a protein, and the binary relation over it induced by the pivot move, as a 2D Protein Transition System. Then, we implemented model checking algorithms for CTL and LTL to verify whether any given 2D Protein Transition System satisfies or not relevant properties of protein conformations, including those specified by formulae F1-F4. The goal of this solution was essentially exploratory, that is, it was aimed at understanding the distinctive features of transition systems for 2D proteins as well as at identifying the better way of specifying and checking their relevant properties. As an example, for most 2D Protein Transition Systems we showed the existence of states with an intermediate energy value that may reach in one step a state with a maximum energy value which, in its turn, may reach in one or more steps (how many, it depends on the length of the considered protein) a state with a minimum energy value. This allows us to conclude that, in order to reduce the number of edges of the 2D Protein Transition System, it is not sound to cut edges connecting states where the energy value of the source state is lower than the energy value of the destination state, because from the destination state it is often possible to reach states with a minimum energy value in a very few steps.

The major weakness of such a solution, as expected, was that we had to confine ourselves to model check transition systems for proteins whose length was at most 10. A protein of length n, indeed, gives rise to a transition system with a number of states equal to $\mathbf{O}(3^{n-1})$, that presents obvious time and space problems from the point of view of computational complexity. To cope with these problems, we took into consideration *symbolic model checking* [92]. Symbolic model checking has been proposed as a possible way out to the state-explosion problem. It basically replaces fixed-point calculation over individual states by manipulations of suitable (compact) definitions of sets of state based on Ordered Binary Decision Diagrams (OBDDs). In the worst case, the original transition system and its OBDD representation have the same size; however, this is usually not the case whenever the system presents some "regularities". We provided an encoding of 2D Protein Transition Systems in the input language required by the symbolic model checker NuSMV [26] and we tested properties F1-F4. Unfortunately, we did not achieve any significant improvement (we were still able to deal with proteins of length at most 10 only). This was probably due to the lack of regularities in the considered transition systems.

An alternative approach to the state-explosion problem is that of on-the-fly model checking [29, 49]. As already pointed out, in many cases it allows one to avoid the construction of the entire state space of the system by letting the property to check drive the construction of the system. In particular, when a state that falsifies the considered property is reached, the construction is stopped. Only in the worst case, that is, when the property is satisfied, the entire system must be explicitly built. We provided an encoding of 2D Protein Transition Systems in Promela, the specification language of the on-the-fly model checker Spin [68], and we tested (the LTL formulae for) properties F1-F4. Once more, we did not achieve any significant improvement (we were still unable to go beyond the above threshold for protein length). A possible explanation of this failure is that we cannot fully exploit the Spin mechanism for message exchange at the user's disposal.

3.4.2 A viable alternative: an ad-hoc on-the-fly model checker

In this section we describe an on-the-fly model checker that we developed to formally specify and verify the properties of 2D Protein Transition Systems. It is written in SICStus Prolog and it benefits from some distinctive features of logic programming, such us backtracking and nondeterminism.

The proposed on-the-fly model checker basically executes a depth-first visit of the conformation space. Such an exploration strategy presents several advantages. First, the properties of conformations we are interested in can be expressed by formulae of the form $prop \wedge(\vee) EX\ prop \ldots \wedge(\vee) EX \ldots EX\ prop$, where *prop* is either an atomic proposition or a conjunction/disjunction of atomic propositions, which can be directly checked by means of a depth-first visit. Moreover, depth-first visit makes it easy to record the path leading to a state that satisfies the property under consideration. Such a path contains pieces of information that can be exploited to understand the properties (and the dynamics) of protein conformations. Finally, a natural implementation of a depth-first visit strategy can be obtained by taking advantage of backtracking.

In the following, we describe the pseudo-code of our algorithm (we consider its instantiation dealing with property F1, but the structure of the resulting instance remains essentially the same for the majority of the properties we are interested in).

Algorithm 8 *On_the_fly_model_checker*

```
1: fly(Node, Primary, K, Min_en) :-
2:     length(Node,N),
3:     coord_noloop(Node,N,C),
4:     energy_constraint(Primary,C,E),
5:     exist([Node,E],Primary,K,Min_en,N,[[Node,E]],Path),!,
6:     write('The node satisfies the property. Witness path: '),
7:     write_reverse(Path).

8: fly(_ ,_ ,_,_):-
9:     write('The node does not satisfy the property').
```

The input of the main predicate *fly* consists of (i) the conformation of the protein we are interested in, expressed as a list on the alphabet $\{l, f, r\}$, that is, a node of the transition system for the protein; (ii) the primary structure of the protein, that is, a list of amino acids either on the alphabet $\{H, P\}$ or on the 20-letter alphabet $\mathcal{A}$; (iii) a bound k on the number of steps within which the property under consideration must be satisfied (since the transition system for the protein is ergodic, as stated by Proposition 3.3.3, any state can be reached from any other one and thus reachability must be replaced with bounded reachability, that is, we must introduce an upper bound to the length of the path from the input state to a state that satisfies the considered property, if any); (iv) an estimate of the minimum energy value for the considered protein, computed by means of the technique described in Section 3.2 (such a value can possibly be ignored or replaced with another one in order to test properties different from F1).

Predicate *fly* behaves as follows. First, it computes the length N of the input node, that is, the number of labels it is made of. Then, it determines the coordinates of the conformation on the 2D lattice and it exploits them, together with information about the primary structure of the protein, to compute the energy of the conformation. Next, it executes predicate *exist* that checks the existence of a path of length at most k from the input state to a state with minimum energy. In case of success, it returns a witness path as output and predicate *fly* concludes its execution by printing it. In case of failure, predicate *fly* terminates its execution by printing a message notifying that the property is not satisfiable starting from the given input node.

Algorithm 9 *Exist*

```
exist(Conformation,_,_,Min_en,_,Visitednodes,Visitednodes):-
    prop(Conformation,Min_en),!.

exist([Node,_],Primary,K,Min_en,N,Visitednodes,Path):-
    $K > 0$, $N1$ is $N - 1$, $K1$ is $K - 1$,
    $I$ in $1 \dots N1$, indomain($I$),
    succ(Node,I,Succ),
    coord_noloop(Succ,N,C),
    notmember(Succ, Visitednodes),
    energy(C,Primary,Esucc),
    exist([Succ,Esucc],Primary,K1,Min_en,N,[[Succ,Esucc] | Visitednodes],Path).
```

Algorithm 10 *Succ*

```
succ([A|Innode], I, [A|Outnode]) :-
    $I > 1$, !, $I1$ is $I - 1$,
    succ(Innode,I1,Outnode)

succ([E|Innode], 1, [N|Innode]) :-
    ($E = s \rightarrow (N = l; N = r)$;
    $E = l \rightarrow (N = r; N = s)$;
    $E = r \rightarrow (N = l; N = s)$).
```

Predicate *exist* actually encodes and checks the property specified by the temporal logic formula (in our example, property F1) on the given protein transition system. It receives as input a conformation *Conformation* of a given protein, its energy, its primary structure, a bound K on the number of allowed steps, an estimate *Min_en* of its minimum energy value in the 2D lattice, the length N of the conformation, and the list *Visitednodes* of the already visited nodes. If the current conformation satisfies predicate (*prop(Folding,Min_en)*), that is, if the conformation energy is (approximately) equal to *Min_en*, then predicate *exist* returns the list *Visitednodes* of the nodes visited to reach the current one. Otherwise, it proceeds as follows. First, it checks whether $K > 0$. If this is the case, it nondeterministically selects a position I in $\{1, \dots, N-1\}$. Then, it exploits predicate *succ* to compute a successor *Succ* of the current configuration *Configuration*. *Succ* is obtained from *Configuration* through a pivot move that changes the label in position I. The choice of the label to insert (each label in $\{l, f, r\}$ can be replaced with two other ones, e.g., the label l can be replaced with f or r) is nondeterministic as well. Then, predicate *exist* computes the coordinates of the successor node *Succ*, it checks whether it is a conformation (that is, it includes no loops) and it does not belong to the list of visited nodes *Visitednodes*, and it computes its energy value *Esucc*. The execution of predicate *exist* terminates with a recursive call with the following input: the node *Succ*, its energy *Esucc*, the primary structure of the protein, the bound

$K-1$, the estimate *Min_en* of the minimum energy value for the considered protein, the length N of conformations, and the list of visited nodes *Visitednodes* extended with the node *Succ*. The result will be returned as the value of the output variable *Path* (the output variable of the original goal).

3.4.3 Properties and refinements of the proposed model checker

In this section we analyze the distinctive features of the proposed on-the-fly model checker and we discuss some refinements of it.

An ingredient that contributes to the efficiency of the proposed solution is the use of a single predicate (predicate *coord_noloop*) to compute the coordinates of a given node and to check whether it is a conformation (that is, it does not contain loops). New coordinates are added dynamically to the Prolog database. The predicate computes the coordinates of nodes one at a time and every time it determines a new coordinate, that is, a new abscissa/ordinate pair, it checks whether or not it was already in the database. If the new coordinate matches an existing one, the predicate fails (the given conformation contains a loop); otherwise, it adds the new coordinate to the database and then it focuses its attention on the next coordinate to determine. In such a way, the predicate computes coordinates and checks conformations for loops in a time which is linear in the length of the input node. Furthermore, as soon as it detects the presence of a loop, it stops the computation, thus avoiding to compute the complete set of coordinates.

To choose the position where label substitution must be applied during the computation of the successor of a given node, we use the *clpfd* predicate *indomain*. When both the nodes obtained from such a replacement lead to failure, Prolog exploits backtracking to choose a different position for label substitution. To speed up the computation, we developed an alternative version of the model checker that takes advantage of a random choice, in place of predicate *indomain*, that does not allow backtracking in case of failure. This makes the procedure sound, but incomplete: on the one hand, when the answer is positive, we have a remarkable time acceleration; on the other hand, if at the end of the computation a property turns out to be not satisfied, we cannot trust such an outcome (and thus we must resort to the basic procedure).

Among the various property of protein transition systems that the proposed model checker allows one to verify, we would like to focus our attention on those stating the existence of a path connecting two given configurations whose length is less than or equal to a given threshold. To check properties in this class, it suffices to slightly modify the definition of predicate *exist* as follows.

Algorithm 11 *Exist* (modified version)

```
exist(Node1,Node2,K,_,Visitednodes,Visitednodes,K):-
    Node1=Node2,!.

exist(Node1,Node2,K,N,Visitednodes,Path,Krem):-
    K > 0,
    random(1,N,I),
    succ(Node,I,Succ),
    coord_noloop(Succ,N,_),
    notmember(Succ, Visitednodes),
    K1 is K − 1,
    exist(Succ,Node2,K1,N, [[Succ| Visitednodes],Path, Krem).
```

The new instance of predicate *exist* receives as input two conformations *Node1* and *Node2*, a bound K on the number of steps, the conformation length N, and the list *Visitednodes* of the already visited nodes. If predicate *exist* succeeds, it returns as output a witness path and the number of remaining steps *Krem* (the effective length of the path is thus K − *Krem*). The base case consists in testing whether the two input conformations *Node1* and *Node2* coincide. If this is the case, predicate *exist* returns the list *Visitednodes* as the witness path and K as the number of remaining steps. If this is not the case, predicate *exist* computes a successor *Succ* of *Node1*, then it checks whether *Succ* is a conformation which has not been visited yet, and, finally, it recursively calls predicate *exist* with the following input: *Succ*, *Node2*, $K - 1$, N, and the list *Visitednodes* extended with node *Succ*. To speed up execution time, we can replace predicate *indomain* with predicate *random* as explained above.

The problem of computing a sequence of steps leading from a given conformation to another one is a classical one in protein folding. As mentioned in Section 3.1, Madras and Sokal proved that it is always possible to find a path of length at most n^2 connecting any two given conformations, where n is the conformation length. We addressed such a problem by model checking bounded reachability properties for various proteins, taking into consideration different values for the bound k. More precisely, we started model checking by assigning the value n^2 to k and then we repeatedly halved such a value, until we were not able to find a path of length k between the two conformations under consideration. The outcome of our experimentation can be summarized as follows: for any protein transition system and any pair of its conformations, we have been able to find a linear connecting path instead of a quadratic one (as a matter of fact, the possibility of strengthening the n^2 upper bound was already conjectured by Madras and Sokal in [90]).

The last feature of the proposed model checker that we would like to mention is its ability to take advantage of those parts of proteins whose structure remains unchanged during the evolution process that leads proteins to their native conformation. These portions of proteins often assume a helix form (see Section 3.1). We incorporated in the model checker information about helices of the form *(frrfll)*$^+$, that is, helices consisting of one or more repetitions of the string *frrfll*. To impose the preservation of these structures during the folding process, we excluded the corresponding positions from the ones that can be affected by a label substitution when calculating the successor of a given conformation. To this end, it suffices to include in the input for the model checker a list of intervals identifying those protein portions that cannot change during the folding process, e.g., by associating the list $[[7, 24], [49, 72], [85, 90]]$ with a given protein (conformation), we mean that substrings in *(frrfll)*$^+$ occur in the intervals 7-24, 49-72, and 85-90 and thus, when looking for a label to substitute, we cannot choose labels in those positions, and to modify predicate *exist* accordingly (if n is the length of the conformation and $[[a_1, b_1], \ldots, [a_j, b_j]]$ is the list of preserved intervals, natural numbers in $[a_i, b_i]$, with $1 \leq i \leq j$, are removed from the domain over which the variable I takes its value). Notice that while the number of successors of a conformation in the general (unrestricted) case is $2 \cdot n$, in the presence of the preserved intervals $[[a_1, b_1], \ldots, [a_j, b_j]]$ it becomes $2 \cdot (n - (b_1 - a_1 + 1) - \ldots - (b_j - a_j + 1))$, which results in a significant improvement of the performance.

3.5 Experimental evaluation

We experimented the on-the-fly model checker we developed on some (fragments of) proteins selected from the Protein Data Bank (PDB), a repository containing information about known proteins and nucleic acids. Data in PDB, typically obtained by X-ray crystallography or NMR spectroscopy, are submitted by biologists and biochemists from all around the world and can be accessed for free.

We checked a large number of meaningful properties of conformations, which can be expressed by relatively simple formulae, and the outcomes of the experiments are quite promising. In particular, we have been able to verify basic properties of conformations for proteins whose length is greater than 400. As an example, given a protein whose length is in between 50 and 450 and whose primary structure is given with respect to the 20-letter alphabet of amino acids, the model checker has been able to establish in a few milliseconds whether or not it is possible to reach in less than 10 steps a conformation with energy value lower than -1000 starting from the all-straight conformation fd_0 (fd_0 is the conformation *fff . . . fff*). Figure 3.7 reports the model checker execution time for proteins of increasing length (from 50 to 450). In addition, in case of success, the model checker outputs a witness path. In achieving such a performance, the fact that the model checker is on-the-fly plays a fundamental role: without such a capability (or an equivalent one), to

Protein length	Execution time (ms)
50	0
100	15
150	32
200	47
250	94
300	125
350	141
400	203
450	250

Figure 3.7: Model checking execution time for proteins of increasing length.

Energy value	Execution time (ms)
-1000	0
-5000	78
-10000	703
-15000	1265
-20000	1875
-25000	2313
-30000	2906

Figure 3.8: Energy value and model checking execution time.

deal with a large protein, to say, of length 400, a model checker would need to build a transition system with $\mathbf{O}(3^{400})$ states, which constitutes a barrier to the possibility of testing any reasonable property on proteins of a significative length.

We also compared the execution time of model checking on proteins with and without unchangeable parts. In various cases, constraining some parts to be preserved results in a significative improvement of the performance; however, this is not always the case. Preventing a protein from assuming some specific conformations may indeed force the model checker to follow long alternative paths to reach a conformation which satisfies a given property.

As a general rule, the more complex the property to check is, the higher the execution time is. As an example, suppose that we want to reach a configuration with an energy value E close to the minimum one starting from the all-straight conformation fd_0[5]. We have that the closer to the minimum energy value E is, the higher the computation time is. In case of proteins with a considerable length, the resulting degradation in performance is not negligible. In Figure 3.8, we report the execution times needed to check the existence of a path of length at most 10 from the initial conformation fd_0 to a conformation with an energy value lower than a threshold E, with E varying from -1000 to -30000, for a protein of length 50 whose minimum energy value is approximately equal to -45000.

3.6 Conclusions and future work

In this chapter, we argued that model checking can be successfully exploited in solving classical problems such as the protein structure prediction and protein folding problems. The model checking machinery can indeed be used to establish meaningful properties of protein conformations. Such

[5]We can reasonably assume that if E is sufficiently close to the minimum energy value, then the corresponding configuration is sufficiently close to that/those of minimum energy, as witnessed by the cases in which we actually know the configuration(s) with minimum energy value.

properties can then be encoded as additional constraints that constraint solvers dealing with the above-mentioned problems can use to reduce the solution space.

The technical contribution of the work can be summarized as follows. First, we addressed and solved the problem of providing an estimate of the minimum energy value of a protein. Then, for any given protein, we showed how to build a transition system whose states represent its possible conformations in the 2D-lattice and whose transitions model transformations between pairs of valid conformations. Finally, to verify whether the resulting transition system satisfies some meaningful properties expressed by temporal logic, we took advantage of on-the-fly model checking. More precisely, we developed an on-the-fly model checker in SICStus Prolog and we extensively experimented it on a number of representative proteins and properties.

As for future work, besides further improving the evaluation of the strength and limitations of the proposed model checker by applying it to additional proteins and/or temporal logic formulae, our current research is developing in two different directions. On the one hand, we are trying to improve the performance of the model checker by parallelizing its operation as much as possible. On the other hand, we are integrating it with an existing constraint solver, namely, COLA [34], to measure its benefits on concrete instances of the protein structure prediction and protein folding problems.

4
The sequence alignment problem

In nature, new sequences are adapted from pre-existing sequences rather then invented de novo. This is very fortunate for computational sequence analysis. We can often recognize a significant similarity between a new sequence and a sequence about which something is already known; when we do this we can transfer information about structure and/or function to the new sequence. We say that the two related sequences are *homologous*.

At at first glance, deciding that two biological sequences are similar is no different from deciding that two text strings are similar. One set of methods for biological sequence analysis is therefore rooted in computer science, where there is an extensive literature on string comparison methods [39, 60]. The concept of *alignment* is crucial. Evolving sequences accumulate insertions and deletions as well substitutions, so before the similarity of two sequences can be evaluated, one typically begins by finding a plausible alignment between them.

Almost all alignment methods find the optimal alignment between two strings under some *scoring model*. Since we want a scoring model to give the biologically most likely alignment the highest score, we must take into account the fact that biological molecules have evolutionary histories, three-dimensional folded structures, and other features which constrain their primary sequence evolution.

In this chapter we will present the state of the art relative to sequence alignment. We will start introducing algorithms for pairwise alignment and then we will pass to the construction of multiple alignments of families of sequences.

4.1 Pairwise alignment

In this section we deal with the problem of deciding if a pair of sequences is evolutionary related or not. We examine traditional pairwise sequence alignment and comparison algorithms which use dynamic programming to find optimal alignments. The basic mutational processes that are considered are *substitutions*, *insertions*, and *deletions*. Insertions and deletions are together referred to as *gaps*. In Figure 4.1 there is a possible alignment of two protein sequences. Identical positions are indicated with letters, similar positions[1] with a plus sign, and gaps with a minus sign.

[1]Similar pairs of residues are those which have a positive score in the substitution matrix used to score the alignment; we will briefly introduce substitution matrices later.

```
HBA_HUMAN   GSAQVKGHGKKVADALTNAVAHV---D--DMPNALSALSDLHAHKL
            ++ ++++H+ KV   + +A  ++          +L+ L+++H+ K
LGB2_LUPLU  NNPELQAHAGKVFKLVYEAAIQLQVTGVVVTDATLKNLGSVHVSKG
```

Figure 4.1: An alignment example.

4.1.1 The scoring model

Models for alignment evaluations are commonly based on the following assumptions:

- mutations in different sites of a sequence are independent;
- conservative mutations are more likely in related sequences and less probable in random alignments;
- the length of a gap is not related with elements aligned with the gap.

Consequently, the total score to assign to an alignment will be a sum of terms for each aligned pair of residues, plus terms for each gap.

Substitution matrices

First of all, let us consider the global alignment without gaps of two sequences with the same length. Let $x = x_1 \dots x_n$ and $y = y_1 \dots y_n$ be two strings on an alphabet Σ, where Σ consists of the four bases in the case of DNA and the twenty amino acids in the case of proteins. We introduce two probabilistic models, one for unrelated sequences (*random model*) and one for related sequences (*match model*).

The random model R assumes that each $a \in \Sigma$ occurs independently with some frequency q_a, hence the probability that x and y are aligned is given by the product of the frequencies of single elements:

$$Pr(x, y|R) = \prod_{i=1}^{n} q_{x_i} q_{y_i} \tag{4.1.1}$$

The alternative match model M assumes that each coupling of elements a and b occurs with a joint probability p_{ab}. The whole alignment has the following probability:

$$Pr(x, y|M) = \prod_{i=1}^{n} p_{x_i y_i} \tag{4.1.2}$$

Thus it is possible to calculate and compare the probability that a given alignment occurs by chance and the probability the alignment points out a relation between the two sequences. In order to arrive at an additive scoring system, we take the logarithm of the ratio of the probabilities relative to the two models:

$$S(x, y) = \log \prod_{i=1}^{n} \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} = \sum_{i=1}^{n} \log \frac{p_{x_i y_i}}{q_{x_i} q_{y_i}} = \sum_{i=1}^{n} s(x_i, y_i), \tag{4.1.3}$$

where $s(x_i, y_i) = \log[p_{x_i y_i}/(q_{x_i} q_{y_i})]$. Each $s(a, b)$ represents the score associated to the alignment of elements a and b. The $s(a, b)$ scores can be arranged in a matrix, the *substitution matrix.* In Figure 4.2 there is the BLOSUM50 substitution matrix, which is very used in implementations. Positive scores are associated to likely coupling in related alignments while negative scores correspond to coupling which often occur by chance.

Gap functions

As far as gaps are concerned, associated scores must be negative. The score of a gap of length g is usually calculated in one of the two following ways:

$$\gamma_1(g) = -gd \tag{4.1.4}$$
$$\gamma_2(g) = -d - (g-1)e, \tag{4.1.5}$$

where d and e are two parameters. These scores can be derived from a probabilistic model too.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	5	−2	−1	−2	−1	−1	−1	0	−2	−1	−2	−1	−1	−3	−1	1	0	−3	−2	0
R	−2	7	−1	−2	−4	1	0	−3	0	−4	−3	3	−2	−3	−3	−1	−1	−3	−1	−3
N	−1	−1	7	2	−2	0	0	0	1	−3	−4	0	−2	−4	−2	1	0	−4	−2	−3
D	−2	−2	2	8	−4	0	2	−1	−1	−4	−4	−1	−4	−5	−1	0	−1	−5	−3	−4
C	−1	−4	−2	−4	13	−3	−3	−3	−3	−2	−2	−3	−2	−2	−4	−1	−1	−5	−3	−1
Q	−1	1	0	0	−3	7	2	−2	1	−3	−2	2	0	−4	−1	0	−1	−1	−1	−3
E	−1	0	0	2	−3	2	6	−3	0	−4	−3	1	−2	−3	−1	−1	−1	−3	−2	−3
G	0	−3	0	−1	−3	−2	−3	8	−2	−4	−4	−2	−3	−4	−2	0	−2	−3	−3	−4
H	−2	0	1	−1	−3	1	0	−2	10	−4	−3	0	−1	−1	−2	−1	−2	−3	2	−4
I	−1	−4	−3	−4	−2	−3	−4	−4	−4	5	2	−3	2	0	−3	−3	−1	−3	−1	4
L	−2	−3	−4	−4	−2	−2	−3	−4	−3	2	5	−3	3	1	−4	−3	−1	−2	−1	1
K	−1	3	0	−1	−3	2	1	−2	0	−3	−3	6	−2	−4	−1	0	−1	−3	−2	−3
M	−1	−2	−2	−4	−2	0	−2	−3	−1	2	3	−2	7	0	−3	−2	−1	−1	0	1
F	−3	−3	−4	−5	−2	−4	−3	−4	−1	0	1	−4	0	8	−4	−3	−2	1	4	−1
P	−1	−3	−2	−1	−4	−1	−1	−2	−2	−3	−4	−1	−3	−4	10	−1	−1	−4	−3	−3
S	1	−1	1	0	−1	0	−1	0	−1	−3	−3	0	−2	−3	−1	5	2	−4	−2	−2
T	0	−1	0	−1	−1	−1	−1	−2	−2	−1	−1	−1	−1	−2	−1	2	5	−3	−2	0
W	−3	−3	−4	−5	−5	−1	−3	−3	−3	−3	−2	−3	−1	1	−4	−4	−3	15	2	−3
Y	−2	−1	−2	−3	−3	−1	−2	−3	2	−1	−1	−2	0	4	−3	−2	−2	2	8	−1
V	0	−3	−3	−4	−1	−3	−3	−4	−4	4	1	−3	1	−1	−3	−2	0	−3	−1	5

Figure 4.2: The BLOSUM50 substitution matrix.

$$\begin{array}{llll} \mathtt{I\ G\ A}\ x_i & \quad \mathtt{A\ I\ G\ A}\ x_i & \quad \mathtt{G\ A}\ x_i\ \mathtt{-\ -} \\ \mathtt{L\ G\ V}\ y_j & \quad \mathtt{G\ V}\ y_i\ \mathtt{-\ -} & \quad \mathtt{S\ L\ G\ V}\ y_j \end{array}$$

Figure 4.3: The three ways an alignment can be extended up to x_i, y_j.

4.1.2 Needleman-Wunsch algorithm

Needleman-Wunsch algorithm allows to determine the *optimal global*[2] alignment of two sequences [100]. The key idea consists in recursively computing the optimal alignment for subsequences of increasing length. This is possible thanks to score independence and additivity.

Let $x = x_1 \dots x_n$ and $y = y_1 \dots y_n$ be two sequences. Let $F(i,j)$ be the score of the best alignment of subsequences $x_1 \dots x_i$ and $y_1 \dots y_j$. If $F(i-1,j-1)$, $F(i-1,j)$, and $F(i,j-1)$ are known, it is possible to compute $F(i,j)$. The alignment can be extended up to x_i, y_j in the following three ways: either x_i is aligned with y_j or x_i is aligned with a gap or y_j is aligned with a gap. The three positions are illustrated in Figure 4.3. The best alignment between $x_1 \dots x_i$ and $y_1 \dots y_j$ is obtained by choosing the greatest score among the three options.

Assuming 4.1.4 as gap score, we can obtain the following recursive equation:

$$F(i,j) = max \begin{cases} F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{4.1.6}$$

For each $i \in \{1, \dots, n\}$ and for each $j \in \{1, \dots, m\}$, $F(i,0)$ and $F(0,j)$ values correspond to alignments of a prefix of a sequence with an initial gap of the other sequence. Thus we set $F(i,0) = -id$ and $F(0,j) = -jd$. Starting from these values, it is possible to build a $n \times m$ matrix using equation 4.1.6. For construction, $F(n,m)$ value represents the score of the optimal alignment between x and y. For each $F(i,j)$, we keep a pointer back to the element from which it was calculated. It is possible to reconstruct the alignment by following the pointers up to element $F(0,0)$ according to the following rules:

- $(i,j) \rightarrow (i-1,j-1)$ (x_i is aligned with y_j);
- $(i,j) \rightarrow (i-1,j)$ (x_i is aligned with a gap);
- $(i,j) \rightarrow (i,j-1)$ (y_j is aligned with a gap).

In Figure 4.4 there is an example of a global optimal alignment obtained through this procedure.

We have a particular case when it is known that either one of the two sequences is contained in the other or there is a partial overlapping between the two sequences. In this case extremity gaps should not be penalized, thus we set $F(i,0) = F(0,j) = 0$ for each i and j. The alignment is

[2] An alignment is global when it refers to entire sequences.

$$
\begin{array}{c|ccccccccccccccc}
 & & & P & & A & & W & & H & & E & & A & & E \\
\hline
 & \mathbf{0} & \leftarrow & -8 & \leftarrow & -16 & \leftarrow & -24 & \leftarrow & -32 & \leftarrow & -40 & \leftarrow & -48 & \leftarrow & -56 \\
 & \uparrow & \nwarrow & & & & & & \nwarrow & & & & & & & \\
H & \mathbf{-8} & & -2 & \leftarrow & -10 & \leftarrow & -18 & & -14 & \leftarrow & -22 & \leftarrow & -30 & \leftarrow & -38 \\
 & \uparrow & \nwarrow & & \nwarrow & & & & \nwarrow & & \nwarrow & & & & & \\
E & \mathbf{-16} & & -9 & & -3 & \leftarrow & -11 & & -18 & & -8 & \leftarrow & -16 & \leftarrow & -24 \\
 & \uparrow & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & & & \uparrow & \nwarrow & & & \\
A & -24 & & \mathbf{-17} & & -4 & & -6 & & -13 & & -16 & & -3 & \leftarrow & -11 \\
 & \uparrow & & \uparrow & & \uparrow & \nwarrow & & \nwarrow & & & & \nwarrow & & \nwarrow & \\
G & -32 & & \mathbf{-25} & & -12 & & -7 & & -8 & \leftarrow & -16 & & -11 & & -6 \\
 & \uparrow & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & \\
A & -40 & & -33 & & \mathbf{-20} & & -15 & & -9 & & -9 & & -11 & & -12 \\
 & \uparrow & & \uparrow & & \uparrow & \nwarrow & & & & \nwarrow & & \nwarrow & & \nwarrow & \\
W & -48 & & -41 & & -28 & & \mathbf{-5} & \leftarrow & -13 & & -12 & & -12 & & -14 \\
 & \uparrow & & \uparrow & & \uparrow & & \uparrow & \nwarrow & & & & \nwarrow & & \nwarrow & \\
G & -56 & & -49 & & -36 & & \mathbf{-13} & & -7 & \leftarrow & -15 & & -12 & & -15 \\
 & \uparrow & & \uparrow & & \uparrow & & \uparrow & \nwarrow & & \nwarrow & & & & \nwarrow & \\
H & -64 & & -57 & & -44 & & -21 & & \mathbf{-3} & & -7 & \leftarrow & -15 & & -12 \\
 & \uparrow & \nwarrow & & & \uparrow & & \uparrow & & \uparrow & \nwarrow & & & & \nwarrow & \\
E & -72 & & -65 & & -52 & & -29 & & -11 & & \mathbf{3} & \leftarrow & \mathbf{-5} & & -9 \\
 & \uparrow & \nwarrow & & & \uparrow & & \uparrow & & \uparrow & \nwarrow & & \nwarrow & & \nwarrow & \\
E & -80 & & -73 & & -60 & & -37 & & -19 & & -5 & & 2 & & \mathbf{1}
\end{array}
$$

```
--P-AW-HEAE
HEAGAWGHE-E
```

Figure 4.4: An example of execution of Needleman-Wunsch algorithm.

built by following back pointers from the element with maximum score (which is either in the last row or in the last column) up to an element situated in the final row or column (not necessarily $F(0,0)$).

4.1.3 Smith-Waterman algorithm

Another frequently recurring problem consists in searching the best correspondence between *subsequences* of x and y. In this case we deal with *local* alignment. The algorithm we report, which is known as Smith-Waterman algorithm [120], is a variant of Needleman-Wunsch algorithm. The recursive equation is the following:

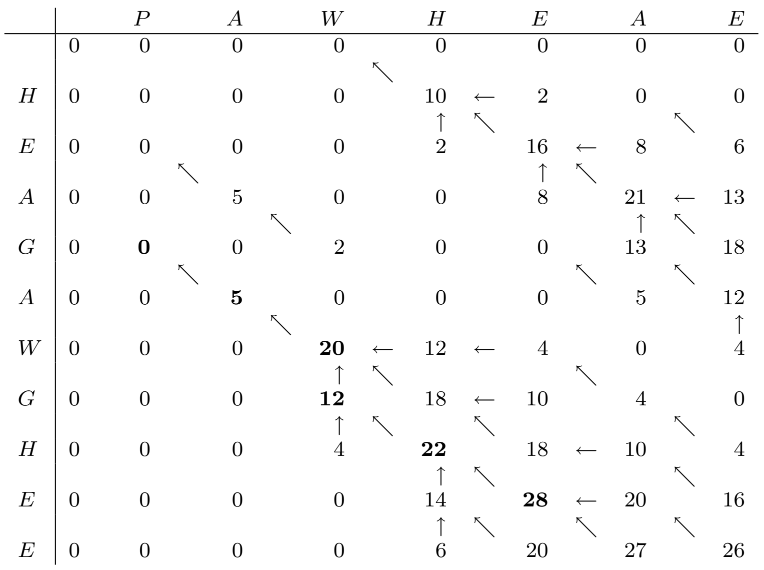
$$F(i,j) = max \begin{cases} 0 \\ F(i-1,j-1) + s(x_i, y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{4.1.7}$$

Elements $F(i,0)$ and $F(0,i)$ are inizialized to 0. In this case there are not negative scores. The intuitive reason is that, if an alignment up to a certain point has a negative score, then it is better to begin a new one than to extend the old one.

Once matrix F has been built, the alignment can be obtained starting from the identification of the maximum element of the matrix and following back pointers up to a 0, which denotes the beginning of the alignment. Unlike Needleman-Wunsch algorithm, in this case an alignment can begin and end at any point of the matrix. In Figure 4.5 there is an example of local alignment computed in such a way.

4.1.4 Repeated local alignments

In many cases we are interested in *all* the possible local alignments of a sequence with respect to another sequence. The algorithm we propose is able to find one or more copies of parts of a



```
AW-HE
AWGHE
```

Figure 4.5: An example of execution of Smith-Waterman algorithm.

sequence, called *motif*, within another sequence.

Let $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ be two sequences and let as assume that we are only interested in matches scoring higher than some threshold T. If we assume y to be the motif, then at the end of the execution of the algorithm x will be partitioned in regions locally aligned with y (*alignment regions*) and regions where there is no correspondence. We define the score of an alignment region as the difference between the score computed in the usual way and the threshold T. All these scores will be positive.

For $j > 0$, $F(i,j)$ is defined as the maximum sum of local alignment scores to $x_1 \dots x_i$, assuming that x_i is situated in an alignment region ending in x_i and y_j (this does not necessarily mean that x_i and y_j are aligned). For $j = 0$, $F(i,0)$ is the maximum sum of completed alignment scores to the subsequence $x_1 \dots x_i$, assuming that x_i is situated in an unmatched region. If we set $F(0,j) = 0$, for each $j \in \{0,\dots,m\}$, elements of the first column can be calculated as follows:

$$F(i,0) = max \begin{cases} F(i-1,0), \\ F(i-1,j) - T, \text{ with } j = 1,\dots,m. \end{cases} \tag{4.1.8}$$

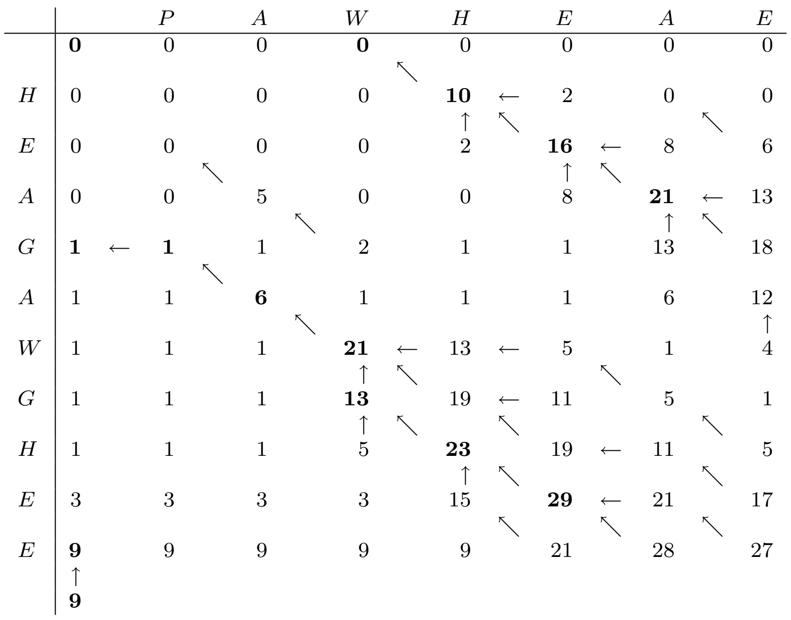
Remaining elements can be calculated according to the following recursive equation:

$$F(i,j) = max \begin{cases} F(i,0)F(i-1,j-1) + s(x_i,y_j), \\ F(i-1,j) - d, \\ F(i,j-1) - d. \end{cases} \tag{4.1.9}$$

Equation 4.1.8 allows to determine unmatched regions and to establish when an alignment region ends: it happens only when the score relative to subsequence $x_1 \dots x_{i-1}$ is at least T. Equation 4.1.9 allows to determine the beginning and the continuation of alignment regions.

The total score can be obtained by adding element $F(n+1,0)$, which is computed using equation 4.1.8. $F(n+1,0)$ equals 0 if there are not alignments with score greater than T. Otherwise, alignments can be obtained by following back pointers from $F(n+1,0)$ to $F(0,0)$. At each step we can determine if and how each x element is aligned following Needleman-Wunsch algorithm.

The difference is that x elements corresponding to positions $(i, 0)$ are outside alignment regions. In Figure 4.6 there is a repeated local alignment calculated in this way. There are two alignment regions and each point indicates unmatched regions. Every time an element of the first column is reached, the current alignment region ends.



		P	A	W	H	E	A	E
	0	0	0	**0**	0	0	0	0
H	0	0	0	0	**10**	2	0	0
E	0	0	0	0	2	**16**	8	6
A	0	0	5	0	0	8	**21**	13
G	**1**	**1**	1	2	1	1	13	18
A	1	1	**6**	1	1	1	6	12
W	1	1	1	**21**	13	5	1	4
G	1	1	1	**13**	19	11	5	1
H	1	1	1	5	**23**	19	11	5
E	3	3	3	3	15	**29**	21	17
E	**9**	9	9	9	9	21	28	27
	9							

```
HEA.AW-HE.
HEAGAWGHEE
```

Figure 4.6: An example of repeated local alignment.

4.1.5 Models with different gap functions

All the algorithms previously proposed build a $n \times m$ matrix where each element can be calculated in constant time. Their complexity is therefore $O(nm)$. If the two sequences have the same order of length (which is quite typical), algorithms have quadratic complexity. This depends on the gap function which has been chosen: function 4.1.4 only requires the knowledge of three precursors to compute an element $F(i, j)$. A generic gap function $\gamma(g)$ has a linear cost because it can require the knowledge of all precursors. Such a function leads to cubic algorithm complexity.

Using 4.1.5 as gap function, complexity remains $O(nm)$. In this case we must distinguish, for each x_i, y_j couple, the case of a gap of length one from the case of longer gaps. Therefore $F(i, j)$ value is not anymore enough. As an example, let us consider the algorithm for global alignment and let as refer to Figure 4.3. Let $M(i, j)$ be the best score for subsequences $x_1 \dots x_i$ and $y_1 \dots y_j$, supposing that x_i is aligned with y_j; let $I_x(i, j)$ be the best score, supposing that x_i is aligned with a gap; let $I_y(i, j)$ be the best score, supposing that y_j is aligned with a gap. Equations necessary to compute the best score become the following:

$$M(i,j) = max \begin{cases} M(i-1, j-1) + s(x_i, y_j), \\ I_x(i-1, j-1) + s(x_i, y_j), \\ I_y(i-1, j-1) + s(x_i, y_j); \end{cases} \tag{4.1.10}$$

$$I_x(i,j) = max \begin{cases} M(i-1, j) - d, \\ I_x(i-1, j) - e; \end{cases} \tag{4.1.11}$$

$$\begin{array}{ccccc} \texttt{G} & \texttt{A} & x_{i-1} & - & x_i \\ \texttt{S} & \texttt{L} & \texttt{G} & y_j & - \end{array}$$

Figure 4.7: A case of a deletion immediately followed by an insertion.

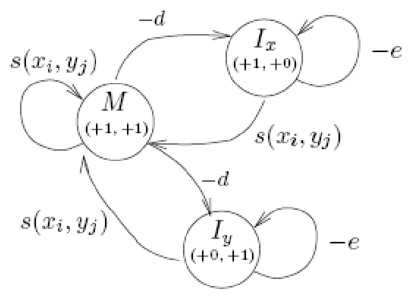


Figure 4.8: Automaton corresponding to equations (4.1.10), (4.1.11), and (4.1.12).

$$I_y(i,j) = max \begin{cases} M(i,j-1) - d, \\ I_y(i,j-1) - e. \end{cases} \tag{4.1.12}$$

The shape of the last two equations is due to the fact that we assume that a deletion can never be followed by an insertion, as shown in Figure 4.7. This is true for the best alignment if $-d - e$ is lower than the lowest score in the substitution matrix.

The precedent equations can be naturally represented through a *finite state automaton* whose states are M, I_x, and I_y and whose transitions describe the score increase (decrease) in the passage from a state to the following one. Transitions also specify a suitable increase of i, j indices. In Figure 4.8 there is such an automaton. Such a representation can be used as a starting point for generalizing alignment algorithms. As an example, it may be possible to distinguish with different states regions with a high degree of similarity and without gaps from regions with a low degree of similarity and with gaps, giving different substitution scores in the two cases. In this case, the automaton implicitly maintains additional information on the kind of region (high/low similarity).

4.1.6 Linear space alignments

When dealing with true biological sequences, the previously described algorithms can be too complex with respect to both time and space. As far as time is concerned, heuristic techniques are often adopted. These techniques improve performances but not always find the optimal solution. Among the most used tools we outline BLAST (http://www.ncbi.nlm.nih.gov/blast) and FASTA (http://fasta.bioch.virginia.edu). As far as space is concerned, there exist some techniques which allow to find the optimal alignment in space $O(n+m)$ (with a time increase of at most two factors). As an example, let us consider the global alignment problem with gap function given by 4.1.4. Let $x = x_1 \dots x_n$ and $y = y_1 \dots y_m$ be two sequences and let $u = \lfloor n/2 \rfloor$. If we can establish, for some $v \in \{0, \dots, m\}$, whether $F(u, v)$ is in the path which identifies the optimal alignment, then we can divide F in two sub-matrices, the first one corresponding to initial subsequences from $(0, 0)$ to (u, v), and the second one corresponding to final subsequences from (u, v) to (n, m). The problem can now be solved recursively: recursion ends when we reach a sequence of length 0. Alternatively, when matrices are "enough" small, it is possible to directly apply Needleman-Wunsch algorithm.

The main problem consists in identifying v. For $i > u$, we define $c(i, j)$ in a way that $F(u, c(i, j))$ is in the optimal path from $F(1, 1)$ to $F(i, j)$. It is possible to update the value of $c(i, j)$ while computing $F(i, j)$. If $F(i', j')$ is the element used to compute $F(i, j)$, we set $c(u, j) = j'$ and $c(i, j) = c(i', j')$. This operation can be locally executed: it is enough to maintain the previous row of c and the previous row of F. The desired value is $v = c(n, m)$.

```
1.  VTISCTGSSSNIGAG-NHVKWYQQLPG
2.  VTISCTGTSSNIGS--ITVNWYQQLPG
3.  LRLSCSSSGFIFSS--YAMYWVRQAPG
4.  LSLTCTVSGTSFDD--YYSTWVRQPPG
5.  PEVTCVVVDVSHEDPQVKFNWYVDG--
6.  ATLVCLISDFYPGA--VTVAWKADS--
7.  AALGCLVKDYFPEP--VTVSWNSG---
8.  VSLTCLVKGFYPSD--IAVEWESNG--
```

Figure 4.9: A multiple alignment of 8 fragments of immunoglobulin sequences.

4.2 Multiple alignment

Aligning pairs of sequences allows to capture similarities between *two* structures. However sequences are often grouped in *families*: differences within a family are usually viewed as a consequence of mutations from an ancestor during evolution. Sequences belonging to the same family usually have a high functional similarity but can be very different from a structural[3] point of view.

It is very useful to have some methods which allow to establish the relation between a given sequence and a family of sequences. This often makes it possible to infer functional characteristic of the sequence. Such a survey can be carried out through the research of the optimal multiple alignment (*multiple sequence alignment problem*). An example of multiple alignment is present in Figure 4.9. In general, we need a statistical method to evaluate and compare alignments and an algorithm to find the best evaluation. In the following, we will mostly refer to protein alignments (even if the method can also be applied to DNA sequences).

4.2.1 A model for the evaluation of a multiple alignment

We proceed giving a formal definition of the problem. For $k > 2$, let $x_1, \dots, x_k$ be k strings on an alphabet Σ (we can suppose Σ is the alphabet of amino acids). A *multiple alignment* of $x_1, \dots, x_k$ is a table M with elements over $\Sigma \cup \{-\}$ and with the following characteristics:

1. M has k rows;
2. ignoring gaps ($-$), the i-th row coincides with x_i;
3. each row contains at least a character different from $-$;
4. each column contains at least a character different from $-$.

We denote with ℓ the number of columns of M, with m_j the j-th column, and with m_{ij} the j-th symbol of the i-th row (which is either an element of x_i or a gap). Furthermore, given a sequence x, we denote with $|x|$ the number of symbols of x and, for some $a, b \in \{1, \dots, |x|\}$, we denote with $x[a..b]$ the subsequence of x starting in position a and ending in position b. Given a function S which associates a numeric value to each alignment M of $x_1, \dots, x_k$, the *optimal multiple alignment problem* consists in minimizing/maximazing S.

In order to obtain a biologically significant alignment, the choice of function S is fundamental. Ideally, S should encode all known information about structural and functional aspects and about the evolutionary history of sequences. In practice, at least the following two characteristics are considered:

- statistically, residuals in specific positions of sequences of a family are more conserved with respect to residuals in other positions (*position-specific scoring*);
- sequence are not independent, they belong to a *phylogenetic tree* which describes evolution.

[3] Regarding the spatial conformation of the molecule.

As a further simplification, we initially consider models which ignore these two aspects. In this way it is possible to independently compute a score for each column of the alignment. The standard way to do this is based on the addition of scores relative to pairs of elements and is called *SP* (*Sum of Pairs*) *cost*. Given a function s which associates a score to each pair of symbols (for example, a substitution matrix), the score of the j-th column can be computed as follows:

$$S(m_j) = \sum_{i=1}^{k} \sum_{i<l\leq k} s(m_{ij}, m_{lj}). \tag{4.2.1}$$

The score of an alignment M is[4]

$$S(M) = \sum_{j=1}^{\ell} S(m_j). \tag{4.2.2}$$

Function s must be defined even when one or both its arguments are gaps. If the gap function is $\gamma(g) = -gd$ for some d, then it is sufficient to set $s(-,-) = 0$ and, for each $a \in \Sigma$, $s(-,a) = s(a,-) = -d$. If the gap function is different, we must set $s(-,a) = s(a,-) = 0$ and manage gaps separately by means of an additional term in 4.2.2 (the shape of the term depends on the function chosen).

Observe that, even if this formulation is simple, it has some disadvantages. First of all, since it does not keep trace of phylogenetic relations among sequences, it has not a probabilistic justification. Secondly, it has a fault that can be illustrated through the following example. In Figure 4.9, the score associated to the 5-th column (the one containing cysteine in all the $k = 8$ sequences) is $s(\mathtt{C},\mathtt{C})k(k-1)/2$, where $k(k-1)/2$ is the number of different pairs of `C` in the column (`C` in different columns are considered different). If we use BLOSUM50 substitution matrix, we have $s(\mathtt{C},\mathtt{C}) = 13$. If in the last sequence there was `G` instead of `C`, the score of the column would be decremented of $(k-1)(13+3)$ because $s(\mathtt{G},\mathtt{C}) = -3$. The ratio between the two scores would be:

$$\frac{16(k-1)}{13k(k-1)/2} = \frac{32}{13k}.$$

Notice the dependence from k: as the sequence number grows, the relative difference between a correct alignment and an incorrect one decreases. Intuitively, the ratio should increase because, as k grows, it is more likely to have a cysteine conserved in that position.

4.2.2 Standard dynamic programming algorithm

In order to obtain a naïve algorithm for the determination of optimal multiple alignments, it is possible to generalize the one for pairs of sequences. Let $\alpha(i_1, \ldots, i_k)$ be the maximum score of an alignment of initial subsequences $x_1[1..i_1], \ldots, x_k[1..i_k]$. The score of the optimal alignment can be recursively computed as follows:

$$\alpha(i_1,\ldots,i_k) = \max \begin{cases} \alpha(i_1-1, i_2-1, \ldots, i_k-1) & + \quad S(m_{1i_1}, m_{2i_2}, \ldots, m_{ki_k}) \\ \alpha(i_1, i_2-1, \ldots, i_k-1) & + \quad S(-, m_{2i_2}, \ldots, m_{ki_k}) \\ \alpha(i_1-1, i_2, \ldots, i_k-1) & + \quad S(m_{1i_1}, -, \ldots, m_{ki_k}) \\ & \vdots \\ \alpha(i_1-1, i_2-1, \ldots, i_k) & + \quad S(m_{1i_1}, m_{2i_2}, \ldots, -) \\ \alpha(i_1, i_2, i_3-1\ldots, i_k-1) & + \quad S(-,-,m_{3i_3}, \ldots, m_{ki_k}) \\ & \vdots \\ \alpha(i_1, i_2-1, \ldots, i_{k-1}-1, i_k) & + \quad S(-, m_{2i_2}, \ldots, -) \\ & \vdots \end{cases}$$

[4] We apply the same function S both to single alignments and to single columns of alignments. The context solves ambiguities.

In the precedent equation, there are all the $2^k - 1$ gap combinations except the one containing only gaps. Proceeding in the same manner we do for sequence pairs, using such an equation we can build a k-dimensional matrix with $|x_1||x_2|\cdots|x_k|$ elements. Each element requires the computation of $2^k - 1$ values. Total computation time is then $O(2^k \prod_{i=1}^{k} |x_i|) \cdot O(\text{computation of } S)$, while space required to memorize the k-dimensional matrix is $O(\prod_{i=1}^{k} |x_i|)$. It has been proved that the multiple alignment problem with SP cost is NP-complete [125].

4.2.3 Carrillo-Lipman algorithm

Exploring all the k-dimensional matrix is too expensive. Now we will introduce a technique which reduces the number of elements to compute. Observe that equation 4.2.2 can be written as follows:

$$S(M) = \sum_{i=1}^{k} \sum_{i<l\leq k} \sum_{j=1}^{\ell} s(m_{ij}, m_{lj}). \tag{4.2.3}$$

Equation 6.2.3 can be interpreted as the score sum of all alignments of sequence pairs (they are defined starting from the multiple alignment through projection on two sequences). Given an alignment M of $x_1, \ldots x_k$, we denote with $M|_{i,l}$ the projection of M on the sequence pair x_i, x_l. Therefore equation 6.2.3 becomes

$$S(M) = \sum_{i=1}^{k} \sum_{i<l\leq k} S(M|_{i,l}). \tag{4.2.4}$$

Score $S(M|_{i,l})$ can be computed in the way used for sequence pair alignment.

Let M^{opt} be the optimal alignment of $x_1, \ldots, x_k$. We want to establish, for each sequence pair x_p, x_q, a lower bound to the score of the projection of $M^{\text{opt}}|_{p,q}$ on x_p and x_q. If we are able to do this, thanks to the interpretation given by equation 6.2.3 we can exclude all multiple alignments such that at least one of the projections has a score lower than the bound.

Let us suppose to know a heuristic alignment $\hat{M}$ of $x_1, \ldots, x_k$ which is "close to" the optimal one. We denote with $d(x_i, x_l)$ the score of the optimal alignment of the sequence pair x_i, x_l. For definition, it holds that $S(M^{\text{opt}})S(\hat{M})$. If we take advantage of equation 4.2.4, this is equivalent to

$$\sum_{i=1}^{k} \sum_{i<l\leq k} \big(S(M^{\text{opt}}|_{i,l}) - S(\hat{M}|_{i,l})\big)0. \tag{4.2.5}$$

Notice that, in general, the only thing we can say about the scores of projections $M^{\text{opt}}|_{i,l}$ and $\hat{M}|_{i,l}$ is that they can be at most $d(x_i, x_l)$. If we fix two particular sequences x_p and x_q, inequality 4.2.5 can be rewritten as follows:

$$\Big(\sum_{\substack{i=1\\i\neq p}}^{k} \sum_{\substack{i<l\leq k\\l\neq q}} \big(S(M^{\text{opt}}|_{i,l}) - S(\hat{M}|_{i,l})\big)\Big) + S(M^{\text{opt}}|_{p,q}) - S(\hat{M}|_{p,q})0, \tag{4.2.6}$$

and thus as follows:

$$S(\hat{M}|_{p,q}) - \Big(\sum_{\substack{i=1\\i\neq p}}^{k} \sum_{\substack{i<l\leq k\\l\neq q}} \big(S(M^{\text{opt}}|_{i,l}) - S(\hat{M}|_{i,l})\big)\Big) \leq S(M^{\text{opt}}|_{p,q}). \tag{4.2.7}$$

Since $S(M^{\text{opt}}|_{i,l}) \leq d(x_i, x_l)$, we can obtain the following inequality:

$$S(\hat{M}|_{p,q}) - \Big(\sum_{\substack{i=1\\i\neq p}}^{k} \sum_{\substack{i<l\leq k\\l\neq q}} \big(d(x_i, x_l) - S(\hat{M}|_{i,l})\big)\Big) \leq S(M^{\text{opt}}|_{p,q}). \tag{4.2.8}$$

The left term of inequality 4.2.8, which is called *Carrillo-Lipman bound* [21], is the desired bound to the score of projection $M^{\mathrm{opt}}|_{p,q}$ on x_p, x_q.

To compute the lower bounds it is only necessary to know a heuristic alignment (we will see how to obtain it in the sequel) and the $k(k-1)/2$ optimal pair alignments $d(x_i, x_l)$. The next step consists in individuating, for each x_p, x_q, a set of index pairs (i_p, i_q) such that the score of the best alignment of x_p and x_q passing through (i_p, i_q) is at least the lower bound. This can be done in quadratic time. Lastly, the algorithm described in the preceding subsection is applied only to the values obtained through the intersection of all the sets previously computed.

4.2.4 Progressive alignment methods

Probably the most commonly used approach to multiple sequence alignment is *progressive alignment*. This works by constructing a succession of pairwise alignments. Initially, two sequences are chosen and aligned by standard pairwise alignment; this alignment is fixed. Then, a third sequence is chosen and aligned to the first alignment, and this process is iterated until all sequences have been aligned. Such an approach is heuristic and does not guarantee to find the optimal alignment. However, it is efficient and often produces reasonable results.

The most important heuristic of progressive alignment algorithms is to align the most similar pairs of sequences first. This choice is justified by the fact that, more two sequences are similar, more they are likely to be recently derived from a common ancestor, and thus more reliable is the information given by their alignment. In particular, gap positions in more related sequences are generally more precise with respect to the ones relative to less similar sequences. This leads to formulate an heuristic rule which consists in preserving gaps of initial alignments when new sequences are aligned.

Most algorithms build *guide trees*, binary trees whose leaves are labeled by sequences and whose internal nodes represent clusters of sequences.

Feng-Doolittle algorithm

Feng-Doolittle algorithm [48] is one of the first progressive alignment algorithms proposed in literature. In overview, it is as follows:

1. Calculate the $k(k-1)/2$ pairwise alignments of the k sequences and convert scores to distances.

2. Use an incremental clustering algorithm to construct a guide tree starting from distances computed in the previous step;

3. Starting from the first node added to the three, align the child nodes (which may be two sequences, a sequence and an alignment, or two alignments). Repeat for all other nodes in the order that they were added to the tree (i.e., from most similar pairs to least similar pairs) until all sequences have been aligned.

Thompson-Higgins-Gibson algorithm

A problem with Feng-Doolittle approach is that all alignments are determined by pairwise sequence alignments. Once an aligned group has been built up, it is advantageous to use position-specific information from the group multiple alignment to align a new sequence to it. The degree of sequence conservation at each position should be taken into account and mismatches at highly conserved positions penalized more stringently than mismatches at variable positions. Gap penalties in positions might be reduced where lots of gaps occur in the cluster alignment, and increased where no gaps occur. In the sequel we describe an algorithm which uses a position-dependent evaluation schema called *profile*.

Given a multiple alignment M with ℓ columns, a profile for M is a table $|\Sigma \cup \{-\}| \times \ell$ where each column contains numbers indicating the occurrence frequency of each symbol in that position. In Figure 4.10 there is an example of profile.

	1	2	3	4	5	6	7
A	.25	.125					
R		.125					
C					1		
E		.125					
G				.125			.25
H							
I			.25				.125
L	.25		.625			.375	
P	.125						
S		.25		.375		.125	.125
T		.375		.375		.375	
V	.375		.125	.125		.125	.5

Figure 4.10: Profile corresponding to the first 7 positions of the alignment of figure 4.9.

Aligning a sequence to a profile consists in aligning the sequence to the profile columns (allowing gaps). Figure 4.11 shows a possible alignment of string `VTICL` to the profile of Figure 4.10. The score associated to the alignment of a character to a column of the profile can be computed as the weighted sum of all possible symbol coupling. If we refer to Figure 4.11 and we assume to use the BLOSUM50 matrix to evaluate couplings, we can see that the score of `V` in first position is given by $.25 \cdot 0 + .25 \cdot 1 - .125 \cdot 3 + .375 \cdot 5 = 1.75$. The total score is the sum of values of single columns. More formally, let $P = (p_{ij})$ be a profile, a_i the symbol labeling the i-th row of the profile, and $\alpha \in \Sigma \cup \{-\}$ a character. The score of the alignment of α to the j-th column of P is thus

$$v(\alpha, j) = \sum_{i=1}^{|\Sigma \cup \{-\}|} p_{ij} s(a_i, \alpha). \tag{4.2.9}$$

The score of the alignment of a sequence $x = \alpha_1 \cdots \alpha_\ell$ to a profile P with ℓ columns is

$$S(x, P) = \sum_{j=1}^{\ell} v(\alpha_j, j). \tag{4.2.10}$$

The optimal alignment of a sequence x to a profile P is computed by using a dynamic programming algorithm similar to the ones used for pairwise alignments. Let $F(i,j)$ be the score of the best alignment of $x[1..i]$ to the first j columns of P. Equations to compute the optimal sequence-profile alignment are the following:

$$F(0,j) = \sum_{k=1}^{j} v(-,k), \tag{4.2.11}$$

$$F(i,0) = \sum_{k=1}^{i} s(x[k..k], -), \tag{4.2.12}$$

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + v(x[i..i], j) \\ F(i-1, j) + s(x[i..i], -) \\ F(i, j-1) + v(-, j) \end{cases} \quad \text{for } i,j > 0. \tag{4.2.13}$$

```
1 2 3 4 5 6 7
V T I − C L −
```

Figure 4.11: A possible alignment of sequence `VTICL` to the profile of Figure 4.10.

It is also possible to align *a profile to a profile* by extending in a suitable way the score computation and recursive equations. In case of two profiles, gaps must be inserted in the entire column of a profile. In this way it is possible to preserve the already existing multiple alignment within the two groups of sequences.

A well-known implementation of progressive alignment through profiles is the program CLUS TAL W [122]. We outline the main steps of the algorithm:

1. Calculate the $k(k-1)/2$ pairwise alignments of the k sequences and convert scores to distances.

2. Use a neighbour-joining clustering algorithm to construct a guide tree starting from distances computed in the previous step;

3. Progressively align at nodes in order of decreasing similarity, using sequence-sequence, sequen ce-profile, and profile-profile alignment.

Iterative refinement methods: Barton-Stenberg algorithm

One problem with progressive alignment algorithms is that sub-alignments are frozen. That is, once a group of sequences has been aligned, their alignment to each other cannot be changed at a later stage as more data arrive. *Iterative refinement* algorithms attempt to circumvent this problem.

In iterative refinement, an initial alignment is generated, for instance as outlined above; then one sequence (or a set of sequences) is taken out and realigned to a profile of the remaining aligned sequences. If a meaningful score is being optimized, this either increases the overall score or results in the same score. Another sequence is chosen and realigned, and so on, until the alignment does not change. The procedure is guaranteed to converge to a local maximum of the score provided that all the sequences are tried and a maximum score exists, simply because the sequence space is finite.

Barton-Stenberg algorithm is an example of how some of the methods mentioned so far can be combined [9]. It works as follows:

1. Given k sequences, compute the scores of all pairwise alignments. Align sequences x_1 and x_2 with maximum degree of similarity.

2. For each $i = 3, \ldots, k$, let x_i be the sequence with the best score in the sequence-profile alignment with respect to profile $x_1, \ldots, x_{i-1}$. Align x_i to this profile.

3. For each $i = 1, \ldots, k-1$, remove x_i from the alignment and realign x_i to the profile of remaining sequences.

4. Repeat the previous realignment step a fixed number or times, or until the alignment score converges.

The ideas of profile alignment and iterative refinement come quite close to the formulation of probabilistic hidden Markov model approaches for multiple alignment [65, 78, 79], which we will treat in the next section.

4.3 Hidden Markov models for multiple alignment

As already outlined, the simultaneous comparison of many sequences often makes it possible to infer the evolutive history of a family of sequences. Furthermore, it allows to detect biologically relevant patterns. A well-known approach to multiple alignment is based on *hidden Markov models* and is discussed in this section.

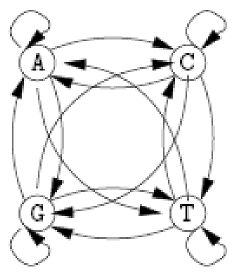


Figure 4.12: A stochastic automaton for DNA sequences.

4.3.1 Markov chains and hidden Markov models

Hidden Markov models are an extension of Markov chains, the simplest model where the probability of an event only depends on the preceding event. We start defining markovian processes.

Definition 4.3.1 *A* ***discrete-time stochastic process*** *is a succession of random variables X_i: $\Omega \to \mathbb{R}$, with $i \in I \subseteq \mathbb{N}$, defined on a sample space Ω. If variables are discrete, the process is called* ***chain****. The support of variables is called* ***state space*** *and each element of the support is called* ***state****.*

Definition 4.3.2 *A chain $\{X_i\}_{i \in I \subseteq \mathbb{N}}$ satisfies the* ***Markov property*** *if, for each $n \in I$ and each sequence of states $i_1, \ldots, i_n$, it holds that*

$$\Pr(X_n = i_n \mid X_{n-1} = i_{n-1}, \ldots, X_1 = i_1) = \Pr(X_n = i_n \mid X_{n-1} = i_{i-1}) \tag{4.3.1}$$

A ***Markov chain*** *is a chain which satisfies the Markov property.*

If there is a finite number of states, a Markov chain can be represented through a stochastic automaton, that is a graph whose nodes are labeled with states and whose edges are labeled with probabilities to pass to the next state. An example of Markov chain for DNA strings is present in figure 4.12. Such an automaton is said to be *generative* because a path in the graph determines a string.

Using probabilities labeling edges in the path and exploiting the Markov property it is possible to compute the probability of a given sequence $x = x_1 \ldots x_n$ as follows. For $s, t \in \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$, let $a_{st} = \Pr(x_i = t \mid x_{i-1} = s)$. Then it holds

$$\begin{aligned}
\Pr(x) &= \Pr(x_1, x_2, \ldots, x_n) = \\
&= \Pr(x_n \mid x_{n-1}, \ldots, x_1) \Pr(x_{n-1} \mid x_{n-2}, \ldots, x_1\} \cdots \Pr(x_1) = \\
&= \Pr(x_n \mid x_{n-1}) \Pr(x_{n-1} \mid x_{n-2}) \cdots \Pr(x_1) = \\
&= \Pr(x_1) \prod_{i=2}^{n} a_{x_{i-1}x_i}.
\end{aligned} \tag{4.3.2}$$

In a Markov chain there is a bijective correspondence between symbols emitted from the automaton and corresponding states. In hidden Markov models states are hidden: the observer can only see a sequence of symbols according to which she can infer the probability of corresponding states.

Definition 4.3.3 *A* ***hidden Markov model (HMM)*** *is defined by:*

- *a finite set $S = \{1, 2, \ldots, m\}$ of states;*
- *a conditioned distribution of probability $A = \{a_{ij}\}$ specifying, for each $k, l \in S$, the probability to pass from state k to state l:*

$$a_{kl} = \Pr(\pi_i = l \mid \pi_{i-1} = k);$$

For $k \in S$, we denote by a_{0k} the probability that the initial state is k and by a_{k0} the probability that k is a final state;

- *a set Σ of observable symbols*
- *a conditioned distribution of probability $B = \{b_{ij}\}$ specifying, for each $b \in \Sigma$ and for each $k \in S$, the probability to observe the emission of symbol b when we are in state k:*

$$e_k(b) = \Pr(x_i = b \mid \pi_i = k).$$

Probability computation with HMM

Given a HMM, the joint probability of an observed sequence $x = x_1 \dots x_n$ generated from a state sequence $\pi = \pi_1, \dots, \pi_n$ is

$$\Pr(x, \pi) = a_{0\pi_1} \prod_{i=1}^{n} e_{\pi_i}(x_i)\, a_{\pi_i \pi_{i+1}}, \tag{4.3.3}$$

where, for convention, $\pi_{n+1} = 0$. Since the state sequence is not known, equation 4.3.3 is not really useful. A way to infer the HMM path that led to the generation of a given string consists in determining the *most probable path* (*cpp*) associated to the string, that is, the state sequence

$$\pi^* = \arg\max_{\pi} \Pr(x, \pi). \tag{4.3.4}$$

The Markov property allows to determine π^* through a dynamic programming algorithm, *Viterbi algorithm.* Let $v_k(i)$ be the probability of the cpp which finishes in state k at position i. These probabilities can be recursively computed as follows:

$$\begin{aligned} v_0(0) &= 1; && (4.3.5)\\ v_l(i+1) &= e_l(x_{i+1}) \max_k (v_k(i)\, a_{kl}). && (4.3.6) \end{aligned}$$

Keeping, for each state l and for each position i, a back pointer $\text{ptr}_i(l)$ to the preceding state, it is possible to reconstruct backwards the most probable path. Viterbi algorithm is the following:

1. Initialization: $v_0(0) = 1, v_k(0) = 0$, for $k > 0$.
2. Recursion:
$$v_l(i) = e_l(x_i) \max_k (v_k(i-1)\, a_{kl});$$
$$\text{ptr}_i(l) = \arg\max_k (v_k(i-1)\, a_{kl}).$$
3. Termination:
$$\Pr(x, \pi^*) = \max_k (v_k(n)\, a_{k0});$$
$$\pi_n^* = \arg\max_k (v_k(n)\, a_{k0}).$$
4. Traceback:
$$\pi_{i-1}^* = \text{ptr}_i(\pi_i^*).$$

The probability of a string x, independently from the path, can be computed as the sum of $\Pr(x, \pi)$ on all paths:

$$\Pr(x) = \sum_{\pi} \Pr(x, \pi). \tag{4.3.7}$$

Such a value is equivalent to the computation of (4.3.2) in the case of Markov chains. However, the number of paths increases exponentially with respect to the sequence length. A solution to the problem consists in ignoring all paths except the cpp, and thus defining $\Pr(x) = \Pr(x, \pi^*)$. This approximation often gives good results. However, it is not necessary to resort to it. In fact, using the Markov property, it is possible to obtain a dynamic programming algorithm similar to Viterbi one. Let $f_k(i) = \Pr(x_1 \cdots x_i, \pi_i = k)$ be the probability to observe (sub)sequence $x_1 \dots x_i$ and to be in state k. The following algorithm, which is called *forward algorithm*, allows to recursively compute the probability of $x = x_1 \dots x_n$. The steps are the following:

1. Initialization: $f_0(0) = 1$, $f_k(0) = 0$, per $k > 0$;
2. recursion: $f_l(i) = e_l(x_i) \sum_k (f_k(i-1)\, a_{kl})$;
3. Termination: $\Pr(x) = \sum_k (f_k(n)\, a_{k0})$.

The inverse problem is interesting to: given a sequence x, which is the probability $\Pr(\pi_i = k \mid x)$ that the HMM is in state k at time i? This value can be obtained recursively too. We start from the formula of compound probability $\Pr(x, \pi_i = k)$:

$$\begin{aligned} \Pr(x, \pi_i = k) &= \Pr(x_1 \cdots x_i, \pi_i = k) \Pr(x_{i+1} \cdots x_n \mid x_1 \cdots x_i, \pi_i = k) \\ &= \Pr(x_1 \cdots x_i, \pi_i = k) \Pr(x_{i+1} \cdots x_n \mid \pi_i = k) = \\ &= f_k(i) b_k(i). \end{aligned} \tag{4.3.8}$$

Notice that we set $b_k(i) = \Pr(x_{i+1} \cdots x_n \mid \pi_i = k)$. The backward recursive computation of $b_k(i)$ allows to compute $\Pr(\pi_i = k \mid x)$: since $\Pr(x, \pi_i = k) = f_k(i)b_k(i)$, by applying the definition of conditioned probability we have:

$$\Pr(\pi_i = k \mid x) = \frac{f_k(i)b_k(i)}{\Pr(x)}. \tag{4.3.9}$$

Probability $\Pr(x)$ can be computed by means of the forward algorithm. Steps of the backward algorithm to compute $b_k(i)$ are the following:

1. Initialization: $b_k(n) = a_{k0}$, for each k;
2. Recursion: $b_k(i) = \sum_l a_{kl}\, e_l(x_{i+1})\, b_l(i+1)$.

Estimate of HMM parameters

One of the most difficult problems faced when using HMMs consists in determining transition and emission probabilities a_{kl} and $e_k(b)$. Let us suppose we have a set of sample sequences, known as *training sequences*, which we assume to be independent. We want to model the training sequences by means of a HMM. If paths associated to sequences are known, we can proceed by computing maximum likelihood estimators for a_{kl} and $e_k(b)$. Let A_{kl} be the number of times we pass from state k to state l in the sample and let $E_k(b)$ be the number of times we observe the emission of symbol b in state k. Then we set

$$a_{kl} = \frac{A_{kl}}{\sum_{l'} A_{kl'}} \quad \text{and} \quad e_k(b) = \frac{E_k(b)}{\sum_{b'} E_k(b')}. \tag{4.3.10}$$

This estimation method is sensible to statistical distribution and to the dimension of the sample. Furthermore, if a given state is never present in the sample, the relative equations are undefined. To get round of this last defect, we add some *pseudocount* values to A_{kl} and $E_k(b)$.

If the state sequences associated to the sample are not known, we need an iterative method to estimate parameters. The standardly used procedure is Baum-Welch algorithm. Such an algorithm estimates A_{kl} and $E_k(b)$ considering cpps relative to the training set, which are computed using the current values of a_{kl} and $e_k(b)$. It is possible to prove that, at each iteration, the log likelihood of the model increases and the procedure converges to a local maximum. Usually there is more than one local maximum; therefore the reached value strongly depends on initial parameter values.

4.3.2 Profile HMMs for sequence families

Let us focus again on the multiple alignment problem and try to understand how to use HMMs to solve such a problem. HMMs for multiple alignment are called *profile HMMs* because they are related to the notion of standard profile.

1. `VTISCTG`
2. `VTISCTG`
3. `LRLSCSS`
4. `LSLTCTV`
5. `PEVTCVV`
6. `ATLVCLI`
7. `AALGCLV`
8. `VSLTCLV`

	1	2	3	4	5	6	7
A	.25	.125					
R		.125					
C					1		
E		.125					
G				.125			.25
H							
I			.25				.125
L	.25		.625			.375	
P	.125						
S		.25		.375		.125	.125
T		.375		.375		.375	
V	.375		.125	.125		.125	.5

Figure 4.13: A region without gaps and the corresponding profile.

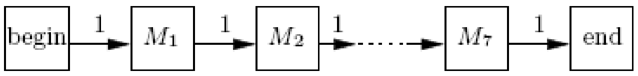


Figure 4.14: The profile of figure 4.13 interpreted as a HMM.

The standard profile relative to a multiple alignment is a model directly built from information extracted from aligned sequences. It is used to evaluate the alignment of a further sequence x. Let us consider the profile associated to a region without gaps of a multiple alignment, like the one present in Figure 4.13. Such a profile can be viewed as the trivial HMM depicted in Figure 4.14, where the number of *match states* M_i equals the columns of the profile, the emission probability of each symbol equals the values of each column, and the transition probability from one state to the following one equals 1.

It is possible to extend such a HMM in order to deal with gaps. At this aim, we consider insertions and deletions separately. In order to manage insertions, that is, portions of x which have no correspondence in the model, we introduce in the HMM a set of new states I_i: each I_i models the fact that the x character which follows the corresponding one in the i-th column of the profile is aligned to a gap. Figure 4.15 shows one of such states with relative transitions. Emission probabilities in a state I_i generally equal occurrence probabilities q_a of each symbol a in a random model.

In order to manage deletions, that is, segments of the multiple alignment which have no correspondence in the sequence x, we introduce a set of *silent* states D_j. As shown in Figure 4.16, silent states do not emit symbols.

The topology of the resulting is depicted in Figure 4.17, where squares indicate match states, rhombs stay for insertion states, and circles outline deletion states. Each triple M_j, I_j, D_j is called *module*. The number of modules defines the *length* of the model. If the profile HMM has M

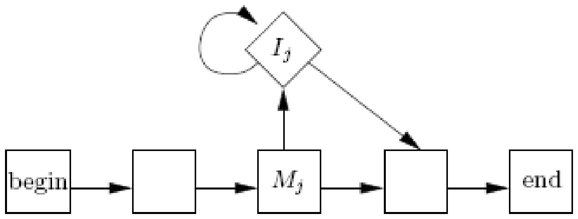


Figure 4.15: A HMM model for insertion.

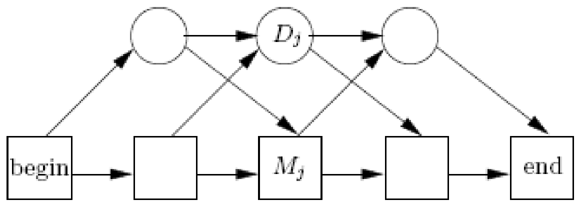


Figure 4.16: A HMM model for deletion.

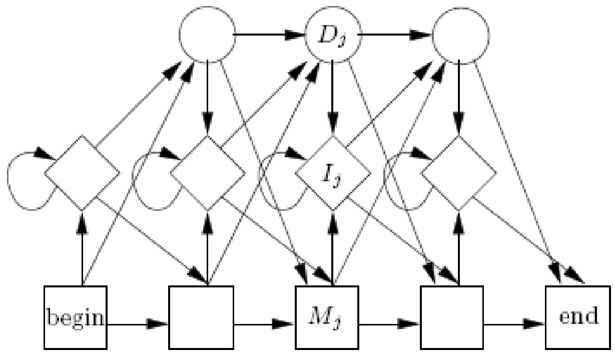


Figure 4.17: The structure of a profile HMM.

modules, then the number of states is $3M+3$. Notice that transitions between insertion states and deletion states have been added. They are quite improbable but they are useful in the construction of the model.

The key idea on the structure of a profile HMM consists in capturing the specific characteristics of a family of sequences by modulating parameters in a suitable way. If emission and transition probabilities are positive, the profile HMM models any possible sequence in the given alphabet, and thus defines a distribution of probability in the sequence space. The correct parameter attribution aims at adapting such a distribution of probability in a way that it has peaks in correspondence of sequences belonging to the considered family. At this aim we can use the algorithms explained in the previous section. For example, if a multiple alignment is available, parameters can be estimated using (4.3.10); otherwise, if we start from a set of non aligned sequences, we can use the Baum-Welch algorithm.

Besides controlling the shape of the distribution of probability, which is determined by a_{kl} and $e_k(b)$, we can rule the length of the model. If we are in possession of a multiple alignment, we can heuristically consider as insertions the columns of the alignment containing gaps for more than half of the length: we will have as many modules as remaining columns. As an example, consider Figure 4.18: the profile HMM derived using the heuristic rule will have 8 modules. Residues in 4-th and 5-th positions of last sequence will be treated as insertions.

If we have a set of non aligned sequences, we can choose as length of the model the average

```
1. VGA--HAGEY
2. V----NVDEV
3. VEA--DVAGH
4. VKG------D
5. VYS--TYETS
6. FNA--NIPKH
7. IAGADNGAGV
```

Figure 4.18: A multiple alignment of 7 sequences.

1	2	3	4	5	6	insert.	7	8	9	10	11
F	P	H	F	–	D	LS	H	G	S	A	Q
F	E	S	F	G	D	LSTPDAV	M	G	N	P	K
F	D	R	F	K	H	LKTEAEM	K	A	S	E	D
F	T	Q	F	A	G	KDLESI	K	G	T	A	P
F	G	–	F	S	G	AS	–	–	D	P	G

Figure 4.19: Cpps of 5 sequences through a profile with 11 modules.

length of sequences. There exists more refined techniques to build an optimal model too.

Score computation with HMM

Profiles HMMs can be used instead of standard profiles in progressive or iterative alignment algorithms. They replace the ad hoc schema based on SP cost. There exist at least two ways to evaluate the relation between a sequence x and a given profile HMM. Viterbi algorithm allows to determine the most probable alignment π^* and its probability $\Pr(x, \pi^*)$. Alternatively, it is possible to use the forward algorithm to compute $\Pr(x)$ on all paths. In any case, in practice, the adopted score is the logarithm of the ratio between one of the preceding probabilities and the probability of x given a random model, that is, $\Pr(x \mid R) = \prod_i q_{x_i}$.

We show the variant of Viterbi algorithm which allows to compute the score of the best alignment of x to the profile HMM. Let $V_j^M(i)$ be the score of the best alignment of the initial subsequence $x_1 \dots x_i$ to the profile, which terminates by x_i emitted by state M_j. Analogously, let $V_j^I(i)$ be the score of the best path ending by x_i emitted by I_j, and let $V_j^D(i)$ the score of the best path ending in state D_j. Then Viterbi equations assume the following shape:

$$\begin{aligned}
V_j^M(i) &= \log \frac{e_{M_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_{j-1}^M(i-1) + \log a_{M_{j-1}M_j}, \\ V_{j-1}^I(i-1) + \log a_{I_{j-1}M_j}, \\ V_{j-1}^D(i-1) + \log a_{D_{j-1}M_j}; \end{cases} \\
V_j^I(i) &= \log \frac{e_{I_j}(x_i)}{q_{x_i}} + \max \begin{cases} V_j^M(i-1) + \log a_{M_j I_j}, \\ V_j^I(i-1) + \log a_{I_j I_j}, \\ V_j^D(i-1) + \log a_{D_j I_j}; \end{cases} \\
V_j^D(i) &= \max \begin{cases} V_{j-1}^M(i) + \log a_{M_{j-1}D_j}, \\ V_{j-1}^I(i) + \log a_{I_{j-1}D_j}, \\ V_{j-1}^D(i) + \log a_{D_{j-1}D_j}. \end{cases}
\end{aligned} \tag{4.3.11}$$

Since emission probabilities of insertion states equal frequencies of the random model, the logarithm in the equation for $V_j^I(i)$ is typically 0. As far as initialization is concerned, we set $V_0^M(0) = 0$ to allow the alignment to begin or end in an insertion or deletion state. The construction of the effective alignment only requires to follow back pointers.

If the profile HMM is available, the construction of a multiple alignment is simple: it is enough to individually align each sequence to the model through Viterbi algorithm. Figure 4.19 shows the cpps of five sequences through a given profile HMM. Observe that the profile HMM does not attempt to align elements corresponding to insertion states. It is arbitrary how to arrange these symbols. This reflects the biologically realistic situation in which parts of sequences (even belonging to the same family) are atypical, non conserved and actually not alignable.

If the profile HMM is not built, it is necessary to adopt a different approach which consists in estimating both a model and an alignment starting from non aligned sequences. The procedure can be described as follows:

1. choose the length of the profile HMM (using either a heuristic rule or a more sophisticated system) and initialize parameters;

2. give an estimation of the parameters of the model using Baum-Welch algorithm (or an analogous algorithm);

3. align all the sequences to the obtained model using Viterbi algorithm and build the multiple alignment as seen before.

The choices of the length of the model and of initial values of parameters are extremely important. In particular, since Baum-Welch algorithm finds local maxima, it is suggested to execute the algorithm starting from different initial values to see if there is convergence to the same value. Furthermore, it is appropriate to look for “reasonable” transition probabilities. An an example, transitions among correspondence states should have higher probability with respect to other transitions.

5

Game theory approaches to sequence comparison

In the previous chapter we presented some state-of-the-art algorithms for evaluating the similarity of strings. However, it is worth noticing that classical alignment methods sometimes reveal themselves to be inadequate to compare biological sequences, e.g., to compare genome assemblies of the same organism obtained by different programs or to detect evolutionary related biological strings. As for assemblies, they are derived from several millions short strings (approximately 700 letters long), called “reads”, by chaining overlapping ones. The result is a set of longer sequences, called “contigs”: usually, it is not possible to get a single contig (the genome), because gaps remain. Different algorithms may produce different assemblies, mainly owing to the errors in the input and to the repetitions within the genome. A left-to-right comparison is not suitable to compare two assemblies, because different parts may be ordered differently (inversions) and some other segments may not be related. For the same reason a standard comparison sometimes is not appropriate to compare evolutionary related biological strings, that can contain inversions too.

Let us consider, for instance, the two DNA sequences *agggagtttttaga* and *agttagtttagaagggga*, that clearly exemplify the issue. A standard left-to-right comparison algorithm would assign to the strings a low score, as pointed out in the alignment below.

$$\begin{array}{cccccccccccccccccc}
a & g & g & g & a & g & t & t & t & t & t & a & - & - & - & - & g & a \\
| & | & & & | & | & | & | & | & & & | & & & & & | & | \\
a & g & t & t & a & g & t & t & t & a & g & a & a & g & g & g & g & a
\end{array}$$

If we have a careful look at the two sequences, we can notice they contain some similar subsequences, i.e., the red and blue subsequences in the following colored representation: *agggagtttttaga*, *agttagtttagaagggga*. Such a representation underlines that the two colored subsequences underwent a process of inversion. It would be beneficial to dispose of a way of comparing sequences that assigns a high score to sequences like the ones above.

In this chapter we present a flexible way to determine how and where two sequences differ based on the use of comparison games. In literature, such games are mainly used to prove inexpressibility results or to establish normal forms for logics [5, 71, 115]. In [95], Montanari et al. have taken a different point of view: they consider structures provided with the successor relation s, they define a criterion to measure the degree of similarity of labeled successor structures, and they develop an algorithm to compute it. Alternatively, one could replace s with the linear order relation $<$; however, $<$ does not preserve locality, and thus phenomena such as inversions cannot be dealt with. In this thesis, we assume a more general point of view by considering a relation that lies in between s and $<$. More precisely, we study first-order Ehrenfeucht-Fraïssé games (*EF-games* for short) played on structures with a limited ordering $<_p$, called *labeled $<_p$-structures*, which is defined as follows: given a pair of positions i and j, we have that $i <_p j$ if and only if $i < j$ and $j - i \leq p$. Relations s and $<$ can be recovered as special cases of $<_p$ for p equal to 1 and p greater than or equal to the maximum of the lengths of the two sequences, respectively.

The *playground* for a *comparison game* (on strings) is defined by a pair of strings w, w'. A *round* consists in a Spoiler's move followed by a Duplicator's one (hereafter we will abbreviate

Spoiler as **I** and Duplicator as **II**). At each round, **I** chooses one of the two strings and a position in it; **II** replays choosing a position in the other string. A *configuration* is a "snapshot" of the playground at a given stage of the game. We will write $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ to indicate the configuration where $\mathbf{i}^n$ (resp., $\mathbf{j}^n$) is the tuple of positions already chosen in w (resp. w'). A game with q rounds on $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$ is denoted by $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$; a game that goes on until either **I** repeats a move (**II** wins) or the current position is not a partial isomorphism (**I** wins) is denoted by $\mathcal{G}((w, \mathbf{i}^n), (w', \mathbf{j}^n))$. In the following, we will write $\mathbf{I}(\mathcal{G}_q)$ (resp., $\mathbf{II}(\mathcal{G}_q)$) to state that **I** (resp., **II**) has a winning strategy in q rounds. The *remoteness* $R(\mathcal{G})$ of $\mathcal{G}$ is the minimum q such that $\mathbf{I}(\mathcal{G}_q)$ (if such a q does not exist, we assume the remoteness to be infinite). A move by **I** in $\mathcal{G}$ is *optimal* if, whatever **II** replies, the new game $\mathcal{G}'$ has remoteness $R(\mathcal{G}') \leq R(\mathcal{G}) - 1$; similarly, **II**'s reply is optimal if $R(\mathcal{G}')R(\mathcal{G}) - 1$ no matter how **I** has played. The algorithmic complexity of EF-games has not been thoroughly investigated. In [104], it is proved that, given two (finite) structures, determining whether $\mathbf{II}(\mathcal{G}_q(\mathfrak{A}, \mathfrak{B}))$ is PSPACE-complete when the vocabulary contains at least one binary and one ternary relation symbol. Efficient algorithms for EF-games on specific classes of structures (equivalence relations, trees, unary relation structures, Boolean algebras, and some natural extensions of them) are given in [72]. In this chapter, we provide a characterization of EF-games on labeled $<_p$-structures that allows us to answer the following questions in polynomial time: given $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$, what is the remoteness of $\mathcal{G}((w, \mathbf{i}^n), (w', \mathbf{j}^n))$? What are the sets of **I**'s and **II**'s optimal moves?

In [95, 96], Montanari et al. propose an algorithm to solve EF-games on labeled successor structures (LSSs) $\mathfrak{A}, \mathfrak{B}$ in $O(n \log n)$ time, where $n = |\mathfrak{A}| + |\mathfrak{B}|$. The solution consists in combining a local and a global strategy. More precisely, $\mathbf{II}(\mathcal{G}_q)$ iff the game is *q-locally safe* and *q-globally safe.* The former property ensures that corresponding distinguished positions have the same relative positions up to a threshold distance and that suitable corresponding neighborhoods spell equal substrings in both words, so that **II** can correctly play q rounds within such neighborhoods (local moves). The latter property is based on counting the multiplicity and the degree of scattering of substrings "falling far away" from distinguished positions. In the case of $<_p$, the global strategy remains essentially unchanged, while the local strategy is quite different. Let us suppose **I**'s moves are coerced to be local. We first provide a necessary and sufficient condition for **II** to win games in q rounds on structures devoid of unary predicates (*q-distance safety*). Roughly speaking, distinguished positions give rise to "rigid" and "elastic" intervals. Suppose that **I** and **II** play the game $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$. If **I** chooses a new position i_{n+1} inside one of the rigid intervals induced by a distinguished position $i_h \in \mathbf{i}^n$, then **II** must choose a new position j_{n+1} such that $j_{n+1} - j_h = i_{n+1} - i_h$. If the new position is chosen inside one of the elastic intervals induced by i_h, then **II** must reply by choosing a new position inside the elastic interval induced by j_h, but not necessarily at the same distance. This guarantees q-distance safety. To deal with unary predicates, we introduce an additional condition (*$<_p$-safety for q-colors*). The basic ingredient is a recursive notion of *q-color*, which presents some similarities with the one given in [5]. $<_p$-safety for q-colors is guaranteed by constraining corresponding positions in $\mathbf{i}^n, \mathbf{j}^n$ to have the same q-color. As a matter of fact, for $p = 1$ the above conditions are equivalent to those for the successor relation s (as expected), while if $p \max(|w|, |w'|)$, then $<_p$ coincides with $<$. Similarly to the case of LSSs, the global strategy essentially consists in comparing the *multiplicity* and the *scattering* of r-colors, with $0 \leq r \leq q - 1$, in the portions of w and w' that are far away from $\mathbf{i}^n$ and $\mathbf{j}^n$, respectively.

This chapter is organized as follows. In Section 5.1 we provide some preliminary definitions. In Section 5.2 we introduce the main uses of EF-games in literature. In Section 5.3 we analyze the basic case of local games on $<_p$-structures. In Section 5.4 we take into consideration local games on labeled $<_p$-structures. Finally, global games on labeled $<_p$-structures are the subject of Section 5.5. Conclusions provide an assessment of the achieved results and outline future work directions. Detailed proofs can be found in Appendix 1.

5.1 Basic definitions

Let $\langle R, n\rangle$ denote a relation symbol R with arity n and let $\tau = \{\langle R_1, n_1\rangle, \ldots, \langle R_k, n_k\rangle\}$ be a finite relational vocabulary. A *τ-structure* $\mathfrak{A}$ is a tuple consisting of a set A, called the *domain*, and an n_i-ary relation $R_i^A \subseteq A^{n_i}$ for each $\langle R_i, n_i\rangle \in \tau$. We denote τ-structures with symbols $\mathfrak{A}$, $\mathfrak{B}$, etc... In this work, we assume that every vocabulary implicitly contains a symbol $=$ interpreted as equality. Besides, we restrict our attention to finite structures expanded with constants. A structure $\mathfrak{A}$ with distinguished elements $a_1, \ldots, a_k \in A$ is denoted by $(\mathfrak{A}, \mathbf{a}^k)$. Given a structure $\mathfrak{A}$ and $A' \subseteq A$, the *substructure* of $\mathfrak{A}$ *induced by* A' is the structure with domain A' whose relations are the relations of $\mathfrak{A}$ restricted to A'.

Two structures $(\mathfrak{A}, \mathbf{a}^h)$ and $(\mathfrak{B}, \mathbf{b}^k)$, with $h, k \geq 0$, are *isomorphic*, $(\mathfrak{A}, \mathbf{a}^h) \cong (\mathfrak{B}, \mathbf{b}^k)$ for short, if $h = k$ and there is an isomorphism f between $\mathfrak{A}$ and $\mathfrak{B}$ such that $f(a_j) = b_j$, for $1 \leq j \leq k$. The *isomorphism type* of a structure is the class of structures isomorphic to it. A *partial isomorphism* between $(\mathfrak{A}, \mathbf{a}^k)$ and $(\mathfrak{B}, \mathbf{b}^k)$, with $k \geq 0$, is an isomorphism of the substructures of $\mathfrak{A}$ and $\mathfrak{B}$ induced by $\mathbf{a}^k$ and $\mathbf{b}^k$, respectively.

Since we are interested in structures associated with strings, we give further definitions for this special case. Let Σ be a fixed alphabet, $w \in \Sigma^*$, and $p \geq 1$. A *labeled $<_p$-structure* is a pair $(w, \mathbf{i}^n)$, where $w = (\{1, \ldots, |w|\}, <_p, (P_a)_{a\in\Sigma})$, $i <_p j$ if and only if $0 < j - i \leq p$ for all $i, j \in \{1, \ldots, |w|\}$, and $i \in P_a$ if and only if $w[i] = a$ for all $i \in \{1, \ldots, |w|\}$, and $\mathbf{i}^n$ are distinguished positions $i_1, \ldots, i_n \in \{1, \ldots, |w|\}$.

Finally, given $i, j \in \{1, \ldots, |w|\}$, the *distance* $\delta(i, j)$ between $w[i]$ and $w[j]$ is $|i - j|$. The *k-distance* $\delta_k(i, j) : \mathbb{N} \times \mathbb{N} \to \mathbb{N} \cup \{\infty\}$ is a "truncated"distance defined as follows: $\delta_k(i, j) = \delta(i, j)$ if $\delta(i, j) \leq k$, $\delta_k(i, j) = \infty$ otherwise.

5.2 Review of literature

The rules of an EF-game usually have a counterpart in a logic, in a way that the possession of a winning strategy by one of the players relates to the ability of certain formulae of that logic to distinguish the structures used in the game. As a consequence, in literature the main use of EF-games is to prove inexpressibility results. In this context, the notion of *m-equivalence* is crucial.

Definition 5.2.1 *Let $\mathfrak{A}$ and $\mathfrak{B}$ be two structures over a vocabulary τ and let $\boldsymbol{a}^k$ and $\boldsymbol{b}^k$ be two k-tuples of elements of A and B, respectively. The structures $(\mathfrak{A}, \boldsymbol{a}^k)$ and $(\mathfrak{B}, \boldsymbol{b}^k)$ are* m-equivalent, *written $(\mathfrak{A}, \boldsymbol{a}^k) \equiv_m (\mathfrak{B}, \boldsymbol{b}^k)$, if $(\mathfrak{A}, \boldsymbol{a}^k) \models \varphi(x) \Leftrightarrow (\mathfrak{B}, \boldsymbol{b}^k) \models \varphi(x)$ for all first-order formulae $\varphi(x)$ with quantifier depth at most m.*

Definition 5.2.2 *A set of sentences Φ defines a class of structures $\mathcal{K}$ if, for every τ-structure $\mathfrak{A}$, $\mathfrak{A} \models \Phi \Leftrightarrow \mathfrak{A} \in \mathcal{K}$.*

The following result relates m-equivalence to definability in first-order logic.

Theorem 5.2.3 *A class $\mathcal{K}$ of finite structures is first-order definable if and only if there is $m \in \mathbb{N}$ such that, whenever $\mathfrak{A} \in \mathcal{K}$ and $\mathfrak{B} \notin \mathcal{K}$, then $\mathfrak{A} \not\equiv_m \mathfrak{B}$.*

Theorem 5.2.3 is usually used in a negative form to prove inexpressibility results: if for all $m \in \mathbb{N}$ there exist two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ such that $\mathfrak{A} \in \mathcal{K}$, $\mathfrak{B} \notin \mathcal{K}$, and $\mathfrak{A} \equiv_m \mathfrak{B}$, then $\mathcal{K}$ is not first order definable.

In literature different ways for describing m-equivalence have been proposed. Fraïssé's work was concerned with the following notion of m-isomorphism, which is a weakening of isomorphism well suited for capturing m-equivalence.

Definition 5.2.4 *Two structures $(\mathfrak{A}, \boldsymbol{a}^k)$ and $(\mathfrak{B}, \boldsymbol{b}^k)$ are* m-isomorphic, *written $(\mathfrak{A}, \boldsymbol{a}^k) \cong_m (\mathfrak{B}, \boldsymbol{b}^k)$, if there is a sequence of nonempty sets $I_0, \ldots, I_m$ of partial isomorphisms such that $\{(a_i, b_i)\}_{1\leq i\leq k} \in I_m$ and satisfying, for each $k = 1, \ldots, m$, the following back-and-forth property:*

- *(FORTH PROPERTY) for each $p \in I_k$ and for each $a \in A$ there is $b \in B$ such that $p \cup \{(a,b)\} \in I_{k-1}$;*
- *(BACK PROPERTY) for each $p \in I_k$ and for each $b \in B$ there is $a \in A$ such that $p \cup \{(a,b)\} \in I_{k-1}$.*

The following theorem gives an algebraic characterization of m-equivalence based on the notion of m-isomorphism.

Theorem 5.2.5 *(Fraïssé, [52]). For structures $(\mathfrak{A}, \boldsymbol{a}^k)$ and $(\mathfrak{B}, \boldsymbol{b}^k)$, and $m \geq 0$,*

$$(\mathfrak{A}, \boldsymbol{a}^k) \cong_m (\mathfrak{B}, \boldsymbol{b}^k) \Leftrightarrow (\mathfrak{A}, \boldsymbol{a}^k) \equiv_m (\mathfrak{B}, \boldsymbol{b}^k).$$

A game-theoretic characterization of m-equivalence is given in the following result.

Theorem 5.2.6 *(Ehrenfeucht, [40]). For structures $(\mathfrak{A}, \boldsymbol{a}^k)$ and $(\mathfrak{B}, \boldsymbol{b}^k)$, and $m \geq 0$,*

$$\boldsymbol{II}(\mathcal{G}_m((\mathfrak{A}, \boldsymbol{a}^k), (\mathfrak{B}, \boldsymbol{b}^k))) \Leftrightarrow (\mathfrak{A}, \boldsymbol{a}^k) \equiv_m (\mathfrak{B}, \boldsymbol{b}^k).$$

Corollary 5.2.7 *A class $\mathcal{K}$ of τ-structures is first order definable if and only if there is $m \in \mathbb{N}$ such that, whenever $\mathfrak{A} \in \mathcal{K}$ and $\mathfrak{B} \notin \mathcal{K}$, then $\boldsymbol{I}(\mathcal{G}_m(\mathfrak{A}, \mathfrak{B}))$.*

Corollary 5.2.7 can be used to prove that a property is not first-order definable: it is sufficient to show that **II** has a winning strategy in suitable EF-games.

It is possible to give a logical description of the equivalence classes of relation $\cong_m$, and thus of relation $\equiv_m$, by formulae of quantifier depth at most m, that is, for each structure $(\mathfrak{A}, \mathbf{a}^k)$, a formula $\varphi^m_{(\mathfrak{A},\mathbf{a}^k)}(x)$ (Hintikka formula) that holds exactly in structures m-isomorphic to $(\mathfrak{A}, \mathbf{a}^k)$. Each first-order formula $\varphi(x)$ is equivalent to the disjunction of a finite number of Hintikka formulae, each one describing a $\cong_m$ class. Another interesting characterization of first-order logic is the one given by Gaifman in [53], where he proves that every first-order sentence is equivalent to a boolean combination of "local" sentences, whose evaluation only requires the inspection of a set of small substructures which are far one from another. The locality of first order logic has consequences on the form assumed by the strategies of the corresponding EF-games. The "high-level" strategy of **II** in such games consists in miming **I** if he plays in the neighborhoods of already chosen elements; otherwise, **II** must find an element with a sufficiently large "equivalent" neighborhood outside all current neighborhoods. The hard part in characterizing a winning strategy lies in the precise meaning of "miming" and finding "equivalent" neighborhoods.

A difficult task in EF theory consists in giving an explicit description of $\equiv_m$ classes. The following result assesses the complexity of the problem of deciding whether two τ-structures are m-equivalent.

Theorem 5.2.8 *(Pezzoli, [104]). Let τ be a vocabulary containing at least one binary and one ternary relation. Given two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ and $m \in \mathbb{N}$, the problem of determining whether $\boldsymbol{II}(\mathcal{G}_m(\mathfrak{A}, \mathfrak{B}))$ is PSPACE-complete.*

Although the problem is hard in general, for specific classes of structures better results can be obtained. As an example, we describe (unlabeled) linear orderings, one of the few cases for which the complete picture is known. For a given $n \in N$, the linear ordering of size n is the relational structure $\mathfrak{L}_n = ([1, n], <)$.

Theorem 5.2.9 *Let $\mathfrak{L}_{n1}$ and $\mathfrak{L}_{n2}$ be two linear orderings, and $m \in \mathbb{N}$. Then*

$$\boldsymbol{II}(\mathcal{G}_m(\mathfrak{L}_{n1}, \mathfrak{L}_{n2})) \Leftrightarrow \mathfrak{L}_{n1} \cong \mathfrak{L}_{n2} \vee n1, n2 \geq 2^m - 1.$$

Thus, determining the winner of a game simply requires to compare the size of the structures.

One of the first conditions to be developed for guaranteeing **II** to win can be linked to the work of Hanf. His work is concerned with elementary equivalence (two structures are *elementary* *equivalent* if they satisfy the same first-order sentences). Before stating Hanf's main theorem (Theorem 5.2.11), we give some introductory definitions.

The *Gaifman graph* of a structure $\mathfrak{A}$ is an undirected graph whose set of nodes is A, having an edge (a_1, a_2) whenever a_1 and a_2 occur in the same tuple of some relation of $\mathfrak{A}$. Since we have assumed that every structure contains equality, the Gaifman graph of a structure $\mathfrak{A}$ contains all edges of the form (a, a) for $a \in A$. The Gaifman graph allows one to define a metric on every structure $\mathfrak{A}$: let $\delta(a_1, a_2)$ be the length of the shortest path between a_1 and a_2 in the Gaifman graph of $\mathfrak{A}$ (we set $\delta(a_1, a_2) = \infty$ if no connecting path between a_1 and a_2 exists). We can extend the notion of distance from pairs of elements to pairs of tuples as follows: given two tuples $\mathbf{a}^k$ and $\mathbf{b}^l$, $\delta(\mathbf{a}^k, \mathbf{b}^l) = \min_{1\le i\le k, 1\le j\le l} \delta(a_i, b_j)$. Given a structure $\mathfrak{A}$ and a tuple $\mathbf{a}^k$ of elements of A, the *ball* $B_r^{\mathfrak{A}}(\mathbf{a}^k)$ of radius r around $\mathbf{a}^k$ is the set $B_r^{\mathfrak{A}}(\mathbf{a}^k) = \{c \in A \mid \delta(c, \mathbf{a}^k) \le r\} \subseteq A$. The *$r$-neighborhood* $N_r^{\mathfrak{A}}$ around $\mathbf{a}^k$ is the substructure of $\mathfrak{A}$ induced by $B_r^{\mathfrak{A}}(\mathbf{a}^k)$. Given two tuples $\mathbf{a}^k$ and $\mathbf{b}^k$, we say that $N_r^{\mathfrak{A}}(\mathbf{a}^k)$ is *isomorphic* to $N_r^{\mathfrak{A}}(\mathbf{b}^k)$, written $N_r^{\mathfrak{A}}(\mathbf{a}^k) \cong N_r^{\mathfrak{A}}(\mathbf{b}^k)$, if there is an isomorphism between the two neighborhoods mapping a_i to b_i, for $1 \le i \le k$. The *isomorphism type* of a neighborhood is the class of neighborhoods isomorphic to it.

Definition 5.2.10 *Given two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, we write $\mathfrak{A} \leftrightarrows_r \mathfrak{B}$ if there is a bijection $f : A \to B$ such that $N_r^{\mathfrak{A}}(a) \cong N_r^{\mathfrak{B}}(f(a))$ for each $a \in A$.*

Theorem 5.2.11 *(Hanf, [61]). Two structures $\mathfrak{A}$ and $\mathfrak{B}$ are elementary equivalent if $\mathfrak{A} \leftrightarrows_r \mathfrak{B}$ for all $r \in \mathbb{N}$.*

Thus elementary equivalence is guaranteed provided that structures realize the same multiset of isomorphism types of r-neighborhoods. However, Theorem 5.2.11 is not very interesting for finite structures, because two finite structures are elementarily equivalent if and only if they are isomorphic. Fagin et al. [47] have adapted the theorem to the finite: since elementary equivalence is a notion too strong, it is replaced by a weaker form of equivalence, based on the existence of a winning strategy for **II** in an EF-game. By Theorem 5.2.6, this notion coincides with m-equivalence. Intuitively, they require that every isomorphism type occurs equally often in both structures or it occurs very often in both structures. This is sufficient to ensure a winning strategy for **II**. Given a τ-structure $\mathfrak{C}$ and $r \in \mathbb{N}$, let $C_r^t = \{c \in C \mid |[N_r^{\mathfrak{C}}(c)]_{\cong}| \le t\}$. The set C_r^t contains the elements of the domain whose r-neighborhood does not occur very often (no more than t times) in the structure.

Definition 5.2.12 *Given two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$, we write $\mathfrak{A} \leftrightarrows_r^t \mathfrak{B}$ if there is a bijection $f : A_r^t \to B_r^t$ such that $N_r^{\mathfrak{A}}(a) \cong N_r^{\mathfrak{B}}(f(a))$ for each $a \in A_r^t$.*

If $\mathfrak{A} \leftrightarrows_r^t \mathfrak{B}$, then isomorphic r-neighborhoods occur the same number of times in both structures or they occur more than t times in both structures.

We define the *degree* $d(a)$ of an element a as the number of its adjacent elements in the Gaifman graph, that is, $d(a) = |B_1^{\mathfrak{A}}(a)| - 1$. The degree of a structure $\mathfrak{A}$ is the maximum degree of its elements.

Theorem 5.2.13 *(Sphere lemma, [47]). Let $\mathfrak{A}$ and $\mathfrak{B}$ be two τ-structures with degree at most d, and let $m \in \mathbb{N}$. If $\mathfrak{A} \leftrightarrows_r^t \mathfrak{B}$ for $r = 3^{m-1}$ and $t = m \cdot d^{3^m}$, then $\mathfrak{A} \equiv_m \mathfrak{B}$.*

In general, two (sub)structures do not need to be isomorphic for **II** to be able to win a game. So, the requirement of having isomorphic neighborhoods is too strong. In [5], Arora and Fagin exploit this fact to give a more versatile sufficient condition for the second player to have a winning strategy. Their result is still based on a multiplicity argument, but elements having the same "approximately isomorphic" neighborhoods are taken into consideration. Additional hypotheses, however, are needed to counterbalance this less restrictive condition, the main of which regards the fact that neighborhoods are assumed to be tree-like structures.

Although Arora and Fagin' s theorem holds for structures over arbitrary languages, for the sake of simplicity we will refer to directed colored graphs, that is we fix a language $\gamma = \{(E, 2), (P_1, 1), \ldots, (P_k, 1)\}$, where E is the edge relation of the graph, and the unary predicates that label the nodes determine their *color*. Note that there are at most 2^k different colors, one for each possible combination of the truth values of the unary predicates. A color can be viewed as a description of the "type" of a node. The following definition generalizes this idea from single points to r-neighborhoods around points, and at the same time it weakens Definition 5.2.12.

Definition 5.2.14 *Let $\mathfrak{G}$ be a γ-structure, and m, $r \in \mathbb{N}$. The (m, r)-color of a node $a \in G$ is inductively defined as follows:*

1. *the $(m, 0)$-color of a is its color along with a description of whether it has a self-loop (that is, whether $E^{\mathfrak{G}}(a, a)$ holds) and whether it is a distinguished element;*
2. *the $(m, r + 1)$-color of a is its (m, r)-color together with the following triples of values, one for each possible (m, r)-color τ:*
   - *the number of nodes b having (m, r)-color τ such that $E^{\mathfrak{G}}(a, b)$ holds and $E^{\mathfrak{G}}(b, a)$ does not hold, or ∞ if such a number is at least m;*
   - *the number of nodes b having (m, r)-color τ such that $E^{\mathfrak{G}}(a, b)$ does not hold and $E^{\mathfrak{G}}(b, a)$ holds, or ∞ if such a number is at least m;*
   - *the number of nodes b having (m, r)-color τ such that both $E^{\mathfrak{G}}(a, b)$ and $E^{\mathfrak{G}}(b, a)$ hold, or ∞ if such a number is greater than m.*

Given two γ-structures $\mathfrak{G}$ and $\mathfrak{H}$, we write $\mathfrak{G} \rightleftharpoons^m_r \mathfrak{H}$ if there is a bijection $f : G \to H$ such that a and $f(a)$ have the same (m, r)-color, for all $a \in G$.

So, the (m, r)-color of a node partially describes the r-neighborhood around that node. In the following result, the (m, r)-color of an edge is the ordered pair of the (m, r)-colors of the corresponding nodes.

Theorem 5.2.15 *Let $\mathfrak{G}$ and $\mathfrak{H}$ be two γ-structures with degree at most d, and let $m \in \mathbb{N}$. Then $\textbf{\textit{II}}(\mathcal{G}_m(\mathfrak{G}, \mathfrak{H}))$ if the following conditions are satisfied:*

1. *$\mathfrak{G} \rightleftharpoons^m_r \mathfrak{H}$ for $r = 3^{2m}$;*
2. *$\mathfrak{G}$ and $\mathfrak{H}$ do not have undirected cycles of length less than r;*
3. *if $E^{\mathfrak{G}}(a, b)$ holds but $E^{\mathfrak{H}}(f(a), f(b))$ does not hold, or viceversa, then there are at least d^r edges in both structures having the same (m, r)-color as (a, b) (resp., as $(f(a), f(b))$).*

As in the case of the Sphere lemma (Theorem 5.2.13), the condition expressed by Theorem 5.2.15 is essentially based on counting the multiplicity of "equivalent"substructures, where equivalence is formalized as "having the same (m, r)-color" that is, roughly speaking, as having approximately the same local tree-like substructures.

5.3 Local games on $<_p$ structures

In this section we start analyzing EF-games on structures provided with with a $<_p$ relation: we consider the simplified case of local games on strings devoid of unary predicates. We provide necessary and sufficient conditions for $\mathbf{II}(\mathcal{G}_q)$, for any given q. The first concept we introduce is the one of *pstep*, which can be used to measure the "signed distance" between two elements in terms of the number of intervals of length p separating them.

Definition 5.3.1 *Let $i, j, k, p \in \mathbb{N}$, with $i, j, p > 0$ and $k \geq p$. We define the function $pstep^{(p)}_k(i, j)$ as follows:*

$$pstep^{(p)}_k(i, j) = \begin{cases} 0 & \text{if } i = j \\ \lceil \frac{\delta(i,j)}{p} \rceil & \text{if } \delta_k(i, j) < \infty \text{ and } i < j \\ \lfloor -\frac{\delta(i,j)}{p} \rfloor & \text{if } \delta_k(i, j) < \infty \text{ and } i > j \\ \infty & \text{if } \delta_k(i, j) = \infty. \end{cases}$$

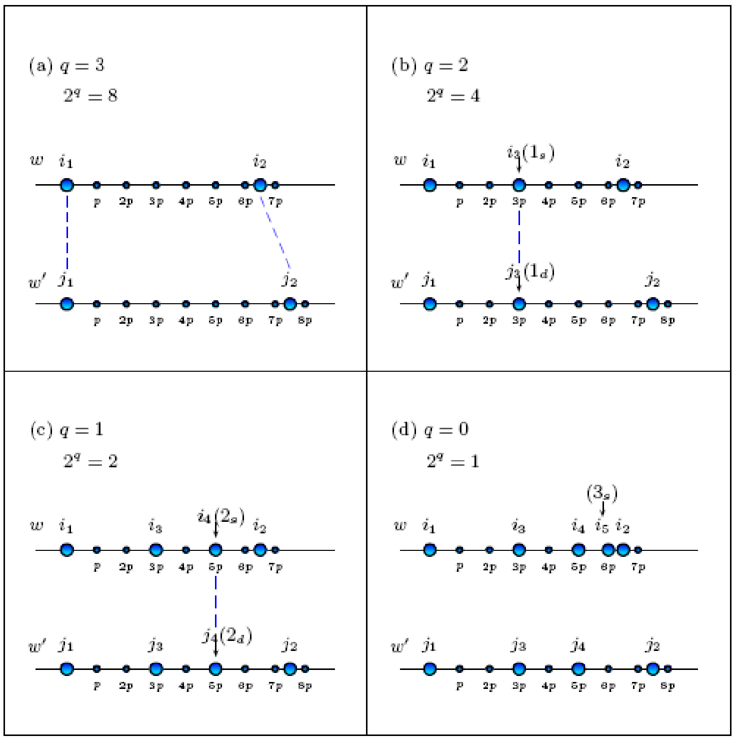


Figure 5.1: *pstep*-safety.

The next definition expresses *pstep-safety* of configurations (in the k-horizon). Condition (1) refers to internal positions, while conditions (2) and (3) deal with positions belonging to the prefix and the suffix, respectively.

Definition 5.3.2 *Let $p, k \in \mathbb{N}$, with kp. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is pstep-*safe *in the k-horizon if the following conditions hold:*

1. $pstep_k^{(p)}(i_r, i_s) = pstep_k^{(p)}(j_r, j_s)\ \forall r, s \in \{1 \dots n\}$;
2. $pstep_{k-p}^{(p)}(0, i_r) = pstep_{k-p}^{(p)}(0, j_r)\ \forall r \in \{1 \dots n\}$;
3. $pstep_{k-p}^{(p)}(i_r, |w| + 1) = pstep_{k-p}^{(p)}(j_r, |w'| + 1)\ \forall r \in \{1 \dots n\}$.

Local moves are defined on the basis of *pstep-regions*.

Definition 5.3.3 *Let $q > 0$, $w \in \Sigma^\star$, and let $\boldsymbol{i}^n$ be a set of positions in w. The pstep-*region *$Pstepreg_q^p(w, \boldsymbol{i}^n)$ is the set $\{j | \delta(0, j) \leq p \cdot (2^{q-1} - 1) \vee \delta(j, |w| + 1) \leq p \cdot (2^{q-1} - 1) \vee \exists r (1 \leq r \leq n \wedge \delta(i_r, j) \leq p \cdot 2^{q-1})\}$.*

Definition 5.3.4 *Let $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ be a configuration and let $q > 0$ be the number of remaining moves. A move is* local *if it is played within $Pstepreg_q^p(w, \boldsymbol{i}^n)$ or $Pstepreg_q^p(w', \boldsymbol{j}^n)$.*

Lemma 5.3.6 generalizes the condition of local safety for LSSs [95] by substituting intervals of length p for single positions. If two positions in a string are sufficiently close to each other, the corresponding positions in the other string must be at the same "*pstep* distance". In Example 5.3.5, we provide a configuration which is not *pstep-safe* in the $(p \cdot 2^3)$-horizon. Moreover, we outline a strategy **I** can adopt to win in 3 rounds.

Example 5.3.5 Consider the configuration in Figure 5.1. It is immediate to verify that is not *pstep-safe* in the $(p \cdot 2^3)$-horizon. We show a strategy **I** can adopt to win in 3 rounds.

(a) $q = 3$: $pstep^{(p)}_{8p}(i_1, i_2) = 7$ and $pstep^{(p)}_{8p}(j_1, j_2) = 8$. Hence, the configuration is not *pstep-safe* in the $(p \cdot 2^3)$-horizon and thus $\mathbf{I}(\mathcal{G}_3((w, \mathbf{i}^2), (w', \mathbf{j}^2)))$. An optimal move for **I** is to choose $i_3 = i_1 + \lfloor \frac{7}{2} \rfloor \cdot p = i_1 + 3p$ in w. An optimal reply from **II** is to choose $j_3 = j_1 + 3p$ in w'.

(b) $q = 2$: $pstep^{(p)}_{4p}(i_3, i_2) = 4$ and $pstep^{(p)}_{4p}(j_3, j_2) = \infty$. Again, the configuration is not *pstep-safe* in the $(p \cdot 2^2)$-horizon and thus $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^3), (w', \mathbf{j}^3)))$. If **I** chooses $i_4 = i_3 + \frac{4}{2} \cdot p = i_3 + 2p$ in w, **II** can reply by choosing $j_4 = j_3 + 2p$ in w'.

(c) $q = 1$: $pstep^{(p)}_{2p}(i_4, i_2) = 2$ and $pstep^{(p)}_{2p}(j_4, j_2) = \infty$. Once more, the configuration is not *pstep-safe* in the $(p \cdot 2^1)$-horizon and thus $\mathbf{I}(\mathcal{G}_1((w, \mathbf{i}^4), (w', \mathbf{j}^4)))$. If **I** chooses $i_5 = i_4 + p$ in w, it holds that $i_4 <_p i_5$ and $i_5 <_p i_2$. Hence, **II** is not able to find an element j_5 such that $j_4 <_p j_5$ and $j_5 <_p j_2$.

(d) $q = 0$: whatever **II**'s move at the previous round has been, **I** is the winner.

■

Lemma 5.3.6 *Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not pstep*-safe *in the $(p \cdot 2^q)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

The next lemma shows that if two positions in a string are sufficiently close to each other, that is, at a distance less than or equal to $2^q - 1$, the corresponding positions in the other string must exactly at the same distance (as a matter of fact, such a property holds for $<$ as well).

As a preliminary step, for every $k > 0$, we introduce a function ϑ_k that computes the truncated signed distance between two positions.

Definition 5.3.7 *Let $i, j, k \in \mathbb{N}$, with $i, j, k > 0$. We define the function $\vartheta_k(i, j)$ as follows:*

$$\vartheta_k(i,j) = \begin{cases} i - j & \text{if } \delta_k(i,j) < \infty \\ \infty & \text{otherwise.} \end{cases}$$

Definition 5.3.8 *A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is ϑ*-safe *in the k-horizon if the following conditions hold:*

1. $\vartheta_k(i_r, i_s) = \vartheta_k(j_r, j_s)$ $\forall r, s \in \{1 \dots n\}$;
2. $\vartheta_k(0, i_r) = \vartheta_k(0, j_r)$ $\forall r \in \{1 \dots n\}$;
3. $\vartheta_k(i_r, |w| + 1) = \vartheta_k(j_r, |w'| + 1)$ $\forall r \in \{1 \dots n\}$.

Lemma 5.3.9 *Let $w, w' \in \Sigma^\star$ and $q > 0$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not ϑ*-safe *in the $(2^q - 1)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Observe that if we replace $<_p$ with s, that is, if we take $p = 1$, Lemma 5.3.9 becomes a special case of Lemma 5.3.6. In Example 5.3.10 we provide a configuration which is not *ϑ-safe* in the $(2^3 - 1)$-horizon. Moreover, we outline a strategy **I** can adopt to win in 3 rounds.

Example 5.3.10 Consider the configuration in Figure 5.2. It is immediate to verify that it is not *ϑ-safe* in the $(2^3 - 1)$-horizon. We show a strategy **I** can adopt in order to win in 3 rounds.

(a) $q = 3$: $\delta(i_1, i_2) = 6$ and $\delta(j_1, j_2) = 7$. It follows that the configuration is not *ϑ-safe* in the $(2^3 - 1)$-horizon and thus $\mathbf{I}(\mathcal{G}_3((w, \mathbf{i}^2), (w', \mathbf{j}^2)))$. An optimal move for **I** is to choose $j_3 = \lfloor \frac{j_1 + j_2}{2} \rfloor = j_1 + 3$ in w'. An optimal reply for **II** is to choose $i_3 = \frac{i_1 + i_2}{2} = i_1 + 3$ in w.

(b) $q = 2$: $\delta(i_3, i_2) = 3$ and $\delta(j_3, j_2) = 4$. Again, the configuration is not *ϑ-safe* in the $(2^2 - 1)$-horizon and thus $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^3), (w', \mathbf{j}^3)))$. If **I** chooses $j_4 = \lfloor \frac{j_3 + j_2}{2} \rfloor = j_3 + 2$ in w', **II** can reply by choosing $i_4 = \lfloor \frac{i_3 + i_2}{2} \rfloor = i_3 + 1$ in w.

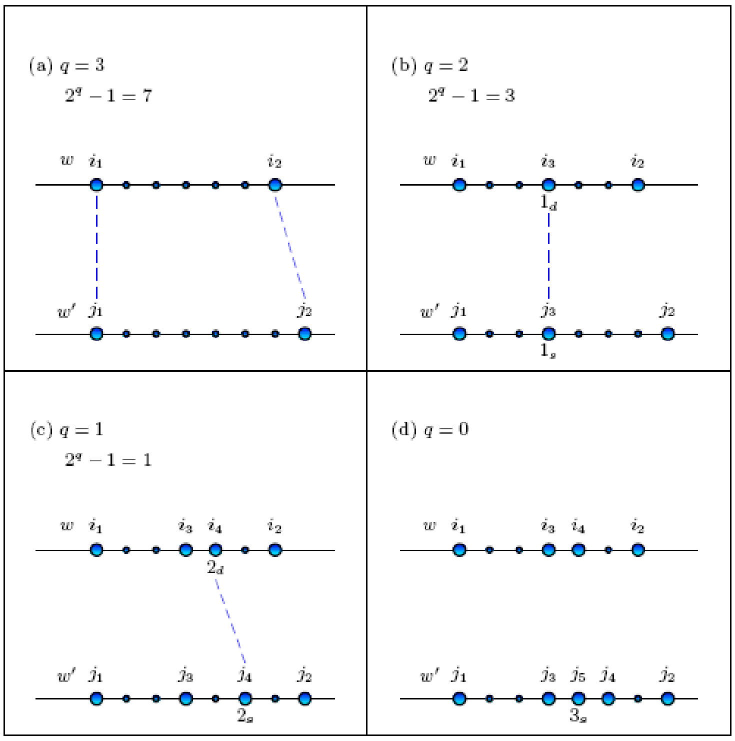


Figure 5.2: ϑ-safety.

(c) $q = 1$: $\delta(i_3, i_4) = 1$ and $\delta(j_3, j_4) = 2$. Once more, the configuration is not *ϑ-safe* in the $(2^1 - 1)$-horizon and thus $\mathbf{I}(\mathcal{G}_1((w, \mathbf{i}^4), (w', \mathbf{j}^4)))$. If **I** chooses $j_5 = j_3 + 1 = j_4 - 1$ in w', **II** is not able to find a position between i_3 and i_4 in w.

(d) $q = 0$: whatever **II**'s move at the previous round has been, **I** has won.

■

Lemma 5.3.6 and Lemma 5.3.9 basically capture the features that $<_p$ has in common with s and $<$, respectively. However, $<_p$ cannot simply be viewed as the composition of s and $<$. The distinctive features of $<_p$, that differentiate it from s and $<$, are captured by the following definition and will be taken into consideration by Theorem 5.3.17.

Let us now introduce the key notions of "rigid" and "elastic" intervals induced by the set of positions $\mathbf{i}^n$ (resp., $\mathbf{j}^n$).

Definition 5.3.11 *Let $q > 1$ and $i \in \mathbb{N}$. The 0th q-rigid interval induced by position i is the closed interval $\rho^+_{0,q}(i) = \rho^-_{0,q}(i) = [i - \alpha^0_q, i + \alpha^0_q]$, where $\alpha^0_q = 2^{q-1} - 1$. The kth* right (resp., left) q-rigid interval induced by position i, *with $0 < k \leq 2^{q-2}$, is the interval $\rho^+_{k,q}(i) = (c - \alpha^z_q, c + \alpha^z_q]$ (resp., $\rho^-_{k,q}(i) = [c - \alpha^z_q, c + \alpha^z_q)$) where $c = i + kp$ (resp., $c = i - kp$) and $\alpha^z_q = 1 + \sum_{j=z-1}^{q-2}(2^j - 1) = 2^{q-1} - 2^{z-1} - q + z + 1$, where $z = \lceil \log_2 k \rceil + 1$. The kth* right (resp., left) q-elastic interval induced by position i *is the interval between the $(k-1)$th and the kth q-rigid right (resp., left) interval.*

It is worth noting that elastic intervals come into play only when p is large enough with respect to q. More precisely, in a game $\mathcal{G}_q$, for all $1 \leq k \leq 2^{q-2}$, the kth (right or left) q-elastic interval induced by any position in $\mathbf{i}^n$ is not empty if (and only if) $p > \alpha^z_q + \alpha^{\overline{z}}_q - 1$, where $z = \lceil \log_2(k-1) \rceil + 1$ for $k > 1$ and $z = 0$ for $k = 1$, and $\overline{z} = \lceil \log_2 k \rceil + 1$. In Figure 5.3 we show the right 5-rigid (resp., 5-elastic) intervals induced by i, which are represented in black (resp., gray).

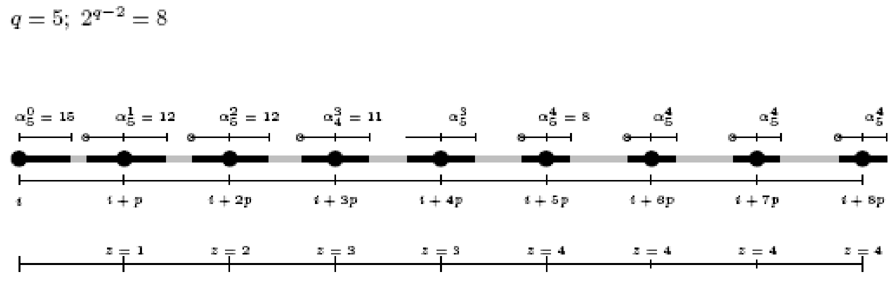


Figure 5.3: Rigid and elastic intervals ($q = 5$; $2^{q-2} = 8$).

The role of rigid and elastic intervals in the evolution of a game is expressed by Theorem 5.3.17. It can be intuitively explained as follows. Consider a game $\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n))$, with $q > 1$. If **I** chooses a new position i_{n+1} inside a rigid interval induced by the position $i_r \in \mathbf{i}^n$, that is, if $|i_{n+1} - i_r| \in (kp - \alpha_q^z, kp + \alpha_q^z]$ for a suitable k, then **II** must choose a new position j_{n+1} such that $j_{n+1} - j_r = i_{n+1} - i_r$. If **I** chooses a new position i_{n+1} inside an elastic interval induced by i_r, then **II** must reply by choosing a new position inside the corresponding elastic interval induced by j_r, but not necessarily at the same distance.

Definition 5.3.12 *Let $q > 0$. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is p-int-*safe *in the k-horizon if the following conditions hold:*

1. *$\forall r, s \in \{1, \ldots, n\}$, with $r < s$, if there exists $0 \le h \le 2^{k-1}$ such that $i_s \in \rho^+_{h,k+1}(i_r)$ or $j_s \in \rho^+_{h,k+1}(j_r)$, then $i_s - i_r = j_s - j_r$;*

2. *$\forall r \in \{1, \ldots, n\}$, if there exists $0 \le h \le 2^{k-1}$ such that $\delta(0, i_r) \in [hp + 1, hp + \alpha^z_{k+1}]$ or $\delta(0, j_r) \in [hp + 1, hp + \alpha^z_{k+1}]$, where $z = \lceil \log_2 h \rceil + 1$ for $h > 0$ and $z = 0$ for $h = 0$, then $\delta(0, i_r) = \delta(0, j_r)$;*

3. *$\forall r \in \{1, \ldots, n\}$, if there exists $0 \le h \le 2^{k-1}$ such that $\delta(i_r, |w| + 1) \in [hp + 1, hp + \alpha^z_{k+1}]$ or $\delta(j_r, |w'| + 1) \in [hp + 1, hp + \alpha^z_{k+1}]$, where $z = \lceil \log_2 h \rceil + 1$ for $h > 0$ and $z = 0$ for $h = 0$, then $\delta(i_r, |w| + 1) = \delta(j_r, |w'| + 1)$.*

Lemma 5.3.13 *Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is p-int-*safe *in the q-horizon, with $q > 0$, then it is ϑ-*safe *in the $(2^q - 1)$-horizon.*

Lemma 5.3.14 *Let $w, w' \in \Sigma^\star$ and $q > 0$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not p-int-*safe *in the q-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Proof. The proof is by induction on q.
Base case: $q = 1$. We distinguish three sets of cases:

1. (a) $\exists r, s$ such that $\delta(i_r, i_s) = 1$, but $\delta(j_r, j_s) \neq 1$ (or $\delta(i_r, i_s) \neq 1$, but $\delta(j_r, j_s) = 1$).
   (b) $\exists r, s$ such that $\delta(i_r, i_s) = p$, but $\delta(j_r, j_s) \neq p$ (or $\delta(i_r, i_s) \neq p$, but $\delta(j_r, j_s) = p$).
   (c) $\exists r, s$ such that $\delta(i_r, i_s) = p + 1$, but $\delta(j_r, j_s) \neq p + 1$ (or $\delta(i_r, i_s) \neq p + 1$, but $\delta(j_r, j_s) = p + 1$).

2. (a) $\exists r$ such that $\delta(0, i_r) = 1$, but $\delta(0, j_r) \neq 1$ (or $\delta(0, i_r) \neq 1$, but $\delta(0, j_r) = 1$).
   (b) $\exists r$ such that $\delta(0, i_r) = p+1$, but $\delta(0, j_r) \neq p+1$ (or $\delta(0, i_r) \neq p+1$, but $\delta(0, j_r) = p+1$).

3. (a) $\exists r$ such that $\delta(i_r, |w| + 1) = 1$, but $\delta(j_r, |w'| + 1) \neq 1$ (or $\delta(i_r, |w| + 1) \neq 1$, but $\delta(j_r, |w'| + 1) = 1$).
   (b) $\exists r$ such that $\delta(i_r, |w| + 1) = p + 1$, but $\delta(j_r, |w'| + 1) \neq p + 1$ (or $\delta(i_r, |w| + 1) \neq p + 1$, but $\delta(j_r, |w'| + 1) = p + 1$).

We provide the details for the first set of cases; the others are similar and thus omitted.
(a) The configuration is not ϑ-*safe* in the (2^1-1)-horizon and thus, by Lemma 5.3.9, $\mathbf{I}(\mathcal{G}_1((w,\mathbf{i}^n),(w',\mathbf{j}^n)))$.
(b) If $\delta(j_r,j_s) > p$, then the configuration is not a partial isomorphism. Hence, we restricted our attention to the case in which $\delta(j_r,j_s) < p$. Furthermore, we assume that $i_r - i_s$ and $j_r - j_s$ have the same sign (if this is not the case, the configuration is not *pstep-safe* in the $(p \cdot 2^0)$-horizon, and thus it is not a partial isomorphism). Without loss of generality, let $i_r < i_s$ and $j_r < j_s$. An optimal move for **I** is to choose $j_{n+1} = j_s - p$. If **II** chooses $i_{n+1} < i_r$, then $i_s - i_{n+1} > p$; if **II** chooses $i_{n+1} > i_r$, then $j_{n+1} < j_r$ but $i_{n+1} > i_r$.
(c) If $\delta(j_r,j_s) \le p$, the configuration is not a partial isomorphism. Hence, let $\delta(j_r,j_s) > p+1$. Furthermore, let $i_r - i_s$ and $j_r - j_s$ have the same sign. Without loss of generality, let $i_r < i_s$ and $j_r < j_s$. An optimal move for **I** is to choose $j_{n+1} = j_r + p + 1 (< j_s)$. If **II** replies with $i_{n+1} \ge i_s$, the configuration is not a partial isomorphism; if **II** replies with $i_{n+1} < i_s$, then $\delta(i_r, i_{n+1}) \le p$ while $\delta(j_r, j_{n+1}) = p+1$ and thus the configuration is not a partial isomorphism.

Inductive step: $q > 1$. As in the base case, we distinguish three sets of cases:

1. (a) $\exists r,s \in \{1,\dots,n\}$ such that $\delta(i_r,i_s) \le 2^q - 1$ or $\delta(j_r,j_s) \le 2^q - 1$ but $i_r - i_s \ne j_r - j_s$.
   (b) $\exists r,s \in \{1,\dots,n\}$, $\exists 1 \le k \le 2^{q-1}$ such that $i_s \in \rho^+_{k,q+1}(i_r)$ or $j_s \in \rho^+_{k,q+1}(j_r)$ but $i_r - i_s \ne j_r - j_s$.
2. (a) $\exists r \in \{1,\dots,n\}$ such that $\delta(0,i_r) \le 2^q - 1$ or $\delta(0,j_r) \le 2^q - 1$, but $\delta(0,i_r) \ne \delta(0,j_r)$.
   (b) $\exists r \in \{1,\dots,n\}$, $\exists 1 \le k \le 2^{q-1}$ such that $\delta(0,i_r) \in [kp+1, kp+\alpha^z_{q+1}]$ or $\delta(0,j_r) \in [kp+1, kp+\alpha^z_{q+1}]$, where $z = \lceil \log_2 k \rceil + 1$, but $\delta(0,i_r) \ne \delta(0,j_r)$.
3. (a) $\exists r \in \{1,\dots,n\}$ such that $\delta(i_r, |w|+1) \le 2^q - 1$ or $\delta(j_r, |w'|+1) \le 2^q - 1$, but $\delta(i_r, |w|+1) \ne \delta(j_r, |w'|+1)$.
   (b) $\exists r \in \{1,\dots,n\}$, $\exists 1 \le k \le 2^{q-1}$ such that $\delta(i_r, |w|+1) \in [kp+1, kp+\alpha^z_{q+1}]$ or $\delta(j_r, |w'|+1) \in [kp+1, kp+\alpha^z_{q+1}]$, where $z = \lceil \log_2 k \rceil + 1$, but $\delta(i_r, |w|+1) \ne \delta(j_r, |w'|+1)$.

As in the base case, we provide the details for the first set of cases; the others are similar and thus omitted.
(a) Since the configuration is not ϑ-*safe* in the (2^q-1)-horizon, by Lemma 5.3.9 $\mathbf{I}(\mathcal{G}_1((w,\mathbf{i}^n),(w',\mathbf{j}^n)))$.
(b) Let us suppose that $i_s \in \rho^+_{k,q+1}(i_r)$. We partition the rigid interval $\rho^+_{k,q+1}(i_r)$ into five parts and we follow a different strategy for each of them. Let c be the center of $\rho^+_{k,q+1}(i_r)$ and let α^z_{q+1}, where $z = \lceil \log k \rceil + 1$, be its radius. Without loss of generality, we assume that $i_r < i_s$ and $j_r < j_s$. From left to right, the five subintervals of $\rho^+_{k,q+1}(i_r)$ we are going to consider are the following:

1. $(c - \alpha^z_{q+1}, c - 2^{q-1}]$
2. $[c - 2^{q-1} + 1, c]$
3. $(c, c + 2^{q-1} - 1]$
4. $[c + 2^{q-1}, c + \alpha^z_{q+1})$
5. $c + \alpha^z_{q+1}$ (the right endpoint of $\rho^+_{k,q+1}(i_r)$)

Strategy 1: Let $\delta(i_r,i_s) = kp - s$, with $s \in [2^{q-1}, \alpha^z_{q+1})$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r,i_s) > \delta(j_r,j_s) \ge kp - (p+1)$ (if the last condition is not satisfied, *pstep-safety* is violated and the thesis immediately follows). **I** chooses $j_{n+1} = j_r - (2^{q-1} - 1)$. If **II** replies with $i_{n+1} \ne i_r - (2^{q-1} - 1)$, the configuration is not ϑ-*safe* in the 2^{q-1}-horizon, then $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. If **II** replies with $i_{n+1} = i_r - (2^{q-1} - 1)$, then $(kp >)\delta(i_{n+1}, i_s) = kp - s + (2^{q-1} - 1) \ge kp - \alpha^z_{q+1} + 1 + (2^{q-1} - 1) = kp - \alpha^z_q + 1$. From $\delta(i_{n+1}, i_s) > \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$.

Strategy 2: Let $\delta(i_r, i_s) = kp - s$, with $s \in [0, 2^{q-1} - 1]$, and let $\delta(i_r, i_s) > \delta(j_r, j_s) \geq kp - (p+1)$. **I** chooses $j_{n+1} = j_s - kp$. If **II** replies with $i_{n+1} = i_s - kp$, then $0 \leq \delta(i_{n+1}, i_r) \leq 2^{q-1} - 1$ and $\delta(j_{n+1}, j_r) > \delta(i_{n+1}, i_r)$. It immediately follows that the configuration is not *ϑ-safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with i_{n+1} such that $\delta(i_{n+1}, i_r) = \delta(j_{n+1}, j_r)$, then $pstep_{p \cdot 2^{q-1}}(i_{n+1}, i_s) \neq pstep_{p \cdot 2^{q-1}}(j_{n+1}, j_s)$. As a consequence, the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 3: Let $\delta(i_r, i_s) = kp + s$, with $s \in (0, 2^{q-1} - 1]$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $i_{n+1} = i_s - kp$. If **II** replies with $j_{n+1} = j_s - kp$, then $0 < \delta(i_r, i_{n+1}) \leq 2^{q-1} - 1$. From $\delta(j_r, j_{n+1}) > \delta(i_r, i_{n+1})$, it follows that the configuration is not *ϑ-safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with j_{n+1} such that $\delta(i_r, i_{n+1}) = \delta(j_r, j_{n+1})$, then $pstep_{p \cdot 2^{q-1}}(i_{n+1}, i_s) \neq pstep_{p \cdot 2^{q-1}}(j_{n+1}, j_s)$. As a consequence, the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 4: Let $\delta(i_r, i_s) = kp + s$, with $s \in [2^{q-1}, \alpha^z_{q+1})$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $i_{n+1} = i_r + 2^{q-1} - 1$. If **II** replies with $j_{n+1} \neq j_q + 2^{q-1} - 1$, the configuration is not *ϑ-safe* in the 2^{q-1}-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with $j_{n+1} = j_r + 2^{q-1} - 1$, then $(kp <)\delta(i_{n+1}, i_s) = kp + s - (2^{q-1} - 1) \leq kp + \alpha^z_{q+1} - (2^{q-1} - 1) = kp + \alpha^z_q$. From $\delta(i_{n+1}, i_s) < \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

Strategy 5: Let $\delta(i_r, i_s) = kp + \alpha^z_{q+1}$, where $z = \lceil \log_2 k \rceil + 1$, and let $\delta(i_r, i_s) < \delta(j_r, j_s) \leq (k+1)p$. **I** chooses $j_{n+1} = j_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, where $z' = \lceil \log_2 \lceil k/2 \rceil \rceil + 1$. By definition, $j_{n+1} \in \rho^+_{\lceil k/2 \rceil, q}(j_r)$. If **II** replies with $i_{n+1} \neq i_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** replies with $i_{n+1} = i_r + \lceil k/2 \rceil p + \alpha^{z'}_q$, it is possible to show that $i_{n+1} \in \rho^-_{\lfloor k/2 \rfloor, q}(i_s)$. From $\delta(i_{n+1}, i_s) \neq \delta(j_{n+1}, j_s)$, it follows that the configuration is not *p-int-safe* in the $(q-1)$-horizon and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

■

In Example 5.3.15, we provide an example of a configuration which is not *p-int-safe* in the 3-horizon and we show that **I** can win in 3 rounds (by applying at the first round Strategy 4).

Example 5.3.15 Consider the configuration in Figure 5.4. It is immediate to verify that it is not *p-int-safe* in the 3-horizon. We show a strategy **I** can adopt in order to win in 3 rounds.

(a) $q = 3$: $\delta(i_1, i_2) = 2p + \alpha^2_4 - 1 = 2p + 4$ and $\delta(j_1, j_2) = 2p + 5$. It follows that the configuration is not *p-int-safe* in the 3-horizon and thus $\mathbf{I}(\mathcal{G}_3((w, \mathbf{i}^2), (w', \mathbf{j}^2)))$. Notice that $pstep^{(p)}_{8p}(i_1, i_2) = pstep^{(p)}_{8p}(j_1, j_2) = 3$, so the configuration is *pstep-safe* in the $(p \cdot 2^3)$-horizon. Furthermore, notice that $\vartheta_7(i_1, i_2) = \vartheta_7(j_1, j_2) = \infty$, so the configuration is *ϑ-safe* in the $(2^3 - 1)$-horizon. An optimal move for **I** is to choose $i_3 = i_1 + 2^{q-1} - 1 = i_1 + 3$ in w. An optimal reply from **II** is to choose $j_3 = j_1 + 3$ in w'.

(b) $q = 2$: $\delta(i_3, i_2) = 2p + \alpha^2_3 - 1 = 2p + 1$ and $\delta(j_3, j_2) = 2p + 2$. Again, the configuration is not *p-int-safe* in the 2-horizon and thus $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^3), (w', \mathbf{j}^3)))$. If **I** chooses $i_4 = i_3 + 2^{q-1} - 1 = i_3 + 1$ in w, **II** can reply by choosing $j_4 = j_3 + 1$ in w'.

(c) $q = 1$: $pstep^{(p)}_{2p}(i_4, i_2) = 2$ and $pstep^{(p)}_{2p}(j_4, j_2) = \infty$. Once more, the configuration is not *pstep-safe* in the $(p \cdot 2^1)$-horizon and thus $\mathbf{I}(\mathcal{G}_1((w, \mathbf{i}^4), (w', \mathbf{j}^4)))$. If **I** chooses $i_5 = i_4 + p$ in w, it holds that $i_4 <_p i_5$ and $i_5 <_p i_2$. Hence, **II** is not able to find a position j_5 such that $j_4 < j_5$ and $j_5 <_p j_2$.

(d) $q = 0$: whatever **II**'s move at the previous round has been, **I** has won.

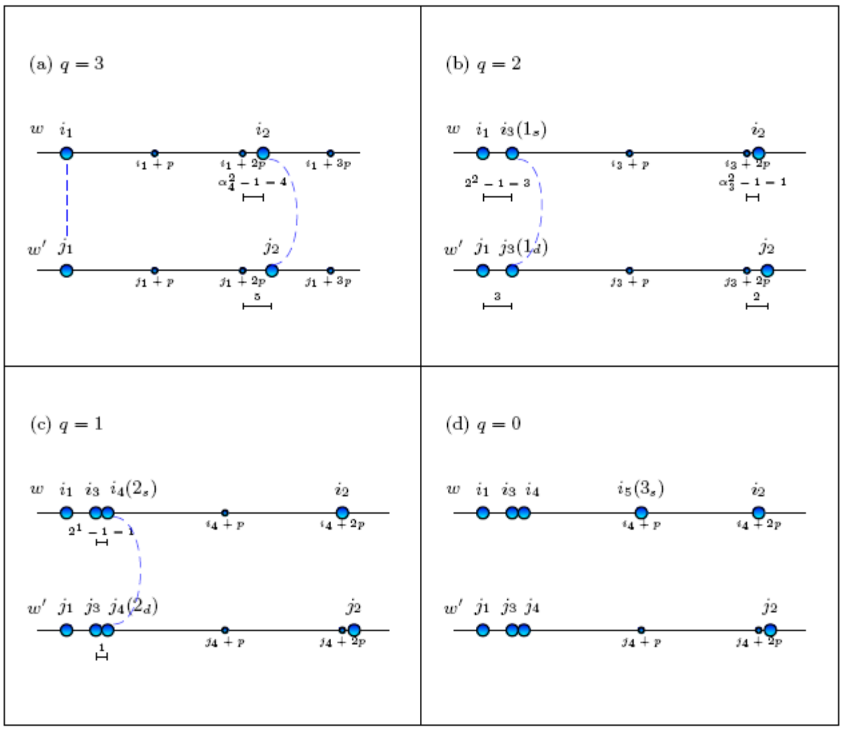


Figure 5.4: p-*int*-safety.

■

Definition 5.3.16 *A configuration* $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ *is* q-distance-safe *if it is pstep*-safe *in the* $(p \cdot 2^q)$-*horizon and, if* $q > 0$, *it is p-int*-safe *in the* q-*horizon.*

The next theorem takes advantage of previous lemmas to provide a necessary condition for **II** to win.

Theorem 5.3.17 *[Necessary condition for **II** to win]*
Let $w, w' \in \Sigma^\star$. *If* $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ *is not* q-distance-safe, *then* $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n))$.

Proof. If $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not *pstep-safe* in the $(p \cdot 2^q)$-horizon, then, by Lemma 5.3.6, $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$; if $q > 0$ and $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not *p-int-safe* in the q-horizon, then, by Lemma 5.3.14, $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$. ■

The next theorem states that *q-distance-safety* is also a sufficient condition for **II** to win (in the restricted setting we are considering, where only local moves are allowed).

Theorem 5.3.18 *[Sufficient condition for **II** to win]*
Let $w, w' \in \Sigma^\star$. *If* $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ *is q-distance-safe, then* $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

5.4 Local games on labeled $<_p$ structures

In this section, we consider the effects of adding unary predicates, that is, of associating a label with each string position, to the local games studied in the previous section. To this end, we introduce the notions of *q-color of a position* and of *q-color of an interval*, which gives a recursive characterization of the labels occurring in a suitable neighborhood of the position.

Definition 5.4.1 *Let $i, j, p \in \mathbb{N}$, with $i, j, p \geq 1$. We define $p\text{-}int^+_j(i) = [i + (j-1)p + 1, i + jp]$ and $p\text{-}int^-_j(i) = [i - jp, i - (j-1)p - 1]$.*

In the following definition, the q-color of positions is defined in terms of the $(q-1)$-color of intervals, which in its turn is defined in terms of the $(q-1)$-color of positions. Moreover, we distinguish between the q-color of an internal interval and the q-color of prefix and suffix intervals. Finally, to keep definitions as simple as possible, we assume all (fictitious) positions before position 1 and after position $|w|$ to be labeled by the special symbol \$.

Definition 5.4.2 *Let $w \in \Sigma^*$, $q, p \in \mathbb{N}$, with $p > 1$, and $i \in \mathbb{Z}$. The* q-color of position i in w, *denoted by $q\text{-col}_w(i)$, is inductively defined as follows:*

- *the 0-color of i in w is the label $w[i]$ for $i \in \{1 \dots |w|\}$ and \$ otherwise;*
- *the (q+1)-color of i in w is the ordered tuple $\sigma^w_{2^q} \cdots \sigma^w_1 w[i] \tau^w_1 \dots \tau^w_{2^q}$ where for all $1 \leq j \leq 2^q$, τ^w_j is the q-color of $p\text{-}int^+_j(i)$ and σ^w_j is the q-color of $p\text{-}int^-_j(i)$.*

The q-color *of the jth right (resp., left) interval $[a, b] = p\text{-}int^+_j(i)$ (resp., $p\text{-}int^-_j(i)$) induced by i, abbreviated* q-col-right-int$^j_w(a, b)$ *(resp.,* q-col-left-int$^j_w(a, b)$*), with $1 \leq j \leq 2^q$, is the ordered tuple $t^w_a \dots t^w_{a+\gamma_1-1}\{t^w_{a+\gamma_1} \dots t^w_{b-\gamma_2}\}t_{b-\gamma_2+1} \dots\ t^w_b$ (resp., $t^w_a \dots t^w_{a+\gamma_2-1}\{t^w_{a+\gamma_2}\ \dots t^w_{b-\gamma_1}\}t_{b-\gamma_1+1} \dots t^w_b$)), where for all $a \leq k \leq b$, t^w_k =q-col$_w(k)$ and*

$$\gamma_1 = \begin{cases} \alpha^z_{q+1} & \textit{if } q \neq 0 \textit{ and } j-1 \leq 2^{q-1}, \textit{ where } z = \lceil \log_2(j-1) \rceil + 1 \textit{ for } j > 1 \\ \quad \textit{and} & z = 0 \textit{ for } j = 1, \\ 0 & \textit{if } q = 0 \textit{ or } j - 1 > 2^{q-1}. \end{cases}$$

$$\gamma_2 = \begin{cases} \alpha^z_{q+1} - 1 & \textit{if } q \neq 0 \textit{ and } j \leq 2^{q-1}, \textit{ where } z = \lceil \log_2 j \rceil + 1 \\ 0 & \textit{if } q = 0 \textit{ or } j > 2^{q-1}. \end{cases}$$

For $q > 0$, the q-color *of the jth prefix (resp., suffix) interval $[a, b] = p\text{-}int^+_j(0)$ (resp., $p\text{-}int^-_j(|w|+1)$), abbreviated* q-col-pref$^j_w(a, b)$ *(resp.,* q-col-suff$^j_w(a, b)$*), with $1 \leq j \leq 2^q - 1$, $q > 0$, is the ordered tuple $t^w_a \dots t^w_{a+\gamma_1-1}\{t^w_{a+\gamma_1} \dots t_b\}$ (resp., $\{t^w_a \dots t^w_{b-\gamma_1}\}t^w_{b-\gamma_1+1} \dots t_b$), where $\forall a \leq i \leq b$, t^w_i =q-col$_w(i)$ and*

$$\gamma_1 = \begin{cases} \alpha^z_{q+1} & \textit{if } q \neq 0 \textit{ and } j-1 \leq 2^{q-1}, \textit{ where } z = \lceil \log_2(j-1) \rceil + 1 \textit{ for } j > 1 \\ \quad \textit{and} & z = 0 \textit{ for } j = 1, \\ 0 & \textit{if } j - 1 > 2^{q-1}. \end{cases}$$

The next definition introduces the notion of $<_p$*-safety for q-colors.*

Definition 5.4.3 *Let $w, w' \in \Sigma^*$ and $p, n, q \in \mathbb{N}$, with $p > 0$. A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* $<_p$-safe for q-colors *if the following conditions hold:*

- *$\forall r \in \{1, \dots, n\}$, q-col$_w(i_r)$ =q-col$_{w'}(j_r)$;*
- *$\forall 1 \leq j \leq 2^{q-1} - 1$, with $q > 1$, $(q-1)$-col-pref^j_w($p\text{-}int^+_j(0)$) = $(q-1)$-col-pref$^j_{w'}$($p\text{-}int^+_j(0)$);*
- *$\forall 1 \leq j \leq 2^{q-1} - 1$, with $q > 1$, $(q-1)$-col-suff^j_w($p\text{-}int^-_j(|w|+1)$) = $(q-1)$-col-suff$^j_{w'}$($p\text{-}int^-_j(|w'|+1)$).*

Example 5.4.4 In Figures 5.5 and 5.6 we show two configurations which are not $<_p$*-safe for 2-colors*. Moreover, for each of them, we outline a strategy **I** can adopt to win in 2 rounds.

As for Figure 5.5, a condition relative to the prefix of an interval is violated. We have that 2-*col*$^w_2(i_1) = \sigma^w_2 \sigma^w_1 w[i_1] \tau^w_1 \tau^w_2$ and 2-*col*$^{w'}_2(j_1) = \sigma^{w'}_2 \sigma^{w'}_1 w'[j_1] \tau^{w'}_1 \tau^{w'}_2$. If we consider only the portions of w and w' on the right of i_1 and j_1, we have that $\tau^w_1 = \dots..a\{a,b\}\{..b\{a,b\}\}$, $\tau^{w'}_1 = \dots..a\{a,b\}\{..b\{a,b\}\}$, $\tau^w_2 = \dots..a\{a,b\}\{\dots\}$, and $\tau^{w'}_2 = \dots..b\{a,b\}\{\dots\}$. Since 1-*col*$_w(i_1+p+1) \neq$ 1-*col*$_{w'}(j_1+p+1)$, it holds that 2-*col*$_w(i_1) \neq$ 2-*col*$_{w'}(j_1)$. Then $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^1), (w', \mathbf{j}^1)))$. An optimal move for **I** is to choose $i_2 = i_1 + p + 1$ in w. **II** must choose a position within $[j_1 + p + 1, \dots, j_1 + 2p]$ in w' labeled by a. Let us suppose **II** chooses $j_2 = j_1 + p + 2$. The new configuration is not *p-int-safe* in the 1-horizon. If **I** chooses $j_3 = j_1 + 1$, it holds that $j_3 \not<_p j_2$. Hence, **II** is not able to find a position i_3 in w such that $i_3 > i_1$ and $i_3 \not<_p i_2$.

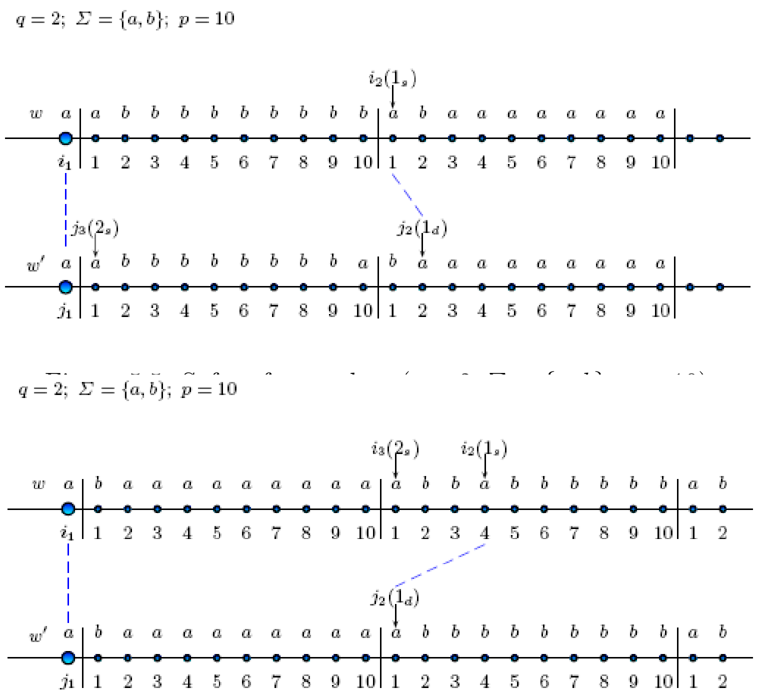


Figure 5.6: Safety for q-colors ($q = 2$, $\Sigma = \{a, b\}$, $p = 10$).

As for Figure 5.6, a condition relative to the interior of an interval is violated. We have that 2-$col_2^w(i_1) = \sigma_2^w \sigma_1^w w[i_1] \tau_1^w \tau_2^w$ and 2-$col_2^{w'}(j_1) = \sigma_2^{w'} \sigma_1^{w'} w'[j_1] \tau_1^{w'} \tau_2^{w'}$. If we consider only the portions of w and w' on the right of i_1 and j_1, we have that $\tau_1^w = \ldots..b\{a\}\{..a\{a,b\}\}$, $\tau_1^{w'} = \ldots..b\{a\}\{..a\{a,b\}\}$, $\tau_2^w = \ldots..a\{a,b\}\{..b\{a,b\}, ..a\{a,b\}\}$, and $\tau_2^{w'} = \ldots..a\{a,b\}\{..b\{a,b\}\}$. Since $\tau_2^w \neq \tau_2^{w'}$ (in particular, $\{b\{a,b\}, a\{a,b\}\} \neq \{b\{a,b\}\}$), it holds that 2-$col_w(i_1) \neq$ 2-$col_{w'}(j_1)$. Then $\mathbf{I}(\mathcal{G}_2((w, \mathbf{i}^1), (w', \mathbf{j}^1)))$. An optimal move for **I** is to choose $i_2 = i_1 + p + 4$, where $w[i_1 + p + 4] = a$. **II** must choose a position within $[j_1 + p + 1, \ldots, j_1 + 2p]$ labeled by a. He can only choose $j_2 = j_1 + p + 1$. The new configuration is not *p-int-safe* in the 1-horizon. If **I** chooses $i_3 = i_1 + p + 1$, it holds that $i_1 \not<_p i_3$. Hence, **II** is not able to find a position j_3 in w' such that $j_1 < j_3 < j_2$ and $j_1 \not<_p j_3$. ■

The following theorem provides a necessary condition for **II** to win a local game on labeled $<_p$ structures.

Theorem 5.4.5 *[Necessary condition for **II** to win]*
Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p > 1$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not $<_p$-safe for q-colors, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

The next theorem gives a sufficient condition for **II** to win a local game on labeled $<_p$ structures. It pairs the condition of *q-distance-safety* (Theorem 5.3.18) with that of *$<_p$-safety for q-colors.*

Definition 5.4.6 *A configuration $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is* q-locally-safe *if it is* q-distance-safe *in the $(p \cdot 2^q)$-horizon and* $<_p$-safe for q-colors.

Theorem 5.4.7 *[Sufficient condition for **II** to win]*
Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p > 1$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is q-locally-safe, then $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

To summarize, we have that *q-local safety* and *$<_p$-safety for q-colors* are necessary and sufficient conditions for **II** to have a winning strategy in q rounds when **I**'s moves are coerced to be local.

We conclude the section by showing that it is possible to check in polynomial time whether the q-colors of two positions in the two strings are equal or not. We take advantage of a suitable

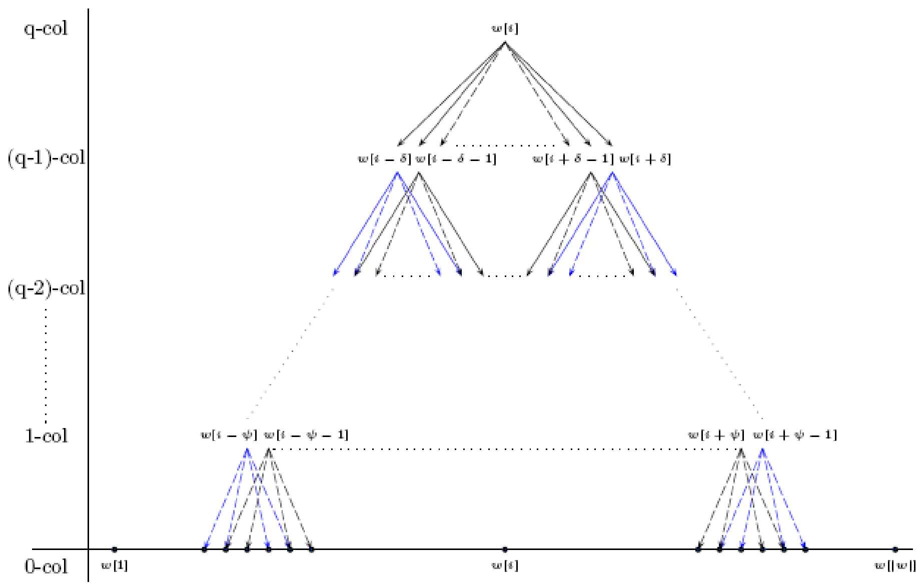

q-col
(q-1)-col
(q-2)-col
1-col
0-col
$w[i]$
$w[i-\delta]$ $w[i-\delta-1]$
$w[i+\delta-1]$ $w[i+\delta]$
$w[i-\psi]$ $w[i-\psi-1]$
$w[i+\psi]$ $w[i+\psi-1]$
$w[1]$
$w[i]$
$w[|w|]$

Figure 5.7: The proposed representation of q-col$_w(i)$.

representation of q-colors, which is graphically depicted in Figure 5.7. The x-axis refers to string positions, while the y-axis refers to q-colors. The q-color of a position i is represented by the label $w[i]$ plus a set of pointers (arrows) to the relevant $(q-1)$-colors, that is, to the $(q-1)$-colors of positions $i-\delta, \ldots, i-1, i+1, \ldots, i+\delta$, with $\delta = p \cdot 2^{q-1}$. Labeled positions and pointers give rise to a directed layered graph. The number of nodes and edges of such a graph can be computed as follows. Let $|w| = n$. For $0 \leq k < q$, the number of nodes of the graph at level k equals the number of k-colors to be computed, that is, $2 \cdot p \cdot \sum_{l=k}^{q-1} 2^l = O(n)$. Thus, the total number of nodes is $O(n \cdot q)$. As far as the number of edges is concerned, for $1 \leq k \leq q$, at level k the number of outgoing edges is $(2 \cdot p \sum_{l=k}^{q-1} 2^l) \cdot (2 \cdot p \cdot 2^{k-1}) = O(n^2)$. Hence, the total number of edges is $O(n^2 \cdot q)$. We partition the set of edges in two classes: *continuous* edges, that point to colors that must occur at the same position within the corresponding p-intervals (of the two strings) and *dashed* edges, that point to colors whose position within the corresponding p-intervals (of the two strings) is irrelevant. To compare the q-colors of two given positions (in the two strings), we build the corresponding graphs and we visit them in a bottom-up fashion, that is, we start from nodes representing 0-colors and we move upwards level-by-level. For both graphs, we compute, at each level k, with $0 \leq k < q$, a table that, for every k-color, keeps track of the its occurrences in the other graph. More precisely, for each position $1 \leq i \leq 2 \cdot p \cdot \sum_{l=k}^{q-1} 2^l$, we compute all the occurrences of *k-col(i)* in the other graph. As a matter of fact, it suffices to search for the occurrences of *k-col(i)* in the interval $[i-(q-k)\cdot(p-1), \ldots, i+(q-k)\cdot(p-1)]$. Moreover, to build the table at level k, we only need to look at the table at level $k-1$. Hence, once the former has been computed, the latter can be deleted. The table at level 0 can be easily computed by looking for the occurrences of each 0-color in the other string. To build the table at level k, with $0 < k < q$, we need to compare k-colors in the two graphs. More precisely, for every position we must check the presence of every $(k-1)$-color it points to in the corresponding p-interval of the other string. In fact, if a $(k-1)$-color is reached by a continuous arrow, it suffices to check whether it occurs exactly at the same position within the corresponding p-interval (in the other string). This can be done by accessing the table at level $k-1$. Once the first "k-comparison" has been done, the following ones can exploit the outcomes of the "$(k-1)$-searches" already performed, because the computation of the k-colors of two neighbor positions requires the computation of the $(k-1)$-colors of two portions of the string which only differ for a 1-position shift.

The resulting algorithm has the following complexity. To build the table at level k, we compare each of the $O(n)$ k-colors of a string with each of the $O(n)$ k-colors in the other string. The comparison of two k-colors takes time $O(p^2 2^k)(= O(p^2 n))$, because it requires to take into consideration

$p2^k$ positions, that is, entries of the table at level k-1, and to scan (part of) their occurrence list of length $O(p)$. Thus, the table at level k can be built in time $O(p^2n^3)$. Since we need to build q-1 tables, the overall time complexity is $O(p^2n^3q)$. Given that q is logarithmic in n (it is not difficult to show that if **I** has a winning strategy, then he has a winning strategy in $O(\log n)$ rounds), the complexity of the algorithm is $O(p^2n^3 \log n)$, which is polynomial in the length of the strings.

5.5 Global games on labeled $<_p$ structures

In this section, we consider the general case, where we do not constrain **I**'s moves to be local. The solution consists in combining a local and a global strategy. The global strategy essentially requires to compare the *multiplicity* and *scattering* of r-colors τ in the portions of w and w' that are far away from already selected positions. In particular, to guarantee that **II** may find a suitable reply for any possible choice of **I**, not only w and w' must have the same r-colors, but there must be enough occurrences (otherwise both strings must have the same number of occurrences), distributed in a similar way in both strings.

Definition 5.5.1 *Let $q, p \in \mathbb{N}^+$, $\boldsymbol{i}^n$ be a set of positions in w and τ be a $(q-1)$-color. The $(q$,$p)$-* free-multiplicity *of τ in w, abbreviated $\rho^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau)$, is the number of occurrences of τ in w which fall in $Free^p_q(w, \boldsymbol{i}^n)$, where $Free^p_q(w, \boldsymbol{i}^n) = \{1, \ldots, |w|\} \setminus Pstepreg^p_q(w, \boldsymbol{i}^n)$.*

Definition 5.5.2 *Let $P \subseteq \mathbb{N}$ be a finite set. A* k-blurred partition *$\mathcal{P}$ of P is a partition of P such that (i) for each $A \in \mathcal{P}$ and for each $a, b \in A$, $\delta(a, b) \leq k$, and (ii) there is not a partition $\mathcal{P}'$ satisfying (i) such that $|\mathcal{P}| > |\mathcal{P}'|$. The number of classes of $\mathcal{P}$ is called k-*blurring.

Definition 5.5.3 *Let $q, p \in \mathbb{N}^+$, $\boldsymbol{i}^n$ be a set of positions in w and τ be a $(q-1)$-color. The $(q$,$p)$-* free-scattering *of τ in w, abbreviated $\sigma^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau)$, is the $(p2^q)$-blurring of $\{i | \ (q-1)\text{-}color_w(i) = \tau \wedge i \in Free^p_q(w, \boldsymbol{i}^n)\}$, where $Free^p_q(w, \boldsymbol{i}^n) = \{1, \ldots, |w|\} \setminus Pstepreg^p_q(w, \boldsymbol{i}^n)$.*

Lemma 5.5.4 *Let $k, q, p \in \mathbb{N}$, with $q, p > 0$, $w \in \Sigma^*$, $\boldsymbol{i}^n$ be a set of positions in w and τ be a $(q-1)$-color in w. $\sigma^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau) \leq k$ if and only if $\exists i_{n+1}, \ldots, i_{n+k}$ such that all the occurrences of τ fall inside $Pstepreg^p_q(w, \boldsymbol{i}^{n+k}) = Ent^p_q(w, \boldsymbol{i}^{n+k})$.*

Let $\Delta^{(w,\mathbf{i}^n)}_{(w',\mathbf{j}^n)} = \{\tau \mid \tau \text{ is a (q-1)-color}, q > 0, \text{ and } \sigma^{(w,\mathbf{i}^n)}_{(q,p)}(\tau) \neq \sigma^{(w',\mathbf{j}^n)}_{(q,p)}(\tau) \vee \rho^{(w,\mathbf{i}^n)}_{(q,p)}(\tau) \neq \rho^{(w',\mathbf{j}^n)}_{(q,p)}(\tau)\}$ be the set of words that **I** can exploit to win. The next theorem provides a necessary condition for **II** to win.

Theorem 5.5.5 *Let $q, p, n \in \mathbb{N}$, with $q, p > 0$, and let $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ be a configuration. If there exists a $q-1$-color τ such that $\tau \in \Delta^{(w,\boldsymbol{i}^n)}_{(w',\boldsymbol{j}^n)}$, then*
$\boldsymbol{I}(\mathcal{G}_{q+min\{\sigma^{(w,\boldsymbol{i}^n)}_{(q,p)}(\tau),\sigma^{(w',\boldsymbol{j}^n)}_{(q,p)}(\tau)\}}((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

A (necessary and) sufficient condition for **II** to win is given by the following theorem.

Theorem 5.5.6 *[Main Theorem]*
Let $w, w' \in \Sigma^*$ and $p, q \in \mathbb{N}$, with $p > 1$. $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$ if and only if the following conditions hold:

1. *$(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is q-locally-safe;*

2. *for all $(r-1)$-color $\tau \in \Delta^{(w,\boldsymbol{i}^n)}_{(w',\boldsymbol{j}^n)}$, with $1 \leq r \leq q$, $\sigma^{(w,\boldsymbol{i}^n)}_{(i,p)}(\tau) > q - r$ and $\sigma^{(w',\boldsymbol{j}^n)}_{(i,p)}(\tau) > q - r$.*

Condition 2. means that **I** cannot detect any difference in q rounds if the *scattering* of the $(r-1)$-colors that have different multiplicity or scattering in $(w, \mathbf{i}^n)$ and $(w', \mathbf{j}^n)$ is sufficiently high in both structures, where the thresholds depend on the size of the $(r-1)$-colors (higher thresholds for smaller $(r-1)$-colors). The remoteness of $\mathcal{G}$ is thus $r + \min(\sigma^{(w,\mathbf{i}^n)}_{(r,p)}, \sigma^{(w',\mathbf{j}^n)}_{(r,p)})$ and it can be intuitively explained as follows: given a suitable $\tau \in \Delta^{(w,\mathbf{i}^n)}_{(w',\mathbf{j}^n)}$, **I** can first choose $\min(\sigma^{(w,\mathbf{i}^n)}_{(r,p)}, \sigma^{(w',\mathbf{j}^n)}_{(r,p)})$ occurrences of τ and then play a local move; after that, it is guaranteed that the reached position is not $(r-1)$-locally safe, so **I** can win the game by playing $r-1$ other local moves.

To compute the remoteness of two strings w and w', we can thus proceed as follows. Let $n = min\{|w|, |w'|\}$. As we already pointed out, this gives an upper bound $m = \log n$ to the value of remoteness (unless the two strings coincide). Hence, for $i = 1, \ldots, m$, we search for an $(i-1)$-color with a different scattering or multiplicity in w and w' (if any). As a preliminary step, we construct a layered graph that represents the $(m-1)$-color of all the positions of the string w. Then, we repeat the same construction for the string w'. Both the resulting graphs consists of $O(n \log n)$ nodes and $O(n^2 \log n)$ edges. Next, for $i = 1, \ldots, |w|$ and $k = 0, \ldots, m-1$, we build a table that keeps track of the occurrences of the k-color of position i in w and w' (in fact, we must also deal with the k-colors of the fictitious positions preceding position 1 and following position $|w|$; however, the treatment of such cases does not affect the complexity of the building procedure). Then, we repeat the same construction for w'. Both constructions take time $O(p^2 n^3 \log n)$. The resulting tables can then be exploited to compute multiplicity and scattering of each k-color in time $O(n)$. This last step globally takes time $O(n^2 \log n)$. The overall complexity of the computation is thus $O(p^2 n^3 \log n)$.

5.6 Conclusions and future work

In this chapter, we analyzed *EF-games* on labeled $<_p$ structures. The $<_p$ relation features a mix of the characteristics of the successor relation s and the linear order relation $<$. From s, it borrows the condition of *pstep-safety* (such a condition is trivially satisfied in the case of $<$). Moreover, it shares the condition of *θ-safety* with $<$ (in the case of s, such a condition immediately follows from *pstep-safety*). In addition, it features some distinctive characteristics, such as the partition of the neighborhoods of selected positions in rigid and elastic intervals.

The work identified necessary and sufficient winning conditions for **I** and **II**, that allow one to compute the remoteness of a game and optimal strategies for both players. Moreover, it provides a polynomial algorithm for the crucial step in the computation of the remoteness, namely, checking whether the q-colors of two distinct positions are equal.

Biological strings can be modeled by labeled $<_p$ structures and the remoteness of a game played on such structures can be viewed as a measure of their degree of similarity, thus proving a natural approach for spotting differences. We are currently comparing the values of the remoteness of real biological sequences (in particular, DNA sequences of the plasmodium parasites), as well as of artificial sequences, for different values of the parameter p. We are also studying the case in which, instead of fixing the value of p in advance, we make p a function of the size of the input sequences, e.g., a logarithmic function.

6 Models for cell cycle, protein p53, and Irinotecan

Irinotecan is an anti-cancerogenic medicament which started to be used in clinics approximately twenty years ago. Scientists are currently trying to optimize its therapy in order to understand how to limit its toxicity and to increase its effectiveness. In this context, it is crucial to comprehend how the presence of this medicament influences cellular proliferation. In the next chapter we will present our contribution to the problem, which consists in modeling the mammalian cell cycle and the behaviour of irinotecan by means of the rule-based language of the biochemical abstract machine Biocham [46, 19], assembling such models into a consensus model, formalizing some biological properties of the resulting model through temporal logics, and automatically checking their satisfaction thanks to the use of model-checking.

In this chapter we will provide the needed biological background. First of all, we will focus on the most commonly used representation systems for biological networks, with particular attention to the diagrams utilized by Novák and Tyson to model mammalian cell division. Next, we will concentrate on the mammalian cell cycle, describing the four phases in which it is usually subdivided and the biochemical reactions characterizing it. Then we will introduce the model of the cell cycle proposed by Novák and Tyson in 2004 [102], to which we will refer for our work. We will proceed by describing the behaviour of protein p53, a tumor-suppressor protein that is normally present in cell nucleuses in small quantities and whose concentration increases when DNA damage occurs. Afterwards we will outline the metabolic pathways of irinotecan, which acts by damaging the cells where it is injected. Finally, we will introduce the set of phenomenological kinetic differential equations proposed in 2007 by Dimitrio to model the joint behaviour of protein p53 and irinotecan [38], that we will exploit in our coupling work. We will conclude the chapter by providing some simulations of such a model.

6.1 Representation systems for biological networks

A cornerstone aspect of present biology is the huge quantity of information generated from the new technologies for data acquisition. In order to rigorously analyze biological processes and, in particular, to simulate them at the computer, it is fundamental to represent such data in a suitable way. In the last years there has been an outbreak of languages/diagrams to represent *biological pathways*, that is, relations between biological elements.

The most popular machine-readable format for representing biochemical quantitative models is the *Systems Biology Markup Language* (*SBML*) ([69]), which has been evolving since mid-2000 with the purpose to provide a common intermediate format, i.e., a lingua franca, able to capture the most essential aspects of models. By supporting SBML as a format for reading and writing models, different software tools (including programs for building and editing models, simulation programs, databases, and other systems) can directly communicate and store the same computable representation of those models. From a practical point of view, SBML can encode models consisting of molecular species linked by reactions to form biochemical networks. Fundamental constituents of each model are reactant species, product species, reactions, reaction rates, and parameters in the rate expressions. Each reaction gives rise to (one or more) kinetic differential equations. To obtain a simulation of the model, it suffices to solve the resulting system of Ordinary Differential Equations (ODE) by means of numerical integration methods [83].

As far as graphical notations to represent biological pathways are concerned, in literature the scenery is vast. A typology of diagrams commonly encountered in molecular biology is given by *qualitative Nets* (*qNET*s), that is, graphs whose nodes represent biological entities (genes, ions, molecules, protein complexes, etc.) and whose edges represent directed binary interactions [1]. These graphs only distinguish two kinds of interactions: positive ones, such as activation and induction, and negative ones, such as inhibition. The limited expressivity of qualitative nets is overcome in *Metabolic Nets* (*MBN*s), where each edge is labeled by the enzyme catalyzing the corresponding reaction. However, these nets can only be used to represent metabolic reactions. Another example of diagrams with a limited modeling potential are *Genic Regulation Nets* (*GRN*s), which can only capture regulation relations among genes and among transcription factors.

One of the first attempts to develop a graphic notation able to represent each kind of molecular interaction are Kohn's *Molecular Interaction Maps* (MIMs) ([75]). MIMs are entity-relationship diagrams that allow one to represent molecular modifications, enzymatic activities, complex formations, and much more. They make it possible to model both qualitative aspects of biological systems and mechanistic interaction details. In fact, they comprise many pieces of information but they are often hardly readable, i.e., in the case of big maps.

The necessity to represent the temporal order of reactions led to the introduction of *Process Diagrams* (*PD*s), that is, state transition diagrams where each state of a biological element is represented by a different node ([73]). Edges connecting different states of the same element exist. Since these graphs explicit all kinds of reactions, they can be easily analyzed and simulated at the computer. However, an explosion of the number of states is unavoidable. To solve such problem, the *Edinburgh Pathway Notation* (EPN) has been conceived: it avoids graph over-crowding by representing within an only object all the states of a given biological element [97].

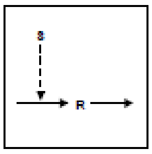
Finally, the desire to standardize the graphical/visual representation of essential biochemical and cellular processes studied in systems biology led to the recent introduction of the *Systems Biology Graphical Notation* (*SBGN*), which defines a comprehensive set of symbols with precise semantics, together with detailed syntactic rules defining their use. SBNG is made of three languages: process diagrams, entity-relationship diagrams, and activity flow diagrams. Currently, only process diagrams have been defined ([85]). It is worth mentioning that the conversion from SBGM diagrams to SBML programs and viceversa is meant to be completely automatical.

The next subsection is devoted to the description of the diagrams used by Novák and Tyson to represent molecular networks. These graphs, which can be catalogued as process diagrams, have been chosen by the two authors to model the mammalian cell cycle (cf. Subsection 6.2.3).

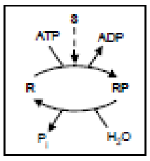
6.1.1 The process diagrams of Novák and Tyson

A detailed and exhaustive description of the notation introduced by Novák and Tyson for the representation of molecular networks is present in [123]. In this review paper, the authors show how simple signaling pathways can be embedded in networks to generate more complex behaviours, which are the basic building blocks of the dynamic behaviour shown by non-linear control systems. They present a precise vocabulary for describing these phenomena and some memorable examples of each of them. In Figures 6.1 and 6.2 we report some simple pathway examples: in each tableau, the left column is devoted to diagrams while the right one to the corresponding kinetic differential equations. As far as Figure 6.1 is concerned, we have (a) linear response, (b) hyperbolic response, (c) sigmoidal response, (d) perfect adaptation, (e) mutual activation, (f) mutual inhibition, and (g) homeostasis. As far as Figure 6.2 is concerned, we have (a) negative feedback, (b) activator-inhibitor, and (c) substrate-depletion oscillators. For further details, such us rate curves and signal-response curves, the reader can refer to [123].

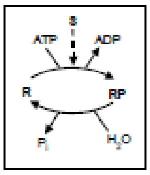
(a)



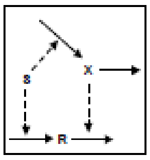
(b)



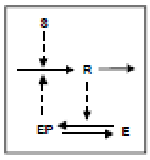
(c)



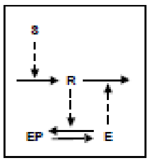
(d)



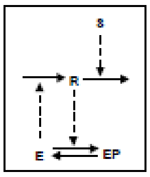
(e)



(f)



(g)



a) $\frac{dR}{dt} = k_0 + k_1 S - k_2 R$;

b) $\frac{dR_P}{dt} = k_1 S(R_T - R_P) - k_2 R_P$;

c) $\frac{dR_P}{dt} = \frac{k_1 S(R_T - R_P)}{k_{m1} + R_T - R_P} - \frac{k_2 R_P}{k_{m2} + R_P}$;

d) $\frac{dR}{dt} = k_1 S - k_2 X \cdot R$; $\frac{dX}{dt} = k_3 S - k_4 X$;

e) $\frac{dR}{dt} = k_0 E_P(R) + k_1 S - k_2 X \cdot R$;

f) $\frac{dR}{dt} = k_0 + k_1 S - k_2 R - k_2' E(R) \cdot R$;

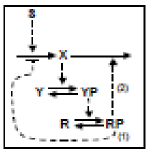
g) $\frac{dR}{dt} = k_0 E(R) - k_2 S \cdot R$.

Both in the diagrams and in the equations, the following abbreviations are used:
S: signal strength (e.g., concentration of mRNA);
R: response magnitude (e.g., concentration of protein);
RP: phosphorylated form of R;
$R_P = [RP]$;
$R_T = R + R_P$;
X: indirect signaling pathway;
P_i: inorganic phosphate;
E: a protein involved with R in mutual activation or inhibition;
EP: phosphorylated form of E.

In the above equations, $E_P(R) = G(K_3 R, K_4, J_3, J_4)$ and $E(R) = G(K_3, K_4 R, J_3, J_4)$, where the Goldbeter- Koshland function G is defined as follows: $G(u, v, J, K) = \frac{2uK}{v - u + vJ + uK + \sqrt{(v - u + vJ + uK)^2 - 4(v - u)uK}}$.

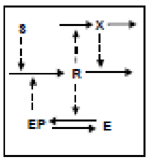
Figure 6.1: Signal-response elements

(a)



a) $\frac{dX}{dt} = k_0 + k_1 S - k_2 X + k_2' R_P \cdot X;$
$\frac{dY_P}{dt} = \frac{k_3 X (Y_T - Y_P)}{K_{m3} + Y_T - Y_P} - \frac{k_4 Y_P}{K_{m4} + Y_P};$
$\frac{dR_P}{dt} = \frac{k_5 Y_P (R_T - R_P)}{K_{m5} + R_T - R_P} - \frac{k_6 R_P}{K_{m6} + R_P};$

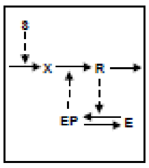
(b)



b) $\frac{dR}{dt} = k_0 E_P(R) + k_1 S - k_2 R - k_2' X \cdot R;$
$\frac{dX}{dt} = k_5 R - k_6 X;$

c) $\frac{dX}{dt} = k_1 S - [k_0' + k_0 E_P(R)] \cdot X;$
$\frac{dR}{dt} = [k_0' + k_0 E_P(R)] \cdot X - k_2 R.$

(c)



Both in the diagrams and in the equations, the same abbreviations of Figure 6.1 are taken into account. The only new component is Y, which in (a) introduces a time delay in the feedback loop.

Figure 6.2: Oscillatory networks

6.2 Mammalian cell-division cycle

Cells reproduce by duplicating their contents and then dividing in two. This cell-division cycle is the fundamental means by which all living things are propagated. The details of the cell cycle may vary from species to species, but certain requirements are universal. First and foremost, to produce a pair of genetically identical daughter cells, the DNA must be faithfully replicated, and the replicated chromosomes must be segregated into two separate cells, see Figure 6.3 on the left.

Our understanding of the cell cycle has undergone a revolution in recent years. In the past the cell cycle was monitored by observing the events of chromosome segregation with a light microscope and by following DNA replication by measuring the incorporation of radioactive precursors into DNA. The focus of attention, therefore, was on the chromosomes, and there seemed to be large differences between the cell cycles of different organisms and different types of cells. Recent experiments have provided a new and simpler perspective, revealing a cell-cycle control system that coordinates the cycle as a whole. The proteins of this control system first appeared over a billion years ago and have been so well conserved in evolution that many of them function perfectly when transferred from a human cell to a yeast cell. Biologists can therefore study the control system in a variety of eucaryotic organisms and use the findings from all of them to assemble a unified picture of how cells grow and divide.

The duration of the cell cycle varies greatly from one cell type to another. In the following we describe the sequence of events in a mammalian cell with a cycle time of about 24 hours. The cycle is traditionally divided into four distinct phases [2]:

- **G1-phase**. It is the temporal gap between the completion of mitosis and the beginning of DNA synthesis. During this phase the cell monitors its environment and its own size and, when the time is ripe, it takes a decisive step that commits it to DNA replication and completion of a division cycle.
- **S-phase (synthesis)**. It is the period of DNA replication. During synthesis, from one double-stranded DNA molecule (chromosome), two identical double stranded DNA molecules (called sister chromatids) are formed and held together by cohesin proteins.
- **G2-phase**. It is the temporal gap between the end of DNA synthesis and the beginning of mitosis. This phase allows the cell to ensure that DNA replication is complete before it

plunges into mitosis.

- **M-phase (mitosis)**. It is the most dramatic phase, when replicated DNA molecules are segregated to daughter cells. The sister chromatids separate so that the daughter cells get one copy of each chromosome.

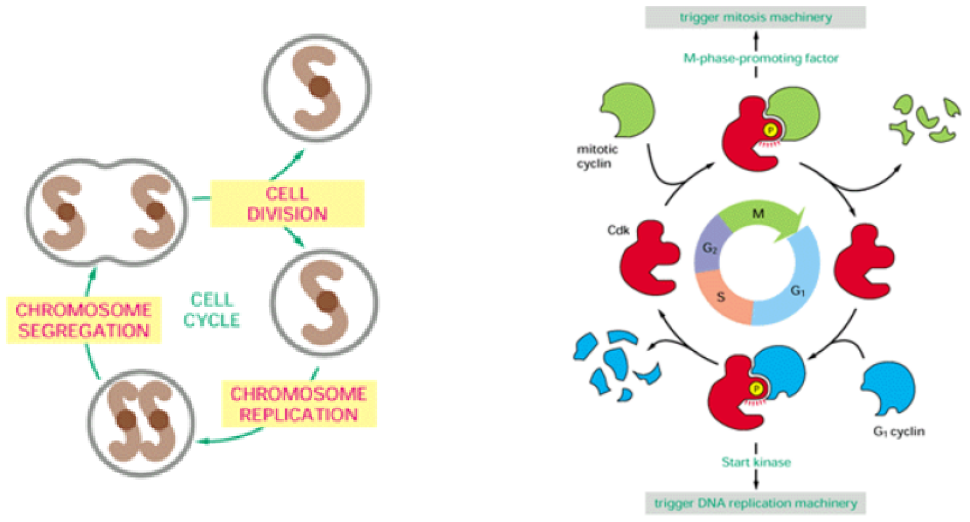


Figure 6.3: The cell-division cycle.

For a graphical overview of the four traditional subdivisions of the cell cycle, see Figure 6.3 on the right.

The proper alternation between synthesis and mitosis is coordinated by a complicated network that regulates the activity of a family of key proteins. These proteins are composed of two subunits: a catalytic subunit, the cyclin-dependent kinase, *cdk* for short, and a regulatory subunit, a *cyclin*. Cdk has to associate with a cyclin partner to form a dimer and has to be appropriately phosphorylated in order to be active. The Cdk/cyclin dimers have a central role in the cell cycle. On top of ensuring the alternation between synthesis and mitosis, the dimers also control cell size and DNA replication. The progression through cell cycle is orchestrated by the rise and fall of the Cdk/cyclin dimers. There are different ways to regulate the activity of the dimers: through synthesis of the cyclins by transcription factors, through degradation of the cyclins, through association with stoichiometric inhibitors, and through phosphorylation and dephosphorylation of cdk.

6.2.1 Cell cycle regulation.

As just remarked, the cell cycle is primarily catalyzed by complexes containing cdks and cyclins [114]. Cdks are serine/threonine protein kinases that are activated at specific points in the cell cycle. There are at least seven cdks in mammalian cells. The cdks are critical for progression through the cell cycle because their inactivation prevents mitosis. Much of the cell cycle regulation is determined by phosphorylation state. Cdc2, also known as cdk1, is the prototype cdk. Cdc2 was originally identified in fission yeast, with the cdc designation used to name cell division cycle genes and proteins. Cdk1 is now the preferred name.

Cdks are regulated by several methods, one of which is phosphorylation on threonine and tyrosine residues. Cdk1 has phosphorylation sites that are both inhibitory and stimulatory. Phosphorylation of cdk1 on threonine 161 by cdk-activating kinase is necessary for cdk1 kinase activity. Phosphorylation of cdk1 at tyrosine 15 (Y15) and threonine 14 (T14), which are located within the active site of the kinase, prevents kinase activity.

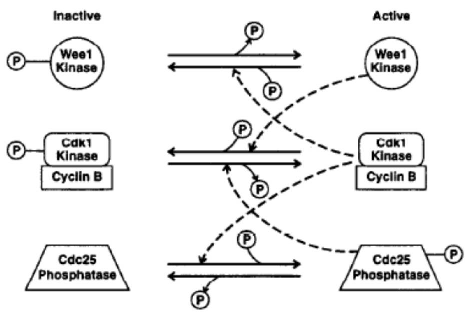


Figure 6.4: Wee1 kinase and cdc25 phosphatase interactions with cdk1-cyclin B complexes.

This phosphorylation is performed by the wee1, mik1, and mytl protein kinases. The wee1 kinase phosphorylates cdk1 upon entry of the cdk1-cyclin complex into the nucleus, thereby protecting the nucleus from premature exposure to an active mitotic kinase. When the cell is ready to divide, cdc25C dephosphorylates cdk1 at Y15 and T14, thereby activating the cdk1 kinase. Wee1 and cdc25C are in turn regulated by phosphorylation, thereby forming an activation cascade. Cdk1 is one of the kinases that may phosphorylate wee1 and cdc25C, forming positive and negative feedback loops (see Figure 6.4). Phosphorylation and dephosphorylation of the active site probably play similarly important roles in cdks other than cdk1; cdc25A and cdc25B, relatives of cdc25C, are active in the G1 and S phases of the cycle, respectively.

Cyclins perform multiple regulatory functions. In addition to the appropriate phosphorylation state of cdks, binding to a cyclin is necessary for the activation of cdks. Cyclins target the cdks to the nucleus; they contain nuclear localization signals that are lacking in cdks. Some cdks bind more than one cyclin. The cyclin bound by a given cdk probably provides some substrate specificity. The cyclins were so named because of their cyclic expression during the cell cycle (see Figure 6.5). Because of this cyclic expression, cdks can be activated only at specific times during the cell cycle. Part of the cyclic expression is due to regulated degradation. Cyclins contain PEST sequences, protein motifs rich in proline (P), glutamate (E), serine (S), and threonine (T), which target them for degradation by ubiquitination at specific times. Thus, a cycling cell enters and exits cell cycle phases in association with the synthesis and degradation of specific cyclins (Figure 6.5). In general, before a cell can enter the next cell cycle phase, the appropriate cyclin of the previous phase is degraded, and the cyclin of the next phase is synthesized.

Cdks phosphorylate a variety of substrates, thereby catalyzing the process of cell division. In phases G2 and M, cdk substrates include nuclear lamins and microtubules that form the nuclear cytoskeleton. Cdk1 can phosphorylate its own regulators wee1 and cdc25C in vitro and probably in vivo, although this ability is unproven (see Figure 6.4).

In G1, an important target of the cdks is the retinoblastoma protein (Rb), and the G1 cyclin-cdk complexes phosphorylate Rb on multiple residues. Hypophosphorylated Rb binds the E2F transcription factor, making it unavailable for transcription. Once the cyclin-cdk phosphorylates Rb, Rb releases E2F, freeing it to participate in transcription of proteins necessary for the progression of the cell cycle. A schematic representation of the activation of E2F by cdk-cyclin complexes is present in Figure 6.6.

Rb remains hyperphosphorylated through the remainder of the cell cycle. Cyclin A- and cyclin B-dependent kinases (cdk2 and cdk1) probably maintain Rb in its hyperphosphorylated form because Rb does not revert to the hypophosphorylated form until the end of mitosis.

Negative regulators of the cell cycle differ among stages. Inhibition of the cell cycle in G2 is thought to occur largely by phosphorylation of cdk1 by wee1. Cdc25C counteracts wee1 by dephosphorylating Y15 and T14 of cdk1, and failure of activation of cdc25C may also result in failure of activation (dephosphorylation) of cdk1 (see Figure 6.4). In the G1 and S phases of the cell cycle, the most extensively investigated inhibitors, and what appear to be the most important

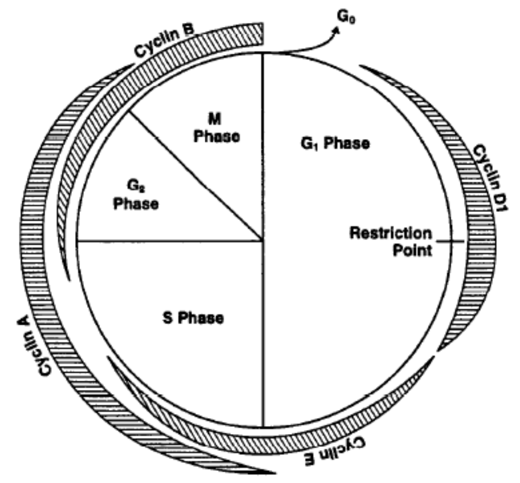


Figure 6.5: Schematic drawing of cell cycle-dependent levels of cyclins.

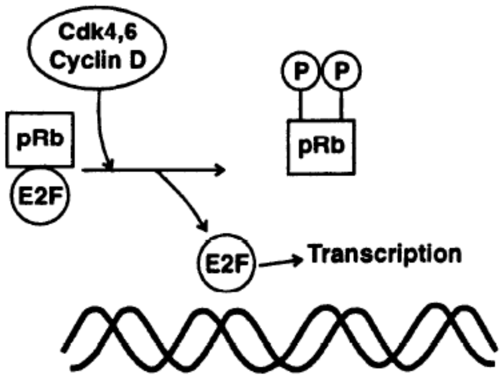


Figure 6.6: Schematic drawing of activation of EF2 by cdk-cyclin complexes.

inhibitors, are the p16 and p21 families. The p16 family includes p16, p15, p18, and p19, which inactivate only the G1 cdks, cdk4 and cdk6. These proteins form stable complexes with the cdks before they are bound to cyclins, and excess expression of cyclin will not dissociate them from the cdk. By binding cdk4 and cdk6, the p16 family members prevent the phosphorylation and inactivation of Rb.

The p21 family includes waf1 (also known as p21, cip1, and pic1), p27 (kip1), and p57 (kip2). The p21 family proteins are of primary importance in G1 regulation. These proteins can inhibit most cdks, possibly excepting cdk1. The p21 family proteins bind cyclins, thereby preventing the cdk from phosphorylating Rb and inducing its dissociation from E2F. In contrast to p16, cell cycle inhibition by p21 can be prevented by increasing cellular concentrations of cyclins. p21, but not p27 or p57, also binds and inhibits proliferating cell nuclear antigen (PCNA), a subunit of DNA polymerase δ, which has roles in DNA replication and repair. The p53 protein transcriptionally induces p21 expression. This is one method by which p53 induces cell cycle arrest.

6.2.2 The restriction point and checkpoints

Grow factors primarily act on cells in G1 and when they are not actively cycling (this phase is usually referred as G0). The point in G1 after which cells no longer respond to withdrawal of growth factors has been termed the *restriction point.* Growth factors stimulate the entry of cells into the cell cycle from G0. Early in G1, the removal of growth factors will result in cells returning to G0. However, in the later parts of G1 when the cells have passed the restriction point, despite the removal of the growth factor, cells will continue to progress into S phase. Therefore, the restriction point is the time after which the cell is committed to entering the cell cycle. The restriction point is thought to be largely regulated by Rb. The complete pathway from growth factors to Rb is not known.

Cell cycle *checkpoints* have been defined as "biochemical pathways that ensure dependence of one process upon another process that is biochemically unrelated"[43]. As an example, the completion of cellular DNA replication is a checkpoint that must be passed before the biochemically unrelated event of chromosome separation in mitosis can begin. Although the term checkpoint, and the above example, suggest a discrete point or time in the cell cycle, checkpoints are often more nebulous.

Cell cycle arrest, also referred to as delay, is produced by a variety of factors that may be intrinsic or extrinsic and may affect several different checkpoints. An example of an intrinsic factor is cell size. An example of an extrinsic factor is cell nutrition, which determines whether and the rate at which a cell progresses through the cell cycle. Cultured cells often become quiescent under conditions of starvation, exit the cell cycle, and enter G0. On the whole-animal scale, starvation will induce certain organs and tissues to atrophy due to cell shrinkage and loss. These cells enter back into the cell cycle only after nutrition is reestablished.

DNA damaging agents trigger checkpoints that produce arrest in G1 and G2 stages of the cell cycle. Cells can also arrest in S, which amounts to a prolonged S phase with slowed DNA synthesis. Arrest in G1 allows repair before DNA replication, whereas arrest in G2 allows repair before chromosome separation in mitosis.

Protein p53 has multiple functions related to the G1 stage arrest. p53 recognizes and binds several types of DNA damage including single-stranded DNA, insertion/deletion mismatches, and free DNA ends. In addition to these functions, p53 has sequence-specific DNA binding and transcriptional activity. Following DNA damage, p53 stimulates the transcription of p21. Protein p21 then inhibits G1 cdks. Protein p53 also has an autoregulatory feedback loop in which it stimulates the transcription of the mdm-2 oncogene, whose protein product inhibits p53. Another function of p53 is the p53-dependent stimulation of *apoptosis*, that is, cell death, following severe DNA damage. All of these p53 functions indicate roles in DNA damage checkpoints.

6.2.3 Mammalian cell cycle model by Novák and Tyson

In this subsection we present the model of mammalian cell division proposed by Novák and Tyson in [102], model that we chose for our coupling work. The process diagram proposed by the authors to represent the molecular network regulating the mammalian cell cycle is graphically depicted in Figures 6.7 and 6.8. The authors justify the model in four stages.

Antagonism between CycB/Cdk1 and Cdh1/APC

Entry into mitosis is triggered by the activity of Cdk1 in combination with B-type cyclins. The kinase subunit is present in excess, so dimer level is determined by cyclin availability. CycB[1] is absent in G1 and accumulates in S/G2/M phases of the cycle. CycB degradation is initiated by the anaphase-promoting complex (APC), which ubiquitinates its substrates, thus targeting them for proteolysis by proteasomes. At the heart of the model is an antagonistic relation between CycB and the APC. APC-dependent degradation of mitotic cyclins is mediated by "adaptors", Cdc20 and Cdh1, which apparently recognize CycB and present it to the APC core for ubiquitination. The adaptors are regulated differently during the cell cycle. Cdc20 is activated (indirectly) by CycB/Cdk1. Cdh1 is inhibited by CycB/Cdk1 (and other cyclin/Cdk holoenzymes) and functions chiefly during G1, when Cdk activity is low. Hence, between CycB/Cdk1 and Cdh1/APC there is a fundamental antagonism, which creates two stable steady states: a G1 state with active Cdh1 and low CycB activity, and an S/G2/M state with high CycB level and Cdh1 turned off.

Progress through the cell cycle is, in essence, periodic switching between these two stable states. The transition from G1 to S (i.e., commitment to a new round of DNA synthesis and division) is referred as START; the reverse transition (i.e., the completion of the chromosome cycle, which occurs when each remaining cyclin is finally destroyed by the APC) is referred as FINISH. At START, Cdh1 must be inactivated so that mitotic cyclins may reappear. This is the job of CycA/Cdk2. At FINISH, Cdh1 must be reactivated. This process depends on Cdc20. In addition to degrading cyclins A and B at phase M, Cdc20 indirectly activates Cdh1. To model the fact that Cdc20 accumulates in S/G2/M and disappears in G1, the authors assume that it is synthesized in a CycB-dependent manner. Newly synthesized Cdc20 is inactive. It becomes active phase M, in a process that depends indirectly on CycB/Cdk1 and can be delayed if chromosomes are not properly aligned on the mitotic spindle.

Inhibitory tyrosine-phosphorylation of Cdk1 subunits plays an important role in the transition from G2 to M-phase. However, since the model focuses on events in G1 phase rather than S/G2/M, the authors have chosen to neglect tyrosine phosphorylation of Cdk1 for the time being.

Early and delayed-response genes

GFs bind to specific receptors in the plasma membrane, stimulating an intracellular signal-transduc tion pathway (RasRafMAP kinase) that activates so called *early response genes*, which in turn activate a second group, the *delayed-response genes*. Among the delayed-response genes there are those for D-type cyclins, which can be considered as GF sensors. D-type cyclins, combined with Cdk4 and Cdk6, set in motion the cell-cycle engine that drives rounds of DNA replication, mitosis and cell division.

GF stimulates the synthesis of two classes of transcription factors: ERG and DRG (early and delayed-response gene products). ERG stimulates DRG synthesis and DRG inhibits ERG synthesis. In addition, DRG feeds back and activates the kinase in its own signaling pathway. Finally, DRG stimulates CycD synthesis (look at differential equations (6.2.1) and (6.2.2) in Table 6.1).

The retinoblastoma protein

CycD/Cdk4 stimulates cell growth and division by phosphorylating the retinoblastoma protein, Rb. Rb is a general inhibitor of RNA polymerases and a specific inhibitor of E2F, a transcription

[1]Following the authors' conventions, hereafter cyclin is abbreviated with cyc.

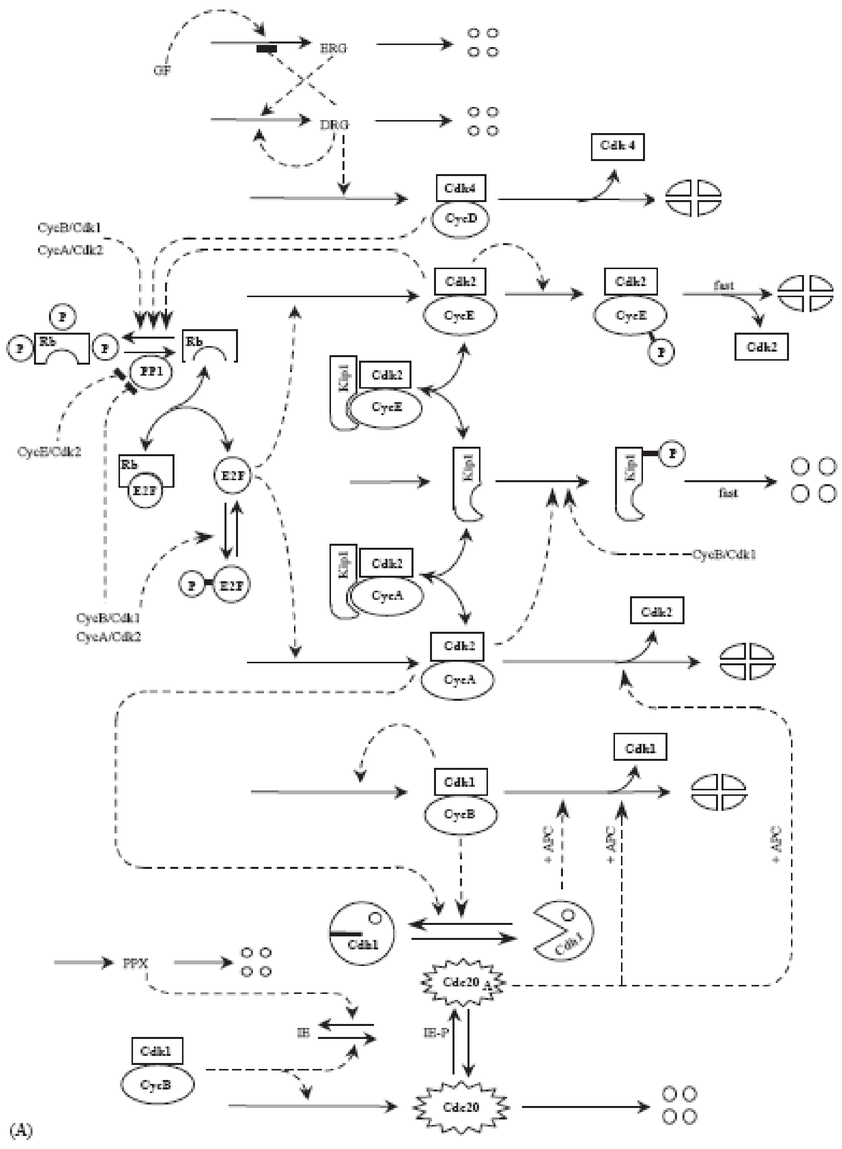


Figure 6.7: Molecular network regulating the progression of mammalian cells through the cell cycle (part A).

$$\frac{d[ERG]}{dt} = \varepsilon \frac{k_{15}}{1 + ([DRG]/J_{15})^2} - k_{16}[ERG] \tag{6.2.1}$$

$$\frac{d[DRG]}{dt} = \varepsilon (k'_{17}[ERG] + \frac{k_{17}([DRG]/J_{17})^2}{1 + ([DRG]/J_{17})^2}) - k_{18}[DRG] \tag{6.2.2}$$

$$\frac{d[cycD]}{dt} = \varepsilon k_9[DRG] + V_6[CycD : Kip1] + k_{24r}[CycD : Kip1] - k_{24}[CycD][Kip1] - k_{10}[CycD] \tag{6.2.3}$$

$$\frac{d[cycD : Kip1]}{dt} = k_{24}[CycD][Kip1] - k_{24r}[CycD : Kip1] - V_6[CycD : Kip1] - k_{10}[CycD : Kip1] \tag{6.2.4}$$

$$\frac{d[cycE]}{dt} = \varepsilon(k'_7 + k_7[E2F_A]) - V_8[CycE] - k_{25}[CycE][Kip1] + k_{25r}[CycE : Kip1] + V_6[CycE : Kip1] \tag{6.2.5}$$

$$\frac{d[cycE : Kip1]}{dt} = k_{25}[CycE][Kip1] - k_{25r}[CycE : Kip1] - V_6[CycE : Kip1] - V_8[CycE : Kip1] \tag{6.2.6}$$

$$\frac{d[cycA]}{dt} = \varepsilon k_{29}[E2F_A][mass] - k_{30}[Cdc20][CycA] - k_{25}[CycA][Kip1] + k_{25r}[CycA : kip1] + V_6[CycA : Kip1] \tag{6.2.7}$$

$$\frac{d[cycA : Kip1]}{dt} = k_{25}[CycA][Kip1] - k_{25r}[CycA : Kip1] - V_6[CycA : Kip1] - k_{30}[Cdc20][CycA : Kip1] \tag{6.2.8}$$

$$\begin{aligned}\frac{d[Kip1]}{dt} = \varepsilon k_5 - V_6[Kip1] - k_{24}[CycD][Kip1] + k_{24r}[CycD : Kip1] + k_{10}[CycD : Kip1] + \\ -k_{25}[Kip1]([CycE] + [CycA]) + k_{25r}([CycE : Kip1] + [CycA : Kip1]) + V_8[CycE : Kip1] + \\ +k_{30}[Cdc20][CycA : Kip1]\end{aligned} \tag{6.2.9}$$

$$\frac{d[E2F]}{dt} = k_{22}([E2F_T] - [E2F]) - (k'_{23} + k_{23}([CycA] + [CycB]))[E2F] \tag{6.2.10}$$

$$\frac{d[CycB]}{dt} = \varepsilon(k'_1 + \frac{k_1([CycB]/J_1)^2}{1 + ([CycB]/J_1)^2}) - V_2[CycB] \tag{6.2.11}$$

$$\frac{d[Cdh1]}{dt} = (k'_3 + k_3[Cdc20])\frac{1 - [Cdh1]}{J_3 + 1 - [Cdh1]}) - V_4\frac{[Cdh1]}{J_4 + [Cdh1]} \tag{6.2.12}$$

$$\frac{d[Cdc20_T]}{dt} = \varepsilon(k'_{11} + k_{11}[CycB]) - k_{12}[Cdc20_T] \tag{6.2.13}$$

$$\frac{d[Cdc20]}{dt} = k_13[IEP]\frac{[Cdc20_T] - [Cdc20]}{J_{13} + [Cdc20_T] - [Cdc20]} - k_{14}\frac{[Cdc20]}{J_{14} + [Cdc20]} - k_{12}[Cdc20] \tag{6.2.14}$$

$$\frac{d[PPX]}{dt} = \varepsilon k_{33} - k_{34}[PPX] \tag{6.2.15}$$

$$\frac{d[IEP]}{dt} = k_{31}[CycB]\frac{1 - [IEP]}{J_{31} + 1 - [IEP]} - k_{32}[PPX]\frac{[IEP]}{J_{32} + [IEP]} \tag{6.2.16}$$

$$\frac{d[GM]}{dt} = k_{27}[mass]H(\frac{[Rb_{hypo}]}{[Rb_T]}) - k_{28}[GM] \tag{6.2.17}$$

$$\frac{d[mass]}{dt} = \varepsilon\mu[GM] \tag{6.2.18}$$

Steady-state relations

$$[PP1_A] = \frac{[PP1_T]}{1 + K_{21}(\phi_E([CycE] + [CycA]) + \phi_B[CycB])} \tag{6.2.19}$$

$$[RB_{hypo}] = \frac{[Rb_T]}{1 + \frac{k_{20}(\lambda_D[CycD_T] + \lambda_E[CycE] + \lambda_A[CycA] + \lambda_B[CycB])}{k'_{19}([PP1_T] - [PP1_A]) + k_19[PP1_A]}} \tag{6.2.20}$$

$$[E2F_A] = \frac{([E2F_T] - [E2F : Rb])[E2F]}{[E2F_T]} \tag{6.2.21}$$

$$[E2F : Rb] = \frac{2[E2F_T][Rb_{hypo}]}{[E2F_T] + [Rb_{hypo}] + L + \sqrt{([E2F_T] + [Rb_{hypo}] + L)^2 - 4[E2F_T][Rb_{hypo}]}} \tag{6.2.22}$$

Table 6.1: Mathematical model of mammalian cell-cycle controls.

Definitions

$$V_2 = k_2'(1-[Cdh1]) + k_2[Cdh1] + k_2''[Cdc20] \tag{6.2.23}$$

$$V_4 = k_4(\gamma_A[CycA] + \gamma_B[CycB]) \tag{6.2.24}$$

$$V_6 = k_6' + k_6(\eta_E[CycE] + \eta_A[CycA] + \eta_B[CycB]) \tag{6.2.25}$$

$$V_8 = k_8' \frac{k_8(\psi_E([CycE]+[CycA]) + \psi_B[CycB])}{J_8 + [CycE_T]} \tag{6.2.26}$$

$$L = \frac{k_{26r}}{k_{26}} + \frac{k_{20}}{k_{26}}(\lambda_D[CycD] + \lambda_E[CycE] + \lambda_A[CycA] + \lambda_B[CycB]) \tag{6.2.27}$$

Rate constants (h^{-1})

$k_1' = 0.1$, $k_1 = 0.6$, $k_2' = 0.05$, $k_2 = 20$, $k_2'' = 1$, $k_3' = 7.5$, $k_3 = 140$, $k_4 = 40$, $k_5 = 20$, $k_6' = 10$, $k_6 = 100$, $k_7' = 0$, $k_7 = 0.6$, $k_8' = 0.1$, $k_8 = 2$, $k_9 = 2.5$, $k_{10} = 5$, $k_{11}' = 0$, $k_{11} = 1.5$, $k_{12} = 1.5$, $k_{13} = 5$, $k_{14} = 2.5$, $k_{15} = 0.25$, $k_{16} = 0.25$, $k_{17}' = 0.35$, $k_{17} = 10$, $k_{18} = 10$, $k_{19}' = 0$, $k_{19} = 20$, $k_{20} = 10$, $k_{22} = 1$, $k_{23}' = 0.005$, $k_{23} = 1$, $k_{24} = 1000$, $k_{24r} = 10$, $k_{25} = 1000$, $k_{25r} = 10$, $k_{26} = 10000$, $k_{26r} = 200$, $k_{27} = 0.2$, $k_{28} = 0.2$, $k_{29} = 0.05$, $k_{30} = 20$, $k_{31} = 0.7$, $k_{32} = 1.8$, $k_{33} = 0.05$, $k_{34} = 0.05$, $\mu = 0.061$.

Dimensionless constants

$J_1 = 0.1$, $J_3 = J_4 = 0.01$, $J_8 = 0.1$, $J_{13} = J_{14} = 0.005$, $J_{15} = 0.1$ $J_{17} = 0.3$, $J_{31} = J_{32} = 0.01$, $K_{21} = 1$, $[E2F_T] = 5$, $[PP1_T] = 1$, $[Rb_T] = 10$, $\phi_E = 25$, $\phi_B = 2$, $\gamma_A = 0.3$, $\gamma_B = 1$, $\eta_E = \eta_A = 0.5$, $\eta_B = 1$, $\lambda_D = 3.3$, $\lambda_E = 5$, $\lambda_A = 3$, $\lambda_B = 5$, $\psi_E = 1$, $\psi_B = 0.05$, $\varepsilon = 1$.

Notes on equations

Novák and Tyson make a list of assumptions that we report in the following. First of all, they write DEs for proteins only and they neglect mRNAs. That is, they assume rapid message turnover, so that mRNAs are always in steady state. As a consequence, although the rate of synthesis of each protein is proportional to the level of its message, [mRNA] never shows up in the equations.
All the rate-of-synthesis terms for proteins have a factor ε, which represents the translation efficiency of the ribosomes. ε is a number between 0 and 1; its value is influenced by growth factors and by translation inhibitors like cycloheximide.
(6.2.3), (6.2.5), (6.2.7), (6.2.11). They assume that all the Cdks are in excess over their cyclin partners, so their concentrations are not rate-limiting in the formation of cyclin/Cdk complexes. For this reason, the concentrations of Cdks do not show up in the equations, and each cyclin/Cdk complex is named for its cyclin subunit.
(6.2.9) They do not write an extra DE for the phosphorylated form of Kip1, because they assume that it is rapidly ubiquitinated and degraded.
(6.2.10) [E2F] is the total concentration of unphosphorylated E2F (free E2F, and E2F complexed with Rb).
(6.2.15) IE is an intermediary enzyme that creates a time delay between CycB accumulation and Cdc20 activation. IEP is dephosphorylated by a phosphatase whose basal activity $=k_3 2$. They assume that the activity of this phosphatase (in the nucleus, where it opposes the action of CycB/Cdk1) is proportional to translation efficiency, ε. Hence, when ε drops to 0.5, both kinase and phosphatase acting on IE are halved.
(6.2.16) $H([Rb_{hypo}]/[Rb_T])$ is a Heaviside function, which equals 0, if $[Rb_{hypo}]/[Rb_T] > 0.8$, or 1, if $[Rb_{hypo}]/[Rb_T] \leq 0.8$.
(6.2.17) They assume that a cell divides, $[mass] \to [mass]/2$, when $[Cdh1]$ crosses 0.2 from below.
(6.2.18) $[PP1_A]$ is the active (dephosphorylated) form of PP1. $[CycE] = [CycE_T][CycE : Kip1]$.
(6.2.19) $[Rb_{hypo}]$ is the total concentration of dephosphorylated forms of Rb, including complexes with E2F and E2FP. $[CycD_T] = [CycD] + [CycD : Kip1]$, because binding of Kip1 does not inhibit CycD/Cdk4 complexes.
(6.2.20) $E2F_A$ is the active form of E2F, i.e. unphosphorylated and not complexed with Rb.
(6.2.21) [E2F:Rb] represents E2F (either phosphorylated or unphosphorylated) complexed with Rb.

Table 6.2: Mathematical model of mammalian cell-cycle controls (continuation).

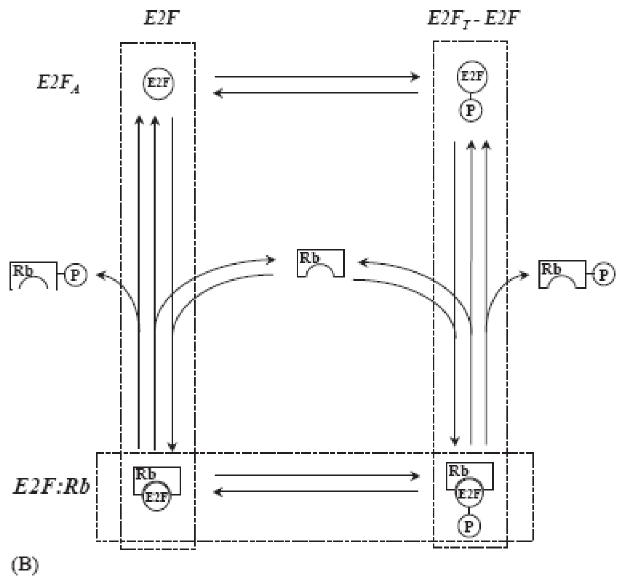


Figure 6.8: Molecular network regulating the progression of mammalian cells through the cell cycle (part B).

factor for the CycA and CycE genes. Phosphorylated Rb releases its hold on E2F, which stimulates synthesis of CycA and CycE. Since these cyclins, in combination with Cdk2, can phosphorylate Rb, CycA and CycE activate their own synthesis. Indeed, Novák and Tyson assume that all four cyclin/Cdk complexes (A, B, D, and E) phosphorylate Rb, although their efficiencies may differ (see equation (6.2.20)). They assume that the total levels of E2F and Rb do not fluctuate ($E2F_T$ and Rb_T are constants). Rb is distributed between two forms: an active, hypophosphorylated form (Rb_{hypo}), which binds to and inhibits E2F, and an inactive, phosphorylated form (Rb_T-Rb_{hypo}), which does not bind E2F.

Dephosphorylation of Rb is catalysed by a type-1 protein phosphatase (PP1), whose activity is inhibited by cyclin/Cdk complexes. PP1 phosphorylation could be started by CycE/Cdk2, with CycA/Cdk2 and CycB/Cdk1 keeping PP1 phosphorylated until the end of mitosis. Down-regulation of PP1 activity could be an essential event at the restriction point, since constitutively active PP1 arrests proliferation of Rb^+ cells. In the model we are describing, the total concentration of PP1 ($PP1_T$) is constant throughout the cycle and distributed between more active ($PP1_A$) and less active forms($PP1_T PP1_A$) in a Cdk-dependent manner (see equation (6.2.19)). Furthermore, the phosphorylation and dephosphorylation of Rb is assumed to be fast enough that the hypo- and hyper-phosphorylated forms are always in equilibrium (equation (6.2.20)).

Another role of Rb is to repress the transcription of housekeeping genes by inhibiting RNA polymerases I and III. By this mechanism, Rb interferes with general cell growth as well as the synthesis of E2F-specific gene products. To model this effect, equation (6.2.17) describes the production of general machinery (GM) for protein synthesis in an Rb-dependent manner, which machinery is then used (equation (6.2.18)) to increase the overall mass of the cell.

Antagonism between Kip1 and cyclins A and E

$p27^{Kip1}$ binds to CycA/Cdk2 and CycE/Cdk2 dimers to form inactive trimers. In the model, Kip1 is synthesized at a constant rate. Degradation of Kip1 depends on ubiquitination by a protein complex called the SCF. Cdk-catalysed phosphorylation of threonine-187 of Kip1 promotes its ubiquitination. CycA-, CycE- and CycB-complexes are allowed to phosphorylate Kip1 (although with different efficiencies, see equation (6.2.25) in Table 6.2). Kip1 and the CycA- and CycE-complexes are mutual antagonists, much like Cdh1/APC and CycB/Cdk1. Either Kip1 is abundant and the cyclins are inactive, or Kip1 is absent and at least one of the cyclins is active.

During G1 phase, several events occur in succession. First, CycD/Cdk4 helps to rid the cell of the enemies of CycA and CycE, by phosphorylating Rb and by binding to and titrating away Kip1. A little help from CycD is enough to allow rapid, autocatalytic rise of cyclins A and E, as they phosphorylate Kip1 and Rb, thus destroying their inhibitor and turning on their own transcription factor, E2F. As CycA rises, it initiates DNA synthesis and turns off Cdh1, allowing CycB to accumulate, so that the cell will eventually be able to enter mitosis.

Ubiquitin-mediated degradation of CycE is also a phosphorylation-dependent process, mediated presumably by CycE/Cdk2 itself and perhaps by other Cdk complexes (equation (6.2.26)). In addition, cyclins A and B have a negative effect on E2F-dependent transcription, by phosphorylating DP1 (the partner of E2F) and thereby down-regulating synthesis of cyclins A and E. Rather than introducing another variable for DP1, the model equations regulate E2F by phosphorylation (equation (6.2.10)). Rb is assumed to bind to both unphosphorylated and phosphorylated forms of E2F, and only the free, unphosphorylated form of E2F is transcriptionally active, (equation (6.2.21)). (In these equations, the concentration of all forms of E2F is $E2F_T$, of all non-phosphorylated forms is E2F, of all phosphorylated forms is $E2F_T E2F$, and of the only active form is $E2F_A$). Notice that, when E2F turns off, CycE level will drop precipitously because it is heavily phosphorylated, but CycA- and CycB-associated kinases will take over CycE role in phosphorylating Rb and Kip1.

Simulation of the cell cycle

Figure 6.9 shows numerically simulated cell cycles of normal mammalian cells, growing exponentially in the presence of GFs, based on the differential equations and parameter values in Tables 6.1 and 6.2. We see that, for a newborn cell ($t = 0$) in early G1, CycD/Cdk4 represents the only kinase phosphorylating Rb, and cyclins A, B and E all lose to their antagonists, Kip1, Rb and Cdh1. However, CycE synthesis is increasing because E2F is slowly recovering from its phosphorylated state caused by CycA- and CycB-kinase in the previous cycle ($t < 0$). As a consequence, at approximately 3 h after cell division, Rb phosphorylation by CycD-kinase (fixed level) and CycE-kinase (increasing level) reaches a threshold, and the positive-feedback loop in CycE transcription turns on. The explosive rise in CycE-dependent kinase activity is reinforced by the simultaneous elimination of its inhibitor Kip1 (a second positive-feedback loop). Since E2F is active at this time, CycA rises and turns off Cdh1 at about half-way through the cycle. Together, CycA- and E-dependent kinases stimulate DNA synthesis. For lack of a better event marker, the authors presume that the G1/S transition is roughly coincident with the inactivation of Cdh1. Somewhat later, high CycB-kinase activity drives the cell into mitosis. After a suitable time delay, introduced by the intermediary enzyme (IE), Cdc20 is activated, enabling cells to exit from mitosis. The cycle then repeats itself.

An essential feature of this model is balanced growth and division, i.e. interdivision time=mass doubling time. In the model, the rate of mass increase is determined by the level of "general machinery"; hence, by the dynamics of Rb and in turn by the presence or absence of GF. No matter how fast or slow cells are growing, cell division is always driven by a doubling of mass. Growth and division are connected by a built-in requirement that cells reach a critical size before starting DNA synthesis. If mass at birth is larger than average (one arbitrary unit), then cells have a shorter cycle time than average .

Cell size enters the model through the "mass"factor in the synthesis of CycA (equation (6.2.7) in Table 6.1). This factor causes the nuclear concentration of CycA to increase as the cell grows, thereby introducing size control at the G1/S boundary.

6.3 Protein p53

In this section we will describe the metabolic pathways of p53, a tumour suppressor protein. Such a protein is activated in reply to DNA damage. In normal conditions, the concentration of p53

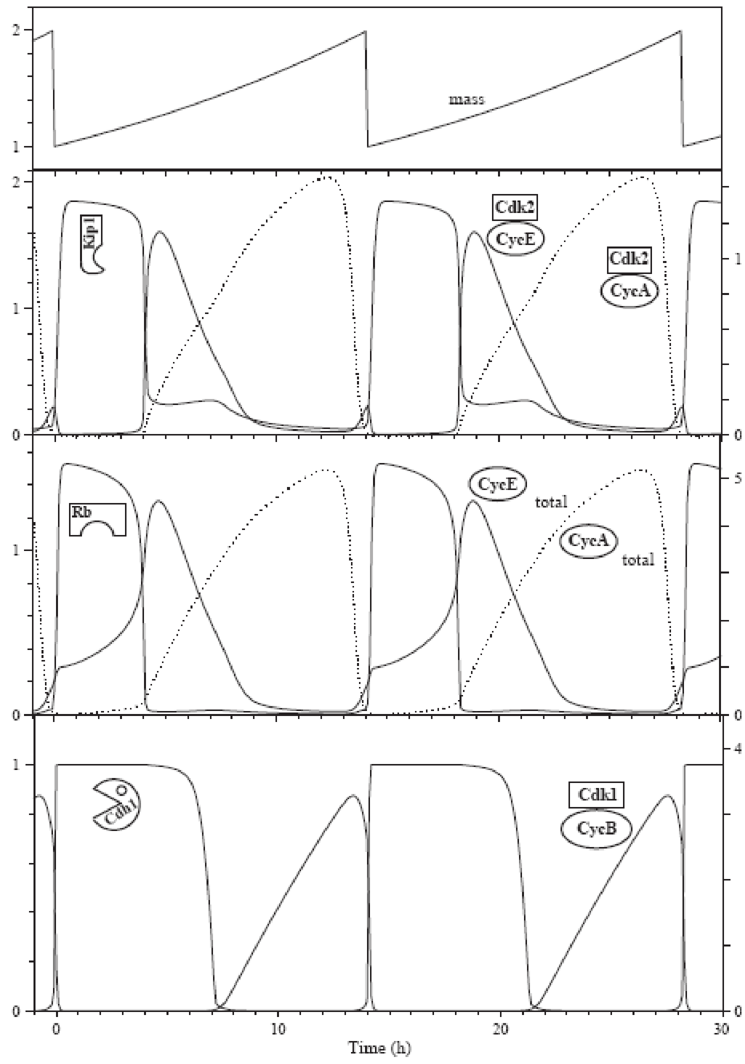


Figure 6.9: Numerical simulation of the mammalian cell cycle).

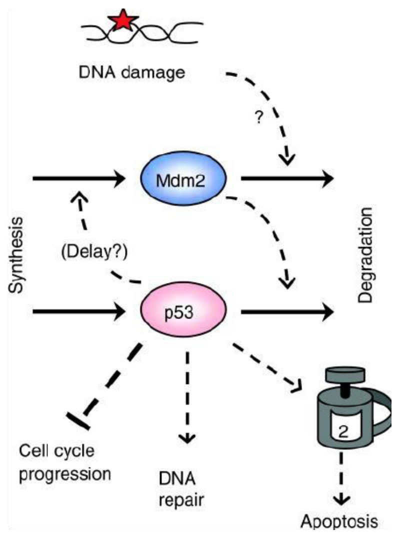


Figure 6.10: The couple p53-Mdm2: a negative loop.

in the nucleus of a cell is feeble: its level is controlled by another protein, Mdm2. These two proteins present a loop of negative regulation. In fact, p53 actives the transcription of Mdm2 while the latter accelerates the degradation of the former (see Figure 6.10). DNA damage increases the degradation rate of Mdm2 so that the control of this protein on p53 becomes weaker and p53 can exercise its functions. The concentration of p53 in sound cells is feeble because this protein is responsible for the activation of many mechanisms that could be dangerous for cells. In an indirect way, it stops the DNA synthesis process, it activates the production of proteins charged with DNA reparation, and it can lead to apoptosis. The activation ways of all these mechanisms are multiple and complex. The first function of p53 is to be a transcription factor. Genes whose transcription is due to p53 (protein $p21^{cip1}$ in particular) are responsible either for the arrest of the cell cycle or for apoptosis. Furthermore, during apoptosis p53 moves to the mitochondria, where it directly actives certain pro-apoptosis proteins and it induces apoptosis independently and more rapidly than through the process activated for transcription.

Protein Mdm2 inhibits the action of p53 both in a direct way, by degrading the protein, and in an indirect way, for competition at the level of a bond site to p21. As far as the degradation of p53 is concerned, it is possible if a first ubiquitin[2] molecule binds to p53: this is the role of Mdm2. Recently it has been discovered that another protein, namely p300, is in competition with Mdm2 to bind to the same site of p53 and carry out the poly-ubiquination of p53, that in such a way will be definitively degraded [76]. Such a process, which requires subsequent bonds of Mdm2 and p300 to p53, can be inhibited by the phosphorylation of p53 or Mdm2 through a kinase of family ATM. This causes a slowing down of p53 degradation. Another way for Mdm2 to inhibit the action of p53 consists in directly binding to it (in concurrence with p300) at the level of the transcription activation site. This way to act is faster but reversible. Furthermore, M2m2 binds to p21, protein which inhibits the G1-S transition and which is one of the targets of p53, and promotes its degradation, stimulating in this way the transition G1-S. Finally, p53 activity can be inhibited for sequestration. Once "monoubiquinated" by Mdm2, p53 (or at least a part of the total concentration of the protein) moves to the cytosol. Here p53 can inhibit the pro-apoptosis protein Bak, preventing in such a way from a possible premature orientation of the cell to apoptosis.

[2]Ubiquitin is a protein which serves to mark proteins that must be eliminated.

6.3.1 A “digital” response

When DNA is damaged, the concentration of p53 must increase to protect the cell, either stopping the cell cycle to allow DNA repair, or provoking apoptosis, if damage is too heavy. This is possible if the control of Mdm2 on p53 weakens. As written above, in most of cases this happens for phosphorylation of p53 and Mdm2 through ATM. When Mdm2 loosens its influence on p53, it is possible to observe some oscillations of p53 and Mdm2 concentrations. The answer to a stronger damage is a bigger number of oscillations. Oscillations have a very regular period.

The first observations of such a phenomenon, which referred to a population of tumor cells, showed damped p53 oscillations owing to a damage due to gamma irradiation. Further observations by Lahav et al. [82] revealed that, on the contrary, if we consider one single cell, the oscillatory behaviour is different: oscillations are not damped and have a constant period. Such observations took a short time period (16 hours) and did not allow to observe more than 2 peaks. More recent observations [54] showed that oscillations, which have a period of approximately 5-6 hours, can last several days. The authors hypothesize that the probability that oscillations become permanent increases with DNA damage. They observe a variability in oscillation amplitude but a regularity in the period. On the contrary, even after strong gamma irradiations, the 40% of observed cells does not show a regular oscillatory behaviour, but rather irregular fluctuations. In this connection, we must remember that observations have been performed on a population of tumor cells; this could explain the non-physiologic behaviour of these cells.

The observed differences on the cellular response to irradiation could be partially explained through the stochastic behaviour of the reparation process and the instability of the oscillator p53-Mdm2. This system is in an equilibrium state if DNA is not damaged but it can pass to an oscillatory state after some variations in its molecular environment. We can suppose that, for one of the parameters, there exists a threshold whose overcoming brings the system to an oscillatory state. If this threshold is not overcome, the system with oscillations will return to its equilibrium state.

6.3.2 Modeling the oscillating behaviour of the protein

Several models have been proposed in literature to model the oscillatory behaviour of proteins p53 and Mdm2. Modelers know that a single negative feedback can only cause damped oscillations, therefore the relation between p53 and Mdm2 can not be described by a negative loop. In order to obtain the desired oscillations, researches try to understand which other mechanism plays a decisive role and creates non-damped and long-lasting oscillations. From a mathematical point of view, to reach the desired result it is possible to combine a positive feedback either with a delay, or with a strongly non-linear response, or with a negative feedback.

In literature, one of the first models of proteins p53 and Mdm2 is due to Lev Bar-Or et al. They consider an only negative loop between p53 and Mdm2 but this loop is composed by three elements: p53, Mdm2, and an intermediate element that could be interpreted as Mdm2_{mRNA}. This model creates damped oscillations, as the first observations had shown.

On the contrary, the subsequent models represent what has been defined the “digital response”of p53, i.e., the oscillatory response, which is regular both in the period and in the amplitude of oscillations. The most interesting models are undoubtedly the ones proposed by Chickermane et al. [24], by Ciliberto et al. [25], and by Geva-Zatorsky et al. [54]. These models try to explain how the couple Mdm2-p53 leaves its state of equilibrium and assumes a very regular oscillatory behaviour. ATM, which triggers such a machinery, can either activate the production of p53 or activate the degradation of Mdm2.

6.4 Irinotecan

Camptothecins are substances that can be extracted from the Chinese tree “Camptotheca acuminata Decne”(see Figure 6.11). Such a tree is known from several centuries for its medical properties

Figure 6.11: Leaves of Camptotheca acuminata Decne.

and is mainly used for the treatment of digestive cancers. Camptothecins are present in the cortex and in the leaves of the plant. Their anticancerogenic properties have been discovered at the end of the Fifties in the United States but the first clinical tests have been interrupted owing to heavy effects due to the toxicity of the substances. The most frequent undesirable effects are myelosuppression and diarrhoea.

Biologists continued their researches trying to decrease toxic effects linked to the medicine. The first results have been obtained only in the Eighties, when they discovered that camptothecins are inhibitors of topoisomerase I, essential enzyme for DNA synthesis. Afterwards, researches started to focus on some semi-synthetic derivative of water-soluble camptothecins, such as irinotecan and topotecan.

6.4.1 Metabolic pathways of Irinotecan

Irinotecan is pro-medicine and must be transformed in its active metabolite, SN38, to be effectively cytotoxic. In fact the anticancerogenic activity of irinotecan (CPT11) is approximately 100 times less effective than the one of SN38. The activation is due to carboxylesterase (CES), an enzyme mainly located in the liver, in the intestine, and in the tumoral tissues. SN38 is then detoxified through glucorono-conjugation: this realizes uridine diphosphate glucoronosyl transferase 1A1 (UGT1A1). Irinotecan is also oxidized by P-450 3A4 and 3A5: the first one leads either to APC or to NPC; the second one transforms irinotecan into another metabolite (M4) (irinotecan is preferably motabolized by cytochrome P-450 3A4). The efficacy and the toxicity of the medicine are linked to its metabolic pathways: for example, the toxicity level of the medicine is partially determined by the polymorphism of UGT1A1, the enzyme responsible for its detoxification. Another factor of resistance to the irinotecan therapy is the detoxicant effect of MDRs (multidrug resistance proteins), which assures the efflux of endogenous and exogenous substrata of the cell. This kind of proteins is often over-expressed in tumor cells which, on account of their genetic instability, present strong resistances to medicaments. If the cancer develops on cells which have the gene of a transporter but do not express it, then the expression of the gene can be started through exposition to anticancerogenic drugs.

The transporter having SN38 as substratum is ABCG2 but there are different transporters for SN38-G (ABCC1) and for CPT11 (ABCA2). In the model we will consider in the following, there will be an only transporter, namely ABCG2. The metabolic pathways of irinotecan are graphically depicted in Figure 6.12.

Mechanisms through which irinotecan damages the cell are very complex and have not completely been explained yet. As previously said, we know that DNA lesions appear after the inhibition of topoisomerase by SN38.

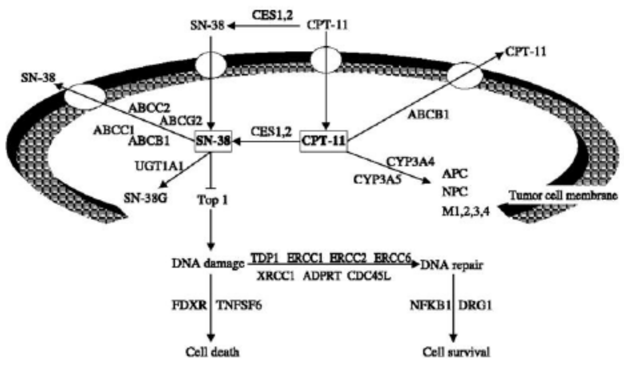


Figure 6.12: Metabolic pathways of irinotecan.

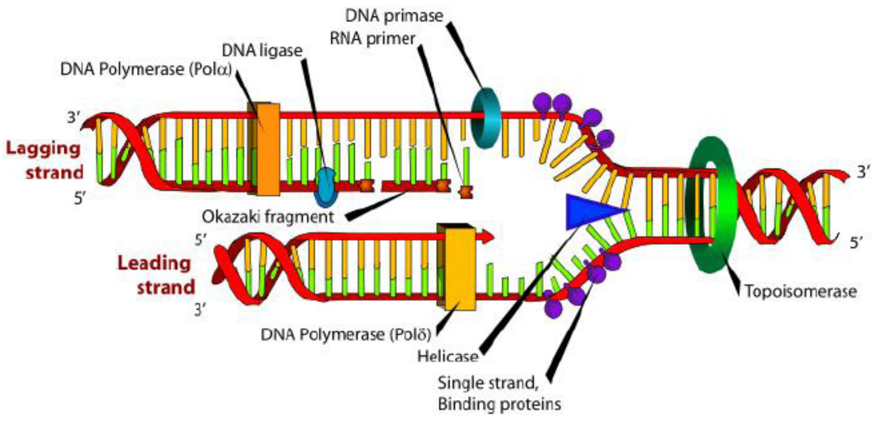


Figure 6.13: The replication fork.

6.4.2 Inhibition of Toposoimerase and DNA damage

Topoisomerase I and II (Top1 and Top2, respectively) are proteins which are present in all living organisms and which check DNA replication and transcription. They intervene to modify the DNA winding degree: Top1 acts on one strand, Top2 on both the strands. In this context we are interested in Top1, which is a target for camptothecins. Top1 links itself to extremity 3' of DNA forming a transitory cleavage complex and cuts a DNA strand, that in such a way is able to unroll. Then such a complex dissociates and a new ligature comes up. In normal conditions, the connection process is favored with respect to the cleavage one. See Figure 6.13 to have a graphical idea of the replication fork.

The target of irinotecan, and above all of its active metabolite SN38, is the complex Top1-DNA. SN38 links to the complex through a covalent bond, preventing in such a way from the ligature of the DNA strand. As clearly written in the title of [107], SN38 acts like a "foot in the door": it keeps opened the DNA strand to which Top1 is linked as to prevent a door from closing. These complexes are still reversible and do not cause DNA lesions. However, they favor them: some lesions can rise as a consequence of the possible collisions with the transcription complexes or with the replication fork. This induces the arrest of the cell cycle. In this case we speak of irreversible complexes. Lesions due to the inhibition of topoisomerase are therefore consecutive to the stages of the cell metabolism. It means that irinotecan injections must be repeated and abundant in order to be effective. Besides irinotecan is more effective during the DNA replication phase [116, 129]. Furthermore, the inhibition of the DNA synthesis takes rapidly place (in a few

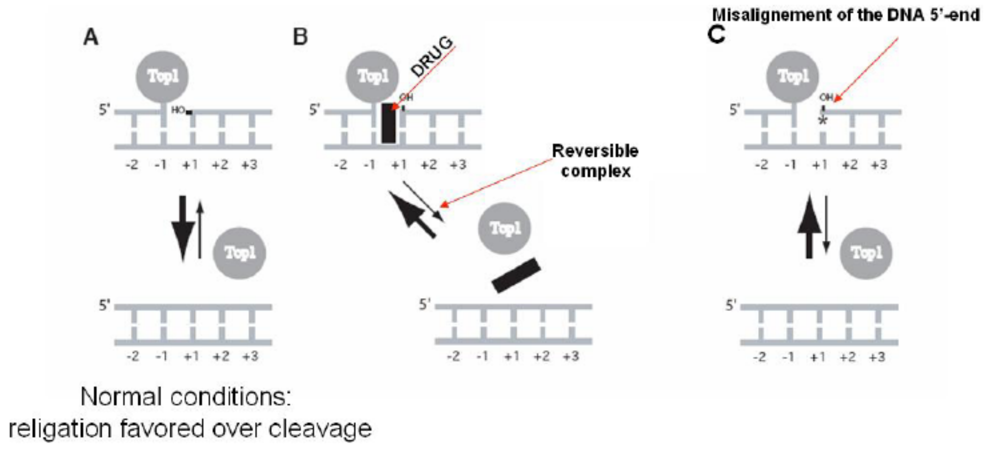


Figure 6.14: Damage process of complex Top1-DNA-SN38.

minutes) and lasts several hours. To see a graphical representation of the damage process of complex Top1-DNA-SN38, look at Figure 6.14.

Defence answers of cells subjected to irinotecan injections are multiple and vary according to the drug dose. The administration of a very light dose suffices to slow down the S phase of the cell cycle and to delay the G2-M transition. If the dose is more substantial, the lag time in the S phase is much more significant and the cell cycle arrest in the G2-M transition can last more than sixty hours or even be permanent. In this latter case, some genes responsible for the cell cycle arrest (as an example, p21) and involved in the aptototic pathway are over-expressed. These genes are activated by p53, and this suggests the intervention of the protein in reply to a DNA damage due to the dissociation of Top1 from DNA [129].

Biologists have not still still answered to the following question: why do camptothecins selectively attack cancer cells? Such a question rises for two reasons: firstly, Top1 is present in all cells, both sound and tumor ones, and secondly, it has not not been proved that Top1 is more sensible to camptothecins in tumor cells. A possible explication could be the defective functioning of reparation and cell cycle arrest pathways in tumor cells. Another hypothesis is the over-expression of Top1 in some tumor cells of the colon [107].

6.5 A partial model: coupling of Irinotecan and p53

In this section we introduce a model developed by Dimitrio [38] which describes the pharmacokinetics/pharmacodinamics (PK/PD) of irinotecan and the behaviour of proteins p53 and Mdm2. For our coupling work we will refer to this model, that takes its place in the European project TEMPO (Temporal genomics for tailored chronotherapeutics), which aims at redefining an irinotecan therapy.

As far as the PK/PD model of a drug is concerned, it takes aim at representing the action of the drug on the body (pharmacodinamic) and the action of the body on the drug (pharmacokinetic), and thus the drug metabolism and its transformations. Such kind of model is useful to give a simple description of observations, to describe the drug action while time evolves, and to make some previsions to optimize the treatment posology in clinic tests. The model we take into consideration deals with the cellular concentrations of irinotecan and its metabolites.

6.5.1 Pharmacokinetics of Irinotecan

The metabolic pathways represented in the pharmacokinetic model taken into consideration are the transformation of CPT11 into SN38 by carboxylesterase, the detoxification due to UGT1A1 which glucoroconjugates SN38 in SN38G, and the transport made by ABCG2. Irinotecan injections are represented by a function which is constant during the administration and then humbles itself. The model is constituted by the following differential equations:

$$\frac{d[CPT11]}{dt} = In(t) - k_1\frac{[CES][CPT11]}{K_{m1}+[CPT11]} - k_{t1}\frac{[ABCG2][CPT11]}{K_{t1}+[CPT11]} \tag{6.5.1}$$

$$\begin{aligned}\frac{d[SN38]}{dt} &= k_1\frac{[CES][CPT11]}{K_{m1}+[CPT11]} - k_{t2}\frac{[ABCG2][SN38]}{K_{t2}+[SN38]} - k_2\frac{[UGT1A1][SN38]^n}{K_{m2}^n+[SN38]^n}+\\ &-k_{compl}[SN38][TOP1][DNA_{free}] + k_{compl1}[CC]\end{aligned} \tag{6.5.2}$$

$$\frac{d[SN38G]}{dt} = k_1\frac{[UGT1A1][SN38]^n}{K_{m1}^n+[SN38]^n} - k_{d1}[SN38G] \tag{6.5.3}$$

$$\frac{d[ABCG2]}{dt} = k_{t2}[ABCG2](\frac{[SN38]}{K_{t2}+[SN38]} + k_{t1}\frac{[CPT11]}{K_{t1}+[CPT11]}) - k_{d2}[ABCG2] \tag{6.5.4}$$

Equation (6.5.1) describes the behaviour of CPT11. Its concentration in the cell depends on irinotecan injections, which have been represented by function $In(t)$. To describe the action of carboxylesterase, an enzymatic Michaelian kinetiks has been used and the activity of such an enzyme has been considered constant. Also the kinetics of the transporter has been described by a function of the same type. This choice is supported by the study presented in [127], where Koeking et al. analyze a transporter of the ABC family in a population of E.Coli bacteria. In the article, the transport rate is represented in function of the drug concentration and its dynamics is Michaelian. This induces to use such a representation for the cellular transporter of irinotecan.

As far as SN38 concentration is concerned, a Hill function with coefficient n has been chosen to describe the action of enzyme UGT1A1. Such an enzyme shows a detoxification action which is supposed to be cooperative. The formation of the ternary complexes SN38-Top1-DNA has been represented by a mass action low. Observe that SN38G transport in equation (6.5.3) is represented by a linear term while for SN38 and CPT11 the transporter has been explicitly represented.

Finally, let us consider equation (6.5.4), which represents the dynamics of the transporter. Such a protein is trans-membranaceous, that is, it transports its substratum through the membrane of the cell. Furthermore, more substratum there is in the cell, more it is expressed. It is not consumed by its substratum and it is saturable. The last term of the equation is a degradation one.

6.5.2 Pharmacodynamics of Irinotecan

In order to represent DNA lesions, it is necassary to describe topoisomerase and the ternary complexes that are formed after irinotecan injections. It is known ([80]) that the topoisomerase expression is regulated by a *circadian rhythm*, that is, it follows the periodicity of the 24-hours cell cycle. The differential equations which describe the pharmacodynamics of irinotecan are the following ones:

$$\begin{aligned}\frac{d[TOP1]}{dt} &= k_{top1}(1+\varepsilon\cos(\frac{2\pi(t-\phi)}{24})) - k_{compl}[SN38][TOP1][DNA_{free}]+\\ &+k_{compl_1}[CC] - k_{dtop1}[TOP1]\end{aligned} \tag{6.5.5}$$

$$\frac{DNA_{free}}{dt} = -k_{compl}[SN38][TOP1][DNA_{free}] + k_{compl1}[CC] + repairDNA([p53_{tot}],[CC_{irr}]) \tag{6.5.6}$$

$$\frac{d[CC]}{dt} = k_{compl}[SN38][TOP1][DNA_{free}] - k_{compl1}[CC] - k_{irr}[CC] \tag{6.5.7}$$

$$\frac{[CC_{irr}]}{dt} = k_{irr}[CC] - repairDNA([p53_{tot}],[CC_{irr}]) \tag{6.5.8}$$

Equation (6.5.5) describes the behaviour of topoisomerase: the first term is the constant production rate while the second one is the circadian production rate. In this second term, a cosine has been chosen to represent the rhythmicity of topoisomerase and ε has been introduced to indicate if the expression of the protein is well cadenced ($\varepsilon = 1$) or not (in tumor cells this natural rhythm can be lost). Equations (6.5.7) and (6.5.8) describe the complexes (reversible and irreversible,

$k_1 = 0.6$	$K_{m1} = 0.2$	$k_{t1} = 0.0001$	$K_{t1} = 2.5$
$K_{t2} = 2$	$k_2 = 0.5$	$K_{m2} = 2.5$	$n = 3$
$k_{compl} = 0.1$	$k_{compl_1} = 0.001$	$k_{d1} = 0.08$	$k_{d2} = 0.02$
$k_{irr} = 0.3$	$k_{t2} = 0.1$	$k_{top1} = 0.5$	$k_{dtop1} = 0.5$

Table 6.3: Parameter values of the PK/PD model of irinotecan.

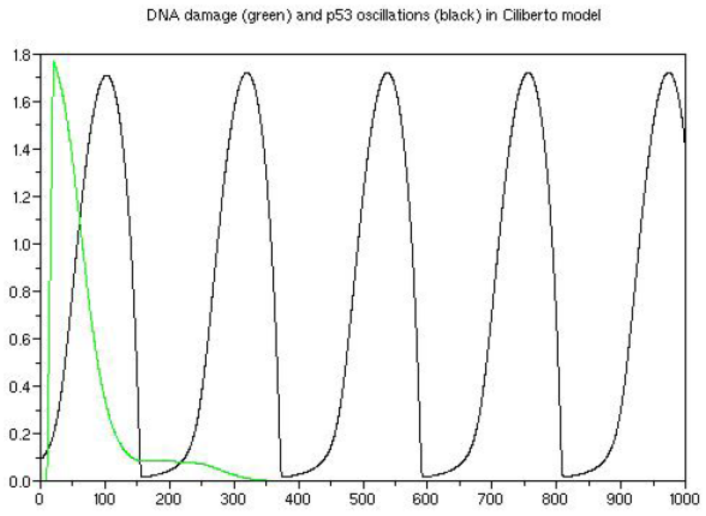


Figure 6.15: Simulation of Ciliberto et al.'s model.

respectively) due to the link between Top1, SN38, and DNA. As far as reversible complexes are concerned, the mass action low has been followed. As far as irreversible complexes, and thus DNA lesions, are concerned, their formation rate is a linear function of the concentration of reversible complexes. Equation (6.5.6) describes the dynamics of free DNA: there is a total quantity of free DNA which decreases in function of the ternary complexes that are formed ($[CC]$) and of DNA lesions ($[CC_{irr}]$). Function "repairDNA"is a reparation function depending on free DNA and on p53 concentration:

$$repairDNA([p53_{tot}], [CC_{irr}]) = -k_{dDNA}[p53_{tot}]\frac{[CC_{irr}]}{J_{DNA} + [CC_{irr}]}.$$

The parameter values of the PK/PD model of irinotecan are present in Table 6.3.

6.5.3 Coupling with p53

The PK/PD model of irinotecan has been coupled by Dimitrio with a model describing the behaviour of protein p53. The aim was to link the DNA damage process to the possible arrest of the cell cycle. As previously said, protein p53 has a strong control role in the life of a cellule: it intervenes in the DNA reparation and in the cell cycle arrest and, if lesions are too heavy, it can lead to apoptosis.

Among the models proposed in literature for protein p53, they chose the one of Ciliberto, Novak, and Tyson [25]. Even if this model is a bit more complicated than the other ones, because it describes several states of proteins p53 and Mdm2, it presents some robustness properties. In function of the bifurcation parameter k_{bif}, which represents the degradation rate of Mdm2, we can observe the following situation: after an initial stable stationary state, for enough big values of k_{bif} it appears an instable stationary state which is surrounded by a cycle. For a simulation of this model, look at the chart in Figure 6.15, where DNA damage is green and p53 is black.

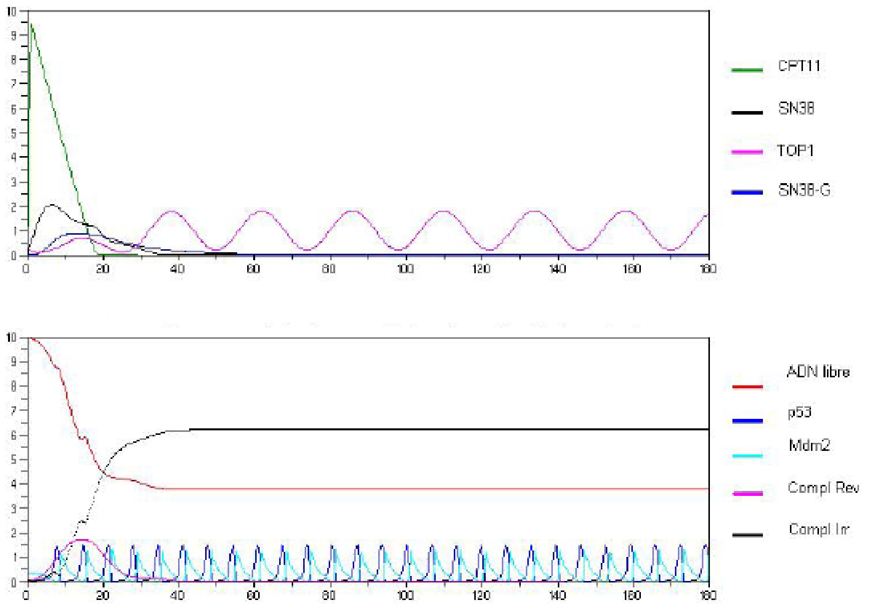


Figure 6.16: One injection made "in phase" suffices to kill the cell.

The differential equations of Ciliberto et al.'s model are the following ones:

$$\frac{d[p53_{tot}]}{dt} = k_{s53} - k_{d53'}[p53_{tot}] - k_{d53}[p53UU] \tag{6.5.9}$$

$$\frac{d[p53U]}{dt} = k_f[Mdm2_{nuc}][p53] + k_r[p53UU] - [p53U](k_r + k_f[Mdm2_{nuc}]) - k_{d53'}[p53U] \tag{6.5.10}$$

$$\frac{d[p53UU]}{dt} = k_f[Mdm2_{nuc}][p53U] - k_r[p53UU] - [p53UU](k_{d53'} + k_{d53}) \tag{6.5.11}$$

$$\frac{d[Mdm2_{nuc}]}{dt} = V_{ratio}(k_i[Mdm2P_{cyt}] - k_0[Mdm2_{nuc}]) - k_{bif}[Mdm2_{nuc}] \tag{6.5.12}$$

$$\frac{d[Mdm2_{cyt}]}{dt} = k_{s2'} + \frac{k_{s2}[p53_{tot}]^3}{J_s^3 + [p53_{tot}]^3} - k_{d2'}[Mdm2_{cyt}] + k_{deph}[MMdm2P_{cyt}] + \\ - \frac{k_{ph}}{J_{ph} + [p53_{tot}]}[Mdm2_{cyt}] \tag{6.5.13}$$

$$\frac{MMdm2P_{cyt}}{dt} = \frac{k_{ph}}{J_{ph} + [p53_{tot}]}[Mdm2_{cyt}] - k_{deph}[Mdm2P_{cyt}] - k_i[MMdm2P_{cyt}] + \\ + k_0[Mdm2_{nuc}] - k_{d2'}[MMdm2P_{cyt}] \tag{6.5.14}$$

where

$$k_{bif} = k_{d2'} + \frac{[CC_{irr}]}{J_{dam} + [CC_{irr}]}k_{d2''}.$$

6.5.4 Simulations

The following simulations aim at showing how an irinotecan injection can damage a cell in a way that can be irreversible or not, depending on the irinotecan dose, on the moment when the medicament is administered (according to the topoisomerase phases), and on the reparation capability of the cell. In Figure 6.16 we can see that one only injection can kill a cell if the administration moment coincides with the topoisomerase phases. On the contrary, if the same dose is administered out of phase, it only creates reversible lesions (see Figure 6.17).

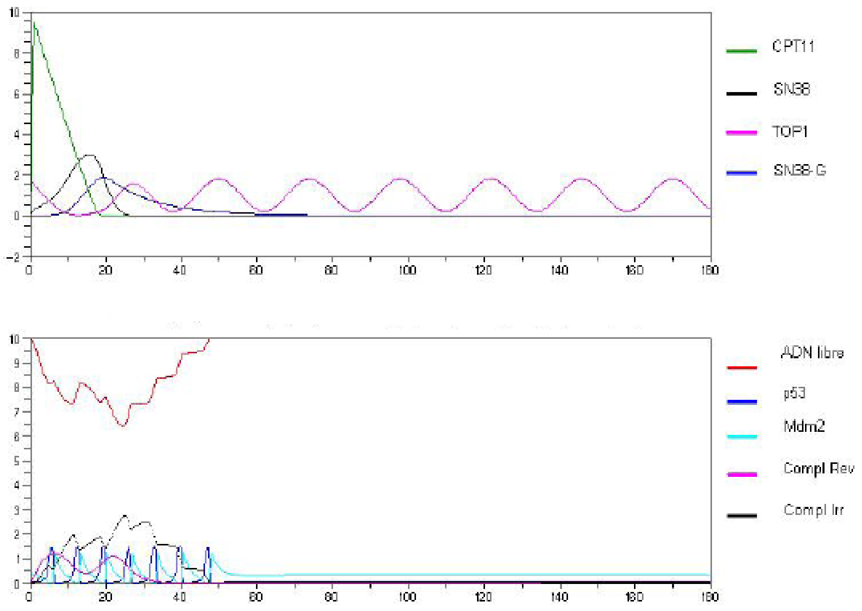


Figure 6.17: When injections are made in phase opposition, lesions are repaired.

In Figure 6.18 the parameter describing the progression of topoisomerase in the circadian rhythm has been changed: we can observe that lesions due to the same dose of irinotecan are repaired. Finally, in the last simulation (Figure 6.19) we can examine the behaviour of two cells with different reparation capability. In one of the two cells, the threshold below which p53 is not able to repair has been decreased. The black line represents the damage of the cell which is not susceptible of reparation: its level remains constant while time evolves.

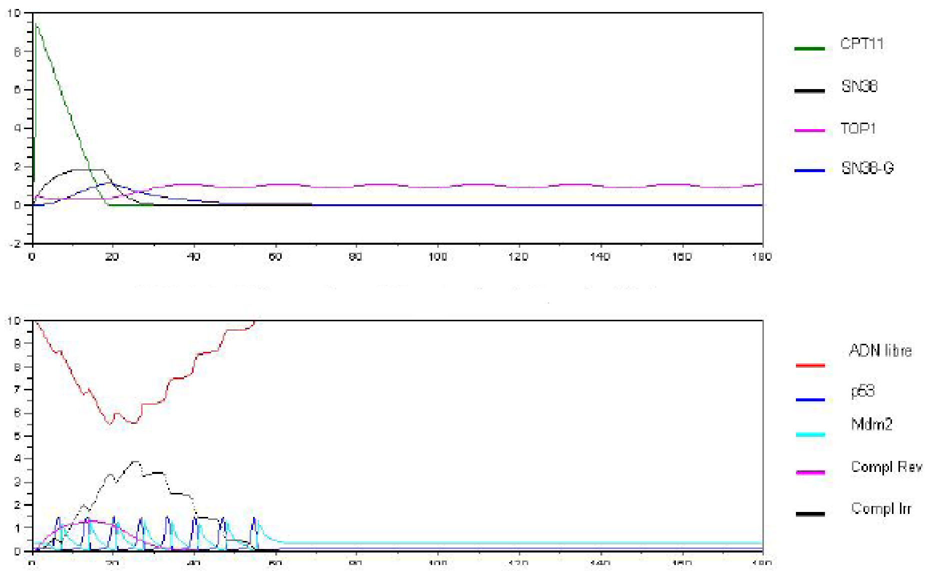


Figure 6.18: Without progression in the circadian rithm, even a strong drug dose does not kill the cell.

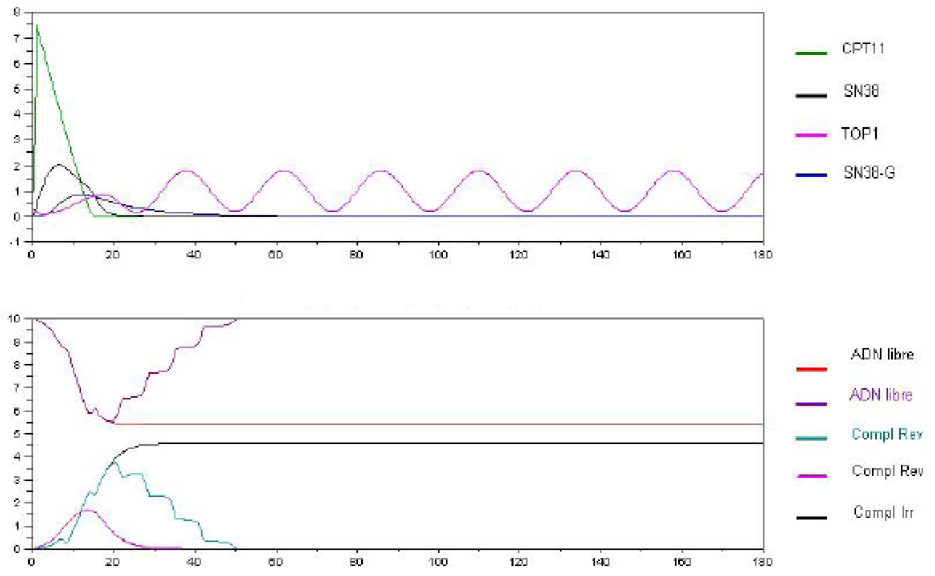


Figure 6.19: Different reparation capability of two cells: lesions are the same, but they can either be repaired or not.

7 Using Biocham to model and verify Irinotecan injections on a cell

Irinotecan shows significant antitumor activity against a variety of solid tumors, including lung, colorectal, and cervical cancers. However, the treatment dose and duration have been severely limited by serious side effects such as granulocytopenia and diarrhea. The maximization of the antitumor effects and the minimization of the toxicity of the drug to normal tissues are very important in cancer chemotherapy, because antitumor drugs call kill normal cells as well as tumor cells. Thus, it is fundamental to study how the medicament acts when injected in normal cells.

In order to understand how irinotecan influences cellular proliferation, we decided to take advantage of the Biochemical Abstract Machine (*BIOCHAM*), a formal modeling environment for network biology which offers automated reasoning tools for querying the temporal properties of a biological system. In particular, our approach consisted in encoding suitable models of mammalian cell cycle, p53/Mdm2, and irinotecan in the rule-based language of Biocham, assembling them into a consensus model, and verifying some biological properties of the resulting model through model checking.

The chapter is organized as follows. In Section 7.1 we will introduce the syntax of Biocham reaction rules, the different semantics that can be associated to such rules, and the temporal logics that can be used to formalize biological properties of a reaction model. In Section 7.2 we will describe how we encoded in Biocham the set of differential equations proposed by Novák and Tyson in [102] to model the mammalian cell cycle and we will focus on the strategy we adopted to couple the three models (cell cycle, p53/Mdm2, and irinotecan, respectively) together. In Section 7.3 we will report some biological properties of the resulting model that we expressed in Constraint-LTL and we will concentrate on model checking results. In Section 7.4 we will conclude by briefly discussing how our approach can be adapted to study the influence of irinotecan on groups of tumor cells.

7.1 Biocham

In the last years, one of the main challenges of computational systems biology became the creation of powerful simulation, analysis, and reasoning tools for biologists to decipher existing data, devise new experiments, and thus understand functional properties of genomes, proteomes, cells, organs, and organisms. Many of the goals of this emerging discipline are achieved in the Biochemical Abstract Machine (*BIOCHAM*) [46, 19], a formal modeling environment for network biology developed at INRIA-Rocquencourt in 2003. Biocham allows one to model biochemical systems, to make simulations, and to check temporal properties. In the same years many other powerful instruments have been created in this context. A fine example is *Simpathica* [4], a computational tool which provides an automaton-based semantics of the temporal evolution of biochemical reactions starting from a representation given as a set of differential equations.

Each tool for network biology must provide a formalism to represent interactions between molecular species and their regulations. A pragmatical example is the Systems Biology Markup

Language (SBML) [69] (see Section 6.1), which uses a syntax of reaction rules with kinetic expressions to define *reaction models*. SBML is nowadays supported by a majority of modeling tools [67, 121]. In this section we explain how to denote objects in a cell, their interaction and transport using the rule-based syntax of Biocham, which is similar to (and compatible with) SBML. Then we present three different semantics for Biocham models, that correspond to three different abstraction levels. Finally, we show how temporal logics can be used to formalize the biological properties of the resulting models and how to check automatically their satisfaction.

7.1.1 Modeling biochemical processes in Biocham

Following SBML and Biocham conventions, a model of a biochemical system is formally a set of reaction rules of the form e `for` $S \Rightarrow S'$, where S is a *solution*, that is, a set of molecules given with their stoichiometric coefficient, S' is the transformed solution, and e is a kinetic expression involving the concentrations of molecules. The reaction rules represent biomolecular interactions between chemical or biochemical compounds, ranging from small molecules to proteins and genes.

The syntax of the formal objects involved and of their reactions is given by the following (simplified) grammar:

```
object = molecule|molecule::location
molecule=name|molecule-molecule|molecule~{name, ...,name}

reaction =solution=>solution|kinetics for solution=>solution
solution = _|object|number*object|solution+solution
```

The basic object is a molecular compound. Thanks to the `::` operator, it can be given a precise location, which is simply a name denoting a (fixed) compartment, such as the nucleus, the cytoplasm, the membrane, etc. The binding operator `-` is used to represent complexations and other forms of intermolecular bindings. The alteration operator `~` makes it possible to attach to a compound a set of modifications, such as the set of phosphorylated sites of a protein. For instance, `A~{p}` denotes a phosphorylated form of the compound `A`, and `A~{p}-B` denotes its complexation with `B`.

Reaction rules transform one formal solution into another one. The following abbreviations are used: `A =[C]=> B` for the catalyzed reaction `A+C => C+B`, and `A <=> B` for the reversible reaction equivalent to the two symmetrical reactions `A => B` and `B => A`. The constant _ represents the empty solution. It is used for instance in protein *degradation rules*, such us `A =>_` , and in *synthesis rules*, such us `_=[G]=> A` for the synthesis of `A` by (activated gene) catalyst `G`. The other main rule schemas are *(de)complexation rules*, such us `A + B => A-B` for the complexation of `A` and `B`, *(de)phosphorylation rules*, such us `A =[B]=> A~{p}` for the phosphorylation of `A` catalyzed by kinase `B`, and *transport rules*, such us `A::nucleus => A::cytoplasm` for the transport of `A` from the nucleus to the cytoplasm.

Reactions can be given kinetic expressions. For instance, `k*[A]*[B] for A=[B]=>A~{p}` specifies a mass action law kinetics with parameter k for the reaction. Classical kinetics expressions are the following ones:

- the **mass action low kinetics** $k * \prod_{i=1}^{n} x_i^{l_i}$, which refers to a reaction with n reactants x_i, where l_i is the stoichiometric coefficient of x_i as a reactant;
- the **Michaelis-Menten kinetics** $V_m * x_s/(K_m + x_s)$, which states for an enzymatic reaction of the form $x_s = [x_e] \Rightarrow x_p$, where[1] $V_m = k * (x_e + x_e * x_s/K_m)$;
- the **Hill's kinetics** $V_m * x_s^n/(K_m^n + x_s^n)$, of which the Michaelis-Menten kinetics is a special case for $n = 1$.

Kinetic expressions can be written either explicitly, allowing any kinetics, or using shortcuts such us `MA(k)` for a Mass Action law with parameter k, or `MM(Vm,Km)` for a Michaelian kinetics.

[1] $x_e * x_s/K_m$ is the concentration of the enzyme-substrate complex, supposed constant in the Michaelian approximation, and $x_e + x_e * x_s/K_m$ is thus the total amount of enzyme.

7.1.2 Semantics of the reaction model

A set of reaction rules $\{e_i$ `for` S_i => $S'_i\}_{i=1,\dots,n}$ over molecular concentration variables $\{x_1, \dots, x_m\}$ can be interpreted under different semantics. The traditional *differential semantics* interpret the rules by the following system of Ordinary Differential Equations (ODE):

$$dx_k/dt = \sum_{i=1}^{n} r_i(x_k) * e_i - \sum_{j=1}^{n} l_j(x_k) * e_j,$$

where $r_i(x_k)$ (resp., l_i) is the stoichiometric coefficient of x_k in the right (resp., left) member of rule i. Given a set of reaction rules, Biocham allows to obtain a simulation of the model by solving the set of corresponding differential equations.

Another interpretation of biochemical reaction models is provided by the *stochastic* semantics. In that semantics, a reaction model is interpreted as a (continuous time) Markov chain, and the kinetic expressions as transition rates. For a given volume V_k of the location where the compound x_k resides, a concentration C_k for x_k is translated into a molecule number $N_k = \lfloor C_k \times V_k \times N_A \rfloor$, where N_A is Avogadro' s number. A state in the stochastic semantics will be a vector of integers indicating the numbers of molecules for each species. Let us consider a reaction model $\{e_i$ `for` l_i => $r_i\}_{i \in I}$, and let us denote by $S \rightarrow_i S'$ the fact that rule i fires in state S resulting in state S', i.e., $S' = S - l_i + r_i$ (if Sl_i). In a given state S, the numbers of molecules are fixed integer values and the kinetic expression e_i evaluates into a (positive) real valued reaction rate, noted $e_i(S)$. If we denote by $\mathcal{S}$ the set of stochastic transitions, then we can associate to a reaction model $\{e_i$ `for` l_i => $r_i\}_{i \in I}$ the stochastic transition model $\{(S, S', e_i(S)) \in \mathcal{S} \mid i \in I, S \rightarrow_i S'\}$.

Finally, the *boolean semantics* associates to a reaction rule model a Kripke structure where the set of states is the set of all tuples of Boolean values denoting the presence or absence of the different biochemical compounds, the transition relation is the union (i.e., disjunction) of the relations associated to the reaction rules, and the labeling function simply associates to a given state the set of biochemical compounds that are present in the state. Reaction rules are asynchronous in the sense that one reaction rule is fired at a time (interleaving semantics), hence the transition relation is the union of the relations associated to the reaction rules. Furthermore, the Boolean abstraction of enzymatic reactions associates several transitions to a single reaction rule, one for each case of possible consumption of the molecules in the left hand side of the rule. The originated Kripke structure is a non-deterministic transition system where the temporal evolution is modeled by the succession of transition steps.

7.1.3 Querying Biocham models in temporal logic

Temporal logics and model-checking algorithms have been proved to be useful to respectively express biological properties of complex biochemical systems and automatically verify their satisfaction in both qualitative and quantitative models, i.e., in continuous [18], stochastic [17], and boolean models [41]. This approach relies on a logical paradigm for systems biology that consists in making the following identifications [45]:

biological model = *transition system*
biological property = *temporal logic formula*
biological validation = *model-checking*

Having a formal language not only for describing models, (i.e., transition systems based on process calculi [110, 105], rules [41], Petri nets [56], etc.), but also for formalizing the biological properties of the system known from biological experiments under various conditions, opens a whole avenue of research for designing automated reasoning tools inspired from circuit and program verification to help the modeler.

The temporal logics CTL (*Computation Tree Logic*), LTL[2] (*Linear Time Logic*) and PLTL (*Probabilistic LTL*) with numerical constraints are used, respectively, in the boolean semantics, in the differential semantics, and in the stochastic semantics of reaction models.

[2]For a review on CTL and LTL, the reader can refer to Subsection 3.3.2.

As far as the use of CTL is concerned, Biocham provides an interface to the symbolic model checker NuSMV [26]. Examples of biological queries that can be expressed in CTL are the following ones.
About reachability: Is there a pathway for synthesizing a protein P, *EF(P)*?
About pathways: Can the cell reach a state s while passing by another state s_2, $EF(s_2 \wedge EF(s))$? Is state s_2 a necessary checkpoint for reaching state s, $\neg E((\neg s_2)Us)$? Can the cell reach a state s without violating certain constraints c, $E(cUs)$? Is it possible to synthesize a protein P without creating or using protein Q, $E(\neg QUP)$?
About stability: Is a certain (partially described) state s of the cell a steady state, $s => EG(s)$? A permanent state, $s => AG(s)$? Can the cell reach a given permanent state s, $EF(AGs)$? Must the cell reach a given permanent state s, $AF(AGs)$? Can the system exhibit a cyclic behaviour with respect to the presence of a product P, $EG((P => EF\neg P)(\neg P => EFP))$? The latter formula expresses that there exists a path where at all time points whenever P is present it becomes eventually absent, and whenever it is absent it becomes eventually present.

As far as the use of LTL is concerned, a version with constraints over the reals, called *Constraint-LTL*, is adopted in Biocham. Constraint-LTL considers first-order atomic formulae with equality, inequality and arithmetic operators ranging over real values of concentrations and of their derivatives. For instance, $F([A] > 10)$ expresses that the concentration of A eventually gets above the threshold value 10 and $G([A]+[B] < [C])$ states that the concentration of C is always greater than the sum of the concentrations of A and B. Oscillation properties, abbreviated as $oscil(M,K)$, are defined as a change of sign of the derivative of M at least K times: $F((d[M]/dt > 0)$ & $F((d[M]/dt < 0)$ & $F((d[M]/dt > 0)\ldots)))$. The abbreviated formula $oscil(M,K,V)$ adds the constraint that the maximum concentration of M must be above the threshold V in at least K oscillations.

Constraint-LTL formulae are interpreted in linear Kripke structures which represent either an experimental data time series or a simulation trace, both completed with loops on terminal states. Given the system of ordinary differential equations (ODE) corresponding to a reaction model, under the hypothesis that the initial state is completely defined, a discrete simulation trace is easily obtained by means of numerical integration methods (such as Runge-Kutta or Rosenbrock method for stiff systems). Since constraints refer not only to concentrations, but also to their derivatives, traces of the form

$$(< t_0, x_0, dx_0/dt, d^2x_0/dt^2 >, < t_1, x_1, dx_1/dt, d^2x_1/dt^2 >, \ldots)$$

are considered, where at each time point t_i, the trace associates the concentration values x_i to the variables, and the values of their first and second derivatives dx_i/dt and d^2x_i/dt^2. It is worth noting that in adaptive step size integration methods of ODE systems, the step size t_{i+1} - t_i is not constant and is determined through an estimation of the error made by the discretization.

Observe that the notion of *next state* refers to the state of the following time point in a discretized trace, and thus does not necessarily imply a real time neighborhood. The rationale is that the numerical trace contains enough relevant points, and in particular those where the derivatives change abruptly, to correctly evaluate temporal logic formulae. This has been very well verified in practice with various examples of published mathematical models [18]. An innovative feature of Biocham is that it places at the user's disposal a procedure `learn_parameters` for finding parameter values such that a given constraint-LTL formula is satisfied. This search procedure actually replicates and automates part of what the modeler currently does by hand: trying different parameter values, between bounds that are thought reasonable, or computed by other methods such as bifurcation diagrams, in order to obtain behaviors in accordance with the experimental knowledge. Biocham provides a way to explore much faster this parameter space, once the effort to formalize the expected behavior in constraint-LTL is done.

For the stochastic semantics, it is natural to consider the *PCTL* logic [62], which basically replaces the path operators of CTL, E and A, by the operator $P_{\bowtie p}$. This operator represents a constraint $\bowtie_p$ on the probability that the formula under $P_{\bowtie p}$ is true. For instance, $A(\psi U \psi')$ becomes $P_1(\psi U \psi')$, i.e., the probability that $\psi U \psi'$ is realized is 1. The atomic formulae considered here are first-order formulae with arithmetic constraints, ranging on integers representing numbers of molecules.

However, existing probabilistic model-checking tools, such us the one used in PRISM [81], do not well handle either highly non-deterministic examples or examples where variables have a large domain, which is the case of Biocham models' stochastic semantics. This led Biocham' authors to actually consider the PLTL fragment of PCTL formulae in which the $P_{\bowtie p}$ operator can only appear once as head of the formula. To evaluate the probability of realization of the underlying LTL formula, Biocham samples a certain number of stochastic simulations using standard algorithms like that of Gillespie [57] or of Gibson [55]. The outer probability is then estimated by counting. In principle, Monte-Carlo algorithm can be used for model-checking and kinetic parameter learning along the same lines as in the differential semantics. However, both the stochastic simulation process and the model-checking process are computationally more expensive than in the differential semantics by several orders of magnitude. For this reason, such search algorithms are currently not practical with the stochastic semantics.

7.2 A consensus model for Irinotecan, protein p53, and mammalian cell cycle

The first step of our approach to the investigation of the influence of irinotecan on the mammalian cell cycle consists in the encoding of the selected models of irinotecan, p53/Mdm2, and mammalian cell cycle in the Biocham rule-based language. Biocham source codes for the three models are reported in Appendix 2. As a matter of fact, we only concentrated on the cell cycle model, because the other two models had previously been encoded in Biocham.

7.2.1 Modeling the mammalian cell cycle in Biocham

Looking in parallel at the set of phenomenological differential equations and at the process diagram prosed by Novák and Tyson to model the mammalian cell cycle (cf. Tables 6.1 and 6.2 and Figures 6.7 and 6.8, respectively), we aimed at obtaining a set of Biocham reaction rules.

It is worth noting that in general it is not possible to map an ODE to a sole set of Biocham rules without looking at the diagrammatic representation. In fact, for some differential equations more than one translation is possible. As an example, let us consider the two equations (1) $dA/dt = (k_2 - k_1)[A]$ and (2) $dB/dt = k_1[A]$ and let us suppose that $k_1 = k_2$, which makes equation (1) zeroize. In this case, at least the two following mappings are possible:
Mapping 1
```
k1*[A] for A=>B.
k2* [A] for _=[A]=>A.
```
Mapping 2
```
k1*[A] for _=[A]=>B.
```
Another complication is due to the fact that in some cases one single Biocham rule may be originated from several differential equations. For instance, in our mapping we obtained the rule
```
(MA(k25),MA(k25r)) for CycE+Kip1<=>CycE-Kip1
```
from the red terms in the following differential equations (that is, equations (6.2.5), (6.2.6), and (6.2.9) of Table 6.1):

$$\frac{d[cycE]}{dt} = \varepsilon(k_7' + k_7[E2F_A]) - V_8[CycE] - k_{25}[CycE][Kip1] + k_{25r}[CycE : Kip1] + V_6[CycE : Kip1]$$

$$\frac{d[cycE : Kip1]}{dt} = k_{25}[CycE][Kip1] - k_{25r}[CycE : Kip1] - V_6[CycE : Kip1] - V_8[CycE : Kip1]$$

$$\frac{d[Kip1]}{dt} = \varepsilon k_5 - V_6[Kip1] - k_{24}[CycD][Kip1] + k_{24r}[CycD : Kip1] + k_{10}[CycD : Kip1] + \\ -k_{25}[Kip1]([CycE] + [CycA]) + k_{25r}([CycE : Kip1] + [CycA : Kip1]) + V_8[CycE : Kip1] + \\ +k_{30}[Cdc20][CycA : Kip1].$$

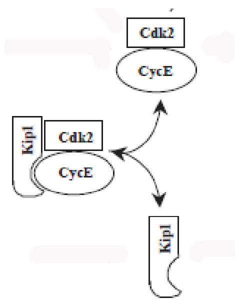


Figure 7.1: Fragment of molecular network relative to CycE and Kip1.

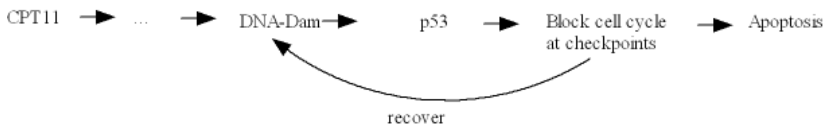


Figure 7.2: Expected behaviour of the consensus model.

In this case, the fragment of molecular network present in Figure 7.1 helped us to disambiguate the interpretation of the equations, and thus to choose the mapping that better fits the model.

It is possible to observe that, even if a mapping algorithm from ODE to Biocham rules can not be given, the following two rules are valid: *positive* terms originate (catalyzed) *synthesis* rules while *negative* terms give rise to (catalyzed) *degradation* rules. As an example, we provide the translation of the following equation (that is, equation (6.2.2) of Table 6.1):

$$\frac{d[DRG]}{dt} = \varepsilon(k'_{17}[ERG] + \frac{k_{17}([DRG]/J_{17})^2}{1+([DRG]/J_{17})^2}) - k_{18}[DRG],$$

which turns out to be this one:

```
MA(epsilon*k17p) for _=[ERG]=>DRG.
epsilon*k17*([DRG]/J17)^ 2/(1+([DRG]/J17)^ 2) for _=[DRG]=>DRG.
MA(k18) for DRG=>_.
```

Biocham puts at the user's disposal an instrument that allows one to check the soundness of the translation: by editing the command `show_kinetics` it is possible to obtain the set of ordinary differential equations corresponding to a set of Biocham rules. In fact, the passage from Biocham rules to ODEs is completely automatic. In our case, after encoding the cell cycle model into a set of Biocham rules, we took advantage of this method to verify the correctness of our mapping.

7.2.2 The complete model

Once the three models of irinotecan, p53/Mdm2, and mammalian cell cycle have been encoded in the Biocham rule-based language, we proceeded by assembling them into a consensus model. The expected behaviour of the resulting model is graphically depicted in Figure 7.2 and can be described as follows. Injections of irinotecan (CPT11) induce DNA Damage. In reply to this, the cell reacts by activating protein p53, which blocks the cell cycle at checkpoints. This arrest is thought to repair critical damage before DNA replications occurs, thereby avoiding the propagation of genetic lesions to progeny cells. Thus, while the cell cycle is arrested, protein p53 will try to recover the damage. If it is possible, the cell cycle will be restarted; otherwise, if the damage is too extensive, the cell will undergo apoptosis.

The strategy we chose to draw the three models together is illustrated in Figure 7.3. As remarked in the previous chapter, in literature we found evidence of the fact that, if irinotecan is injected in a cell during phase S of the cell cycle, then more DNA damage will be caused with respect

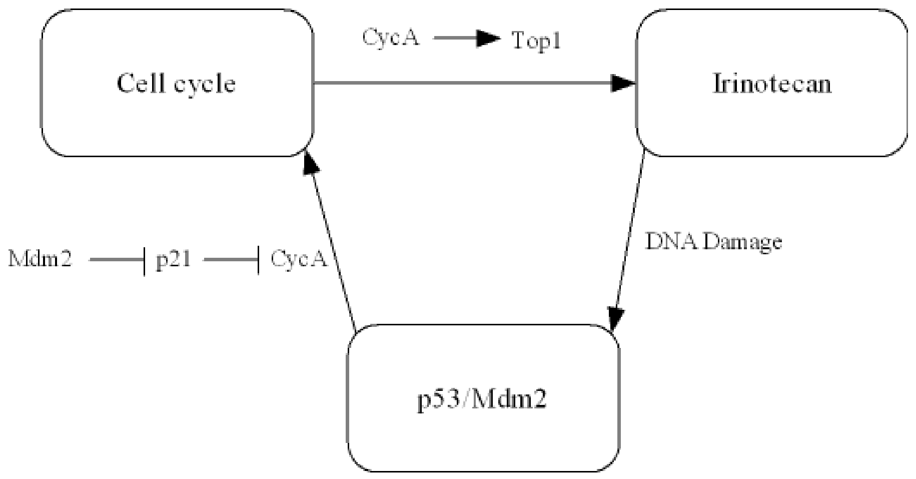


Figure 7.3: Our consensus model.

to the other phases of the cell cycle [116, 129]. Keeping this fact in mind, and remembering that the dimer characterizing phase S is CycA/Cdk2 (CycA for short), we inserted in cell cycle model a rule stating that a high concentration of CycA determines a high concentration of Topoisomerase I (Top1 for short), enzyme that, as previously explained, contributes to cause DNA single-strand breaks in presence of irinotecan. In this way we linked the cell cycle model to the irinotecan one. The link between the irinotecan model and the p53/Mdm2 one is given by DNA damage. In fact, irinotecan injections cause DNA damage, which in turn triggers the activity of protein p53. Finally, in order to link the p53/Mdm2 and cell cycle models, we inserted in the p53/Mdm2 model a rule which fixes that Mdm2 inhibits p21, and a further rule which imposes that p21 inhibits CycA. Roughly speaking, this means that a high concentration of protein p53 determines a low concentration of protein CycA, and thus a block of the cell cycle before phase S.

7.3 Model checking properties of the resulting model

Once the new “linking rules” have been embodied to the models, we proceeded by simulating them. As a matter of fact, at first we simulated the irinotecan and cell cycle models, without the p53/Mdm2 one. Then we concentrated on cell cycle and p53/Mdm2, without irinotecan. Finally, we simulated the three models together. In this last case, we obtained different simulations by varying the lapse of time between consecutive irinotecan injections. All the simulations we made were up to 100 hours (in practise, they lasted approximately one minute). In Figure 7.4 there is a simulation of the resulting model, where one injection every 36 hours is performed.

The plot puts in evidence how DNA damage increases after every injection. The oscillating trend of proteins p53 and Mdm2 is well highlighted. Furthermore, it is possible to notice the irregular behaviour assumed by CycA after irinotecan injections and the dependence of Top1 from CycA.

After having obtained numerical simulations for the models, we started to query them by means of constraint-LTL formulae to study their properties. In the following, we provide some examples of biological properties we were able to check: for each property, first of all we express it through the natural language, then we formalize it in constraint-LTL, and finally we present the results of the corresponding model-checking query.

F1: After an irinotecan injection is performed, is DNA damage able to go under the threshold of 0.1 before the next injection is done?

Constraint-LTL: $G(([CPT11] > 9.45) \vee (([CPT11] \leq 9.45) U ([DNAdam] < 0.1)))$.

Results: Before testing the property, we decided to parameterize the lapse of time between con-

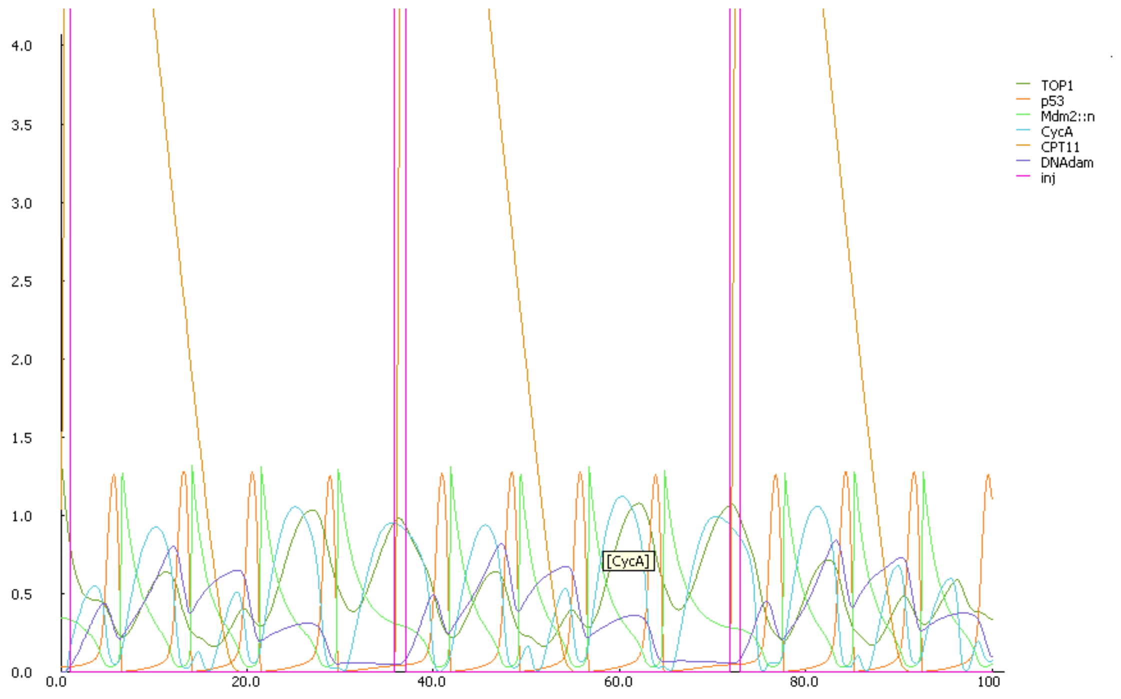


Figure 7.4: Zoomed simulation plot of the resulting model.

secutive irinotecan injections. Then we took advantage of procedure `learn_parameters` to find the minimum k such that, if one injection is performed every k hours, then property F1 is true. We found out that the minimum k multiple of 12 which makes F1 true is 36. Thus, one injection every 36 hours should be performed in order to allow DNA damage to be recovered before the next injection. Then we tried to see what it happens if, at each injection, we double the irinotecan dose. In this case, one injection every 48 hours should be done.

F2: Is is true that, whenever CycA is greater than 1, then Top1 is greater than a value k?
Constraint-LTL: $G([CycA] > 1 \rightarrow [TOP1] > k)$.
Results: In order to exactly know the link between the concentration values of CycA and Top1, and in particular the dependence of the latter one from the former one, we exploited the `learn_parameters` procedure to find out the maximum k such that F2 holds. This value turned out to be 0.55.

F3: Within a time interval of 100 time units, is CycA greater than 1.3 in at least 9 oscillations?
Constraint-LTL: $oscil([CycA], 9, 1.3)$.
Results: This property turned out to be true only in the simulation where the p53/Mdm2 model is not taken into account. In fact, as expected, even if DNA damage occurs, when protein p53 does not act, the cell cycle is not affected, and thus CycA exhibits a regular behaviour. When the p53/Mdm2 model is added, CycA oscillations are not any more regular, which means that the cell cycle is often stopped. If, for instance, we consider the simulation where one injection every 36 hours is performed, the following weaker property is satisfied: $oscil([CycA], 7, 0.9)$.

F4: Is DNA damage an increasing function?
Constraint-LTL: $G(d[DNAdam]/dt \geq 0)$.
Results: As matter of fact, this property only holds when the p53/Mdm2 model is not taken into account, thus DNA damage is not recovered and continuously increases until apoptosis is reached.

Such a model checking approach turned out to be very efficient: we had very fast execution times, that is, lower than one second in almost all cases. Furthermore, it proved to be effective, allowing as to extrapolate many relevant biological properties of the model (and concentration values that make specifications true) that could have not been easily detected by simply adopting simulation techniques.

7.4 Conclusions and future work

In this chapter we proposed a model checking approach to the investigation of the influence of an anti-tumor drug, namely irinotecan, on the cell cycle of mammalian individuals. After encoding suitable models of mammalian cell cycle, irinotecan, and a tumor-suppressor protein in the rule-based language of Biocham, we integrated these models into a consensus model and we exploited the model checking machinery to verify whether such a model satisfies some meaningful properties expressed as formulae of constraint-LTL. Furthermore, in some cases we took advantage of a parameter learning procedure to infer parameter values which make formulae true, extrapolating in such a way interesting features of irinotecan metabolic pathways.

In order to understand how irinotacan interferes with tumor cell proliferation, a further element should be taken into account, that is, the *circadian clock*, which regulates the synchronous progression of cells through each stage of the cell cycle, determining the daily time windows during which cells can traverse certain phases of the cell cycle [91]. The circadian synchronization of cell cycle progression, which characterizes healthy normal tissues, is often altered in cellular tissues affected by malignant tumors [98]. Roughly speaking, it means that at the same time different tumor cells could be in different phases of the cell cycle.

As a matter of fact, the behaviour of a single cell is not sensibly affected by malignant tumors, but the presence of a disease is generally detected by observing groups of at least one hundred cells. At the level of single cells, a slowing down of biochemical reactions is the only observed phenomenon in presence of tumor. An interesting extension of the work so far presented would consist in adding to our consensus model a new “circadian clock module” to synchronize cells and simulating the resulting model on a group of about one hundred cells. The three already present modules would remain almost unchanged, a part from a speed reduction of the reactions inserted in Figure 7.3 to link them.

Conclusions

In this theses we showed how some formal techniques of Computer Science, such as model checking, game theory, and the combination of simulation and model checking, can be successfully applied to find effective solutions for relevant biological problems. More precisely, we presented some novel approaches to three topical classes of issues: the protein folding and protein structure prediction problems, the sequence alignment/comparison problems, and the problems of modeling and checking metabolic pathways. In the following, we summarize the thesis content.

- In Chapter 1 we provided a smattering about fundamental biological notions. In particular, we introduced some basic concepts, such us the ones of DNA, gene, chromosome, and protein, and some relevant computational issues.
- In Chapter 2 we introduced the basic concepts regarding the chemical structure of proteins and the physical process leading to the formation of their native shape (protein folding). Furthermore, we focused on the most common simplified models of proteins and computational techniques that are used in the prediction of the structure of a protein (protein structure prediction).
- In Chapter 3 we showed how model checking techniques can be successfully exploited to verify meaningful properties of protein conformations and of their relationships to be used in solving the protein structure prediction and the protein folding problems. We modeled the space of protein conformations as a finite transition system whose states are all the possible conformations of a protein and whose transitions represent admissible transformations of conformations. Then, we showed how meaningful properties of such a transition system can be expressed in temporal logic. Finally, we used the model checking machinery to algorithmically check them. To cope with the combinatorial blow up of the state space, we resorted to on-the-fly model checking. We developed an on-the-fly model checker in SICStus Prolog and we experimented it on sample proteins and properties, obtaining quite promising results.
- In Chapter 4 we dealt with the problem of deciding whether a pair/group of sequences is evolutionary related or not. We examined traditional sequence alignment and comparison algorithms which use dynamic programming to find optimal alignments. The basic mutational processes taken into account are substitutions, insertions, and deletions.
- In Chapter 5 we showed how Ehrenfeucht-Fraïssé games can be successfully exploited to compare (finite) sequences. More precisely, we gave necessary and sufficient conditions for Spoiler/Duplicator to win games played on finite structures with a limited order relation, that lies in between the successor relation and the usual (linear) order relation, and a finite number of unary predicates. On the basis of such conditions, we outlined a polynomial (in the size of the input strings) algorithm to compute the "remoteness" of a game and to determine the optimal strategies/moves for both players. The remoteness of a game can be used to measure the degree of similarity between two structures, thus providing a natural approach to compare biological strings.
- In Chapter 6, after introducing the most commonly used representation systems for biological networks, we described the mammalian cell cycle, the action of the tumor-suppressor protein p53, and the behaviour of irinotecan, an anti-cancerogenic medicament. Furthermore, we focused on a model of the joint behaviour of protein p53 and irinotecan recently proposed in French laboratories.
- In Chapter 7 we presented our approach to the investigation of the influence of irinotecan on cellular proliferation, which consisted in encoding suitable models of mammalian cell

cycle, protein p53, and irinotecan in the the rule-based language of the Biochemical Abstract Machine (BIOCHAM), assembling them into a consensus model, formalizing some biological properties of the resulting model through temporal logic (Constraint-LTL), and automatically checking their satisfaction thanks to the use of model-checking. We concluded the chapter by mentioning a possible extension of our work that allows to study the influence of irinotecan on groups of tumor cells.

For each one on the three biological problems we dealt with, the use of formal methods of Computer Science turned out to be beneficial. As for the protein folding/protein structure prediction problem, our model-checking approach revealed to be an effective tool to extrapolate interesting properties of protein conformations that can be exploited to improve the solution search in existing constraint-based protein folding algorithms. In general, constraints allow one to easily model minimization problems. Once the constraint model is defined, a constraint solver can indeed be used to search for solutions. This search exploits the constraints to prune the solution space. The properties we detected can be encoded as additional constraints to be used to further reduce the solution space.

The use of model checking proved to be particularly suitable in the investigation of the irinotecan metabolic pathways too. In fact, it let us get a good comprehension of the drug influence on the mammalian cell cycle and infer some properties to be exploited in the drug therapy, such us optimal injection times and doses. The next step of this work will consist in studying the irinotecan action on groups of tumor cells and to compare results with the ones already obtained for healthy cells.

Finally, EF-games give us an effective method to compute the degree of similarity of strings. For instance, such a method can be used to compare evolutionary related biological strings or to compare genome assemblies of the same organism obtained by different programs. Even though a systematic comparison between our game-oriented approach and classical ones is missing, some intuitive examples show (see Section 5) that in many cases our technique is the most adequate one.

To conclude, the result of this work is twofold: on the one hand, it confirms the conviction that formal techniques of Computer Science are really promising to attack biological problems; on the other hand, it shows once again that this field is very ticklish, because the simplifications introduced to model biological entities or processes can sometimes lead to twist information at stake. Thus, the collaboration between biologists and computer scientists is absolutely necessary.

A

Appendix 1

In this appendix chapter we provide detailed proofs for lemmas and theorems of Chapter 5. Before coming to the proofs, we describe *ϑ-regions* and *Interval-regions*, which define local moves with respect to *ϑ-safety* and *p-int-safety*.

Definition A.0.1 *Let $q > 1$, $w \in \Sigma^\star$, and $\boldsymbol{i}^n$ be a set of positions in w. The ϑ-*region *$\vartheta reg_q(w, \boldsymbol{i}^n)$ is the set $\{j | \delta(0, j) \leq 2^{q-1} - 1 \vee \delta(j, |w| + 1) \leq 2^{q-1} - 1 \vee \exists r(1 \leq r \leq n \wedge \delta(i_r, j) \leq 2^{q-1} - 1)\}$.*

Definition A.0.2 *Let $q > 1$, $w \in \Sigma^\star$, and $\boldsymbol{i}^n$ be a set of positions in w. The interval-*region *$Intreg^p_q(w,\ \boldsymbol{i}^n)$ is the set $\{j | \exists 0 \leq k \leq 2^{q-2} s.t. \delta(0, j) \in [kp + 1, \dots, kp + \alpha^z_q]$, where $z = \lceil \log_2 k \rceil + 1$ for $k > 0$ and $z = 0$ if $k = 0 \vee \exists 0 \leq k \leq 2^{q-2}$ $s.t. \delta(j, |w| + 1) \in [kp + 1, \dots, kp + \alpha^z_q] \vee$ $\exists 1 \leq r \leq n\ \exists 0 \leq k \leq 2^{q-2}$ $s.t.$ $j \in \rho^+_{k,q}(i_r) \vee j \in \rho^-_{k,q}(i_r)$.*

Lemma 5.3.6 *Let $w, w' \in \Sigma^\star$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not pstep-*safe *in the $(p \cdot 2^q)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n),\ (w', \boldsymbol{j}^n)))$.*

Proof. The proof is by induction on q.
Base case: $q = 0$. $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not *pstep-safe* in the p-horizon and thus there exist r and s such that $pstep^{(p)}_p(i_r, i_s) \neq pstep^{(p)}_p(j_r, j_s)$. Then the configuration is not a partial isomorphism.

Inductive step: $q > 0$. We distinguish three cases:

1. $\exists r, s \in \{1, \dots, n\}$ such that $pstep^{(p)}_{p \cdot 2^q}(i_r, i_s) \neq pstep^{(p)}_{p \cdot 2^q}(j_r, j_s)$.

2. $\exists r \in \{1, \dots, n\}$ such that $pstep^{(p)}_{p \cdot (2^q - 1)}(0, i_r) \neq pstep^{(p)}_{p \cdot (2^q - 1)}(0, j_r)$.

3. $\exists r \in \{1, \dots, n\}$ such that $pstep^{(p)}_{p \cdot (2^q - 1)}(i_r, |w| + 1) \neq pstep^{(p)}_{p \cdot (2^q - 1)}(j_r, |w'| + 1)$.

We provide the details for the first case; the other ones are similar and thus omitted. Without loss of generality, let us suppose $\alpha = pstep^{(p)}_{p \cdot 2^q}(i_r, i_s) < pstep^{(p)}_{p \cdot 2^q}(j_r, j_s)$ and $i_r < i_s$. Then $\delta(i_r, i_s) \leq p \cdot 2^q$. **I** chooses $i_{n+1} = i_r + \lfloor \frac{\alpha}{2} \rfloor \cdot p$. It holds that (i) $\delta(i_r, i_{n+1}) \leq p \cdot 2^{q-1}$ (for the way i_{n+1} has been chosen) and (ii) $\delta(i_{n+1}, i_s) \leq p \cdot 2^{q-1}$. We prove that (ii) holds. If α is even, it trivially holds. If α is odd, then $\delta(i_r, i_s) \leq p \cdot (2^q - 1)$. Thus $\delta(i_{n+1}, i_s) \leq p \cdot (2^q - 1) - p \cdot \lfloor \frac{2^q - 1}{2} \rfloor = p(\lceil \frac{2^q - 1}{2} \rceil + \lfloor \frac{2^q - 1}{2} \rfloor) - p \cdot \lfloor \frac{2^q - 1}{2} \rfloor = p \cdot \lceil \frac{2^q - 1}{2} \rceil = p \cdot \lceil 2^{q-1} - \frac{1}{2} \rceil = p \cdot 2^{q-1}$. If **II** chooses a position j_{n+1} such that $pstep^{(p)}_{p \cdot 2^{q-1}}(i_r, i_{n+1}) \neq pstep^{(p)}_{p \cdot 2^{q-1}}(j_r, j_{n+1})$, then the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon, and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. If **II** chooses a position j_{n+1} such that $pstep^{(p)}_{p \cdot 2^{q-1}}(i_r, i_{n+1}) = pstep^{(p)}_{p \cdot 2^{q-1}}(j_r, j_{n+1}) = \lfloor \frac{\alpha}{2} \rfloor$, then $pstep^{(p)}_{p \cdot 2^{q-1}}(i_{n+1}, i_s) = \alpha - \lfloor \frac{\alpha}{2} \rfloor$ but $pstep^{(p)}_{p \cdot 2^{q-1}}(j_{n+1}, j_s) > \alpha - \lfloor \frac{\alpha}{2} \rfloor$. It follows that the configuration is not *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon, and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$. ■

Lemma 5.3.9 *Let $w, w' \in \Sigma^\star$ and $q > 0$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not ϑ-*safe *in the $(2^q - 1)$-horizon, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.*

Proof. The proof is by induction on q.
Base case: $q = 1$ and thus $2^q - 1 = 1$. We distinguish three cases:

1. $\exists r, s \in \{1, \dots, n\}$ such that $\vartheta_1(i_r, i_s) \neq \vartheta_1(j_r, j_s)$.
2. $\exists r \in \{1, \dots, n\}$ such that $\vartheta_1(0, i_r) \neq \vartheta_1(0, j_r)$.
3. $\exists r \in \{1, \dots, n\}$ such that $\vartheta_1(i_r, |w| + 1) \neq \vartheta_1(j_r, |w'| + 1)$.

We provide the details for the first case; the other ones are similar and thus omitted.
If $i_r = i_s$ but $j_r \neq j_s$ or viceversa, then the configuration is not a partial isomorphism. Otherwise, it holds that $\delta(i_r, i_s) = 1$ but $\delta(j_r, j_s) > 1$ (or viceversa). If **I** chooses a position between j_r and j_s in w', **II** looses because in w there are no positions between i_r and i_s.

Inductive step: $q > 1$. As in the base case, we distinguish three cases:

1. $\exists r, s \in \{1, \dots, n\}$ such that $\vartheta_{2^q-1}(i_r, i_s) \neq \vartheta_{2^q-1}(j_r, j_s)$.
2. $\exists r \in \{1, \dots, n\}$ such that $\vartheta_{2^q-1}(0, i_r) \neq \vartheta_{2^q-1}(0, j_r)$.
3. $\exists r \in \{1, \dots, n\}$ such that $\vartheta_{2^q-1}(i_r, |w| + 1) \neq \vartheta_{2^q-1}(j_r, |w'| + 1)$.

As in the base case, we provide the details for the first case; the other ones are similar and thus omitted.
If $i_r \leq i_s$ and $j_r > j_s$ or viceversa, it is easy to take back to the base case. So let us suppose $i_r - i_s$ and $j_r - j_s$ have the same sign. Furthermore, without loss of generality we suppose $i_r < i_s$ and $\vartheta_{2^q-1}(i_r, i_s) < \vartheta_{2^q-1}(j_r, j_s)$. Then $\delta(i_r, i_s) \leq 2^q - 1$.
We distinguish two subcases:
Subcase a) $\vartheta_{2^q-1}(j_r, j_s) = \infty$. It means that $\delta(j_r, j_s) \geq 2^q$. **I** chooses $j_{n+1} = \lfloor \frac{j_r + j_s}{2} \rfloor$. Then $\delta(j_r, j_{n+1}) \geq 2^{q-1}$ and $\delta(j_{n+1}, j_s) \geq 2^{q-1}$. Since $\delta(i_r, i_s) \leq 2^q - 1$, anyhow **II** chooses i_{n+1} it holds that $\delta(i_r, i_{n+1}) \leq 2^{q-1} - 1$ or $\delta(i_{n+1}, i_s) \leq 2^{q-1} - 1$. Then the configuration is not ϑ-*safe* in the $(2^{q-1} - 1)$-horizon, and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.
Subcase b) $\vartheta_{2^q-1}(j_r, j_s) < \infty$. It means that $\delta(j_r, j_s) \leq 2^q - 1$. **I** chooses $j_{n+1} = \lfloor \frac{j_r + j_s}{2} \rfloor$.

- If $\delta(j_r, j_s) < 2^q - 1$, then $\delta(j_r, j_{n+1}) \leq 2^{q-1} - 1$ and $\delta(j_{n+1}, j_s) \leq 2^{q-1} - 1$. Anyhow **II** chooses i_{n+1}, it holds that either $\delta(i_r, i_{n+1}) \leq 2^{q-1} - 1$ and $\delta(i_r, i_{n+1}) < \delta(j_r, j_{n+1})$ or $\delta(i_{n+1}, i_s) \leq 2^{q-1} - 1$ and $\delta(i_{n+1}, i_s) < \delta(j_{n+1}, j_s)$. Then the configuration is not ϑ-*safe* in the $(2^{q-1} - 1)$-horizon, and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.
- If $\delta(j_r, j_s) = 2^q - 1$, then $\delta(j_r, j_{n+1}) = 2^{q-1} - 1$ and $\delta(j_{n+1}, j_s) = 2^{q-1}$. Anyhow **II** chooses i_{n+1}, it holds that either $\delta(i_{n+1}, i_s) \leq 2^{q-1} - 1$ or $\delta(i_r, i_{n+1}) < \delta(j_r, j_{n+1})$. Then the configuration is not ϑ-*safe* in the $(2^{q-1} - 1)$-horizon, and thus $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+1}), (w', \mathbf{j}^{n+1})))$.

■

Theorem 5.3.18 *[Sufficient condition for **II**'s to win]*
*Let $w, w' \in \Sigma^\star$. If $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is q-distance-safe, then **II**$(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$.*

Proof. We prove that, if $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is q-*distance*-safe, i.e., it is *pstep-safe* in the $(p \cdot 2^q)$-horizon and (if $q > 0$) *p-int-safe* in the q-horizon, and if **I** can only make local moves, that is, play in $\mathrm{Pstepreg}_q^p(w, \mathbf{i}^n)$ or $\mathrm{Pstepreg}_q^p(w', \mathbf{j}^n)$, then **II** is able to reply so that $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is $(q-1)$-*distance*-safe.
In the following, we denote the region $\mathrm{Intreg}_q^p(w, \mathbf{i}^n) \backslash\ \vartheta reg_q(w, \mathbf{i}^n)$ by $\mathrm{Int}\backslash\vartheta reg_q^p(w, \mathbf{i}^n)$ and the region $\mathrm{Pstepreg}_q^p(w, \mathbf{i}^n) \backslash\ \mathrm{Intreg}_q(w, \mathbf{i}^n)$ by $Pstep\backslash Intreg_q^p(w, \mathbf{i}^n)$. In order to keep the proof not too complicated, we do not consider the case in which **I** plays in a prefix or suffix of w or w'.
Let us suppose **I** chooses $i_{n+1} \in \mathrm{Pstepreg}_q^p(w, \mathbf{i}^n)$. Let $i_{\beta_1}, \dots i_{\beta_b} \in \mathbf{i}^n$ be the (only) positions such that $i_{n+1} \in Int\backslash\vartheta reg_q^p(w, i_{\beta_1}, \dots, i_{\beta_b})$. Let $i_{\alpha_1}, \dots i_{\alpha_a} \in \mathbf{i}^n$ be the (only) positions such that

$i_{n+1} \in Pstep \backslash$ Intreg$_q^p(w, i_{\alpha_1}, \dots, i_{\alpha_a})$. Let $i_z \in \mathbf{i}^n$ be the nearest position to i_{n+1}. Let us suppose, without loss of generality, that $i_z < i_{n+1}$. We can distinguish two cases:
Case a) $\delta_{2^{q-1}-1}(i_z, i_{n+1}) = z < \infty$. In this case **II** must reply with $j_{n+1} = j_z + z$. It is easy to see that $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is *ϑ-safe* in the $(q-1)$-horizon. Now we prove $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is *p-int-safe* in the $(q-1)$-horizon. We show that, $\forall c \in \{\beta_1, \dots, \beta_b\}$, $i_{n+1} - i_c = j_{n+1} - j_c$. Let us suppose i_c is such that $i_z < i_{n+1} < i_c$. We know $i_{n+1} \in Int \backslash \vartheta reg_q^p(i_c)$, thus $\exists 1 \le k \le 2^{q-2}$ such that $\delta(i_{n+1}, i_c) \in (kp - \alpha_q^s, \dots, kp + \alpha_q^s]$, where $s = \lceil \log_2 k \rceil + 1$. The most interesting case is the one in which $i_c = i_{n+1} + kp + x$, with $0 < x \le \alpha_q^s$. It holds that $\delta(i_z, i_c) = \delta(i_z, i_{n+1}) + \delta(i_{n+1}, i_c) \le 2^{q-1} - 1 + kp + \alpha_q^s = kp + \alpha_{q+1}^s$. Then $i_c \in Int \backslash \vartheta reg_{q+1}^p(i_z)$, and thus (since the configuration was *p-int-safe* in the q-horizon), $\delta(i_z, i_c) = \delta(j_z, j_c)$. Furthermore, $\delta(i_z, i_{n+1}) = \delta(j_z, j_{n+1})$. It follows that $\delta(i_{n+1}, i_c) = \delta(j_{n+1}, j_c)$, and thus $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is *p-int-safe* in the $(q-1)$-horizon.
Now we prove $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon. Observe that, $\forall d \in \{\alpha_1, \dots, \alpha_a\}$, $x_d = pstep_{p \cdot 2^{q-1}}^{(p)}(i_{n+1}, i_d) < \infty$ (that is, $i_d \in [i_{n+1} + (x_d - 1) \cdot p + 1, \dots, i_{n+1} + x_d \cdot p]$). We must show that $pstep_{p \cdot 2^{q-1}}^{(p)}(j_{n+1}, j_d) = x_d$. Let us suppose i_d is such that $i_z < i_{n+1} < i_d$. If $\delta(i_z, i_d) = \delta(j_z, j_d)$, then $\delta(i_{n+1}, i_d) = \delta(j_{n+1}, j_d)$ and we concluded. So let $\delta(i_z, i_d) \ne \delta(j_z, j_d)$. Since $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is *q-distance*-safe, it holds that $pstep_{p \cdot 2^q}^{(p)}(i_z, i_d) = pstep_{p \cdot 2^q}^{(p)}(j_z, j_d) = k$, $(k-1) \cdot p + \alpha < \delta(i_z, i_d) < kp - \alpha$, and $(k-1) \cdot p + \alpha < \delta(j_z, j_d) < kp - \alpha$, with $\alpha > 2^{q-1} - 1$ (observe that $k = x_d$). Thus $\delta(i_{n+1}, i_d) = (k-1) \cdot p + r_1 - 2^{q-1} + 1 > (k-1) \cdot p$ and $\delta(j_{n+1}, j_d) = (k-1) \cdot p + r_2 - 2^{q-1} + 1 > (k-1) \cdot p$, with $r_1, r_2 \ge \alpha$. Then $pstep_{p \cdot 2^{q-1}}^{(p)}(j_{n+1}, j_d) = k = x_d$.
Case b) $\delta_{2^{q-1}-1}(i_z, i_{n+1}) = \infty$. It means that $\delta(i_z, i_{n+1}) > 2^{q-1} - 1$. We must distinguish whether $\{i_{\beta_1}, \dots, i_{\beta_b}\} = \emptyset$ or not.
Subcase b1) $\{i_{\beta_1}, \dots, i_{\beta_b}\} \ne \emptyset$. Let $e \in \{\beta_1, \dots, \beta_b\}$ be such that $\delta(i_{n+1}, i_e) = s$ is the minimum distance among i_{n+1} and the positions of the set $\{i_{\beta_1}, \dots, i_{\beta_b}\}$. Furthermore, let us suppose $i_e < i_{n+1}$. **II** must reply with $j_{n+1} = j_e + s$. We need to show that $(w, w', \mathbf{i}^{n+1}, \mathbf{j}^{n+1})$ is *p-int-safe* in the $(q-1)$-horizon and *pstep-safe* in the $(p \cdot 2^{q-1})$-horizon. We show *p-int-safety* (notice that ϑ-safety is trivially satisfied). Let $i_c \in \{\beta_1, \dots, \beta_b\}$. Furthermore, let $i_e < i_{n+1} < i_c$ and let $g = \delta(i_{n+1}, i_c)$. We want to show that $\delta(j_{n+1}, j_c) = g$. It holds that $\delta(i_{n+1}, i_e) \in (kp - \alpha_q^s, \dots, kp + \alpha_q^s]$, where $s = \lceil \log_2 k \rceil + 1$ and $k \le 2^{q-2}$. The most interesting case is the one in which $\delta(i_{n+1}, i_e) = kp + y_1$, with $y_1 \le \alpha_q^s$. Furthermore, it holds that $\delta(i_{n+1}, i_c) \in (hp - \alpha_q^{s'}, \dots, hp + \alpha_q^{s'}]$, where $s' = \lceil \log_2 h \rceil + 1$ and $h \le 2^{q-2}$. The most interesting case is the one in which $\delta(i_{n+1}, i_c) = hp + y_2$, with $y_2 \le \alpha_q^{s'}$. So $\delta(i_e, i_c) \le (k+h)p + \alpha_q^s + \alpha_q^{s'} \le (k+h)p + \alpha_{q+1}^{\max\{s,s'\}}$, with $k + h \le 2^{q-1}$. Then $i_c \in$ Int$\backslash \vartheta reg_{q+1}^p(w, i_e)$, and thus $\delta(i_e, i_c) = \delta(j_e, j_c)$. It follows that $\delta(j_{n+1}, j_c) = \delta(i_{n+1}, i_c) = g$.
Subcase b2) $\{i_{\beta_1}, \dots, i_{\beta_b}\} = \emptyset$. Given $\{i_{\alpha_1}, \dots, i_{\alpha_a}\}$, $\forall i \in \{\alpha_1, \dots, \alpha_a\}$, let $p_i = pstep_{p \cdot 2^{q-1}}^{(p)}(i_{n+1}, i_i) < \infty$ and let $Y_i = \{j_i + (p_i - 1) \cdot p + 1, \dots, j_i + p_i \cdot p\}$. **II** must choose $j_{n+1} \in \bigcap_{i=1}^{\alpha} Y_i \cap \overline{Intreg_q^p}(w', \mathbf{j}^n)$, where $\overline{Intreg_q^p}(w', \mathbf{j}^n) = \{1, \dots, |w'|\} \backslash Intreg_q^p(w', \mathbf{j}^n)$. ■

Theorem 5.4.5 *[Necessary condition for **II** to win]*
Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p > 1$. If $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is not $<_p$-safe for q-colors, then $\boldsymbol{I}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

Proof. The proof is by induction on q.
Base case:

- $q = 0$. $\exists r \in \{1, \dots n\}$ such that 0-col$_w(i_r) \ne$ 0-col$_{w'}(j_r)$. Then $w[i_r] \ne w'[j_r]$, and thus the current configuration is not a partial isomorphism.

- $q = 1$. $\exists r \in \{1, \dots n\}$ such that 1-col$_w(i_r) \ne$ 1-col$_{w'}(j_r)$. It means that $\sigma_1^w w[i_r] \tau_1^w \ne \sigma_1^{w'} w'[j_r] \tau_1^{w'}$, where $\tau_1^w =$ 0-col-int-right$_w^1(i_r + 1, i_r + p)$, $\tau_1^{w'} =$ 0-col-int-right$_{w'}^1(j_r + 1, j_r + p)$, $\sigma_1^w =$ 0-col-int-left$_w^1(i_r - p, i_r - 1)$, and $\sigma_1^{w'} =$ 0-col-int-left$_{w'}^1(j_r - p, j_r - 1)$.
  Without loss of generality, let us consider two cases:
  1) $w[i_r] \ne w'[j_r]$. In this case the current configuration is not a partial isomorphism;
  2) $\tau_1^w \ne \tau_1^{w'}$, where $\tau_1^w = \{w[i_r + 1], \dots, w[i_r + p]\}$ and $\tau_1^{w'} = \{w'[j_r + 1], \dots, w'[j_r + p]\}$. Then $\exists k \in [1, \dots, p]$ such that $\forall j \in [1, \dots, p]$ $w[i_r + k] \ne w'[j_r + j]$. **I** chooses $i_{n+1} = i_r + k$

(so $i_r <_p i_{n+1}$). In order not to violate predicates, **II** must choose j_{n+1} out of interval $[j_r+1,\dots,j_r+p]$. So $i_r <_p i_{n+1}$ but $j_r \not<_p j_{n+1}$, and thus the resulting configuration is not a partial isomorphism.

Inductive step: $q > 1$. There are three cases:

1. $\exists r \in \{1,\dots,n\}$ such that $q\text{-col}_w(i_r) \neq q\text{-col}_{w'}(j_r)$;
2. $\exists 1 \leq j \leq 2^{q-1}-1$ such that $(q-1)\text{-col-int-pref}_w^j(1+(j-1)\cdot p, jp) \neq (q-1)\text{-col-int-pref}_{w'}^j(1+(j-1)\cdot p, jp)$;
3. $\exists 1 \leq j \leq 2^{q-1}-1$ such that $(q-1)\text{-col-int-suff}_w^j(|w|-j\cdot p+1, |w|-(j-1)\cdot p) \neq (q-1)\text{-col-int-suff}_{w'}^j(|w'|-jp+1, |w'|-(j-1)\cdot p)$.

We provide the details for the first case; the other ones are similar and thus omitted. Without loss of generality, we consider the following subcases:
i) $w[i_r] \neq w'[j_r]$. In this case the current configuration is not a partial isomorphism.
ii) $\exists k \in \{1,\dots,2^{q-1}-1\}$ such that $t^w_{i_r+k} \neq t^{w'}_{j_r+k}$, where $t^w_{i_r+k} = (q-1)\text{-col}_w(i_r+k)$ and $t^{w'}_{j_r+k} = (q-1)\text{-col}_{w'}(j_r+k)$. **I** chooses $i_{n+1} = i_r+k$. If **II** replies with $j_{n+1} = j_r+k$, since $(q-1)\text{-col}_w(i_r+k) \neq (q-1)\text{-col}_{w'}(j_r+k)$, for inductive hypothesis $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. If **II** replies with $j_{n+1} \neq j_r+k$, then $\vartheta_{2^{q-1}-1}(i_r,i_{n+1}) \neq \vartheta_{2^{q-1}-1}(j_r,j_{n+1})$, and thus the configuration is not *ϑ-safe* in the $(2^{q-1}-1)$-horizon. By Lemma 5.3.9 $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$.
iii) $\exists 1 < j \leq 2^{q-2}+1$, $\exists x \in \{1,\dots,\alpha_q^z\}$ such that $t^w_{i_r+(j-1)\cdot p+x} \neq t^{w'}_{j_r+(j-1)\cdot p+x}$, where $z = \lceil log_2(j-1)\rceil+1$. Moreover $t^w_{i_r+(j-1)\cdot p+x}=(q-1)\text{-col}_w(i_r+(j-1)\cdot p+x)$ and $t^{w'}_{j_r+(j-1)\cdot p+x}=(q-1)\text{-col}_{w'}(j_r+(j-1)\cdot p+x)$. **I** chooses $i_{n+1} = i_r+(j-1)\cdot p+x$. If **II** replies with $j_{n+1} = j_r+(j-1)\cdot p+x$, since (q-1)-$\text{col}_w(i_r+(j-1)\cdot p+x) \neq$ (q-1)-$\text{col}_{w'}(j_r+(j-1)\cdot p+x)$, for inductive hypothesis $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. If **II** replies with $j_{n+1} \neq j_r+(j-1)\cdot p+x$, then $\delta(i_r,i_{n+1}) \neq \delta(j_r,j_{n+1})$, and thus the configuration is not *p-int-safe* in the $(q-1)$-horizon. By Lemma 5.3.14 $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$.
iv) $\exists 1 \leq j \leq 2^{q-1}$, $\exists k \in \{\gamma_1+1,\dots,p-\gamma_2\}$ such that $\forall h \in \{\gamma_1+1,\dots,p-\gamma_2\}$ $t^w_{i_r+(j-1)\cdot p+k} \neq t^{w'}_{j_r+(j-1)\cdot p+k}$, where

$$\gamma_1 = \begin{cases} \alpha_q^z & \text{if } j-1 \leq 2^{q-2}, z = \lceil log_2(j-1)\rceil+1 \text{ if } j>1, \text{ and } z=0 \text{ if } j=1 \\ 0 & \text{if } j-1 > 2^{q-2}. \end{cases}$$

$$\gamma_2 = \begin{cases} \alpha_q^t & \text{if } j \leq 2^{q-2}, \text{ and } z = \lceil log_2 j\rceil+1, \\ 0 & \text{if } j > 2^{q-2}, \end{cases}$$

and $t^w_{i_r+(j-1)\cdot p+x}=(q-1)\text{-col}_w(i_r+(j-1)\cdot p+x)$ and $t^{w'}_{j_r+(j-1)\cdot p+x}=(q-1)\text{-col}_{w'}(j_r+(j-1)\cdot p+x)$. **I** chooses $i_{n+1} = i_r+(j-1)\cdot p+x$. If **II** replies with $j_{n+1} \in [j_r+(j-1)\cdot p+\gamma_1+1,\dots,j_r+(j-1)\cdot p+p-\gamma_2]$, since $(q-1)\text{-col}_w(i_{n+1}) \neq (q-1)\text{-col}_{w'}(j_{n+1})$, for inductive hypothesis $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. If **II** replies with $j_{n+1} \in [j_r+(j-1)\cdot p+1,\dots,j_r+(j-1)\cdot p+\gamma_1] \vee [j_r+(j-1)\cdot p+p-\gamma_2+1,\dots,j_r+j\cdot p]$, then either the configuration is not *p-int-safe* in the $(q-1)$-horizon (if $j \leq 2^{q-2}+1$) or $(q-1)\text{-col}_w(i_{n+1}) \neq (q-1)\text{-col}_{w'}(j_{n+1})$ (if $j > 2^{q-2}+1$). Then $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. If **II** does not choose j_{n+1} in interval $[j_r+(j-1)\cdot p+1, j_r+j\cdot p]$, then $pstep^{(p)}_{p\cdot 2^{q-1}}(i_r,i_{n+1}) \neq pstep^{(p)}_{p\cdot 2^{q-1}}(j_s,j_{n+1})$, and thus the configuration is not *pstep-safe* in the $(p\cdot 2^{q-1})$-horizon, then by Lemma 5.3.6 $\mathbf{I}(\mathcal{G}_{q-1}((w,\mathbf{i}^{n+1}),(w',\mathbf{j}^{n+1})))$. ■

Theorem 5.4.7 *[Sufficient condition for **II** to win]*
*Let $w, w' \in \Sigma^\star$, and $p, q \in \mathbb{N}$, with $p>1$. If $(w,w',\mathbf{i}^n,\mathbf{j}^n)$ is q-locally-safe, then **II**$(\mathcal{G}_q((w,\mathbf{i}^n),(w',\mathbf{j}^n)))$.*

Proof (Sketch of). We need to prove that, if $(w,w',\mathbf{i}^n,\mathbf{j}^n)$ is q-locally-safe, that is, q-distance-safe in the $(p\cdot 2^q)$ horizon and $<_p$-safe for q-colors, and if **I** can only make local moves, that is, play in $\text{Pstepreg}^p_q(w,\mathbf{i}^n)$ or in $\text{Pstepreg}^p_q(w',\mathbf{j}^n)$, then **II** is able to reply so that $(w,w',\mathbf{i}^{n+1},\mathbf{j}^{n+1})$ is $(q-1)$-distance-safe and $<_p$-safe for $(q-1)$-colors. In the following, we denote the region $\text{Intreg}^p_q(w,\mathbf{i}^n)\setminus \vartheta reg_q(w,\mathbf{i}^n)$ by $Int\setminus\vartheta reg^p_q(w,\mathbf{i}^n)$ and the region $\text{Pstepreg}^p_q(w,\mathbf{i}^n)\setminus \text{Intreg}_q(w,\mathbf{i}^n)$ by

$Pstep \backslash Intreg^p_q(w, \mathbf{i}^n)$. In order to keep the proof not too complicated, we do not consider the case in which **I** plays in a prefix or suffix of w or w'. Let us suppose **I** chooses $i_{n+1} \in \mathrm{Pstepreg}^p_q(w, \mathbf{i}^n)$. Let $i_{\beta_1}, \dots i_{\beta_b} \in \mathbf{i}^n$ be the (only) positions such that $i_{n+1} \in \mathrm{Int}\ \backslash \vartheta reg^p_q(w, i_{\beta_1}, \dots, i_{\beta_b})$. Let $i_{\alpha_1}, \dots i_{\alpha_a} \in \mathbf{i}^n$ be the (only) positions such that $i_{n+1} \in Pstep \backslash\ \mathrm{Intreg}^p_q(w, i_{\alpha_1}, \dots, i_{\alpha_a})$. Let $i_z \in \mathbf{i}^n$ be the nearest position to i_{n+1}. Let us suppose, without loss of generality, that $i_z < i_{n+1}$. We consider the following cases.
Case a) $\delta_{2^{q-1}-1}(i_z, i_{n+1}) = z < \infty$. In this case **II** must reply with $j_{n+1} = j_z + z$. Certainly $w[i_{n+1}] = w'[j_{n+1}]$ because for hypothesis $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is $<_p$-safe for q-colors, therefore in particular $w_{2^{q-1}-1}[i_z] = w'_{2^{q-1}-1}[j_z]$.
Case b)$\delta_{2^{q-1}-1}(i_z, i_{n+1}) = \infty$. It means that $\delta(i_z, i_{n+1}) > 2^{q-1} - 1$. We must distinguish whether $\{i_{\beta_1}, \dots, i_{\beta_b}\} = \emptyset$ or not.
Subcase b1) $\{i_{\beta_1}, \dots, i_{\beta_b}\} \neq \emptyset$. Let $e \in \{\beta_1, \dots, \beta_b\}$ be such that $\delta(i_e, i_{n+1}) = f$ is the minimum distance among i_{n+1} and the positions of the set $\{i_1, \dots, i_\beta\}$. Without loss of generality, let $i_e < i_{n+1}$ and let $f = kp + x$, where $1 \leq k \leq 2^{q-2}$ and $x \leq \alpha^s_q$, with $s = \lceil \log_2 k \rceil + 1$. **II** must reply with $j_{n+1} = j_e + f$. Certainly $w[i_{n+1}] = w'[j_{n+1}]$ because for hypothesis $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is $<_p$-safe for q-colors, therefore in particular $w[i_e + kp + 1, \dots, i_e + kp + \alpha^s_q] = w'[j_e + kp + 1, \dots, j_e + kp + \alpha^s_q]$.
Subcase b2) $\{i_{\beta_1}, \dots, i_{\beta_b}\} = \emptyset$. Given $\{i_{\alpha_1}, \dots, i_{\alpha_a}\}$, $\forall i \in \{\alpha_1, \dots, \alpha_a\}$, let $p_i = pstep^{(p)}_{p \cdot 2^{q-1}}(i_{n+1}, i_i) < \infty$ and let $Y_i = \{j_i + (p_i - 1) \cdot p + 1, \dots, j_i + p_i \cdot p\}$. **II** must choose $j_{n+1} \in \bigcap_{i=1}^{\alpha} Y_i \cap \overline{Intreg}^p_q(w', \mathbf{j}^n)$, where $\overline{Intreg}^p_q(w', \mathbf{j}^n) = \{1, \dots, |w'|\} \backslash Intreg^p_q\ (w', \mathbf{j}^n)$. ■

Theorem 5.5.5 *Let $q, p, n \in \mathbb{N}$, with $q, p > 0$, and let $(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ be a configuration. If there exists a $(q-1)$-color τ such that $\tau \in \Delta^{(w, \boldsymbol{i}^n)}_{(w', \boldsymbol{j}^n)}$, then* $\boldsymbol{I}(\mathcal{G}_{q + min\{\sigma^{(w, \boldsymbol{i}^n)}_{(q,p)}(\tau), \sigma^{(w', \boldsymbol{j}^n)}_{(q,p)}(\tau)\}}((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$.

Proof. Without loss of generality, we can assume $k = \sigma^{(w, \mathbf{i}^n)}_{(q,p)}(\tau) \leq \sigma^{(w', \mathbf{j}^n)}_{(q,p)}(\tau)$. By Lemma 5.5.4, $\exists i_{n+1}, \dots, i_{n+k}$ such that all the occurrences of τ fall inside $\mathrm{Pstepreg}^p_q(w, \mathbf{i}^{n+k})$. **I** makes k moves choosing $i_{n+1}, \dots, i_{n+k}$ and **II** replies respectively with $j_{n+1}, \dots, j_{n+k}$. If $\sigma^{(w, \mathbf{i}^n)}_{(q,p)}(\tau) < \sigma^{(w', \mathbf{j}^n)}_{(q,p)}(\tau)$, then by Lemma 5.5.4 there exists an occurrence of τ in w' that falls inside $Free^p_q(w', \mathbf{j}^{n+k}) = \{1, \dots, |w'|\} \setminus \mathrm{Pstepreg}^p_q(w', \mathbf{j}^{n+k})$ and **I** chooses such an occurrence. Whatever is the reply of **II** in $Free^p_q(w, \mathbf{i}^{n+k})$, it holds that $(q-1)\text{-col}_w(i_{n+k+1}) \neq (q-1)\text{-col}_{w'}(j_{n+k+1})$, then by Theorem 5.4.5 $\mathbf{I}(\mathcal{G}_{q-1}((w, \mathbf{i}^{n+k+1}), (w', \mathbf{j}^{n+k+1})))$, that is, $\mathbf{I}(\mathcal{G}_{k+1+q-1=k+q}((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$. If, on the contrary, $\sigma^{(w, \mathbf{i}^n)}_{(q,p)}(\tau) = \sigma^{(w', \mathbf{j}^n)}_{(q,p)}(\tau)$, it must hold that $\rho^{(w, \mathbf{i}^n)}_{(q,p)}(\tau) \neq \rho^{(w', \mathbf{j}^n)}_{(q,p)}(\tau)$. If there exists an occurrence of τ in $Free^p_q(w', \mathbf{j}^{n+k})$, the strategy is the same of the previous case. If, on the contrary, all the occurrences of τ fall inside $\mathrm{Pstepreg}^p_q(w', \mathbf{j}^{n+k})$, then there exists an index z such that $q\text{-col}_w(i_z) \neq q\text{-col}_{w'}(j_z)$, and thus by Theorem 5.4.5 $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^{n+k}), (w', \mathbf{j}^{n+k})))$, that is, $\mathbf{I}(\mathcal{G}_{k+q}((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$. ■

Theorem 5.5.6 *[Main Theorem]*
Let $w, w' \in \Sigma^*$ and $p, q \in \mathbb{N}$, with $p > 1$. $\boldsymbol{II}(\mathcal{G}_q((w, \boldsymbol{i}^n), (w', \boldsymbol{j}^n)))$ *if and only if the following conditions hold:*

1. *$(w, w', \boldsymbol{i}^n, \boldsymbol{j}^n)$ is q-locally-safe;*
2. *for all $(r-1)$-color $\tau \in \Delta^{(w, \boldsymbol{i}^n)}_{(w', \boldsymbol{j}^n)}$, with $1 \leq r \leq q$, $\sigma^{(w, \boldsymbol{i}^n)}_{(i,p)}(\tau) > q - r$ and $\sigma^{(w', \boldsymbol{j}^n)}_{(i,p)}(\tau) > q - r$.*

Proof. ($\Leftarrow$) If $(w, w', \mathbf{i}^n, \mathbf{j}^n)$ is not q-distance-safe (resp., not $<_p$-safe for q-colors), then $\mathbf{I}(\mathcal{G}_q((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$ by Theorem 5.3.17 (resp. 5.4.5). If 2. does not hold, then $\mathbf{I}(\mathcal{G}_{r+q-r=q}((w, \mathbf{i}^n), (w', \mathbf{j}^n)))$ by Theorem 5.5.5.
($\Rightarrow$) Let us suppose $(w, w', \mathbf{i}^{n+k} \mapsto \mathbf{j}^{n+k})$ satisfies conditions 1. and 2. **I** chooses i_{n+k+1}. There are two cases:

- $i_{n+k+1} \in \mathrm{Pstepreg}^p_{q-k}(w, \mathbf{i}^{n+k})$. In this case the proof is the same of Theorem 5.4.7.

- $i_{n+k+1} \in Free^{p}_{q-k}(w, \mathbf{i}^{n+k})$. Let τ be the $(q-k-1)$-color of i_{n+k+1}. It suffices to choose $j_{n+k+1} \in Free^{p}_{q-k}(w', \mathbf{j}^{n+k})$ such that the $(q-k-1)$-color of j_{n+k+1} is τ. Ab absurdo, let us suppose such j_{n+k+1} does not exist, that is, let us assume all the occurrences of τ in w' fall inside $\mathrm{Pstepreg}^{p}_{q-k}(w', \mathbf{j}^{n+k})$. For Lemma 5.5.4 it holds that $\sigma^{(w',\mathbf{j}^n)}_{(q-k,p)}(\tau) \leq k = q-r$. Then, $\rho^{(w,\mathbf{i}^n)}_{(q-k,p)}(\tau) = \rho^{(w',\mathbf{j}^n)}_{(q-k,p)}(\tau)$. Moreover, for hypothesis $(w, w', \mathbf{i}^{n+k}, \mathbf{j}^{n+k})$ is $<_p$-safe for $(q-k)$-colors, then all the occurrences of τ in w fall inside $\mathrm{Pstepreg}^{p}_{q-k}(w, \mathbf{i}^{n+k})$. This contradicts the fact that $i_{n+k+1} \in Free^{p}_{q-k}(w, \mathbf{i}^{n+k})$. We can conclude that there exists $j_{n+k+1} \in Free^{p}_{q-k}(w', \mathbf{j}^{n+k})$ such that the $(q-k-1)$-color of j_{n+k+1} is τ.

■

A
Appendix 2

Hereafter we report the Biocham models of irinotecan, p53/Mdm2, and mammalian cell cycle. Coupling rules are included.

A.1 irinotecan.bc

```
parameter(injection,10).

add_event(Time>=1,injection,0).
add_event(Time>=36,injection,10).
add_event(Time>=37,injection,0).
add_event(Time>=72,injection,10).
add_event(Time>=73,injection,0).

macro(inj,injection).

injection for _ => CPT11.

% Activation by hydrolysis
parameter(k1,0.6).
parameter(Km1,0.2).
parameter(CES,1).
k1*[CPT11]*CES/(Km1+[CPT11]) for CPT11 => SN38.

% Deactivation by glucoronidation
parameter(UGT1A1,1).
parameter(n,3).
parameter(k2,0.5).
parameter(Km2,2.5).
k2*UGT1A1*[SN38]^n/(Km2^n+[SN38]^n) for SN38 =>SN38G.

% ABC transporters
parameter(kpgp2,0.0001).
parameter(Kpgp2,2.5).
kpgp2*[ABCG2]*[CPT11]/([CPT11]+Kpgp2) for CPT11 =[ABCG2]=> _.
parameter(kpgp1,0.2).
parameter(Kpgp1,2).
kpgp1*[ABCG2]*[SN38]/([SN38]+Kpgp1) for SN38 =[ABCG2]=> _.
```

```
% DNA damage
parameter(kcompl,0.05).
parameter(kdecompl,0.0001).
macro(DNAfree,DNAtot-[TOP1cc]-[DNAdam]).
(MA(kcompl)*DNAfree,MA(kdecompl)) for SN38 + TOP1 <=> TOP1cc.
parameter(kirr,1).
MA(kirr) for TOP1cc => DNAdam.

% degradations
parameter(kd3,0.08).
MA(kd3) for SN38G => _.
parameter(kdtop1,0.5).
MA(kdtop1) for TOP1 => _.

% synthesis of TOP1
parameter(top1,0.1).
top1 for _ => TOP1.

% initial state
present(ABCG2,0.15).
present(TOP1,1.4).
parameter(DNAtot,10).

% Rules of p53Mdm2 and cellcycle models are loaded
add_biocham('C:/Documents and
settings/Elisabetta/Desktop/elisabetta/p53Mdm2.bc').
add_biocham('C:/Documents and
Settings/Elisabetta/Desktop/elisabetta/cellcycle.bc').

% Redefinition of some parameters for p53Mdm2
parameter(ks53,3.3).
parameter(kd53p,0.33).
parameter(kd53,480).
parameter(ks53,3.3).
parameter(kf,528).
parameter(kr,150).
parameter(kdDNA,1.02).
parameter(kd2p,0.6).
parameter(ks2,0.36).
parameter(ks2p,0.09).
parameter(kph,3).
parameter(kdeph,360).
parameter(ki,840).
parameter(ko,30).
parameter(kd2pp,0.6).

% removal of IR
delete_rules(_ => DNAdam).
delete_event(Time>=10,IR,1).
delete_event(Time>=20,IR,0).

show_macros({inj,DNA}).
hide_molecules(?).
show_molecules({CPT11,TOP1,p53,Mdm2::n,DNAdam,CycA}).
```

A.2 p53Mdm2.bc

```
parameter(ratio,15).
volume(n,1/ratio).

%p53
(ks53,MA(kd53p)) for _ <=> p53.
MA(kf) for p53  =[Mdm2::n]=>p53~{u}.
MA(kr) for p53~{u}  => p53.
MA(kd53p) for p53~{u} => _.
MA(kf) for p53~{u}  =[Mdm2::n]=> p53~{uu}.
MA(kr) for p53~{uu}=>p53~{u}.
(kd53+kd53p)*[p53~{uu}] for p53~{uu} => _.
macro(p53tot,[p53]+[p53~{u}]+[p53~{uu}]).
parameter(ks53,0.055).
parameter(kd53p,0.0055).
parameter(kd53,8).
parameter(ks53,0.055).
parameter(kf,8.8).
parameter(kr,2.5).

% DNA damage
(kDNA*IR,MM(kdDNA*p53tot,Jdna)) for _ <=> DNAdam.
parameter(IR,0).
add_event(Time>=10,IR,1).
add_event(Time>=20,IR,0).
parameter(kDNA,0.18).
parameter(kdDNA,0.017).
parameter(Jdna,1).
macro(ir,IR).

% Mdm2
(ks2p,MA(kd2p)) for _ <=> Mdm2::c.
ks2*p53tot^m/(Js^m+p53tot^m) for _ =[p53]=> Mdm2::c.
parameter(kd2p,0.01).
parameter(ks2,0.006).
parameter(ks2p,0.0015).
parameter(Js,1.2).
parameter(m,3).
(kph*[Mdm2::c]/(Jph+p53tot),MA(kdeph)) for Mdm2::c <=> Mdm2~{p}::c.
MA(kd2p) for Mdm2~{p}::c => _.
parameter(kph,0.05).
parameter(Jph,0.01).
parameter(kdeph,6).
(MA(ko),MA(ki)) for Mdm2::n<=> Mdm2~{p}::c.
parameter(ki,14).
parameter(ko,0.5).
kd2p_n*[Mdm2::n] for Mdm2::n => _.
[Mdm2::n]*[DNAdam]*kd2pp_n/(Jdam+[DNAdam]) for Mdm2::n =[DNAdam]=>_.
parameter(kd2pp,0.01).
parameter(Jdam,0.2).
macro(kd2p_n,kd2p/ratio).
macro(kd2pp_n,kd2pp/ratio).
macro(kd2,kd2p+[DNAdam]*kd2pp/(Jdam+[DNAdam])).

% Interaction with cell cycle
present(p21,0.5).
parameter(km221,0.3).
parameter(A21,0.3).
MA(km221) for p21=[Mdm2::n]=>_. %Mdm2 inhibits p21
MA(A21)   for CycA=[p21]=>_. %p21 inhibits CycA

% initial state: steady state, without damaged dna
present (p53,0.04).
present (p53~{u},0.02).
present (p53~{uu},0.01).
present (Mdm2::n,0.33).
present (Mdm2::c,0.12). present
(Mdm2~{p}::c,0.01).
```

A.3 cellcycle.bc

```
% CycD stands for Cdk4-CycD
% CycE stands for Cdk2-CycE
% CycA stands for Cdk2-CycA
% CycB stands for Cdk1-CycB

% Initial state
present(ERG,0.01218087133020163).
present(DRG,0.9005327820777893).
present(CycD,0.012876115970612).
present(CycD-Kip1,0.434299469).
present(Kip1,0.8491207979619503).
present(CycE,0.01693358868360519).
present(CycE-Kip1,0.4828373789787).
present(massT,1.07598838806).
present(Cdc20,0.00001).
present(CycA,0.001462333556264639).
present(CycA-Kip1,0.0403826870955527).
present(E2F,4.957176518440247).
present(CycB,0.0025).
present(Cdh1,0.99).
present(Cdc20_T,0.0001).
present(IEP,0.000017).
present(PPX,1).
present(GM,0.93809711933136).

% TOP1 production
parameter(top1bis,0.5).
top1bis*[CycA] for _=[CycA]=>TOP1.

epsilon*k15/(1+([DRG]/J15)^2)              for _=>ERG.          % eq(1)
MA(k16)                                    for ERG=>_.          % eq(1)
MA(epsilon*k17p)                           for _=[ERG]=>DRG.    % eq(2)
epsilon*k17*([DRG]/J17)^2/(1+([DRG]/J17)^2)  for _=[DRG]=>DRG.  % eq(2)
```

```
MA(k18)                                       for DRG=>_.              % eq(2)
MA(epsilon*k9)                                for _=[DRG]=>CycD.       % eq(3)
MA(V6)                                        for _=[CycD-Kip1]=>CycD.  % eq(3)
MA(k10)                                       for CycD=>_.             % eq(3)
(MA(k24),MA(k24r))                      for CycD+Kip1<=>CycD-Kip1.% eq(3)(4)(9)
MA(V6+k10)                                    for CycD-Kip1=>_.        % eq(4)
epsilon*(k7p+k7*E2F_A)                        for _=>CycE.             % eq(5)
MA(V8)                                        for CycE=>_.             % eq(5)
MA(V6)                                        for _=[CycE-Kip1]=>CycE.  % eq(5)
(MA(k25),MA(k25r))                      for CycE+Kip1<=>CycE-Kip1.% eq(5)(6)(9)
MA(V6+V8)                                     for CycE-Kip1=>_.        % eq(6)
epsilon*k29*E2F_A*mass                        for _=[massT]=>CycA.     % eq(7)
MA(k30)                                       for CycA=[Cdc20]=>_.     % eq(7)
MA(V6)                                        for _=[CycA-Kip1]=>CycA.  % eq(7)
(MA(k25),MA(k25r))                      for CycA+Kip1<=>CycA-Kip1.% eq(7)(8)(9)
MA(V6)                                        for CycA-Kip1=>_.        % eq(8)
MA(k30)                                       for CycA-Kip1=[Cdc20]=>_. % eq(8)
epsilon*k5                                    for _=>Kip1.             % eq(9)
MA(V6)                                        for Kip1=>_.             % eq(9)
MA(k10)                                       for _=[CycD-Kip1]=>Kip1.  % eq(9)
MA(V8)                                        for _=[CycE-Kip1]=>Kip1.  % eq(9)
MA(k30)                     for Cdc20+CycA-Kip1=>Kip1+Cdc20+CycA-Kip1. % eq(9)
k22*E2F_T                                     for _=>E2F.              % eq(10)
MA(k22+k23p)                                  for E2F=>_.              % eq(10)
MA(k23)                                       for E2F=[CycA]=>_.       % eq(10)
MA(k23)                                       for E2F=[CycB]=>_.       % eq(10)
epsilon*k1p                                   for _=>CycB.             % eq(11)
epsilon*k1*([CycB]/J1)^2/(1+([CycB]/J1)^2)    for _=[CycB]=>CycB.      % eq(11)
MA(V2)                                        for CycB=>_.             % eq(11)
(k3p+k3*[Cdc20])*(1-[Cdh1])/(J3+1-[Cdh1])     for _=[Cdc20]=>Cdh1.     % eq(12)
V4*[Cdh1]/(J4+[Cdh1])                         for Cdh1=>_.             % eq(12)
epsilon*k11p                                  for _=>Cdc20_T.          % eq(13)
MA(epsilon*k11)                               for _=[CycB]=>Cdc20_T.   % eq(13)
MA(k12)                                       for Cdc20_T=>_.          % eq(13)
k13*[IEP]*([Cdc20_T]-[Cdc20])/(J13+[Cdc20_T]-[Cdc20])
                                                    for _=[IEP]=>Cdc20. % eq(14)
(k14/(J14+[Cdc20])+k12)*[Cdc20]               for Cdc20=>_.            % eq(14)
epsilon*k33                                   for _=>PPX.              % eq(15)
MA(k34)                                       for PPX=>_.              % eq(15)
k31*[CycB]*(1-[IEP])/(J31+1-[IEP])            for _=[CycB]=>IEP.       % eq(16)
k32*[PPX]*[IEP]/(J32+[IEP])                   for IEP=[PPX]=>_.        % eq(16)
k27*mass*(if Rb_hypo/Rb_T >0.8 then 0 else 1) for _=[massT]=>GM.       % eq(17)
MA(k28)                                       for GM=>_.               % eq(17)
MA(epsilon*mu*nbcells)                        for _=[GM]=>massT.       % eq(18)

% Steady-state relations
macro(PP1_A, PP1_T/(1+K21*(Phi_E*([CycE]+[CycA])+Phi_B*[CycB]))).   % eq(19)
macro(Rb_hypo, Rb_T/(1+(k20*(lambda_D*CycD_T+lambda_E*[CycE]+lambda_A*[CycA]+
+ lambda_B*[CycB]))/(k19p*(PP1_T-PP1_A)+k19*PP1_A))).  % eq(20)
macro(E2F_A,(E2F_T - E2FRb)*[E2F]/E2F_T).   %eq(21)
macro(E2FRb, 2*E2F_T*Rb_hypo/(E2F_T+Rb_hypo+L+
+((E2F_T+Rb_hypo+L)^2 - 4*E2F_T*Rb_hypo)^(1/2))).   % eq(22)

% Definitions
```

```
macro(V2, k2p*(1 - [Cdh1])+k2*[Cdh1]+k2s*[Cdc20]). %eq(23)
macro(V4, k4*(gamma_A*[CycA]+gamma_B*[CycB])). %eq(24)
macro(V6, k6p+k6*(eta_E*[CycE]+eta_A*[CycA]+eta_B*[CycB])). %eq(25)
macro(V8, k8p*(k8*(Psi_E*([CycE]+[CycA])+Psi_B*[CycB]))/(J8+CycE_T)). %eq(26)
macro(L, k26r/k26+k20/k26*(lambda_D*[CycD]+lambda_E*[CycE]+
+lambda_A*[CycA]+lambda_B*[CycB])). %eq(27)

macro(CycE_T, [CycE]+[CycE-Kip1]).
macro(CycD_T,[CycD]+[CycD-Kip1]).

add_event([Cdh1]>0.2,nbcells, nbcells*2).
macro(mass,[massT]/nbcells).
parameter(nbcells,0.45).

% Rate constants (h^-1)
% p stands for prime
% s stands for second
parameter(k1p,0.1).
parameter(k1,0.6).
parameter(k2p,0.05).
parameter(k2,20).
parameter(k2s,1).
parameter(k3p,7.5).
parameter(k3,140).
parameter(k4,40).
parameter(k5,20).
parameter(k6p,10).
parameter(k6,100).
parameter(k7p,0).
parameter(k7,0.6).
parameter(k8p,0.1).
parameter(k8,2).
parameter(k9,2.5).
parameter(k10,5).
parameter(k11p,0).
parameter(k11,1.5).
parameter(k12,1.5).
parameter(k13,5).
parameter(k14,2.5).
parameter(k15,0.25).
parameter(k16,0.25).
parameter(k17p,0.35).
parameter(k17,10).
parameter(k18,10).
parameter(k19p,0).
parameter(k19,20).
parameter(k20,10).
parameter(k22,1).
parameter(k23p,0.005).
parameter(k23,1).
parameter(k24,1000).
parameter(k24r,10).
parameter(k25,1000).
parameter(k25r,10).
parameter(k26,10000).
parameter(k26r,200).
parameter(k27,0.2).
parameter(k28,0.2).
parameter(k29,0.05).
parameter(k30,20).
parameter(k31,0.7).
parameter(k32,1.8).
parameter(k33,0.05).
parameter(k34,0.05).
parameter(mu,0.061).

%Dimensionless constants
parameter(J1,0.1).
parameter(J3,0.01).
parameter(J4,0.01).
parameter(J8,0.1).
parameter(J13,0.005).
parameter(J14,0.005).
parameter(J15,0.1).
parameter(J17,0.3).
parameter(J31,0.01).
parameter(J32,0.01).
parameter(K21,1).
parameter(E2F_T,5).
parameter(PP1_T,1).
parameter(Rb_T,10).
parameter(Phi_E,25).
parameter(Phi_B,2).
parameter(gamma_A,0.3).
parameter(gamma_B,1).
parameter(eta_E,0.5).
parameter(eta_A,0.5).
parameter(eta_B,1).
parameter(lambda_D,3.3).
parameter(lambda_E,5).
parameter(lambda_A,3).
parameter(lambda_B,5).
parameter(Psi_E,1).
parameter(Psi_B,0.05).
parameter(epsilon,1).

macro(CycA_T, [CycA]+[CycA-Kip1]).
macro(Kip1_T,[Kip1]+[CycA-Kip1]+[CycD-Kip1]+[CycE-Kip1]).
```

Bibliography

[1] B.D.Aguda and A.B.Goryachev. From Pathways Databases to Network Models of Switching Behavior. *PLoS Compututational Biology*, 3(9):1674-1678, 2007.

[2] B.Alberts, J.Alexander, L.Julian, R.Martin, and R.Keith. *Molecular biology of the cell.* Garland editor, 2008.

[3] C.B.Anfinsen. Principles that govern the folding of protein chain. *Science*, 181:223 230, 1973.

[4] M.Antoniotti, A.Policriti, N.Ugel, and B.Mishra. Model building and model checking for biochemical processes. *Cell Biochem Biophys*, 38(3):271-86, 2003.

[5] S.Arora and R.Fagin. On winning strategies in Erenfeucht-Fraïssé games. *Theoretical Computer Science*, 174:97-121, 1997.

[6] R.Backofen and S.Will. Excluding symmetries in constraint-based search. *Journal of Constraints*, 7(3):333-349, 2002.

[7] R.Backofen and S.Will. A Constraint-Based Approach to Structure Prediction for Simplified Protein Models that Outperforms Other Existing Methods. *In Proc. of the International Conference on Logic Programming* (ICLP), Springer, pp. 49-71, 2003.

[8] R.Backofen and S.Will. A constraint-based approach to fast and exact structure prediction in three-dimensional protein models. *Journal of Constraints*, 11(1):5-30, 2006.

[9] G.J.Barton and M.J.E.Stenberg. A strategy for the rapid multiple alignment of protein sequences. *Journal of Molecular Biology*, 198:327-337, 1987.

[10] G.Batt, C.Belta, and R.Weiss. Temporal Logic Analysis of Gene Networks under Parameter Uncertainty. *IEEE Transactions of Automatic Control*, 53:215-229, 2008.

[11] M.Berrera, H.Molinari, and F.Fogolari. Amino acid empirical contact energy definitions for fold recognition. *BMC Bioinformatics*, 4(8), 2003.

[12] A.Biere, A.Cimatti, E.M.Clarke, O.Strichman, and Y.Zhu. *Bounded Model Checking.* Advances in Computers, vol. 58, Academic Press, 2003.

[13] R.Bonneau and D.Baker. Ab initio protein structure prediction: progress and prospects. *Annual Review of Biophysics and Biomolecular Structure*, 30:17389, 2001.

[14] C.L.Brooks, M.Karplus, and B.M.Pettitt. Proteins: a Theoretical Perspective of Dynamics, Structure, and Thermodynamics. *Advances in Chemical Physics*, 71, Wiley, New York, 1988.

[15] B.R.Brooks et al. Charmm: A program for macromolecular energy minimization and dynamics. *Journal of Computational Chemestry*, 4:187217, 1983.

[16] T.A.Brown. *Genomes.* BIOS Scientific Publishers Ltd, Second Edition, 2002.

[17] M.Calder, V.Vyshemirsky, D.Gilbert, and R.Orton. Analysis of signalling pathways using the continuous time markow chains. In C.Priami and G.Plotkin,(eds.). *Transactions on Computational Systems Biology VI*, vol. 4220 of LNCS (LNBI), pp. 4467. Springer, Heidelberg, 2006.

[18] L.Calzone, N.Chabrier-Rivier, F.Fages, and S.Soliman. Machine learning biochemical networks from temporal logic properties. In C.Priami and G.Plotkin, (eds.). *Transactions on Computational Systems Biology VI*, vol. 4220 of LNCS (LNBI), pp. 68-94, Springer, Heidelberg, 2006.

[19] L.Calzone, F.Fages, and S.Soliman. BIOCHAM: An Environment for Modeling Biological Systems and Formalizing Experimental Knowledge. *Bioinformatics* 22: 1805-1807, 2006.

[20] N.Carreiro and D.Gelernter. Linda in Context. *Comm. of the ACM*, 32(4):444-458, 1989.

[21] H.Carrillo and D.Lipman. The multiple sequence alignment problem in biology. *SIAM Journal of Applied Mathematics*, 48:1073-1082, 1988.

[22] N.Chabrier and F.Fages. Symbolic model checking of biochemical networks. In *Proc. of the 1st Workshop on Computational Methods in Systems Biology* (CMSB), C.Priami (ed.), vol. 2602 of LNCS, pp. 149-162, Springer, 2003.

[23] H.F.Chan and K.A.Dill. The protein folding problem. *Physics Today*, pp. 24-32, 1993.

[24] V.Chickermane, A.Ray, H.M.Sauro, and A.Nadim. A model for p53 Dynamics Triggered by DNA damage. *Siam Journal on Applied Dynamical Systems*, 6:1, pp.61-78, 2007.

[25] A.Ciliberto, B.Novak, and J.J.Tyson. Steady States on Oscillations in the p53/Mdm2 Network. *Cell Cycle*, 4:3, pp.488-493, 2005.

[26] A.Cimatti, E.Clarke, F.Giunghiglia, E.Giunghiglia, M.Pistore, M.Roveri, R.Sebastiani, and A.Tacchinella. Nusmv 2: An opensource tool for symbolic model checking. In *Proc. of the International Conference on Computer-Aided Verification* (CAV' 02), vol. 2404 of LNCS, pp. 27-31, 2002.

[27] E.M.Clarke, O.Grumberg, and D.A. Peled. Model Checking. Cambridge, Mass.: MIT Press, 1999.

[28] P.Clote and R.Backofen. *Computational Molecular Biology.* John Wiley & Sons, 2001.

[29] C.Courcoubetis, M.Y.Vardi, P.Wolper, and M.Yannakakis. Memory efficient algorithms for the verification of temporal properties. *Formal Methods in System Design*, 1:275-288, 1992.

[30] T.E.Creighton. *Proteins. Structure and Molecular Principles.* Freeman, New York, 1984.

[31] P.Crescenzi, D.Goldman, C.Papadimitriou, A.Piccolboni, and M.Yannakakis. On the complexity of protein folding. *Journal of Computational Biology*, 5:597–603, 1998.

[32] A.Dal Palù, A.Dovier, and F.Fogolari. Constraint logic programming approach to protein structure prediction. *BMC Bioinformatics*, 5(186), 2004.

[33] A.Dal Palù, A.Dovier, and E.Pontelli. A Constraint Logic Programming Approach to 3D Structure Determination of Large Protein Complexes. In *Proc. of the 12th International Conference on Logic Programming and Automated Reasoning* (LPAR), vol. 3835 of LNAI, pp. 48-63, 2005.

[34] A.Dal Palù, A.Dovier, and E.Pontelli. A constraint solver for discrete lattices, its parallelization, and application to protein structure prediction. *Software: Practice and Experience*, 37(13):1405-1449, 2007.

[35] E.De Maria, A.Dovier, A.Montanari, and C.Piazza. Exploiting Model Checking in Constraint-based Approaches to the Protein Folding. In *Proc. of the Workshop on Constraint-Based Methods for Bioinformatics* (WCB), pp.46-54, 2006.

[36] E.De Maria, A.Montanari, and N.Vitacolonna. Games on Strings with a Limited Order Relation. In S.Artemov and A.Nerode, (eds.), *Proc. of Symposium on Logical Foundations of Computer Science* (LFCS09), vol. 5407 of LNCS, pp. 164-179, Springer, 2009.

[37] K.A.Dill, S.Bromberg, K.Yue, K.M.Fiebig, D.P.Yee, P.D.Thomas, and H.S.Chan. Principles of protein folding - a perspective from simple exact models. *Protein Science*, 4:561602, 1995.

[38] L.Dimitrio. Irinotecan: Modelling intracellular pharmacokinetics and pharmacodynamics. M2 master thesis (in French, English summary), University Pierre-et-Marie-Curie and INRIA internal report, June 2007.

[39] R.Durbin, S.Eddy, A.Krogh, and G.Mitchison. Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, 1998.

[40] A.Ehrenfeucht. An application of games to the completeness problem for formalized theory. *Fundamenta Mathematicae*, 49:129-141, 1961.

[41] S.Eker, M.Knapp, K.Laderoute, P.Lincoln, J.Meseguer, and M.K. Sönmez. Pathway logic: Symbolic analysis of biological signaling. In *Proceedings of the seventh Pacific Symposium on Biocomputing*, pp. 400412, 2002.

[42] R.Elber and M.Karplus. Multiple conformational states of proteins: a molecular dynamics analysis of myoglobin. *Science*, 235:318321, 1987.

[43] S.J.Elledge. Cell cycle checkpoints: preventing an identity crisis. *Science*, 274:16641672, 1996.

[44] E.A.Emerson. Temporal and modal logic. In *the Handbook of Theoretical Computer Science*, Volume B (chapter 16), pp. 995-1072, J. van Leeuwen (Ed.), Elsevier Science Publisher, 1990.

[45] F.Fages. Temporal logic constraints in the biochemical abstract machine biocham (invited talk). In P.M.Hill, (ed.), *LOPSTR 2005*, vol. 3901 of LNCS, Springer, Heidelberg, 2006.

[46] F.Fages, S.Soliman, and N. Chabrier-Rivier. Modelling and querying interaction networks in the biochemical abstract machine BIOCHAM. *Journal of Biological Physics and Chemistry* 4(2), pp.64-73, 2004.

[47] R.Fagin, L.Stockmeyer, and M.Y.Vardi. On monadic NP vs. monadic co-NP. *Inform. and Comput.*, 120(1):78-92, 1995.

[48] D.F.Feng and R.F.Doolittle. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. *Journal of Molecular Evolution*, 25:351-369, 1987.

[49] J.C.Fernandez, C.Jard, T.Jeron, and G.Viho. *Using on-the-fly verification techniques for the generation of test suites.* In *Proceedings of the International Conference on Computer-Aided Verification (CAV)*, vol. 1102 of LNCS, pp. 348-359, Springer, 1996.

[50] C.Floudas, J.L.Klepeis, and P.M.Pardalos. Global optimization approaches in protein folding and peptide docking. In F. Roberts, (ed.), *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 1999.

[51] F.Fogolari, G.Esposito, P.Viglino, and S.Cattarinussi. Modeling of polypeptide chains as C-α chains, C-α chains with C-β, and C-α chains with ellipsoidal lateral chains. *Biophysical Journal*, 70:11831197, 1996.

[52] R.Fraïssé. Sur quelques classifications des systmes de relations. *Publications Scientifiques*, 1:35-182, 1954.

[53] H.Gaifman. On local and nonlocal properties. In J.Stern, editor, *Proceedings of the Herbrand Symposium, Logic Colloquium '81*, pp.105-135, North Holland Pub. Co., 1982.

[54] N.Geva-Zatorsky, N.Rosenfeld, S.Itzkovitz, R.Milo, A.Sigal, E.Dekel, T.Yarnitzky, Y.Liton, P.Polak, G.Lahav, and U.Alon. Oscillations and variability in the p53 system. *Molecular System Biology*, 2006.0033, 2006.

[55] M.A.Gibson, and J.Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *Journal of Physical Chemistry* 104, 1876 1889, 2000.

[56] D.Gilbert, M.Heiner, and S.Lehrack. A unifying framework for modelling and analysing biochemical pathways using petri nets. In M.Calder and S.Gilmore, (eds.), *CMSB 2007*, vol. 4695 of LNCS (LNBI), Springer, Heidelberg, 2007.

[57] D.T.Gillespie. General method for numerically simulating stochastic time evolution of coupled chemical-reactions. *Journal of Computational Physics* 22, 403434, 1976.

[58] K.Ginalski. Comparative modeling for protein structure prediction. *Current opinion in structural biology*, 16(2):172177, 2006.

[59] A.Godzik, A.Kolinski, and J.Skolnick. Lattice representations of globular proteins: how good are they? *Journal of Computational Chemistry*, 14:11941202, 1993.

[60] D.Gusfield. Algorithms on strings, trees, and sequences: computer science and computational biology. Cambridge University Press, 1997.

[61] W.Hanf. Model-Theoretic Methods in the Study of Elementary Logic. In J.W.Addison, L.Henkin, and A.Tarsky, (eds.), *The Theory of Models*, pp. 132-145, North-Holland, Amsterdam, 1965.

[62] H.Hansson and B.Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing* 6, 512535, 1994.

[63] W.E.Hart and S.C.Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eighths of optimal. *Journal of Computational Biology*, 3(1):53-96, 1996.

[64] W. E.Hart and S.C.Istrail. Lattice and off-lattice side chain models of protein folding: Linear time structure prediction better than 86% of optimal. *Journal of Computational Biology*, 4(3):241-259, 1997.

[65] D.Haussler, A.Krogh, I.S.Mian, and K.Sjölander. Protein modeling using hidden Markov models: analysis of globins. In *Proc. on the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, 1:792-802, 1993.

[66] J.Heath, M.Kwiatkowska, G.Norman, D.Parker and O.Tymchyshyn. Probabilistic model checking of complex biological pathways. In C.Priami (ed.), *Proc. of the 4th Workshop on Computational Methods in Systems Biology* (CMSB), vol. 4210 of Lecture Notes in Bioinformatics, pp. 32-47, Springer, 2006.

[67] W.S.Hlavacek, J.R.Faeder, M.L.Blinov, R.G.Posner, M.Hucka, and W.Fontana. Rules for modeling signal-transduction systems. *Science STKE*, 344:6,2006.

[68] G.J.Holzmann. The Spin Model Checker: Primer and Reference Manual. Addison-Wesley, 2003.

[69] M.Hucka, et al. The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models. *Bioinformatics*, 19:524531, 2003.

[70] D.T.Jones. Predicting novel protein folds by using FRAGFOLD. *Proteins*, 5:127 131, 2001.

[71] H.J.Keisler and W.B.Lotfallah. Shrinking games and local formulas. *Annals of Pure and Applied Logic*, 128:215-225, 2004.

[72] B.Khoussainov and J.Liu. On Complexity of Ehrenfeucht-Fraïssé Games. In S.Artemov and A.Nerode, (eds.), *Proc. of LFCS 2007*, vol. 4514 of LNCS, pp. 293-309, Springer, 2007.

[73] H.Kitano, A.Funahashi, Y.Matsuoka, and K.Oda. Using process diagrams for the graphical representation of biological networks. *Nature Biotechnology*, 23:961-966, 2005.

[74] P.E.Kloeden and E.Platen. Numerical solution of stochastic differential equations. Springer-Verlag, New York, 1992.

[75] K.W Kohn and M.Aladjem. Circuit diagrams for biological networks. *Molecular Systems Biology* 2:2006.0002, 2006.

[76] K.W.Kohn and Y.Pommier. Molecular interaction map of the p53 and Mdm2 logic elements, which control the Off-Onn switch of p53 in response to DNA damage. *Biochemical and Biophysical Research Communications*, 331:816-827, 2005.

[77] A.Kolinski and J.Skolnick. Reduced models of proteins and their applications. *Polymer*, 45:511-524, 2004.

[78] A.Krogh, M.Brown, I.Mian, K.Sjölander, and D.Haussler. Hidden Markov models in computational biology: Applications to protein modelling. *Journal of Molecular Biology*, 235:1501-31, 1994.

[79] A.Krogh. Hidden Markov models for labeled sequences. In *Proc. of the 12th IAPR International Conference on Pattern Recognition*, pp.140-144, IEEE Computer Society Press, 1994.

[80] Y.Kuramoto, K.Hata, S.Koyonagi, S.Ohdo, H.Shimeno, S.Soeda. Circadian regulation of mouse topoisomerase I gene expression by glucorticoid hormones. *Biochemical Pharmacology*, 71: 1155-1161, 2006.

[81] M.Z.Kwiatkowska, G.Norman, and D.Parker. Prism 2.0: A tool for probabilistic model checking. In *First International Conference on Quantitative Evaluation of Systems* (QEST 2004), pp. 322323, IEEE Computer Society, Los Alamitos, 2004.

[82] G.Lahav, N.Rosenfeld, A.Sigal, N.Geva-Zatorsky, A.J.Levine, M.B.Elowitz, and U.Alon. Dynamics of the p53-Mdm2 feed-back loop in individual cells. *Nature Genetics*, 36:147-150, 2004.

[83] J.D.Lambert. *Numerical Methods for Ordinary Differential Systems*. J.Wiley and Sons, Chichester, 1991.

[84] K.F.Lau and K.A.Dill. A lattice statistical mechanics model of the conformational and sequence spaces of proteins. *Macromolecules*, 22:3986-3997, 1989.

[85] N.Le Novère et al. Systems Biology Graphical Notation: Process Diagram Level 1. *Nature Precedings*: hdl:10101/npre.2008.2320.1, 2008.

[86] C.Levinthal. Are there pathways to protein folding? *Journal de Chimie Physique*, 65:4445, 1968.

[87] A.D.MacKerell, Jr. et al. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *Journal of Physical Chemestry B*, 102:35863616, 1998.

[88] N.Madras and G.Slade. The self avoiding walk. Boston, MA: Birkhäuser, 1993.

[89] N.Madras and A.D.Sokal. Nonergodicity of local, length-conserving Monte Carlo algorithms for the self-avoiding walk. *Journal of Statistical Physics*, 47(3-4):573-595, 1987.

[90] N.Madras and A.D.Sokal. The pivot algorithm: A highly efficient Monte Carlo method for the self-avoiding walk. *Journal of Statistical Physics*, 50(1-2):109-186, 1988.

[91] T.Matsuo, S.Yamaguchi, S.Mitsui, A.Emi, F.Shimoda, and H.Okamura. Control mechanism of the circadian clock for timing of cell division in vivo. *Science*,302:255259, 2003.

[92] K.L.McMillan. Symbolic Model Checking: An Approach to the State Explosion Problem. Kluwer Academic, 1993.

[93] P.G.Mezey. Potential energy hypersurfaces. *Studies in Physical and Theoretical Chemistry*, 53, 1987.

[94] K.M.Misura, D.Chivian, C.A.Rohl, D.E.Kim, and D.Baker. Physically realistic homology models built with ROSETTA can be more accurate than their templates. In *Proc. Natl. Acad. Sci. USA*, 103:5361-5366, 2006.

[95] A.Montanari, A.Policriti, and N.Vitacolonna. An Algorithmic Account of Winning Strategies in Ehrenfeucht Games on Labeled Successor Structures. In G.Sutcliffe and A.Voronkov, (eds.), *Proc. of the 12th International Conference on Logic for Programming, Artificial Intelligence and Reasoning* (LPAR 2005), vol. 3835 of LNCS, pp. 139-153, Springer, December 2005.

[96] A.Montanari, A.Policriti, and N.Vitacolonna. On Devising Algorithms for Ehrenfeucht-Fraïssé Games. In *The Annual GAMES-Meeting*, Lausanne, Switzerland, Sept. 2007.

[97] S.L.Moodie, A.A.Sorokin, I.Goryanin, and P.Ghazal. A graphical notation to describe the logical interactions of biological pathways. *Journal of Integrative Bioinformatics*, 3:36, 2006.

[98] M.C.Mormont, and F.Levi. Circadian system alterations during cancer processes. *International Journal of Cancer*, 70:241247, 1997.

[99] J.Moult. A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction. *Current opinion in structural biology*, 15:285289, 2005.

[100] S.B.Needleman and C.D.Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443-453, 1970.

[101] A.Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39:407460, 1997.

[102] B.Novák and J.J.Tyson. A model for restriction point control of the mammalian cell cycle. *Journal of Theoretical Biology* 230, pp.563-579, 2004.

[103] P.M.Pardalos, D.Shalloway, and G.Xue. Optimization methods for computing global minima of nonconvex potential energy functions. *Journal of Global Optimization*, 4, pp. 117-133, 1994.

[104] E.Pezzoli. Computational Complexity of Ehrenfeucht-Fraïssé Games on Finite Structures. In G-Gottlob, E.Grandjean, and K.Seyr, (eds.), *Proc. of CSL 1998*, vol. 1584 of LNCS, pp. 159-170, Springer, August 1999.

[105] A.Phillips and L.Cardelli. A correct abstract machine for the stochastic pi-calculus. In *Proc. of Concurrent Models in Molecular Biology* (Bioconcur'04), affiliated with CONCUR'04, 2004.

[106] C.Piazza, M.Antoniotti, V.Mysore, A.Policriti, F.Winkler, and B.Mishra. Algorithmic Algebraic Model Checking I: Challenges from Systems Biology. In *Proc. of the 17th International Conference on Computer Aided Verification* (CAV), vol. 3576 of LNCS, pp. 5-19, Springer, 2005.

[107] Y.Pommier. Camptothecins and Topoisomerase I: A Foot in the Door. Targeting the genome beyond Topoisomerase I with camptothecins and Novel Anticancer drugs: importance of DNA Replication, Repair and Cell Cycle Checkpoints. Preprint NIH (NCI), http://discover.nci.nih.gov/pommier/Pommier.Top1.pdf, 2004.

[108] A.Ponitz and P.Tittmann. Improved upper bounds for self-avoiding walks in Z^d. *Electronic J. Combinatorics* 7(1):1-19, 2000.

[109] J.P.Quielle and J.Sifakis. Specification and verification of concurrent systems in CESAR. In *Proc. 5th International Symposium on Programming*, vol. 137 of LNCS, pp. 337-351, Springer, 1982.

[110] A.Regev, W.Silverman, and E.Y.Shapiro. Representation and simulation of biochemical processes using the pi-calculus process algebra. In *Proc. of the sixth Pacific Symposium of Biocomputing*, pp. 459470, 2001.

[111] W.G.Richards. Calculation of conformational free energy of histamine. *Journal of Theoretical Biology*, 43, pp. 389, 1974.

[112] B.Rost, R.Schneider, and C.Sander. Protein fold recognition by prediction-based threading. *Journal of Molecular Biology*, 1996.

[113] A.Šali, E.Shakhnovich, and M.Karplus. How does a protein fold? *Nature*, 369:248 251, 1994.

[114] K.A.Schafer. The Cell Cycle: A Review. *Veterinary Pathology*, 35:461-478, 1998.

[115] T.Schwentick. On winning Ehrenfeucht games and monadic NP. *Annals of Pure and Applied Logic*, 79:61-92, 1996.

[116] S.Ohdo, T.Makinosumi, T.Ishizaki, E.Yukawa, S.Higuchi, S. Nakano, and N. Ogawa. Cell Cycle-Dependent Chronotoxicity of Irinotecan Hydrochloride in Mice. *Journal of Pharmacology and Experimental Terapeutics* 283(3):1383-1388, 1997.

[117] B. Rost. Protein structure prediction in 1D, 2D and 3D. *Encyclopedia of Computational Chemistry*, John Wiley & Sons, pp. 2242–2255, 1998.

[118] W.Sitao, J.Skolnick, and Y.Zhang. Ab initio modeling of small proteins by iterative TASSER simulations. *BMC Biology*, vol 5(17), 2007.

[119] J.Skolnick and A.Kolinski. Reduced models of proteins and their applications. *Polymer*, 45:511524, 2004.

[120] T.F.Smith and M.S.Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195-197, 1981.

[121] S.Soliman, and F.Fages. CMBSlib: a library for comparing formalisms and models of biological systems. In V.Danos and V.Schachter, (eds.), CMSB 2004, vol. 3082 of LNCS (LNBI), pp. 231235, Springer, Heidelberg ,2005.

[122] J.D.Thompson, D.G. Higgins, and T.J.Gibson. CLUSTAL W: improving the sensivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. *Nucleic Acid Research*, 22:4673-4680, 1994.

[123] J.J.Tyson, K.C.Chen, and B.Novák. Sniffers, buzzers, toggles and blinkers: dynamics of regulatory and signaling pathways in the cell. *Current Opinion in Cell Biology*, 15:221231, 2003.

[124] T.Veitshans, D.Klimov, and D.Thirumalai. Protein folding kinetics: timescales, pathways and energy landscapes in terms of sequence-dependent properties. *Folding and Design*, 2:122, 1996.

[125] L.Wang and T.Jiang. On the complexity of multiple sequence alignment. *Journal of Molecular Biology*, 1(4):337-348, 1994.

[126] L.T. Wille and J.Vennik. Computational complexity of the ground-state determination of atomic clusters. *Journal of Physics A: Mathematical and General*, 18(8): L419 L422, 1985.

[127] B.Woebking, G.Reuter, R.A.Shilling, S.Velamakanni, S.Shahi, H.Venter, L.Balakrishnan, and H.W.Van Veen. Drug-Lipid A. Interactions on the Escherichia coli ABC transporter MsbA. *Journal of Bacteriology*, 187:18, 6363-6369, 2005.

[128] Y.Zhang, A.K.Akaraki, and J.Skolnick. TASSER: an automated method for the prediction of protein tertiary structures in CASP6. *Proteins*, 7S:9198, 2005.

[129] Y.Zhou, F.G.Gwadry, W.C.Reinhold, L.D.Miller, L.H.Smith, U.Scherf, E.T.Liu, K.W.kohn, Y.Pommier, and J.N.Weinstein. Transcriptional Regulation of Mitotic Genes by Camptothecin-induced DNA Damage : Microarray Analysis of Doseand Time-dependent Effects. *Cancer Research* 62, 1668-1695, 2002.